UCLA
UCLA Electronic Theses and Dissertations

Title
Teamwork and Exploration in Reinforcement Learning

Permalink
https://escholarship.org/uc/item/9b29203s

Author
Cassano, Lucas

Publication Date
2020

Supplemental Material
https://escholarship.org/uc/item/9b29203s#supplemental

Peer reviewed|Thesis/dissertation

eScholarship.org Powered by the California Digital Library
University of California

UNIVERSITY OF CALIFORNIA

Los Angeles

Teamwork and Exploration in Reinforcement Learning

A dissertation submitted in partial satisfaction
of the requirements for the degree
Doctor of Philosophy in Electrical Engineering

by

Lucas Cesar Eduardo Cassano

2020

© Copyright by
Lucas Cesar Eduardo Cassano
2020

ABSTRACT OF THE DISSERTATION

Teamwork and Exploration in Reinforcement Learning

by

Lucas Cesar Eduardo Cassano
Doctor of Philosophy in Electrical Engineering
University of California, Los Angeles, 2020
Professor Ali H. Sayed, Chair

Reinforcement learning (RL) is a powerful machine learning paradigm that studies the interaction between a single agent with an unknown environment. A plethora of applications fit into the RL framework, however, in many cases of interest, a team of agents will need to interact with the environment and with each other to achieve a common goal. This is the object study of collaborative multi-agent RL (MARL).

Several challenges arise when considering collaborative MARL. One of these challenges is decentralization. In many cases, due to design constraints, it is undesirable or inconvenient to constantly relay data between agents and a centralized location. Therefore, fully distributed solutions become preferable. The first part of this dissertation addresses the challenge of designing fully decentralized MARL algorithms. We consider two problems: policy evaluation and policy optimization. In the policy evaluation problem, the objective is to estimate the performance of a target team policy in a particular environment. This problem has been studied before for the case with streaming data, however, in most implementations the target policy is evaluated using a finite data set. For this case, existing algorithms guarantee convergence at a sub-linear rate. In this dissertation we introduce *Fast Diffusion for Policy Evaluation* (FDPE), an algorithm that converges at linear rate for the finite data set case. We then consider the policy optimization problem, where the objective is for all agents to learn an optimal team policy. This problem has also been studied recently, however, existing solutions are data inefficient and converge to Nash equilibria (whose performance can be

catastrophically bad) as opposed to team optimal policies. For this case we introduce the *Diffusion for Team Policy Optimization* (DTPO) algorithm. DTPO is more data efficient than previous algorithms and does not converge to Nash equilibria. For both of these cases, we provide experimental studies that show the effectiveness of the proposed methods.

Another challenge that arises in collaborative MARL, which is orthogonal to the decentralization problem, is that of scalability. The parameters that need to be estimated when full team policies are learned, grow exponentially with the number of agents. Hence, algorithms that learn joint team policies quickly become intractable. A solution to this problem is for each agent to learn an individual policy, such that the resulting joint team policy is optimal. This problem has been the object of much research lately. However, most solution methods are data inefficient and often make unrealistic assumptions that greatly limit the applicability of these approaches. To address this problem we introduce *Logical Team Q-learning* (LTQL), an algorithm that learns factored policies in a data efficient manner and is applicable to any cooperative MARL environment. We show that LTQL outperforms previous methods in a challenging predator-prey task.

Another challenge is that of efficient exploration. This is a problem both in the single-agent and multi-agent settings, although in MARL it becomes more severe due to the larger state-action space. The challenge of deriving policies that are efficient at exploring the state space has been addressed in many recent works. However, most of these approaches rely on heuristics, and more importantly, they consider the problem of exploring the state space separately from that of learning an optimal policy (even though they are related, since the state-space is explored to collect data to learn an optimal policy). To address this challenge, we introduce the *Information Seeking Learner* (ISL), an algorithm that displays state of the art performance in difficult exploration benchmarks. The fundamental value of our work on exploration is that we take a fundamentally different approach from previous works. As opposed to earlier methods we consider the problem of exploring the state space and learning an optimal policy jointly. The main insight of our approach is that in RL, obtaining point estimates of the quantities of interest is not sufficient and confidence bound estimates are also necessary.

The dissertation of Lucas Cesar Eduardo Cassano is approved.

Lieven Vandenberghe

Jonathan C. Kao

Christina Fragouli

Ali H. Sayed, Committee Chair

University of California, Los Angeles

2020

To my Family.

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF TABLES

ACKNOWLEDGMENTS

I would like to start by thanking Professor Ali H. Sayed for giving me the opportunity to be a member of the Adaptive Systems Laboratory. I would like to thank him for his guidance through my studies. His tireless and meticulous approach to scientific research has been, and still is, an inspiration to me.

I would also like to thank Professor Christina Fragouli, Professor Jonathan C. Kao and Professor Lieven Vandenberghe for agreeing to be a part of my committee.

I am also thankful to Deeona Columbia, Ryo Arreola, Mandy Smith and Patricia Vonlanthen for their help during my time at UCLA and EPFL. During my studies I am lucky to have met good friends, colleagues and collaborators who I also thank: Sulaiman Alghunaim, Stefan Vlaski, Kun Yuan, Chung-Kai Yu, Hawraa Salami, Bicheng Ying, Chengcheng Wang, Roula Nassif, Augusto Santos, Virginia Bordignon, Elsa Rizk, Guillermo Ortiz Jimenez, Mert Kayaalp, Konstantinos Ntemos and Professor Ricardo Merched.

I am thankful to my family, Daniel, Maria, Matías and Tomás, for their support.

This dissertation is based upon work partially supported by the National Science Foundation under grants CCF-1524250 and ECCS-1407712. Any opinions, findings, and conclusions or recommendations expressed in this material are those of the author and do not necessarily reflect the views of the National Science Foundation, the Department of Defense or the U.S. Government.

VITA

2012	Electronics Engineer, Buenos Aires Institute of Technology, Argentina.
2013	Freelance Engineer, Argentina.
2014	Signal Processing Engineer, Satellogic S. A., Buenos Aires City, Argentina.
2015	M.S. in Electrical Engineering, University of California, Los Angeles, CA, USA.
2015	Intern, Mojix Inc., Los Angeles, CA, USA.
2015–2018	Research and Teaching Assistant, Department of Electrical Engineering, University of California, Los Angeles, CA, USA.
2018–2020	Visiting Doctoral Assistant, École Polytechnique Fédérale de Lausanne, Switzerland.

PUBLICATIONS

Lucas Cassano and Ali H. Sayed, "Logical Team Q-learning: An approach towards optimal factored policies in cooperative MARL," submitted for publication, available as arXiv:2006.03553, June 2020.

Lucas Cassano and Ali H. Sayed, "ISL: A novel approach for deep exploration", submitted for publication, available as arXiv:1909.06293, January 2020.

Lucas Cassano and Ali H. Sayed, "ISL: Optimal Policy Learning With Optimal Exploration-Exploitation Trade-Off", NeurIPS Optimization Foundations of Reinforcement Learning

Workshop, Vancouver, Canada, December 2019.

Lucas Cassano, Kun Yuan and Ali H. Sayed, "Multi-Agent Fully Decentralized Value Function Learning with Linear Convergence Rates", in *IEEE Transactions on Automatic Control*, December 2021, to appear.

Lucas Cassano, Kun Yuan and Ali H. Sayed, "Distributed value-function learning with linear convergence rates", in *Proceedings of the European Control Conference* (ECC), pp. 505–511, Naples, Italy, June 2019.

Lucas Cassano, Sulaiman Alghunaim and Ali H. Sayed, "Team Policy Learning for Multi-agent Reinforcement Learning", in *Proceedings of the IEEE International Conference on Acoustics, Speech and Signal Processing* (ICASSP), pp. 3062–3066, Brighton, UK, May 2019.

CHAPTER 1

Introduction

In this chapter we introduce necessary background on reinforcement learning (RL), and provide an outline of the dissertation and its contributions.

1.1 Reinforcement Learning

Reinforcement learning (RL) research goes back to the early works [1–4], it studies the interaction between an agent with an environment that has unknown dynamics. The way this interaction works is as follows: the agent finds itself in a state and has a number of actions to choose from, it then choses an action, obtains a reward and transitions to a new state (see Figure 1.1). This interaction may go on forever or for a limited amount of time. The goal of the agent is to select the actions so as to maximize the long term cumulative rewards.

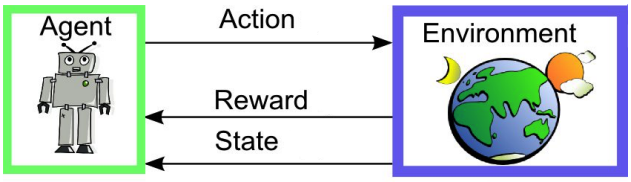


Figure 1.1: Reinforcement Learning loop.

Note that this is a very general model and therefore RL has a plethora of real world applications, considering that most daily activities we perform can be cast in this setting. Some examples of applications are: robots that learn to perform different types of physical tasks (arranging boxes, harvesting fruit, cooking, self-driving, etc.), performing financial

operations (buying/selling stock), playing games (chess, GO, StarCraft, football, etc.). It is this great diversity of high impact applications that motivates RL research. Recent years have seen an explosion in popularity in RL research. One of the main drivers of this interest has been high profile successes [5–11] in games and robotics, made possible by *deep RL* (which refers to the combination of RL algorithms with *deep learning* techniques).

Traditional RL considers a unique agent interacting with the environment. However, in many cases of interest, a multiplicity of agents will need to interact with the environment and with each other. This is the object of study of Multi-agent RL (MARL), which goes back to the work [12, 13]. Within MARL, there are three distinct categories: cooperative, competitive and mixed. Cooperative MARL considers the case where all agents form a team and collaborate to accomplish a unique shared goal (for example, a group of firemen robots trying to put out a fire). In competitive MARL, the agents have individual goals and have to compete to accomplish them (for example, animals competing over food sources). The mixed scenario is one in which some agents form teams and these teams compete against each other. In this dissertation we focus on the cooperative MARL scenario.

1.2 Background

We provide background material on RL and introduce the notation used throughout this manuscript. The background material presented is not intended to be exhaustive, but rather covers the concepts necessary to put into context the contributions introduced in this dissertation.

1.2.1 Notation

Matrices are denoted by upper case letters, while vectors are denoted with lower case. Random variables and sets are denoted with bold font and calligraphic font, respectively. $\rho(A)$ indicates the spectral radius of matrix A. I_M is the identity matrix of size M. $\mathbb{E}_g$ is the expected value with respect to distribution g. $\|\cdot\|_D$ refers to the weighted matrix norm, where D is a diagonal positive definite matrix. We use $\preceq$ to denote entry-wise inequality.

$\text{col}\{v(n)\}_{n=1}^{N}$ is a column vector with elements $v(1)$ through $v(N)$ (where $v(N)$ is at the bottom). The indicator function is denoted as $\mathbb{I}$. And finally $\mathbb{R}$ and $\mathbb{N}$ represent the sets of real and natural numbers, respectively.

1.2.2 The RL problem

As is customary, we model the agent-environment interaction as a Markov Decision Process (MDP). An MDP is defined by $(\mathcal{S},\mathcal{A},\mathcal{P},r)$, where $\mathcal{S}$ is a set of states, $\mathcal{A}$ is a set of actions of size $A = |\mathcal{A}|$, the transition kernel $\mathcal{P}(s'|s,a)$ specifies the probability of transitioning to state $s' \in \mathcal{S}$ from state $s \in \mathcal{S}$ having taken action $a \in \mathcal{A}$, and $r : \mathcal{S} \times \mathcal{A} \times \mathcal{S} \to \mathbb{R}$ is the reward function. Specifically, $r(s,a,s')$ is the reward when the agent transitions to state $s' \in \mathcal{S}$ from state $s \in \mathcal{S}$ having taken action $a \in \mathcal{A}$. The reward $r(s,a,s')$ can be a random variable with distribution $n_{s,a,s'}(r)$. The objective is to find a policy $\pi(a|s)$ that maximizes the discounted cumulative reward (we will refer to such policy as an optimal policy $\pi^\dagger(a|s)$):

$$\pi^\dagger = \arg\max_{\pi} J(\pi) \tag{1.1}$$

$$J(\pi) = \mathbb{E}_{\mathcal{P},\pi,n,d}\left(\sum_{t=0}^{\infty} \gamma^t \boldsymbol{r}(\boldsymbol{s}_t, \boldsymbol{a}_t, \boldsymbol{s}_{t+1}) \right) \tag{1.2}$$

where d is the distribution of initial states $\boldsymbol{s}_0$, $\gamma \in [0,1)$ is some discount factor, and $\boldsymbol{s}_t$ and $\boldsymbol{a}_t$ are the state and action at time t, respectively. We clarify that the notation $\pi(a|s)$ specifies the probability of selecting action a at state s. Most RL algorithms whose aim is to learn $\pi^\dagger$ follow one of three possible approaches: the policy gradient approach, the Bellman optimality equation approach, or the soft RL approach. In the following sections we provide a brief introduction to these approaches.

1.2.3 The Policy Gradient Approach

One strategy to obtain an optimal policy $\pi^\dagger$ is the policy gradient approach. Assume we parameterize the policy via some parameter vector θ (we denote the parameterized policy by $\pi(a|s;\theta)$). The idea then is to estimate the gradient of the objective with respect to θ and

then improve the policy by taking small steps in the ascent direction in parameter space. More specifically,

$$\theta_{i+1} = \theta_i + \mu \nabla_\theta J(\theta) \tag{1.3}$$

where we write $J(\theta)$ instead of $J(\pi(a|s;\theta))$. The gradient is given by the following equivalent expressions [14]:

$$\nabla_\theta J(\pi) = \mathbb{E}_{\pi,d^\pi}(\boldsymbol{r}(\boldsymbol{s},\boldsymbol{a},\boldsymbol{s'}) + \gamma v^\pi(\boldsymbol{s}) - v^\pi(\boldsymbol{s'}))\nabla_\theta \log \pi(\boldsymbol{a}|\boldsymbol{s};\theta) \tag{1.4}$$

$$\nabla_\theta J(\pi) = \mathbb{E}_{\pi,d^\pi}(q^\pi(\boldsymbol{s},\boldsymbol{a}) - v^\pi(\boldsymbol{s}))\nabla_\theta \log \pi(\boldsymbol{a}|\boldsymbol{s};\theta) \tag{1.5}$$

$$v^\pi(s) = \mathbb{E}_{\mathcal{P},\pi,n}\Big(\sum_{t=0}^{\infty} \gamma^t \boldsymbol{r}(\boldsymbol{s}_t,\boldsymbol{a}_t,\boldsymbol{s}_{t+1})\big|\boldsymbol{s}_0 = s\Big) \tag{1.6}$$

$$q^\pi(s,a) = \mathbb{E}_{\mathcal{P},n}(\boldsymbol{r}(s,a,\boldsymbol{s'}) + \gamma v^\pi(\boldsymbol{s'})) \tag{1.7}$$

$$d^\pi(s) = (1-\gamma)\mathbb{E}_{\pi,d}\sum_{t=0}^{\infty} \gamma^t \mathrm{Pr}(\boldsymbol{s}_t = s) \tag{1.8}$$

where $d^\pi(s)$ is the stationary state distribution induced on the MDP by π, and $v^\pi(s)$ and $q^\pi(s,a)$ are known as the state value function and the state-action value function, respectively.[1] In simple terms, $v^\pi(s)$ is the return that is expected if the agent starts at state s and chooses actions according to policy π, while $q^\pi(s,a)$ is the return that is expected if the agent starts at state s, chooses action a and then chooses future actions according to policy π. Notice that expressions (1.4) and (1.5) require knowledge of $v^\pi(s)$ (and possibly $q^\pi(s,a)$), which are unknown. Therefore, to implement a policy gradient loop it is necessary to estimate the state value function. The estimation of the value functions is known as *Policy Evaluation*. Algorithm 1.1 shows a typical policy gradient loop.

There are many algorithms that fall into the *policy gradient* category [15–21]. Generally speaking, the difference among these is how they estimate the gradient. Algorithms that maintain estimates of both the policy and the value function are often referred to as actor-

[1] From now on, when no confusion arises, we will use the term *value function* to refer to either $v^\pi(s)$ or $q^\pi(s,a)$.

Algorithm 1.1 Policy gradient loop

Initialize: the parameters θ.
for Epochs $e = 0, \dots, E$ **do**
  Collect transitions (s, a, r, s') by interacting with the environment following the current policy $\pi(a|s;\theta)$.
  Use the collected transitions to estimate the value function corresponding to the current policy (running a policy evaluation algorithm).
  Update the policy parameters θ using (1.3).
end for

critic algorithms, where *actor* refers to the policy and the *critic* is the value function. Policy gradient algorithms have the quality that they are very stable and have shown impressive performance in very challenging tasks [9–11], however, they have the inconvenience that they are very sample inefficient. Note in algorithm (1.1) that for every gradient step a lot of data has to be collected, which is then discarded in the following iteration.

1.2.4 The Bellman Optimality Equation and Q-learning

The algorithms we introduced in the previous section work by alternating between estimating how good the current policy is and then improving it. A different approach to learn an optimal policy could be to check if there exists some identity that is only satisfied by the optimal policy, and then leverage such property to learn the optimal policy directly. Indeed one such identity exists. The optimal state-action value function $q^\dagger(s, a)$ (which is the state-action value function that corresponds to an optimal policy $\pi^\dagger$) satisfies relation (1.9).

$$q^\dagger(s, a) = \mathbb{E}_{\mathcal{P},n}\left(\boldsymbol{r}(s, a, \boldsymbol{s'}) + \gamma \max_{a'} q^\dagger(\boldsymbol{s'}, a')\right) \tag{1.9}$$

Equation (1.9) is known as the Bellman optimality equation. One important feature of this relation is that, as opposed to (1.7), it does not have any expectation taken with respect to any policy π. It is this characteristic that will allow us to derive an algorithm that can use samples collected following *any* policy. It is easy to check that any policy that assigns positive probability only to actions that maximize $q^\dagger(s, a)$ (i.e. $q^\dagger(s, a) = \max_{a'} q^\dagger(s, a')$) is an optimal policy [22]. Since the optimal policy can be extracted from $q^\dagger$, the problem of

finding $\pi^\dagger$ can be replaced by the problem of finding $q^\dagger$.

From (1.9) we can define the Bellman optimality operator $\mathcal{B}$ as follows

$$\mathcal{B}q(s,a) = \mathbb{E}_{\mathcal{P},n}\left(\boldsymbol{r}(s,a,\boldsymbol{s'}) + \gamma \max_{a'} q(\boldsymbol{s'},a')\right) \tag{1.10}$$

Operator $\mathcal{B}$ is a γ-contraction whose fixed point is $q^\dagger$. Therefore, repeated application of this operator to any initial $q(s,a)$ function converges to $q^\dagger$ with γ-linear rate [22]. Note though that in RL, the dynamics of the environment are assumed to be unknown. Therefore, in an RL setting, this procedure to obtain $q^\dagger$ is not feasible. However, we can obtain an RL algorithm by performing a stochastic approximation to (1.10) that relies on samples. This algorithm is known as *Q-learning* and is due to [23] (see algorithm 1.2). Note that this algorithm utilizes a replay buffer, the purpose of this buffer is simply to be able to reuse samples (as opposed to just using them once to update $q(s,a)$ and then discarding them). Using a replay buffer greatly improves the sample efficiency of the algorithm.[2]

Algorithm 1.2 Q-learning with replay buffer

Initialize: q-function and an empty replay buffer $\mathcal{R}$.
for epochs $e = 0, \dots, E$ **do**
  Collect transitions (s,a,r,s') by interacting with the environment following any policy which assigns strictly positive probability to all actions and store them in the $\mathcal{R}$.
  for iterations $i = 0, \dots, I$ **do**
    Sample a transition (s,a,r,s') from the replay buffer.
    Update $q(s,a) = q(s,a) + \mu(r + \gamma \max_{a'} q(s',a') - q(s,a))$
  end for
end for

There are several variants of *Q-learning* [5, 24–26]. These algorithms have the advantage over policy gradient schemes that they are much more sample efficient. In other words, they require less interactions with the environment to achieve the same level of performance. The disadvantage of thise algorithms is that when combined with function approximators (like neural networks for example), convergence cannot be guaranteed and therefore they can diverge [27].

[2]The more sample efficient an algorithm is, the less interactions with the environment it requires.

1.2.5 Soft RL

Recently, a new approach termed soft RL has developed, which aims at deriving algorithms that combine the stability of policy gradient schemes with the sample efficiency of *Q-learning*. They do so by adding the policy's entropy as a regularizer to the original RL objective (1.2). More specifically,

$$J(\pi) = \mathbb{E}_{\mathcal{P},\pi,f,d}\left(\sum_{t=0}^{\infty}\gamma^t\big(\boldsymbol{r}(\boldsymbol{s}_t,\boldsymbol{a}_t,\boldsymbol{s}_{t+1}) - \lambda\log\pi(\boldsymbol{a}_t|\boldsymbol{s}_t)\big)\right) \tag{1.11}$$

where $\lambda > 0$ is a temperature parameter. The optimal value functions for this modified cost become:

$$v_\lambda^\star(s) = \lambda\log\left(\sum_a \exp\left(\lambda^{-1}\mathbb{E}_{\mathcal{P},n}\big(\boldsymbol{r}(s,a,\boldsymbol{s'}) + \gamma v_\lambda^\star(\boldsymbol{s'})\big)\right)\right) \tag{1.12}$$

$$q_\lambda^\star(s,a) = \mathbb{E}_{\mathcal{P},n}\big(\boldsymbol{r}(s,a,\boldsymbol{s'}) + \gamma v_\lambda^\star(\boldsymbol{s'})\big) \tag{1.13}$$

$$= \mathbb{E}_{\mathcal{P},n}\left(\boldsymbol{r}(s,a,\boldsymbol{s'}) + \gamma\lambda\log\sum_a \exp(\lambda^{-1}q_\lambda^\star(s,a))\right) \tag{1.14}$$

$$\pi_\lambda^\star(a|s) = \frac{\exp\left[\lambda^{-1}\mathbb{E}_{\mathcal{P},n}\left(\boldsymbol{r}(s,a,\boldsymbol{s'}) + \gamma v_\lambda^\star(\boldsymbol{s'})\right)\right]}{\sum_a \exp\left[\lambda^{-1}\mathbb{E}_{\mathcal{P},n}\left(\boldsymbol{r}(s,a,\boldsymbol{s'}) + \gamma v_\lambda^\star(\boldsymbol{s'})\right)\right]} \tag{1.15}$$

Notice that the effect of adding the entropy as a regularizer is that the *max* operator in (1.9) is substituted by a soft approximation (this gives rise to the term soft RL). The key is that expressions (1.12) and (1.15) are the only pair that satisfy the following relation for every (s,a) pair:

$$v_\lambda^\star(s) + \lambda\log\pi_\lambda^\star(a|s) = \mathbb{E}_{\mathcal{P},n}\big(\boldsymbol{r}(s,a,\boldsymbol{s'}) + \gamma v_\lambda^\star(\boldsymbol{s'})\big) \tag{1.16}$$

Note that relation (1.16) has the same feature as (1.9), it does not have an expectation with respect to any policy π. It is this feature that allows soft RL to derive algorithms that are as sample efficient as Q-learning. Several algorithms have been introduced based on this maximum-entropy formulation [28–34]. In algorithm 1.3 we show a simplified version of the *SBEED* algorithm [33].

Algorithm 1.3 simplified SBEED

Initialize: $v(s)$, $\pi(a|s)$ and $\rho(s,a)$, and an empty replay buffer $\mathcal{R}$.
for Epochs $e = 0, \ldots, E$ **do**
  Collect transitions (s,a,r,s') by interacting with the environment following any policy which assigns strictly positive probability to all actions and store them in the $\mathcal{R}$.
  for Iterations $i = 0, \ldots, I$ **do**
    Sample a transition (s,a,r,s') from the replay buffer.
    Update $\rho(s,a) = \rho(s,a) + \mu(r + \gamma v(s') - v(s) - \lambda \log \pi(a|s) - \rho(s,a))$
    Update $\log \pi(s,a) = \log \pi(s,a) + \mu\rho(s,a)$
    Update $v(s) = v(s) + \mu\rho(s,a)$
  end for
end for

Note that algorithm 1.3 is similar to *Q-learning* in that it is able to re-use data using a replay buffer and is similar to actor-critic schemes in that it learns simultaneously a policy (an actor) and a value function (the critic).

1.3 Multi-Agent RL (MARL)

In this section we introduce the fundamentals of MARL; for a recent survey see [35]. Within the context of MARL, three main categories can be enumerated: cooperative, competitive and mixed. Like we clarified before, in this dissertation we study the cooperative case.

1.3.1 Cooperative MARL

We consider a team of K agents that form a network. The network is represented by a graph in which the edges represent the communication links. Agent k communicates only with a subset of the agents in the network which we refer to as the neighborhood of k (denoted by $\mathcal{N}_k$). The topology of the network is determined by some combination matrix L whose kn-th entry (denoted by ℓ_{kn}) is a scalar with which agent n weights information incoming from agent k (note that $\ell_{kn} \neq 0 \iff k \in \mathcal{N}_n$). The agents interact with an environment and with each other. We model this interaction as a Multi-agent MDP (MA-MDP), which we define by the tuple $(K, \mathcal{S}, \mathcal{A}^k, \mathcal{P}, r^k)$. Here, K is the number of agents; $\mathcal{S}$ denotes a set of global states shared by all agents of size $S = |\mathcal{S}|$, and $\mathcal{A}^k$ is the set of actions available

to agent k. We refer to $\bar{\mathcal{A}} = \mathcal{A}^1 \times \cdots \times \mathcal{A}^K$ as the set of team actions. Furthermore, we clarify that we use the notation $\bar{a}$ to refer to the team's action (i.e., the collection of all individual actions), while a^k refers to the individual action of agent k and a^{-k} refers to the collection of all actions except for agent k's action. The transition kernel $\mathcal{P}(s'|s, \bar{a})$ specifies the probability of transitioning to global state $s' \in \mathcal{S}$ from global state $s \in \mathcal{S}$ having taken team action $\bar{a} \in \bar{\mathcal{A}}$, and $r^k : \mathcal{S} \times \bar{\mathcal{A}} \times \mathcal{S} \to \mathbb{R}$ is the reward function of agent k. We assume that reward r^k is only visible by its corresponding agent k. It is possible though, that each reward function r^k is composed of a local component and a global component shared by all agents (i.e., $r^k(s, \bar{a}, s') = r^k_{\text{local}}(s, \bar{a}, s') + r_{\text{global}}(s, \bar{a}, s')$). The global component of the reward can be associated with *team events* (for example, a football team scoring a goal) while the local component with *individual events* (for instance, spending too much energy due to excessive effort). It is important to highlight that the transition probabilities of the global state and the reward functions of the individual agents depend on the actions of all agents. The team's goal is to maximize the aggregated return defined as:

$$J(\bar{\pi}) = \sum_{t=0}^{\infty} \frac{\gamma^t}{K} \left(\sum_{k=1}^{K} \mathbb{E}_{\bar{\pi}, \mathcal{P}, n, d} \left[\boldsymbol{r}^k(\boldsymbol{s_t}, \bar{\boldsymbol{a}}_{\boldsymbol{t}}, \boldsymbol{s_{t+1}}) \right] \right) \tag{1.17}$$

where $\bar{\pi}(\bar{a}|s)$ denotes the team's policy. Note that (1.17) is a simple extension of (1.2) that aggregates the reward functions of all agents. It is well known that in this scenario the team can be regarded as one single agent where the team action consists of the joint actions by all agents [36]. Hence, one possible approach to address cooperative MARL is simply to centralize training and execution, and apply one of the single-agent methods reviewed in the previous section. However, this approach has two fundamental issues:

- Centralization might be undesirable or even unfeasible due to implementation constraints. Then the question arises, whether a fully decentralized approach can be derived that matches the performance of the centralized approach. We address this challenge in Chapters 2 and 3.

- Another issue is that of scalability. Assume that centralized training is a feasible option,

or that a decentralized approach that matches the centralized scheme is available. These approaches require estimation of variables ($q(s, a^1, \cdots, a^K)$ or $\pi(a^1, \cdots, a^K|s)$) that depend upon the join action space which is exponential with the number of agents. For instance, consider a small team with 4 agents each of which has 5 actions to choose from, then the team's action space is of size $5^4 = 625$. This quickly becomes computationally intractable. So the question that arises is whether it is possible to learn factored functions instead of joint functions (i.e., learn K $q^k(s, a^k)$ functions as opposed to learning one $q(s, a^1, \cdots, a^K)$ function). We address this challenge in Chapter 4.

1.4 Organization and Contributions

In this dissertation we make contributions to both the single-agent and multi-agent settings. The manuscript is organized as follows:

- **Chapter 2**: In this chapter we consider the policy evaluation problem when multiple agents are involved. The contribution of this chapter is the introduction of *Fast Diffusion for Policy Evaluation (FDPE)*, a fully decentralized policy evaluation algorithm for the finite data set case, under which all agents have a guaranteed linear convergence rate to the minimizer of the global cost function. The work in this chapter is based on material from references [37, 38].

- **Chapter 3**: We consider the problem of learning an optimal team policy in the cooperative MARL setting and introduce *Diffusion Team Policy Optimization (DTPO)*. This is a fully distributed algorithm that allows agents to converge to optimal team policies. The algorithm achieves this by relying on local communication among the agents. This chapter is based on [39]. Form the work we present in this chapter two new problems are naturally raised, which we consider in the following two chapters.

- **Chapter 4**: In this chapter we address the challenge of learning factored policies in cooperative MARL scenarios. The goal is to obtain factored policies that determine the

individual behavior of each agent so that the resulting joint policy is optimal. In this chapter we make contributions to both the dynamic programming and RL settings. In the dynamic programming case we provide a number of lemmas that prove the existence of such factored policies and we introduce an algorithm (along with proof of convergence) that provably leads to them. Then we introduce *Logical Team Q-learning (LTQL)*, which is a stochastic version of the algorithm for the RL case. This chapter is based on [40]. Our method exhibits state of the art performance on a challenging MARL game.

- **Chapter 5**: From our work in Chapter 3, another challenge becomes apparent, namely, that of exploration. That is, when the team interacts with the environment, how should the agents choose their actions so as to maximize the information learned about the environment? In other words, how should actions be chosen in order to learn an optimal policy with the least amount of interactions with the environment as possible. In this chapter we address this exploration problem. We introduce the *Information Seeking Learner (ISL)* algorithm, which is efficient at exploring the state space of the MDP. Similarly to soft RL, we derive the algorithm by augmenting the traditional RL objective with a novel regularization term. We provide convergence results for the dynamic programming case and empirically show that our method exhibits state of the art performance on a range of challenging deep-exploration benchmarks. This work in based on [41].

- **Chapter 6**: In this final chapter we provide a brief summary of the contributions introduced in this dissertation. We then conclude by discussing future research avenues that naturally follow from the work we introduce.

CHAPTER 2

Fully Decentralized Policy Evaluation

The goal of a policy evaluation algorithm is to estimate the performance that an agent will achieve when it follows a particular policy to interact with an environment. Policy evaluation algorithms are important because, as we briefly mentioned in the previous chapter, they are often key parts of more elaborate solution methods where the ultimate goal is to find an optimal policy for a particular task (one such example is the class of actor-critic algorithms – see [42] for a survey). This chapter studies the problem of policy evaluation in a fully decentralized setting. We consider two distinct scenarios.

In the first case, K independent agents interact with independent instances of the same environment following potentially different behavior policies[1] to collect data and the objective is for the agents to cooperate. In this scenario, each agent only has knowledge of its own states and rewards, which are independent of the states and rewards of the other agents. Various practical situations give rise to this scenario. For example, consider a task that takes place in a large geographic area. The area can be divided into smaller sections, each of which can be explored by a separate agent. This framework is also useful for collective robot learning (see, [43–45]). We clarify that this first scenario is not classified as MARL because the agents do not interact with each other.

The second scenario we consider is that of MARL. In this case a group of agents interact simultaneously with a unique MDP and with each other to attain a common goal. In this setting, there is a unique global state known to all agents and each agent receives distinct local rewards, which are unknown to the other agents.

[1]The *behavior policy* is the policy that is followed while interacting with the environment with the purpose of collecting data. This stands in contrast with the *target policy*, which is the policy whose corresponding value function we wish to estimate.

The contribution of this chapter is twofold. In the first place, we introduce *Fast Diffusion for Policy Evaluation*, a fully decentralized policy evaluation algorithm under which all agents have a guaranteed linear convergence rate to the minimizer of the global cost function. The algorithm is designed for the finite data set case and combines off-policy learning, eligibility traces, and linear function approximation. The eligibility traces are derived from the use of a more general cost function and they allow the control of the bias-variance trade-off we mentioned previously. In our distributed model, a fusion center is not required and communication is only allowed between immediate neighbors. The algorithm is applicable both to distributed situations with independent MDPs (i.e., independent states and rewards) and to MARL scenarios (i.e., global state and independent rewards). To the best of our knowledge, this is the first algorithm that combines all these characteristics. Our second contribution is a novel proof of convergence for the algorithm. This proof is challenging due to the combination of three factors: the distributed nature of the algorithm, the primal-dual structure of the cost function we optimize, and the use of stochastic biased gradients as opposed to exact gradients.

2.1 Related Works

The material we present in this chapter is related to the class of works that study policy evaluation, distributed reinforcement learning, and multi-agent reinforcement learning.

There exist a plethora of algorithms for single-agent policy evaluation such as GTD [46], TDC [47], GTD2 [47], GTD-MP/GTD2-MP [48], GTD(λ) [49], and True Online GTD(λ) [50]. The main feature of these algorithms is that they have guaranteed convergence (for small enough step-sizes) while combining off-policy learning and linear function approximation; and are applicable to scenarios with streaming data. They are also applicable to cases with a finite amount of data. However, in this latter situation, they have the drawback that they converge at a sub-linear rate because a decaying step-size is necessary to guarantee convergence to the minimizer. In most current applications, policy evaluation is actually carried out after collecting a finite amount of data (one example is the recent success in

the game of GO [51]). Therefore, deriving algorithms with better convergence properties for the finite sample case becomes necessary. By leveraging recent developments in variance-reduced algorithms, such as SVRG [52] and SAGA [53], the work [54] presented SVRG and SAGA-type algorithms for policy evaluation. These algorithms combine GTD2 with SVRG and SAGA and they have the advantage over GTD2 in that linear convergence is guaranteed for fixed data sets. The algorithm we present in this chapter is related to [54] in that we too use a variance-reduced strategy, however we base our algorithm in the AVRG strategy [55] which is more convenient for distributed implementations because of an important balanced gradient calculation feature.

Another interesting line of work in the context of distributed policy evaluation is [56], [57]. In [56] and [57] the authors introduce *Diffusion GTD2* and *ALG2*; which are extensions of *GTD2* and *TDC* to the fully decentralized case, respectively. These algorithms consider the situation where independent agents interact with independent instances of the same MDP. These strategies allow individual agents to converge through collaboration even in situations where convergence is infeasible without such collaboration. The *FDPE* algorithm we introduce in this chapter can be applied to this setting as well and has two main advantages over [56] and [57]. First, the proposed algorithm has guaranteed linear convergence, while the previous algorithms converge at a sub-linear rate. Second, while in some instances, the solutions in [56] and [57] may be biased due to the use of the Mean Square Projected Bellman Error (MSPBE) as a surrogate cost (this point is further clarified in Section 2.2), the proposed method allows better control of the bias term due to a modification in the cost function.

There is also a good body of work on MARL. However, most works in this area focus on the policy optimization problem instead of the policy evaluation problem. The work that is closer to the current contribution is [58], which was pursued simultaneously and independently. The goal of the formulation in [58] is to derive a linearly-convergent distributed policy evaluation procedure for MARL. The work [58] does not consider the case where independent agents interact with independent MDPs. In the context of MARL, our proposed *FDPE* technique has three advantages in comparison to the approach from [58].

First, the memory requirement of the algorithm in [58] scales linearly with the amount of data (i.e., $O(N)$), while the memory requirement for the proposed method in this manuscript is $O(1)$), i.e., it is independent of the amount of data. Second, the algorithm of [58] does not include the use of eligibility traces; a feature that is often necessary to reach state of the art performance (see, for example, [59, 60]). Finally, the algorithm from [58] requires all agents in the network to sample their data points in a synchronized manner, while the algorithm we propose in this work does not require this type of synchronization. Another paper that is related to the current work is [61], which considers the same distributed MARL as we do; although their contribution is different from ours. The main contribution in [61] is to extend the policy gradient theorem to the MARL case and derive two fully distributed actor-critic algorithms with linear function approximation for policy optimization. The connection between [61] and our work is that their actor-critic algorithms require a distributed policy evaluation algorithm. The algorithm they use is similar to [56] and [57] (they combine diffusion learning [62] with standard TD instead of GTD2 and TDC as was the case in [56] and [57]). The algorithm we present in this chapter is compatible with their actor-critic schemes (i.e., it could be used as the critic), and hence could potentially be used to augment their performance and convergence rate.

Our work is also related to the literature on distributed optimization. Some notable works in this area include [62–72]. Consensus [63] and Diffusion [62] constitute some of the earliest work in this area. These methods can converge to a neighborhood around, but not exactly to, the global minimizer when constant step-sizes are employed [64, 72]. Another family of methods is based on distributed alternating direction method of multipliers (ADMM) [65]. While these methods can converge linearly fast to the exact global minimizer, they are computationally more expensive than previous methods since they need to optimize a sub-problem at each iteration. An exact first-order algorithm (EXTRA) was proposed in [66] for undirected networks to correct the bias suffered by consensus, (this work was later extended for the case of directed networks [69]). EXTRA and DEXTRA [69] can also converge linearly to the global minimizer while maintaining the same computational efficiency as consensus and diffusion. Several other works employ instead a gradient tracking strategy [67, 68, 73]. These

works guarantee linear convergence to the global minimizer even when they operate over time-varying networks. Recently, the Exact Diffusion algorithm [70, 71] has been introduced for static undirected graphs. This algorithm has a wider stability range than EXTRA (and hence exhibits faster convergence [71]), and for the case of static graphs is more communication efficient than gradient tracking methods since the gradient vectors are not shared among agents. *FDPE* is related to Exact Diffusion since our MARL model is based on static undirected graphs and our distributed strategy is derived in a similar manner to *Exact Diffusion*. We remark that there is a fundamental difference between the *FDPE* algorithm we present and the works in [62–72], namely, our algorithm finds the global *saddle-point* in a primal dual formulation while the cited works solve convex minimization problems.

2.2 Problem Setting

We consider the problem of policy evaluation within the traditional reinforcement learning framework we introduced in the previous chapter. We recall that the objective of a policy evaluation algorithm is to estimate the performance of a known target policy using data generated by either the same policy (this case is referred as *on-policy*), or a different policy that is also known (this case is referred as *off-policy*). As we mentioned in the introduction, we model our setting as an MDP defined by the tuple $(\mathcal{S},\mathcal{A},\mathcal{P},r)$. Even though in this chapter we analyze the distributed scenario, in this section we motivate the cost function for the single agent case for clarity of exposition and in the next section we generalize it to the distributed setting. We thus consider an agent that wishes to learn the value function, $v^{\pi}(s)$, for a target policy of interest $\pi(a|s)$, while following a potentially different behavior policy $\phi(a|s)$. We recall that the value function for a target policy π, starting from some initial state $s \in \mathcal{S}$ at time i, is defined as follows:

$$v^{\pi}(s) = \mathbb{E}_{\mathcal{P},\pi,n}\left(\sum_{t=i}^{\infty} \gamma^{t-i} \boldsymbol{r}(\boldsymbol{s}_t, \boldsymbol{a}_t, \boldsymbol{s}_{t+1}) \Big| \boldsymbol{s}_i = s \right) \tag{2.1}$$

we recall that $\mathbb{E}_n$ is the expectation with respect to the reward distributions $n_{s,a,s'}(r)$ and $0 \leq \gamma < 1$ is the discount factor. Note that since we are dealing with a constant target policy π, the transition probabilities between states, which are given by $p^{\pi}_{s,s'} = \mathbb{E}_{\pi}\mathcal{P}(s'|s, \boldsymbol{a})$, are fixed and hence the MDP reduces to a Markov Rewards Process. In this case, the state evolution of the agent can be modeled with a Markov Chain with transition matrix P^{π} whose entries are given by $(P^{\pi})_{ij} = p^{\pi}_{i,j}$.

Assumption 2.1. *We assume that the Markov Chain induced by the behavior policy $\phi(a|s)$ is aperiodic and irreducible. In view of the Perron-Frobenius Theorem [72], this condition guarantees that the Markov Chain under $\phi(a|s)$ will have a steady-state distribution in which every state has a strictly positive probability of visitation [72].* □

Using the matrix P^{π} and defining:

$$r^{\pi}(s) = \mathbb{E}_{\pi,\mathcal{P},n}\big(\boldsymbol{r}(s, \boldsymbol{a}, \boldsymbol{s'})\big) \tag{2.2}$$

$$r^{\pi} = [r^{\pi}(1), \cdots, r^{\pi}(S)]^T \tag{2.3}$$

$$v^{\pi} = [v^{\pi}(1), \cdots, v^{\pi}(S)]^T \tag{2.4}$$

we can rewrite (2.1) in matrix form as:

$$v^{\pi} = \sum_{n=0}^{\infty} (\gamma P^{\pi})^n r^{\pi} = (I - \gamma P^{\pi})^{-1} r^{\pi} \tag{2.5}$$

Note that the inverse $(I - \gamma P^{\pi})^{-1}$ always exists; this is because $\gamma < 1$ and the matrix P^{π} is right stochastic with spectral radius equal to one. We further note that v^{π} also satisfies the following $h-$stage Bellman equation for any $h \in \mathbb{N}$:

$$v^{\pi} = (\gamma P^{\pi})^h v^{\pi} + \sum_{n=0}^{h-1} (\gamma P^{\pi})^n r^{\pi} \tag{2.6}$$

2.2.1 Definition of cost function

We are interested in applications where the state space is too large (or even infinite) and hence some form of function approximation is necessary to reduce the dimensionality of the parameters to be learned. As we anticipated in the introduction, in this work we use linear approximations.[2] More formally, for every state $s \in \mathcal{S}$, we approximate $v^\pi(s) \approx x_s^T \theta^\star$ where $x_s \in \mathbb{R}^M$ is a feature vector corresponding to state s and $\theta^\star \in \mathbb{R}^M$ is a parameter vector such that $M \ll S$. Defining $X = [x_1, x_2, \cdots, x_S]^T \in \mathbb{R}^{S \times M}$, we can write a vector approximation for v^π as $v^\pi \approx X\theta^\star$. We assume that X is a full rank matrix; this is not a restrictive assumption since the feature matrix is a design choice. It is important to note though that the true v^π need not be in the range space of X. If v^π is in the range space of X, an equality of the form $v^\pi = X\theta^\star$ holds exactly and the value of $\theta^\star$ is unique (because X is full rank) and given by $\theta^\star = (X^T X)^{-1} X^T v^\pi$. For the more general case where v^π is not in the range space of X, then one sensible choice for $\theta^\star$ is:

$$\theta^\star = \arg\min_\theta \|X\theta - v^\pi\|_D^2 = (X^T D X)^{-1} X^T D v^\pi \tag{2.7}$$

where D is some positive definite weighting matrix to be defined later. Although (2.7) is a reasonable cost to define $\theta^\star$, it is not useful to derive a learning algorithm since v^π is not known beforehand. As a result, for the purposes of deriving a learning algorithm, another cost (one whose gradients can be sampled) needs to be used as a surrogate for (2.7). One popular choice for the surrogate cost is the MSPBE (see, e.g., [47, 48, 56, 57]); this cost has the inconvenience that its minimizer θ^o is different from (2.7) and some bias is incurred [47]. In order to control the magnitude of the bias, we shall derive a generalization of the MSPBE which we refer to as $H-$truncated λ-weighted Mean Square Projected Bellman Error (Hλ-MSPBE). To introduce this cost, we start by writing a convex combination of equation (2.6)

[2]We choose linear function approximation, not just because it is mathematically convenient (since with this approximation our cost function is strongly convex) but because there are theoretical justifications for this choice. In the first place, in some domains (for example Linear Quadratic Regulator problems) the value function is a linear function of known features. Secondly, when policy evaluation is used to estimate the gradient of a policy in a policy gradient algorithm, the policy gradient theorem [14] assures that the exact gradient can be obtained even when a linear function is used to estimate v^π.

with different h's ranging from 1 to H (we choose H to be a finite amount instead of $H \to \infty$ because in this chapter we deal with finite data instead of streaming data) as follows:

$$\begin{aligned} v^\pi &= (1-\lambda)\sum_{h=1}^{H-1}\lambda^{h-1}\left((\gamma P^\pi)^h v^\pi + \sum_{n=0}^{h-1}(\gamma P^\pi)^n r^\pi\right) + \lambda^{H-1}\left((\gamma P^\pi)^H v^\pi + \sum_{n=0}^{H-1}(\gamma P^\pi)^n r^\pi\right) \\ &= \Gamma_2(\lambda, H) r^\pi + \rho_1(\lambda, H)\Gamma_1(\lambda, H) v^\pi \end{aligned} \tag{2.8}$$

where we introduced:

$$\rho_1(\lambda, H) = \frac{(1-\lambda)\gamma + (1-\gamma)(\gamma\lambda)^H}{1-\gamma\lambda} \tag{2.9}$$

$$\Gamma_2(\lambda, H) = \sum_{n=0}^{H-1}(\gamma\lambda P^\pi)^n = \left(I - (\gamma\lambda P^\pi)^H\right)(I - \gamma\lambda P^\pi)^{-1} \tag{2.10}$$

$$\Gamma_1(\lambda, H) = \frac{1}{\rho_1(\lambda, H)}\left((1-\lambda)\gamma P^\pi \sum_{n=0}^{H-1}(\gamma\lambda P^\pi)^n + (\gamma\lambda P^\pi)^H\right) \tag{2.11}$$

and $0 \leq \lambda \leq 1$ is a parameter that controls the bias.

Remark 2.1. *Note that* $0 < \rho_1(\lambda, H) \leq \gamma < 1$. □

Remark 2.2. $\Gamma_1(\lambda, H)$ *is a right stochastic matrix because it is defined as a convex combination of powers of* P^π *(which are right stochastic matrices).* □

Note that from now on for the purpose of simplifying the notation, we refer to $\rho_1(\lambda, H)$, $\Gamma_1(\lambda, H)$ and $\Gamma_2(\lambda, H)$ as ρ_1, Γ_1 and Γ_2, respectively. Replacing v^π in (2.8) by its linear approximation we get:

$$X\theta \approx \Gamma_2 r^\pi + \rho_1\Gamma_1 X\theta \tag{2.12}$$

Projecting the right hand side onto the range space of X so that an equality holds, we arrive at:

$$X\theta = \Pi\left[\Gamma_2 r^\pi + \rho_1\Gamma_1 X\theta\right] \tag{2.13}$$

where $\Pi \in \mathbb{R}^{S\times S}$ is the weighted projection matrix onto the space spanned by X, (i.e., $\Pi = X(X^TDX)^{-1}X^TD$). We can now use (2.13) to define our surrogate cost function:

$$S(\theta) = \frac{1}{2}\Big\|\Pi\big(\Gamma_2 r^\pi + \rho_1\Gamma_1 X\theta\big) - X\theta\Big\|_D^2 + \frac{\eta}{2}\big\|\theta - \theta_{\rm p}\big\|_U^2 \tag{2.14}$$

where the first term on the right hand side is the Hλ-MSPBE, $\eta \geq 0$ is a regularization parameter, $U > 0$ is a symmetric positive-definite weighting matrix, and $\theta_{\rm p}$ reflects prior knowledge about θ. Two sensible choices for U are $U = I$ and $U = X^TDX = C$, which reflect previous knowledge about θ or the value function $X\theta$, respectively. The regularization term can be particularly useful when the policy evaluation algorithm is used as part of a policy gradient loop (since subsequent policies are expected to have similar value functions and the value of θ learned in one iteration can be used as $\theta_{\rm p}$ in the next iteration) like, for example, in [74]. One main advantage of using the proposed cost (2.14) instead of the more traditional MSPBE cost is that the magnitude of the bias between its minimizer (denoted as $\theta^o(H,\lambda)$) and the desired solution $\theta^\star$ can be controlled through λ and H. To see this, we first rewrite $S(\theta)$ in the following equivalent form:

$$S(\theta) = \frac{1}{2}\big\|X^TD(I-\rho_1\Gamma_1)X\theta - X^TD\Gamma_2 r^\pi\big\|_{(X^TDX)^{-1}}^2 + \frac{\eta}{2}\big\|\theta-\theta_{\rm p}\big\|_U^2 \tag{2.15}$$

Next, we introduce the quantities:

$$A = X^TD(I-\rho_1\Gamma_1)X \tag{2.16}$$

$$b = X^TD\Gamma_2 r^\pi \tag{2.17}$$

$$C = X^TDX \tag{2.18}$$

Remark 2.3. *A is an invertible matrix.*

Proof. Due to remarks 1 and 2 we have that the spectral radius of $\rho_1(\lambda,H)\Gamma_1(\lambda,H)$ is strictly smaller than one, and hence $I-\rho_1(\lambda,H)\Gamma_1(\lambda,H)$ is invertible. The result follows by recalling that X and D are full rank matrices. □

The minimizer of (2.15) is given by:

$$\theta^o(H,\lambda) = (A^T C^{-1} A + \eta U)^{-1}(\eta U \theta_{\mathrm{p}} + A^T C^{-1} b) \tag{2.19}$$

where $(A^T C^{-1} A + \eta U)^{-1}$ exists and hence $\theta^o(H,\lambda)$ is well defined. This is because ηU is positive-definite and A is invertible. Also note that when $\lambda = 1$, $H \to \infty$ and $\eta = 0$, $\theta^o(H,\lambda)$ reduces to (2.7) and hence the bias is removed. We do not fix $\lambda = 1$ because while the bias diminishes as $\lambda \to 1$, the estimate of the value function approaches a Monte Carlo estimate and hence the variance of the estimate increases. Note from (2.8) and (2.15) that in the particular case where the value function v^π lies in the range space of X (and there is no regularization, i.e., $\eta = 0$) there is no bias $\big($i.e., $\theta^\star = \theta^o(H,\lambda)\big)$ independently of the values of λ and H. This observation shows that when there is bias between $\theta^\star$ and $\theta^o(H,\lambda)$, the bias arises from the fact that the value function being estimated does not lie in the range space of X. In practice, λ offers a valuable bias-variance trade-off, and its optimal value depends on each particular problem. Note that since we are dealing with finite data samples, in practice, H will always be finite. Therefore, eliminating the bias completely is not possible (even when $\lambda = 1$). The exact expression for the bias is obtained by subtracting (2.19) from (2.7). However, this expression does not easily indicate how the bias behaves as a function of γ, λ and H. Lemma 2.1 provides a simplified expression.

Lemma 2.1. *The bias $\|\theta^o(H,\lambda) - \theta^\star\|^2$ is approximated by:*

$$\|\theta^o(H,\lambda) - \theta^\star\|^2 \approx \left(\mathbb{I}\big(v^\pi \neq \Pi v^\pi\big)\frac{\kappa_2 \rho_1}{(1+\kappa_1\eta)(\kappa_3 - \rho_1)} + \frac{\kappa_1 \eta \|\theta_p - \theta^\star\|}{1+\kappa_1\eta}\right)^2 \tag{2.20}$$

where $\mathbb{I}$ is the indicator function and

$$\frac{\rho_1}{\kappa_3 - \rho_1} = \frac{(1-\lambda)\gamma + (1-\gamma)(\gamma\lambda)^H}{\kappa_3(1-\gamma\lambda) - (1-\lambda)\gamma - (1-\gamma)(\gamma\lambda)^H} \tag{2.21}$$

for some constants κ_1, κ_2 and κ_3. See Appendix A for conditions under which the approximation becomes tighter.

Proof. See Appendix 2.C. □

Note that expression (2.20) agrees with our previous discussion and with several intuitive facts. First, due to the indicator function, if v^π lies in the range space of X there is no bias independently of the values of γ, λ and H (as long as $\eta = 0$). Second, if $\lambda = 0$, the bias is independent of H (because when $\lambda = 0$ all terms that depend on H are zeroed). Third, if $H = 1$ then the bias is independent of the value of λ (because when $H = 1$ all terms that depend on λ are zeroed). Furthermore, the expression is monotone decreasing in λ (for the case where $H > 1$) which agrees with the intuition that the bias diminishes as λ increases. Finally, we note that the bias is minimized for $\lambda = 1$ and in this case there is still a bias, which if $\eta = 0$, is on the order of $\mathcal{O}\big(\gamma^H/(\kappa_3 - \gamma^H)\big)$. This explicitly shows the effect on the bias of having a finite H. The following lemma describes the behavior of the variance.

Lemma 2.2. *The variance of the estimate $\widehat{\theta}^o(H, \lambda)$ is approximated by (2.22), for some constants κ_1 and κ_4.*

$$\mathbb{E}\left\|\widehat{\theta}^o(H,\lambda) - \theta^o(H,\lambda)\right\|^2 \approx \frac{\kappa_4}{(1+\kappa_1\eta)^2(N-H)}\left(\frac{1-(\gamma\lambda)^{2H}}{1-(\gamma\lambda)^2}\right) \tag{2.22}$$

Proof. See Appendix 2.D. □

Note that (2.22) is monotone increasing as a function of λ (for $H > 1$) and as a function of H (for $\lambda > 0$). Adding expressions (2.20) and (2.22) shows explicitly the bias-variance trade-off handled by the parameter λ and the finite horizon H. We remark that the idea of an eligibility trace parameter λ as a bias-variance trade-off is not novel to this work and has been previously used in algorithms such as $TD(\lambda)$ [4], $TD(\lambda)$ with *replacing traces* [75], GTD(λ) [49] and True Online GTD(λ) [50]. Note however, that these works derive algorithms for the on-line case (as opposed to the batch setting) using different cost functions. Therefore, the expressions we present in this chapter are different from previous works, which is why we derive them in detail. Moreover, the expressions corresponding to Lemmas 2.1 and 2.2 that quantify such bias-variance trade-off are new and specific for our batch model.

At this point, all that is left to fully define the surrogate cost function $S(\theta)$ is to choose the

positive definite matrix D. The algorithm that we derive in this chapter is of the stochastic gradient type. With this in mind, we shall choose D such that the quantities A, b and C turn out to be expectations that can be sampled from data realizations. Thus, we start by setting D to be a diagonal matrix with positive entries; we collect these entries into a vector d^ϕ and write D^ϕ instead of D, i.e., $D = D^\phi = \text{diag}(d^\phi)$. We shall select d^ϕ to correspond to the steady-state distribution of the Markov chain induced by the behavior policy, $\phi(a|s)$. This choice for D not only is convenient in terms of algorithm derivation, it is also physically meaningful; since with this choice for D, states that are visited more often are weighted more heavily while states which are rarely visited receive lower weights. As a consequence of Assumption 1 and the Perron-Frobenius Theorem [72], the vector d^ϕ is guaranteed to exist and all its entries will be strictly positive and add up to one. Moreover, this vector satisfies ${d^\phi}^T P^\phi = {d^\phi}^T$ where P^ϕ is the transition probability matrix defined in a manner similar to P^π.

Lemma 2.3. *Setting $D = diag(d^\phi)$, the matrices A, b and C can be written as expectations as follows:*

$$A = \mathbb{E}_{d^\phi,\mathcal{P},\pi}\left[\boldsymbol{x}_t\left(\boldsymbol{x}_t - \gamma(1-\lambda)\sum_{n=0}^{H-1}(\gamma\lambda)^n \boldsymbol{x}_{t+n+1} - (\gamma\lambda)^H \boldsymbol{x}_{t+H}\right)^T\right] \tag{2.23a}$$

$$b = \mathbb{E}_{d^\phi,\mathcal{P},\pi}\left[\boldsymbol{x}_t \sum_{n=0}^{H-1}(\gamma\lambda)^n \boldsymbol{r}_{t+n}\right] \tag{2.23b}$$

$$C = \mathbb{E}_{d^\phi}\left[\boldsymbol{x}_t \boldsymbol{x}_t^T\right] \tag{2.23c}$$

where, with a little abuse of notation, we defined $\boldsymbol{x}_t = \boldsymbol{x}_{\boldsymbol{s}_t}$ and $\boldsymbol{r}_t = \boldsymbol{r}^\pi(\boldsymbol{s}_t)$, where $\boldsymbol{s}_t$ is the state visited at time t.

Proof. See Appendix 2.E. □

2.2.2 Optimization problem

Since the signal distributions are not known beforehand and we are working with a finite amount of data, say, of size N, we need to rely on empirical approximations to estimate the

expectations in $\{A, b, C\}$. We thus let $\widehat{A}$, $\widehat{b}$, $\widehat{C}$ and $\widehat{U}$ denote estimates for A, b, C and U from data and replace them in (2.15) to define the following empirical optimization problem:

$$\min_{\theta} J_{\text{emp}}(\theta) = \frac{1}{2}\big\|\widehat{A}\theta - \widehat{b}\big\|^2_{\widehat{C}^{-1}} + \frac{\eta}{2}\big\|\theta - \theta_{\text{p}}\big\|^2_{\widehat{U}} \tag{2.24}$$

Note that whether an empirical estimate for U is required depends on the choice for U. For instance, if $U = I$ then obviously no estimate is needed. However, if $U = C$ then an empirical estimate is needed, (i.e., $\widehat{U} = \widehat{C}$).

To fully characterize the empirical optimization problem, expressions for the empirical estimates still need to be provided. The following lemma provides the necessary estimates.

Lemma 2.4. *For the general off-policy case, the following expressions provide unbiased estimates for A, b and C:*

$$\widehat{A}_n = x_n\left(\rho^H_{n,0}x_n - \gamma(1-\lambda)\sum_{h=0}^{H-1}(\gamma\lambda)^h\xi_{n,n+h+1}x_{n+h+1} - (\gamma\lambda)^H\xi_{n,n+H}x_{n+H}\right)^T \tag{2.25a}$$

$$\widehat{A} = \frac{1}{N-H}\sum_{n=1}^{N-H}\widehat{A}_n \tag{2.25b}$$

$$\widehat{b}_n = x_n\sum_{h=0}^{H-1}(\gamma\lambda)^h\rho^H_{n,h}r_{n+h}, \quad \widehat{b} = \frac{1}{N-H}\sum_{n=1}^{N-H}\widehat{b}_n \tag{2.25c}$$

$$\widehat{C}_n = x_n x_n^T \tag{2.25d}$$

$$\widehat{C} = \frac{1}{N-H}\sum_{n=1}^{N-H}\widehat{C}_n \tag{2.25e}$$

where

$$\rho^H_{t,n} = (1-\lambda)\sum_{h=n}^{H-1}\lambda^{h-n}\xi_{t,t+h+1} + \lambda^{H-n}\xi_{t,t+H} \tag{2.26}$$

$$\xi_{t,t+h} = \prod_{j=t}^{t+h-1}\frac{\pi(a_j|s_j)}{\phi(a_j|s_j)} \tag{2.27}$$

Proof. See Appendix 2.F. □

Note that $\xi_{t,t+h}$ is the importance sample weight corresponding to the trajectory that started at some state s_t and took h steps before arriving at some other state s_{t+h}. Note that even if we have N transitions, we can only use $N - H$ training samples because every estimate of $\widehat{x}_n$ and $\widehat{b}_n$ looks H steps into the future.

2.3 Distributed Policy Evaluation

In this section we present the distributed framework and use (2.24) to derive *Fast Diffusion for Policy Evaluation*. The purpose of this algorithm is to deal with situations where data is dispersed among a number of nodes and the goal is to solve the policy evaluation problem in a fully decentralized manner.

2.3.1 Distributed Setting

We consider a situation in which there are K agents that wish to evaluate a target policy $\pi(a|s)$ for a common MDP. Each agent has N samples, which are collected following its own behavior policy ϕ_k (with steady state distribution matrix D^{ϕ_k}). Note that the behavior policies can be potentially different from each other. The goal for all agents is to estimate the value function of the target policy $\pi(a|s)$ leveraging all the data from all other agents in a fully decentralized manner.

To do this, they form a network in which each agent can only communicate with other agents in its immediate neighborhood. The network is represented by a graph in which the nodes and edges represent the agents and communication links, respectively. The topology of the graph is defined by a combination matrix L whose kn-th entry (i.e., ℓ_{kn}) is a scalar with which agent n scales information arriving from agent k. If agent k is not in the neighborhood of agent n, then $\ell_{kn} = 0$.

Assumption 2.2. *We assume that the network is strongly connected. This implies that there is at least one path from any node to any other node and that at least one node has a self-loop (i.e. that at least one agent uses its own information). We further assume that the*

combination matrix L is symmetric and doubly-stochastic.

Remark 2.4. *In view of the Perron-Frobenius Theorem, assumption 2.2 implies that the matrix L can be diagonalized as $L = H\Lambda H^T$, where one element of Λ is equal to 1 and its corresponding eigenvector is given by $\mathbb{1}/\sqrt{K}$ (where $\mathbb{1}$ is the all ones vector). The remaining eigenvalues of L lie strictly inside the unit circle.*

A combination matrix satisfying assumption 2 can be generated using the Laplacian rule, the maximum-degree rule, or the Metropolis rule (see Table 14.1 in [72]). A sample network is shown in Figure 2.1.

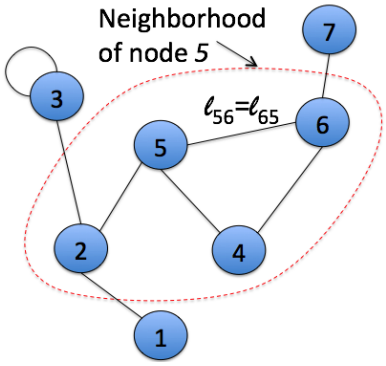


Figure 2.1: Sample network.

2.3.2 Algorithm Derivation

Mathematically, the goal for all agents is to minimize the following aggregate cost:

$$S_{\mathrm{M}}(\theta) = \sum_{k=1}^{K} \tau_k \left(\frac{1}{2} \left\| \Pi\big(\Gamma_2 r^{\pi} + \rho_1 \Gamma_1 X\theta\big) - X\theta \right\|_{D^{\phi_k}}^2 + \frac{\eta}{2} \left\| \theta - \theta_{\mathrm{p}} \right\|_{U_k}^2 \right) \tag{2.28}$$

where the purpose of the nonnegative coefficients τ_k is to scale the costs of the different agents; this is useful since the costs of agents whose behavior policy is closer to the target

policy might be assigned higher weights. For (2.28), we define the matrices D and U to be:

$$D = \sum_{k=1}^{K} \tau_k D^{\phi_k} \tag{2.29}$$

$$U = \sum_{k=1}^{K} \tau_k U_k \tag{2.30}$$

so that equation (2.28) becomes:

$$S_{\mathrm{M}}(\theta) = \frac{1}{2}\Big\|\Pi\big(\Gamma_2 r^{\pi} + \rho_1 \Gamma_1 X\theta\big) - X\theta\Big\|_D^2 + \frac{\eta}{2}\big\|\theta - \theta_{\mathrm{p}}\big\|_U^2 \tag{2.31}$$

Note that (2.31) has the same form as (2.15); the only difference is that in (2.31) the matrices D and U are defined by linear combinations of the individual matrices D^{ϕ_k} and U_k, respectively. Matrices D^{ϕ_k} are therefore not required to be positive definite, only D is required to be a positive definite diagonal matrix. Since the matrices D^{ϕ_k} are given by the steady-state probabilities of the behavior policies, this implies that each agent does not need to explore the entire state-space by itself, but rather all agents collectively need to explore the state-space. This is one of the advantages of our multi-agent setting. In practice, this could be useful since the agents can divide the entire state-space into sections, each of which can be explored by a different agent in parallel.

Assumption 2.3. *We assume that the behavior policies are such that the aggregated steady state probabilities* $\big($*i.e.,* $\sum_{k=1}^{K} \tau_k D^{\phi_k}\big)$ *are strictly positive for every state.*

The empirical problem for the multi-agent case is then given by:

$$\min_{\theta} J_{\mathrm{emp}}(\theta) = \min_{\theta} \frac{1}{2}\big\|\widehat{A}\theta - \widehat{b}\big\|_{\widehat{C}^{-1}}^2 + \frac{\eta}{2}\big\|\theta - \theta_{\mathrm{p}}\big\|_{\widehat{U}}^2 \tag{2.32}$$

$$\widehat{A}_k = \sum_{n=1}^{N-H} \frac{\widehat{A}_{k,n}}{N-H}, \quad \widehat{b}_k = \sum_{n=1}^{N-H} \frac{\widehat{b}_{k,n}}{N-H}, \quad \widehat{C}_k = \sum_{n=1}^{N-H} \frac{\widehat{C}_{k,n}}{N-H} \tag{2.33a}$$

$$\widehat{A} = \sum_{k=1}^{K} \tau_k \widehat{A}_k, \qquad \widehat{b} = \sum_{k=1}^{K} \tau_k \widehat{b}_k, \qquad \widehat{C} = \sum_{k=1}^{K} \tau_k \widehat{C}_k \tag{2.33b}$$

Assumption 2.4. *We assume that $\widehat{C}$ and $\widehat{A}$ are positive definite and invertible, respectively.*

It is easy to show that Assumption 2.4 is equivalent to assuming that each state has been visited at least once while collecting data. Intuitively, this assumption is necessary for any policy evaluation algorithm since one cannot expect to estimate the value function of states that have never been visited. Since we are interested in deriving a distributed algorithm we define local copies $\{\theta_k\}$ and rewrite (2.32) equivalently in the form:

$$\begin{aligned} \min_{\theta} \; & \frac{1}{2}\Big\| \sum_{k=1}^{K} \tau_k \big(\widehat{A}_k \theta_k - \widehat{b}_k\big) \Big\|^2_{\left(\sum_{k=1}^{K} \tau_k \widehat{C}_k\right)^{-1}} + \sum_{k=1}^{K} \tau_k \frac{\eta}{2} \big\|\theta_k - \theta_{\mathrm{p}}\big\|^2_{\widehat{U}_k} \\ \text{s.t} \quad & \theta_1 = \theta_2 = \cdots = \theta_K \end{aligned} \tag{2.34}$$

The above formulation although correct is not useful because the gradient with respect to any individual θ_k depends on all the data from all agents and we want to derive an algorithm that only relies on local data. To circumvent this inconvenience, we reformulate (2.32) into an equivalent problem. To this end, we note that every quadratic function can be expressed in terms of its conjugate function as:

$$\frac{1}{2}\|A\theta - b\|^2_{C^{-1}} = \max_{\omega} \left(-(A\theta - b)^T \omega - \frac{1}{2}\|\omega\|^2_C \right) \tag{2.35}$$

Therefore, expression (2.32) can equivalently be rewritten as:

$$\min_{\theta} \max_{\omega} \sum_{k=1}^{K} \tau_k \left(\frac{\eta}{2}\|\theta - \theta_{\mathrm{p}}\|^2_{\widehat{U}_k} - \omega^T (\widehat{A}_k \theta - \widehat{b}_k) - \frac{1}{2}\|\omega\|^2_{\widehat{C}_k} \right) \tag{2.36}$$

Remark 2.5. *The saddle-point of* (2.36) *is given by*

$$\begin{bmatrix} \widehat{\theta}^o \\ \widehat{\omega}^o \end{bmatrix} = \begin{bmatrix} \left(\widehat{A}^T\widehat{C}^{-1}\widehat{A} + \eta\widehat{U}\right)^{-1}\left(\eta\widehat{U}\theta_p + \widehat{A}^T\widehat{C}^{-1}\widehat{b}\right) \\ \widehat{C}^{-1}\widehat{b} - \widehat{C}^{-1}\widehat{A}\widehat{\theta}^o \end{bmatrix} \tag{2.37}$$

Proof. $\widehat{\theta}^o$ and $\widehat{\omega}^o$ are obtained by equating the gradient of (2.36) to zero and solving for θ and ω. □

Defining local copies for the primal and dual variables we can write:

$$\begin{aligned} &\min_{\theta}\max_{\omega}\sum_{k=1}^{K}\tau_k\left(\frac{\eta}{2}\|\theta_k-\theta_{\mathrm{p}}\|^2_{\widehat{U}_k} - \omega_k^T(\widehat{A}_k\theta_k-\widehat{b}_k) - \frac{1}{2}\|\omega_k\|^2_{\widehat{C}_k}\right) \\ &\text{s.t} \qquad \theta_1=\theta_2=\cdots=\theta_K \qquad \omega_1=\omega_2=\cdots=\omega_K \end{aligned} \tag{2.38}$$

Now to derive a learning algorithm we rewrite (2.38) in an equivalent more convenient manner (the following steps can be seen as an extension to the primal-dual case of similar steps used in [70]). We start by defining the following network-wide magnitudes:

$$\begin{aligned} &\check{\theta} = \mathrm{col}\{\theta_k\}_{k=1}^K, \quad \check{\omega} = \mathrm{col}\{\omega_k\}_{k=1}^K, \quad \check{b} = \mathrm{col}\{\tau_k\widehat{b}_k\}_{k=1}^K \\ &\check{A} = \mathrm{diag}\{\tau_k\widehat{A}_k\}_{k=1}^K, \quad \check{C} = \mathrm{diag}\{\tau_k\widehat{C}_k\}_{k=1}^K, \quad \check{\theta}_{\mathrm{p}} = \mathbb{1}\otimes\theta_{\mathrm{p}} \\ &\check{L} = L\otimes I_M, \quad V = H(I_K-\Lambda)^{1/2}H^T/\sqrt{2}, \quad \check{V} = V\otimes I_M \end{aligned} \tag{2.39}$$

We remind the reader that H and Λ were defined in Remark 2.4. We further clarify that $(I_K-\Lambda)^{\frac{1}{2}}$ is the entrywise square root of the positive definite diagonal matrix $I_K-\Lambda$. The notation $\mathrm{col}\{y\}_{k=1}^K$ refers to stacking vectors y_k from 1 to K into one larger vector. Moreover, $\mathrm{diag}\{Y_k\}_{k=1}^K$ is a block diagonal matrix with matrices Y_k as its diagonal elements.

Remark 2.6. *Due to Remark 2.4, it follows that the bases of the null-spaces of V and $\check{V}$*

are given by $\{\mathbb{1}\}$ *and* $\{\mathbb{1} \otimes I_M\}$*, respectively. Therefore, we get:*

$$\theta_1 = \theta_2 = \cdots = \theta_K \iff \check{V}\check{\theta} = 0 \tag{2.40a}$$

$$\omega_1 = \omega_2 = \cdots = \omega_K \iff \check{V}\check{\omega} = 0 \tag{2.40b}$$

Using (2.40) we transform (2.38) into the following equivalent formulation:

$$\begin{aligned} \min_{\theta} \max_{\omega} \; & \underbrace{\frac{\eta}{2}\|\check{\theta} - \check{\theta}_{\mathrm{p}}\|^2_{\check{U}} - \check{\omega}^T(\check{A}\check{\theta} - \check{b}) - \frac{1}{2}\|\omega\|^2_{\check{C}}}_{=F(\check{\theta},\check{\omega})} \\ \text{s.t} \quad & \check{V}\check{\theta} = 0 \qquad \check{V}\check{\omega} = 0 \end{aligned} \tag{2.41}$$

We next introduce the constraints into the cost by using Lagrangian and extended Lagrangian terms as follows:

$$\min_{\check{\theta}, y^{\omega}} \max_{\check{\omega}, y^{\theta}} F(\check{\theta}, \check{\omega}) + y^{\theta T}\check{V}\check{\theta} - y^{\omega T}\check{V}\check{\omega} + \frac{\|\check{V}\check{\theta}\|^2}{2} - \frac{\|\check{V}\check{\omega}\|^2}{2} \tag{2.42}$$

where y^{ω} and y^{θ} are the dual variables of $\check{\omega}$ and $\check{\theta}$, respectively. Now we perform incremental gradient ascent on $\check{\omega}$ and gradient descent on y^{ω} to obtain the following updates:[3]

$$\psi^{\omega}_{i+1} = \check{\omega}_i + \mu_{\omega}\nabla_{\check{\omega}}F(\check{\theta}_i, \check{\omega}_i) \tag{2.43a}$$

$$\phi^{\omega}_{i+1} = \psi^{\omega}_{i+1} - \mu_{\omega,2}\check{V}^2\psi^{\omega}_{i+1} \overset{\mu_{\omega,2}=1}{=} (I + \check{L})/2\psi^{\omega}_{i+1} \tag{2.43b}$$

$$\check{\omega}_{i+1} = \phi^{\omega}_{i+1} - \mu_{\omega,3}\check{V}y^{\omega} \overset{\mu_{\omega,3}=1}{=} \phi^{\omega}_{i+1} - \check{V}y^{\omega}_i \tag{2.43c}$$

$$y^{\omega}_{i+1} = y^{\omega}_i + \mu_{\omega,4}\check{V}\check{\omega}_{i+1} \overset{\mu_{\omega,4}=1}{=} y^{\omega}_i + \check{V}\check{\omega}_{i+1} \tag{2.43d}$$

[3]In incremental gradient ascent, a gradient update is performed per term of (2.42) instead of doing only one update with the full gradient. We perform incremental updates in our distributed setting because it provides better stability than regular gradient ascent [70, 71].

where in (2.43b) we used $\check{V}^2 = I - \check{L}$. Combining (2.43b) and (2.43c) we get:

$$\psi_{i+1}^{\omega} = \check{\omega}_i + \mu_\omega \nabla_{\check{\omega}} F(\check{\theta}_i, \check{\omega}_i) \tag{2.44a}$$

$$\check{\omega}_{i+1} = (I + \check{L})\psi_{i+1}^{\omega}/2 - \check{V} y_i^{\omega} \tag{2.44b}$$

$$y_{i+1}^{\omega} = y_i^{\omega} + \check{V}\check{\omega}_{i+1} \tag{2.44c}$$

Using (2.44b) to calculate $\check{\omega}_{i+1} - \check{\omega}_i$ we get:

$$\check{\omega}_{i+1} - \check{\omega}_i = (I + \check{L})\left(\psi_{i+1}^{\omega} - \psi_i^{\omega}\right)/2 - \check{V}\left(y_i^{\omega} - y_{i-1}^{\omega}\right) \tag{2.45}$$

Substituting (2.44c) into (2.45) we get:

$$\check{\omega}_{i+1} = (I + \check{L})\left(\psi_{i+1}^{\omega} + \check{\omega}_i - \psi_i^{\omega}\right)/2 \tag{2.46}$$

Which we rewrite as:

$$\psi_{i+1}^{\omega} = \check{\omega}_i + \mu_\omega \nabla_{\check{\omega}} F(\check{\theta}_i, \check{\omega}_i) \tag{2.47a}$$

$$\phi_{i+1}^{\omega} = \psi_{i+1}^{\omega} + \check{\omega}_i - \psi_i^{\omega} \tag{2.47b}$$

$$\check{\omega}_{i+1} = (I + \check{L})\phi_{i+1}^{\omega}/2 \tag{2.47c}$$

Notice that steps (2.43)-(2.47) allow us to get rid of y_i^{ω}. Performing incremental gradient descent on $\check{\theta}$ and gradient ascent on y^{θ} and following equivalently (2.43)-(2.47) we get:

$$\psi_{i+1}^{\theta} = \check{\theta}_i - \mu_\theta \nabla_{\check{\theta}} F(\check{\theta}_i, \check{\omega}_i) \tag{2.48a}$$

$$\phi_{i+1}^{\theta} = \psi_{i+1}^{\theta} + \check{\theta}_i - \psi_i^{\theta} \tag{2.48b}$$

$$\check{\theta}_{i+1} = (I + \check{L})\phi_{i+1}^{\theta}/2 \tag{2.48c}$$

Combining (2.47) and (2.48) and defining $\psi = \mathrm{col}\{\psi^{\omega}, \psi^{\theta}\}$ (and similarly for ϕ) we arrive at algorithm (2.1), which is a fully distributed algorithm.

Algorithm 2.1 Processing steps at node k

Initialize: $\theta_{k,0}$ and $\omega_{k,0}$ arbitrarily and let $\psi_{k,0} = [{\theta_{k,0}}^T, {\omega_{k,0}}^T]^T$.
for $i = 0, 1, 2 \ldots$: **do**

$$\psi_{k,i+1} = \begin{bmatrix} \omega_{k,i} + \tau_k \mu_\omega \big(\widehat{b}_k - \widehat{A}_k \theta_{k,i} - \widehat{C}_k \omega_{k,i}\big) \\ \theta_{k,i} - \tau_k \mu_\theta \big(\eta \widehat{U}_k (\theta_{k,i} - \theta_{\mathrm{p}}) - \widehat{A}_k \omega_{k,i}\big) \end{bmatrix} \tag{2.50a}$$

$$\phi_{k,i+1} = \psi_{k,i+1} + \begin{bmatrix} \omega_{k,i} \\ \theta_{k,i} \end{bmatrix} - \psi_{k,i} \tag{2.50b}$$

$$\begin{bmatrix} \omega_{k,i+1} \\ \theta_{k,i+1} \end{bmatrix} = \Big(\phi_{k,i+1} + \sum_{n \in \mathcal{N}_k} l_{nk} \phi_{n,i+1}\Big)/2 \tag{2.50c}$$

end for

Theorem 2.1. *If Assumption 2.4 is satisfied and the step-sizes μ_ω and μ_θ are small enough while satisfying the following inequality:*

$$\frac{\mu_\omega}{\mu_\theta} > \eta \frac{\lambda_{max}(\widehat{U})}{\lambda_{max}(\widehat{C})} + 2\sqrt{\frac{\mu_\omega}{\mu_\theta} \frac{\lambda_{max}(\widehat{A}\widehat{C}^{-1}\widehat{A}^T)}{\lambda_{max}(\widehat{C})}} \tag{2.49}$$

then the iterates $\theta_{k,i}$ and $\omega_{k,i}$ generated by algorithm 2.1 converge linearly to (2.37).

Proof. See appendix 2.A. □

Condition (2.49) can always be satisfied by making μ_ω/μ_θ sufficiently large. Note that when the step-size is small enough, the convergence rate α depends on two factors: the spectrum of $\rho(I - \mu K^{-1}\Lambda_G)$ (which is also the convergence rate of a centralized gradient descent implementation) and the second biggest eigenvalue of the combination matrix. This implies that when the network is densely connected, the factor that determines the convergence rate of the algorithm is the eigenstructure of the saddle-point matrix G. On the contrary, when the network is sparsely connected, the rate at which the agents' information diffuses across the network is the factor that determines the convergence rate of the algorithm.

Algorithm 2.1 has the inconvenience that at every iteration each agent has to calculate the exact local gradient (i.e., all data samples have to be used), which computationally might be demanding for cases with big data. Therefore we add a variance reduced gradient strategy

Algorithm 2.2 AVRG for N data points and loss function Q

Initialize: θ_0^0 arbitrarily; $g^0 = 0$; $\nabla Q_n(\theta_0^0) \leftarrow 0,\ 1 \leq n \leq N$.
for $e = 0, 1, \ldots$: **do**
  Generate a random permutation function $\boldsymbol{\sigma}^e$ and set $g^{e+1} = 0$
  for $i = 0, 1, \ldots, N-1$: **do**

$$n = \boldsymbol{\sigma}^e(i) \tag{2.52}$$
$$\theta_{i+1}^e = \theta_{k,i}^e - \mu\big(\nabla Q_n(\theta_i^e) - \nabla Q_n(\theta_0^e) + g^e\big) \tag{2.53}$$
$$g^{e+1} \leftarrow g^{e+1} + \nabla Q_n(\theta_i^e)/N \tag{2.54}$$

  end for
  $\theta_0^{e+1} = \theta_N^e$
end for

to Algorithm 2.1. More specifically, we use the AVRG [55] (amortized variance-reduced gradient) technique (Algorithm 2.2).

The AVRG strategy is a single agent algorithm designed to minimize functions of the form:

$$\min_{\theta} \sum_{n=1}^{N} Q_n(\theta) \tag{2.51}$$

where Q_n is some loss function evaluated at the $n-$th data point. The main difference between standard stochastic gradient descent and AVRG is that in AVRG at every epoch the estimated gradients are collected in a vector g, which is used in the following epoch to reduce the variance of the gradient estimates. In the listing corresponding to Algorithm 2 we introduced an epoch index e and a uniform random permutation function $\boldsymbol{\sigma}^e$. The epoch index is due to the fact that AVRG relies on random reshuffling (which is why the permutation function is necessary) and sampling without replacement (hence, one epoch is one pass over each data point). The gradient estimates produced by the AVRG strategy are subject to both bias and variance, however both decay over time and therefore do not jeopardize convergence. The resulting algorithm from the combination of algorithm 2.1 and AVRG (which we refer to as Fast Diffusion for Policy Evaluation) relies on stochastic gradients (as opposed to exact gradient calculations) and as a consequence is more computationally efficient than Algorithm 2.1, while still retaining the convergence guarantees.

Algorithm 2.3 *Fast Diffusion for Policy Evaluation* at node k

Distribute the $N-H$ data points into J mini-batches of size $|\mathcal{J}_j|$; where $\mathcal{J}_j$ is the j-th mini-batch.

Initialize: $\theta_{k,0}^0$ and $\omega_{k,0}^0$ arbitrarily; let $\psi_{k,0}^0 = [{\theta_{k,0}^0}^T, {\omega_{k,0}^0}^T]^T$, $g_k^0 = 0$; $\beta_{k,n}(\theta_{k,0}^0, \omega_{k,0}^0) \leftarrow 0$, $1 \leq n \leq N-H$

for $e = 0, 1, \ldots$: **do**

Generate a random permutation function of the mini-batches $\boldsymbol{\sigma}_k^e$

Set $g_k^{e+1} = 0$

for $i = 0, 1, \ldots, N-1$: **do**

Generate the local stochastic gradients:

$$j = \boldsymbol{\sigma}_k^e(i) \tag{2.55a}$$

$$\beta_k(\theta_{k,i}^e, \omega_{k,i}^e) = g_k^e + \frac{1}{|\mathcal{J}_j|}\sum_{l\in\mathcal{J}_j}\big(\beta_{k,l}(\theta_{k,i}^e, \omega_{k,i}^e) - \beta_{k,l}(\theta_{k,0}^e, \omega_{k,0}^e)\big) \tag{2.55b}$$

$$g_k^{e+1} \leftarrow g_k^{e+1} + \frac{1}{N-H}\sum_{l\in\mathcal{J}_j}\beta_{k,l}(\theta_{k,i}^e, \omega_{k,i}^e) \tag{2.55c}$$

Update $[\theta_{k,i+1}^e, \omega_{k,i+1}^e]^T$ with exact diffusion:

$$\psi_{k,i+1}^e = \begin{bmatrix} \theta_{k,i}^e \\ \omega_{k,i}^e \end{bmatrix} - \tau_k \begin{bmatrix} \mu_\theta & 0 \\ 0 & \mu_\omega \end{bmatrix} \beta_k(\theta_{k,i}^e, \omega_{k,i}^e) \tag{2.56a}$$

$$\phi_{k,i+1}^e = \psi_{k,i+1}^e + \begin{bmatrix} \theta_{k,i}^e \\ \omega_{k,i}^e \end{bmatrix} - \psi_{k,i}^e \tag{2.56b}$$

$$\begin{bmatrix} \theta_{k,i+1}^e \\ \omega_{k,i+1}^e \end{bmatrix} = \left(\phi_{k,i+1}^e + \sum_{n\in\mathcal{N}_k} l_{nk}\phi_{n,i+1}^e\right)/2 \tag{2.56c}$$

end for

$$\begin{bmatrix} \theta_{k,0}^{e+1} \\ \omega_{k,0}^{e+1} \end{bmatrix} = \begin{bmatrix} \theta_{k,J}^e \\ \omega_{k,J}^e \end{bmatrix} \tag{2.57}$$

end for

In the listing of *FDPE*, we introduce $\boldsymbol{\sigma}_k^e$, $\mathcal{J}_j$, and $\beta_{k,j}(\theta, \omega)$, where $\boldsymbol{\sigma}_k^e$ indicates a random permutation of the J mini-batches of the k-th agent, which is generated at the beginning of

epoch e; $\mathcal{J}_j$ is the j-th mini-batch and $\beta_{k,l}(\theta,\omega)$ is defined as follows:

$$\beta_{k,l}(\theta,\omega)=\left[\begin{array}{c}\nabla_\theta F_{k,l}(\theta,\omega)\\ -\nabla_\omega F_{k,l}(\theta,\omega)\end{array}\right]=\left[\begin{array}{c}\eta\widehat{U}_{k,l}(\theta-\theta_{\mathrm{p}})-\widehat{A}_{k,l}^T\omega\\ \widehat{A}_{k,l}\theta-\widehat{b}_{k,l}+\widehat{C}_{k,l}\omega\end{array}\right] \tag{2.58}$$

We remark that algorithm 2.1 is a special case of FDPE, which corresponds to the case where the mini-batch size is selected equal to the whole batch of data. Note that the choice of the mini-batch size provides a communication-computation trade-off. As the number of mini-batches diminishes so do the communication requirements per epoch. However, more gradients need to be calculated per update and hence more gradient calculations might be required to achieve a desired error. Obviously the optimal amount of mini-batches J to minimize the overall time of the optimization process depends on the particular hardware availability for each implementation. Note that the only difference between update equations (2.50) and (2.56) is that the updates which correspond to algorithm 2.1 use exact local gradients, while (2.56) use stochastic approximations obtained through equations (2.55).

Theorem 2.2. *If Assumption 2.4 is satisfied and the step-sizes μ_ω and μ_θ are small enough while satisfying inequality* (2.49)*, then the iterates $\theta_{k,i}^e$ and $\omega_{k,i}^e$ generated by FDPE converge linearly to* (2.37)*.*

Proof. See appendix 2.B. □

The main difference between theorems 2.1 and 2.2 is that there are more constraints on the step sizes due to the gradient noise (i.e., smaller step sizes might be necessary). We clarify that the proof of theorem 2.1 included in appendix 2.A is a special case of the proof of theorem 2.2. We include both of them for convenience of the reader, because the proof of 2.2 is long a demanding, while the other one is much shorter.

2.4 Multi-Agent Reinforcement Learning

In this section we derive a cost for the MARL case that has the same form as (2.32) and therefore shows that algorithms 2.1 and 2.3 are also applicable for this scenario.

The network structure is the same as in the previous section. The difference with the previous section is that in the MARL case the agents interact with a unique environment and with each other, and have a common goal. Therefore, in this section we refer to the collection of all agents as a team. This setup is modeled as a MA-MDP defined by the tuple $(K,\mathcal{S},\mathcal{A}^k,\mathcal{P},r^k)$. We recall that $\mathcal{S}$ is a set of global states shared by all agents and $\mathcal{A}^k$ is the set of actions available to agent k of size $A^k = |\mathcal{A}^k|$. We refer to $\bar{\mathcal{A}} = \prod_{k=1}^{K} \mathcal{A}^k$ as the set of team actions. $\mathcal{P}(s'|s,a)$ is the defined as before but considering global states and team actions, and $r^k : \mathcal{S} \times \bar{\mathcal{A}} \times \mathcal{S} \to \mathbb{R}$ is the reward function of agent k. Specifically, $r^k(s,\bar{a},s')$ is the expected reward of decision maker k when the team transitions to state $s' \in \mathcal{S}$ from state $s \in \mathcal{S}$ having taken team action $\bar{a} \in \bar{\mathcal{A}}$. What distinguishes this model from the one in the previous section is that the transition probabilities and the reward functions of the individual agents depend not only on their own actions but on the actions of all other agents. The goal of all the agents is to maximize the aggregated return and hence in this case the value function is defined as:

$$v^{\pi}(s) = \sum_{t=i}^{\infty} \frac{\gamma^{t-i}}{K} \left(\sum_{k=1}^{K} \mathbb{E}\left[\boldsymbol{r}^k(\boldsymbol{s}_t, \bar{\boldsymbol{a}}_t, \boldsymbol{s}_{t+1}) | \boldsymbol{s}_i = s \right] \right) \tag{2.59}$$

introducing a global reward as $\boldsymbol{r}(\boldsymbol{s}_t, \bar{\boldsymbol{a}}_t, \boldsymbol{s}_{t+1}) = K^{-1} \sum_{k=1}^{K} \boldsymbol{r}^k(\boldsymbol{s}_t, \bar{\boldsymbol{a}}_t, \boldsymbol{s}_{t+1})$ equation (2.59) becomes identical to (2.1). Therefore, with the understanding that in this case the states and the policies, $\pi(\bar{a}|s) = \pi(a^1, \cdots, a^K|s)$ and $\phi(a^1, \cdots, a^K|s)$, are global (and hence also the feature vectors and sampling weights are global[4]), the rest of the derivation follows identically to Section 2.2. Therefore, the empirical problem becomes like (2.24) with the

[4]Note that in this case if sampling weights were to be used for off-policy operation, every agent would require knowledge of the teams behavior policy.

following estimates:

$$\widehat{A} = \sum_{n=1}^{N-H} \frac{\widehat{A}_n}{N-H} \tag{2.60}$$

$$\widehat{b} = \sum_{n=1}^{N-H} \sum_{k=1}^{K} \frac{\widehat{b}_{k,n}}{K(N-H)} \tag{2.61}$$

$$\widehat{C} = \sum_{n=1}^{N-H} \frac{\widehat{C}_n}{N-H} \tag{2.62}$$

Defining $\widehat{A}_{k,n} = \widehat{A}_n$ we can write $\widehat{A}_n = \sum_{k=1}^{K} \widehat{A}_{k,n}/K$ (and similarly for $\widehat{C}_n$). Equations (2.60) become exactly like (2.33) with $\tau_K = K^{-1}$, and therefore both algorithms can be applied to MARL scenarios without changes. We clarify that in this setting the algorithms are still fully decentralized where agents only use their local rewards and are able to leverage all agent's data due to the combination step (2.50c), (2.56c).

2.5 Experiments

In this section we show two simulations corresponding to the two distinct scenarios that *FDPE* can be applied to.

2.5.1 Experiment I

This experiment corresponds to the scenario of Section 2.3. We consider a situation in which the MDP's state space is divided among the agents for exploration.

The MDP's specifications are as follows. The state space is given by a 15×15 grid and the possible actions are: UP, DOWN, LEFT and RIGHT. The reward structure of the MDP and the target policy were generated randomly. We consider a network of 9 agents that divide the state space in 9 regions. The topology of the network and the regions assigned to the agents for exploration are shown in Figure 2.2(a). The feature vectors consist of 26

features, 25 given by radial basis functions[5] (RBF) centered in the red marks show in Figure 2.2(a) plus one bias feature fixed at +1. The behavior policy of every agent is equal to the target policy, except in the edges of its exploration region, where the probabilities of the actions that would take the agent beyond its exploration region are zeroed (the policy is further re-normalized).

In this experiment we show the bias-variance trade-off handled by the eligibility trace parameter λ, the communication-computation trade-off handled by the mini-batch size, and the performance of *FDPE* compared to the existing algorithms that can be applied to this scenario (namely *Diffusion GTD2* and *ALG2*). The hyper-parameters chosen for *FDPE* are $\tau_k = 1/9$, $H = 20$, $\eta = 0$ and $N_k = 2^{15} + H - 1$.

In figures 2.2(b) and 2.2(c) we show the bias and bias+variance curves as functions of λ and its approximations using lemmas 2.1 and 2.2, respectively[6]. Note that the expressions provided in Lemmas 2.1 and 2.2 accurately capture the dependence of the bias and variance as functions of λ. The bias curve was calculated using (2.7) and (2.19). To estimate the combined effects of the bias and variance we calculated $\|\widehat{\theta}^o(H,\lambda) - \theta^\star\|^2$ (using expressions (2.25)) 20 times with independently generated data and averaged the results. Note that the obtained curves agree with our previous discussion on the effect of the parameter λ. In this experiment, the optimal value is approximately $\lambda = 0.6$. Figure 2.2(d) shows how communication and computation can be traded through the use of the mini-batch size. To obtain this figure, we fixed $\lambda = 0.6$ and run *FDPE* until an error smaller than 10^{-10} was obtained for the different batch sizes. For each case, the step-sizes were adjusted to maximize performance for a fair comparison. Note that all points are Pareto optimal, and hence the optimal choice of mini-batch size depends on every particular implementation. The y-axis displays the amount of communication rounds that took place over the entire optimization process, while the x-axis shows the amount of sample gradients calculated. Figure 2.2(e) shows the empirical squared error for the three algorithms (each curve was obtained by

[5]Each RBF is given by $\exp\big(0.5\big((x-x_c)^2+(y-y_c)^2\big)\big)$, where x_c and y_c are the coordinates of the center of that feature and x and y are the coordinates of the agent.

[6]The constants κ_1, κ_2, κ_3 and κ_4 were chosen to better fit the curves.

averaging the squared errors from all the agents), where clearly the linear convergence of our algorithm can be seen. Finally, Figure 2.2(f) shows the mean square deviation (MSD), i.e., $\|\theta_{k,0}^e - \theta^\star\|^2$. As can be seen, our algorithm still outperforms the other algorithms in terms of convergence speed. It also has the advantage that it converges to a lower error (the dashed line) versus the other algorithms which converge to the dotted line, although in this case since the variance is high this advantage is not very significant. Note that as more data becomes available the variance of $\widehat{\theta}^o(H,\lambda)$ becomes smaller (see Lemma 2.2) and hence the advantages (in terms of convergence speed and the minimizer that the algorithms converges to) of using *FDPE* over the other algorithms becomes more pronounced. The remaining hyper-parameters for *FDPE* were: $\tau_k = 1/9$, $J = 2^{10}$ (i.e., batch size equal to 32), $\mu_\theta = 6$ and $\mu_\omega = 15$. For *Diffusion GTD2* and *ALG2* decaying step-sizes were employed to guarantee convergence. The step-sizes decayed as $\mu(1{+}0.01e)^{-1}$, where e is the epoch number (we used this decaying rule because it provided the best results). The initial step-sizes were $\mu_\theta = 1.1$ and $\mu_\omega = 2.75$ for *Diffusion GTD2*, and $\mu_\theta = 2.5$ and $\mu_\omega = 4$ for *ALG2*.

2.5.2 Experiment II

The second experiment relates to the MARL scenario of section 2.4. Similarly to [76], we consider randomly generated MDP's. We consider a random network of $K = 15$ agents. To construct the network, K agents are randomly distributed in a unit square (using a uniform distribution) and agents that are within a distance smaller than $r = 0.27$ become neighbors, which results in a sparsely connected network. The resulting network is shown in Figure 2.3(a). The combination weights are determined according to the Metropolis rule which is given by:

$$\ell_{nk} = \begin{cases} \frac{1}{\max[|\mathcal{N}_n|,|\mathcal{N}_k|]} & n \in \mathcal{N}_k \backslash \{k\} \\ 1 - \sum\limits_{m \in \mathcal{N}_k \backslash \{k\}} \ell_{mk} & n = k \end{cases} \tag{2.63}$$

The generated MDP's have 50 states and 10 actions. The transition probabilities $p(s'|s,a)$ are zero with 0.98 probability and otherwise are sampled from a uniform distribution from

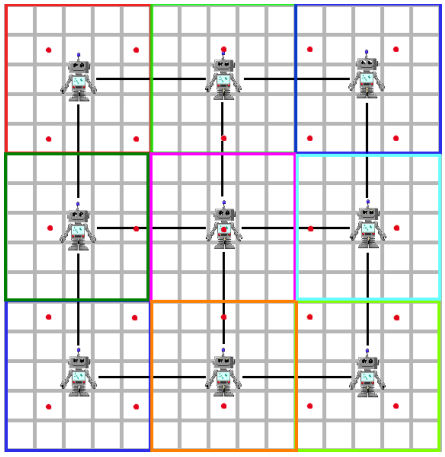
the interval [0,1]. The probabilities are further normalized. With this sampling strategy, we produce realistic MDPs in which from any given state it is only possible to transition to a small subset of the total states. The rewards $r(s'|s,a)$ are zero with 0.99 probability and otherwise are sampled from a Gaussian distribution with zero mean and standard deviation equal to 10. This sampling strategy is also devised to produce more realistic MDPs where rewards are obtained occasionally in specific state-action pairs. The entries of the target policy $\pi(a|s)$ are sampled from a uniform distribution and subsequently normalized. The transition probabilities and target policy are sampled until Assumption 1 is satisfied. We set the discount factor $\gamma = 0.93$ and the length of the feature vectors $M = 5$, where one feature is set to 1 and the remaining ones are sampled from a uniform distribution with interval $[0, 1]$. We generated $N = 2^{18} + H - 1$ team transitions. The remaining parameters for the learning algorithm are the following: $H = 20$, $\eta = 10^{-3}$, $U = I$, $\theta_{\mathrm{p}} = \theta^o(H, \lambda) + \theta_{\mathrm{n}}$ (where θ_{n} is a noise vector whose entries are sampled from a uniform distribution with variance equal to 2.5×10^{-5}) and $\tau_k = 1/15$.

We compare our algorithm with *PD-distIAG* from [58], *Diffusion GTD2* and *ALG2*. In this experiment, we test the on-policy case because *PD-distIAG* only works for this scenario. In Figures 2.3(b) and 2.3(c), we show the bias variance trade-off as a function of λ (the curves were obtained in the same manner as done in the previous experiment). Results are consistent with the ones obtained in the previous section. In this particular case the most convenient value is approximately $\lambda = 0.8$. In Figure 2.3(d) we compare the convergence rates of the different algorithms to solve the empirical problem. The hyper-parameters of all algorithms were tunned to maximize performance. The parameters for *FDPE* were: $J = 2^{12}$ (i.e., batch size equal to 64), $\mu_\theta = 10$ and $\mu_\omega = 10$. For *PD-distIAG* we used the same batch size, and the step-sizes were $\mu_\theta = 1.15$ and $\mu_\omega = 23$. For *Diffusion GTD2* and *ALG2* decaying step-sizes were employed to guarantee convergence. The step-sizes decayed as $\mu(1 + 0.01e)^{-1}$, where e is the epoch number (we used this decaying rule because it provided the best results). The initial step-sizes were $\mu_\theta = 15$ and $\mu_\omega = 7.5$ for *Diffusion GTD2*, and the same values for *ALG2*. In this experiment *FDPE*, *Diffusion GTD2* and *ALG2* show performance in accordance to our theory and the results in the previous section. *PD-distIAG*

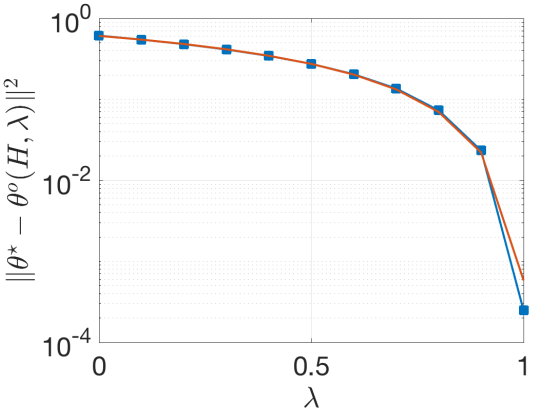
shows a linear convergence rate in accordance to the theory from [58]. However, the rate is slower than the one from our algorithm. Finally, in Figure 2.3(e) we show the MSD. Again, the faster convergence of our algorithm to its empirical minimizer implies a faster convergence in the MSD plot. In this case, the advantage of the parameter λ becomes more noticeable. Note indeed that the minimizer obtained by *FDPE* is approximately one order of magnitude smaller than the minimizer to which the other algorithms will converge (the dotted line).

2.6 Summary

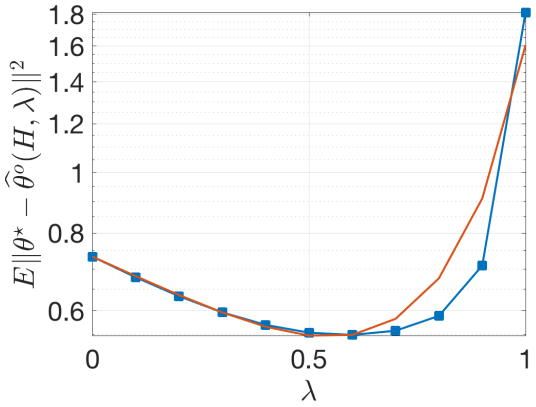
In this chapter we introduced the FDPE algorithm, a distributed policy evaluation algorithm applicable to the MARL setting. This algorithm can be used as part of on-policy algorithms for policy learning in MARL. As we mentioned in the introduction in chapter 1, the disadvantage of this kind of algorithms is that they are data inefficient. This problems is compounded in the case of MARL since the state and action spaces are larger. Therefore, in the next chapter we consider the problem of sample efficient policy learning in MARL.



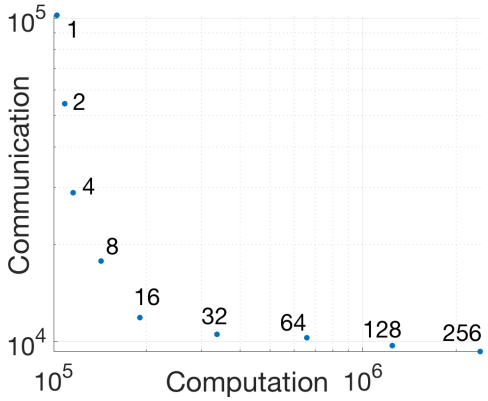
(a) MDP



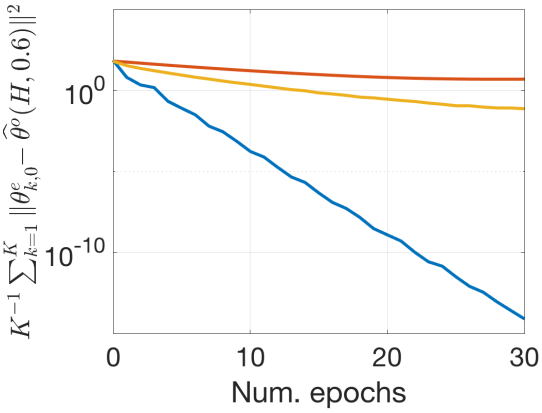
(b) Bias



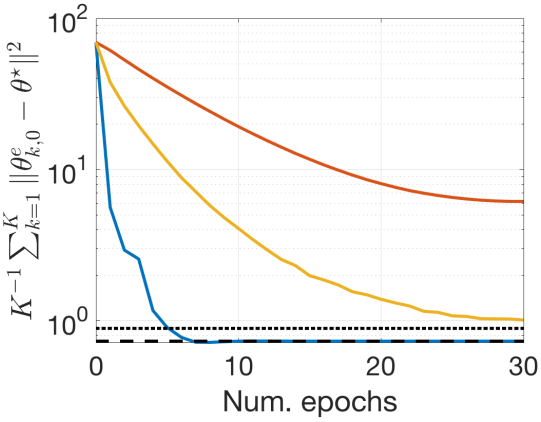
(c) Bias variance trade-off



(d) Communication vs computation

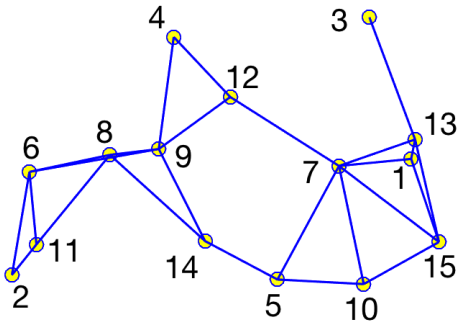


(e) Empirical error

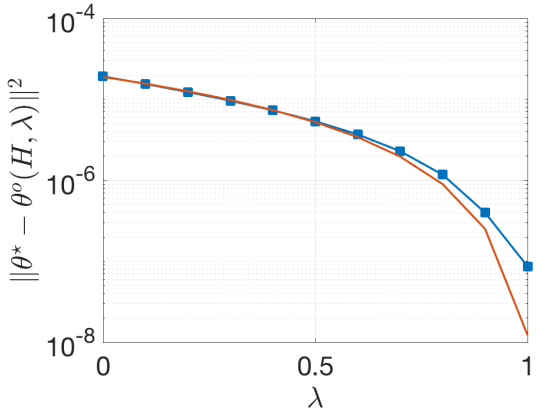


(f) Bias variance trade-off

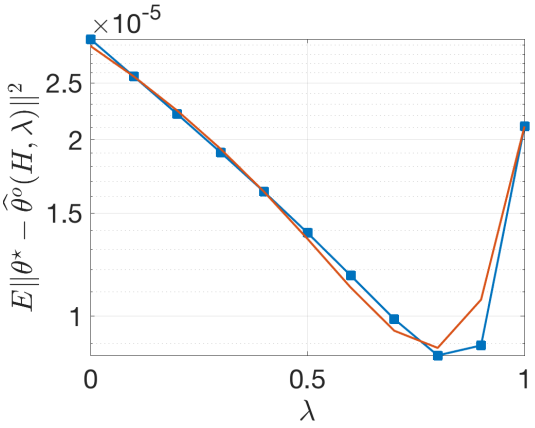
Figure 2.2: In (b) and (c) the red curves are the approximations from lemmas 2.1 and 2.2. In (e) and (f), the blue curve corresponds to *FDPE* and red and yellow correspond to *Diffusion GTD2* and *ALG2*, respectively. In (f) the dotted line is at $\|\widehat{\theta}(H, \lambda = 0) - \theta^\star\|^2$ while the dashed line is at $\|\widehat{\theta}(H, \lambda = 0.6) - \theta^\star\|^2$.



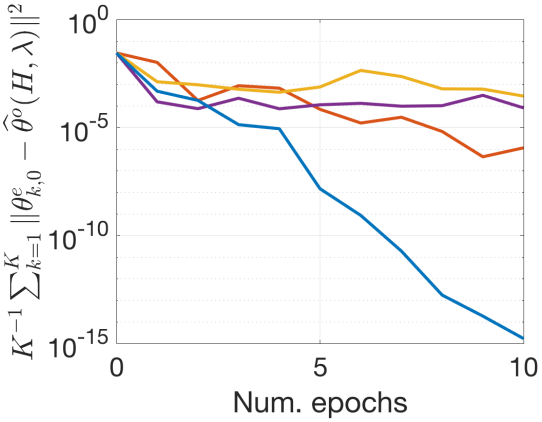
(a) Network



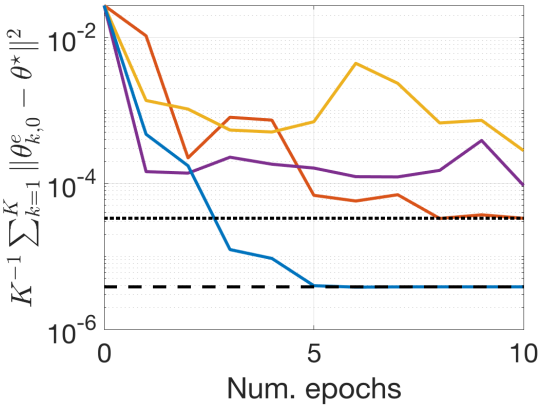
(b) Bias



(c) Bias variance trade-off



(d) Empirical error



(e) Mean square deviation

Figure 2.3: In (b) and (c) the red curves are the approximations from lemmas 2.1 and 2.2. In (d) and (e) the blue, purple, yellow and red curves correspond to *FDPE*, *Diffusion GTD2*, *ALG2* and *PD-distIAG*, respectively. In (e) the dotted line is at $\|\widehat{\theta}(H, \lambda = 0) - \theta^\star\|^2$ while the dashed line is at $\|\widehat{\theta}(H, \lambda = 0.6) - \theta^\star\|^2$.

2.A Proof of theorem 2.1

We start by setting $\mu = \mu_\theta$ and introducing the following definitions:

$$\zeta_{k,i} \triangleq \begin{bmatrix} \theta_{k,i} \\ \frac{1}{\sqrt{\upsilon}}\omega_{k,i} \end{bmatrix}, \quad \zeta_i \triangleq \mathrm{col}\{\zeta_{k,i}\}_{k=1}^K, \quad \upsilon \triangleq \frac{\mu_\omega}{\mu_\theta} \tag{2.64a}$$

$$G_k \triangleq \tau_k \begin{bmatrix} \eta\widehat{U}_k & -\sqrt{\upsilon}\widehat{A}_k^T \\ \sqrt{\upsilon}\widehat{A}_k & \upsilon\widehat{C}_k \end{bmatrix}, \quad \mathcal{G} \triangleq \mathrm{diag}\{G_k\}_{k=1}^K \tag{2.64b}$$

$$G \triangleq \sum_{k=1}^K G_k, \quad p_k \triangleq \tau_k \begin{bmatrix} \eta\widehat{U}_k\theta_{\mathrm{p}} \\ \sqrt{\upsilon}b_k \end{bmatrix}, \quad p \triangleq \mathrm{col}\{p_k\}_{k=1}^K \tag{2.64c}$$

$$\bar{L} \triangleq (L + I_K)/2, \quad \bar{\mathcal{L}} \triangleq \bar{L} \otimes I_{2M}, \quad \mathcal{V} \triangleq V \otimes I_{2M} \tag{2.64d}$$

With these definitions, we write the update equations of Algorithm 1 in the form of equations (2.44) (for both the primal and dual) in the following first order network-wide recursion:

$$\zeta_{i+1} = \bar{\mathcal{L}}\left(\boldsymbol{\zeta}_i - \mu\left(\mathcal{G}\boldsymbol{\zeta}_i - p\right)\right) - \mathcal{V}\mathcal{Y}_i \tag{2.65a}$$

$$\mathcal{Y}_{i+1} = \mathcal{Y}_i + \mathcal{V}\boldsymbol{\zeta}_{i+1} \tag{2.65b}$$

for which $\mathcal{Y}_0 = 0$. Note that the variable $\mathcal{Y}_i$ is a network-wide variable that includes both y^ω and y^θ.

Lemma 2.5. *Recursion* (2.65) *has a unique fixed point* $(\zeta^o, \mathcal{Y}^o)$*, where* $\mathcal{Y}^o$ *lies in the range space of* $\mathcal{V}$*. This fixed point satisfies the following conditions:*

$$\mu\bar{\mathcal{L}}\left(\mathcal{G}\zeta^o - p\right) + \mathcal{V}\mathcal{Y}^o = 0 \tag{2.66a}$$

$$\mathcal{V}\zeta^o = 0 \tag{2.66b}$$

It further holds that $\zeta^o = \mathbb{1}_K \otimes [\widehat{\theta}^{oT}, \widehat{\omega}^{oT}]^T$*, where* $\widehat{\theta}^o$ *and* $\widehat{\omega}^o$ *are given by* (2.37). □

Proof. See appendix 2.G. □

Subtracting ζ^o and $\mathcal{Y}^o$ from (2.65) and defining the error quantities $\widetilde{\zeta}_i = \zeta^o - \zeta_i$ and

$\widetilde{\mathcal{Y}}_i = \mathcal{Y}^o - \mathcal{Y}_i$ we get:

$$\begin{bmatrix} I & 0 \\ -\mathcal{V} & I \end{bmatrix} \begin{bmatrix} \widetilde{\zeta}_{i+1} \\ \widetilde{\mathcal{Y}}_{i+1} \end{bmatrix} = \begin{bmatrix} \bar{\mathcal{L}}\,(I - \mu\mathcal{G}) & -\mathcal{V} \\ 0 & I \end{bmatrix} \begin{bmatrix} \widetilde{\zeta}_i \\ \widetilde{\mathcal{Y}}_i \end{bmatrix} \tag{2.67}$$

Multiplying by the inverse of the leftmost matrix we get:

$$\begin{bmatrix} \widetilde{\zeta}_{i+1} \\ \widetilde{\mathcal{Y}}_{i+1} \end{bmatrix} = \begin{bmatrix} \bar{\mathcal{L}}\,(I - \mu\mathcal{G}) & -\mathcal{V} \\ \mathcal{V}\bar{\mathcal{L}}\,(I - \mu\mathcal{G}) & \bar{\mathcal{L}} \end{bmatrix} \begin{bmatrix} \widetilde{\zeta}_i \\ \widetilde{\mathcal{Y}}_i \end{bmatrix} \tag{2.68}$$

Lemma 2.6. *Through a coordinate transformation applied to* (2.68) *we obtain the following error recursion:*

$$\begin{bmatrix} \bar{x}_{i+1} \\ \hat{x}_{i+1} \end{bmatrix} = \begin{bmatrix} I_{2M} - \mu K^{-1} G & -\mu K^{-\frac{1}{2}} \mathcal{I}^T \mathcal{G} \mathcal{H}_u \\ -\frac{\mu}{\sqrt{K}} (\mathcal{H}_l^T + \mathcal{H}_r^T \mathcal{V}) \bar{\mathcal{L}} \mathcal{G} \mathcal{I} & \mathcal{D}_1 - \mu (\mathcal{H}_l^T + \mathcal{H}_r^T \mathcal{V}) \bar{\mathcal{L}} \mathcal{G} \mathcal{H}_u \end{bmatrix} \begin{bmatrix} \bar{x}_i \\ \hat{x}_i \end{bmatrix} \tag{2.69}$$

where $\mathcal{H}_l, \mathcal{H}_r, \mathcal{H}_u, \mathcal{H}_d \in \mathbb{R}^{2KM \times 4KM - 4M}$ *are some constant matrices,* $\bar{x}_i \in \mathbb{R}^{2M}$, $\hat{x}_i \in \mathbb{R}^{2M}$ *and* $\mathcal{D}_1$ *is a diagonal matrix with* $\|\mathcal{D}_1\|_2^2 = \lambda_2(\bar{L}) < 1$. *Furthermore* $\bar{x}_i$ *and* $\hat{x}_i$ *satisfy:*

$$\|\widetilde{\zeta}_i\|^2 \leq \|\bar{x}_i\|^2 + \|\mathcal{H}_u\|^2 \|\hat{x}_i\|^2, \quad \|\widetilde{\mathcal{Y}}_i\|^2 \leq \|\mathcal{H}_d\|^2 \|\hat{x}_i\|^2 \tag{2.70}$$

□

Proof. See Appendix 2.H. □

Due to Theorem 2.1 from [77], if Assumption 2.4 and the following condition are satisfied:

$$\upsilon \lambda_{\min}(\widehat{C}) > \eta \lambda_{\max}(\widehat{U}) + 2\sqrt{\upsilon \lambda_{\min}(\widehat{C}) \lambda_{\max}(\widehat{A}\widehat{C}^{-1}\widehat{A}^T)} \tag{2.71}$$

then matrix G is diagonalizable with strictly positive eigenvalues. Hence, we can write

$G = Z\Lambda_G Z^{-1}$. Therefore, defining $\check{x}_i = Z^{-1}\bar{x}_i$ we can transform (2.69) into:

$$\begin{bmatrix} \check{x}_{i+1} \\ \hat{x}_{i+1} \end{bmatrix} = \begin{bmatrix} I_{2M} - \mu K^{-1}\Lambda_G & -\mu K^{-\frac{1}{2}} Z^{-1}\mathcal{I}^T\mathcal{G}\mathcal{H}_u \\ \frac{-\mu}{\sqrt{K}}(\mathcal{H}_l^T + \mathcal{H}_r^T\mathcal{V})\bar{\mathcal{L}}\mathcal{G}\mathcal{I}Z & \mathcal{D}_1 - \mu(\mathcal{H}_l^T + \mathcal{H}_r^T\mathcal{V})\bar{\mathcal{L}}\mathcal{G}\mathcal{H}_u \end{bmatrix} \begin{bmatrix} \check{x}_i \\ \hat{x}_i \end{bmatrix} \tag{2.72}$$

Lemma 2.7. *If $\mu < K/\rho(\Lambda_G)$ then the following inequality holds:*

$$\underbrace{\begin{bmatrix} \|\check{x}_{i+1}\|^2 \\ \|\hat{x}_{i+1}\|^2 \end{bmatrix}}_{\triangleq z_{i+1}} \preceq \underbrace{\begin{bmatrix} \rho(I_{2M} - \mu K^{-1}\Lambda_G) & \mu a_2 \\ \mu^2 a_3 & \sqrt{\lambda_2(\bar{L})} + \mu^2 a_4 \end{bmatrix}}_{\triangleq B(\mu)} \begin{bmatrix} \|\check{x}_i\|^2 \\ \|\hat{x}_i\|^2 \end{bmatrix} \tag{2.73}$$

where a_2, a_3 and a_4 are positive constants. □

Proof. See Appendix 2.I. □

Computing the $1-$norm on both sides of the above inequality and using the fact that $\|B(\mu)z_i\|_1 \leq \|B(\mu)\|_1\|z_i\|_1$ we get:

$$\|\check{x}_{i+1}\|^2 + \|\hat{x}_{i+1}\|^2 \leq \|B(\mu)\|_1(\|\check{x}_i\|^2 + \|\hat{x}_i\|^2) \tag{2.74}$$

Iterating we get:

$$\|\check{x}_i\|^2 + \|\hat{x}_i\|^2 \leq \alpha^i(\|\check{x}_0\|^2 + \|\hat{x}_0\|^2) \tag{2.75}$$

$$\alpha = \max\{\rho(I - \mu K^{-1}\Lambda_G) + \mu^2 a_3, \lambda_2(\bar{L})^{\frac{1}{2}} + \mu a_2 + \mu^2 a_4\} \tag{2.76}$$

Recalling that $\check{x}_i = Z^{-1}\bar{x}_i$ and (2.70) we get:

$$\|\widetilde{\zeta}_i\|^2 \leq \max\{\|Z\|^2, \|\mathcal{H}_u\|^2\}\left(\|\check{x}_i\|^2 + \|\hat{x}_i\|^2\right) \leq \alpha^i\kappa \tag{2.77}$$

where $\kappa = \max\{\|Z\|^2, \|\mathcal{H}_u\|^2\}(\|\check{x}_0\|^2 + \|\hat{x}_0\|^2)$. Since $\alpha < 1$ (for small enough μ) we conclude that the iterates $\theta_{k,i}$ and $\omega_{k,i}$ generated by Algorithm 2.1 converge linearly to (2.37) for every agent k; which completes the proof.

2.B Proof of Theorem 2.2

We start by making a simplification in terms of notation for the sake of clarity. Since it is clear that at this point we are dealing with the empirical problem (2.38) we will drop the *hat* and refer to $\widehat{A}_{k,l}$, $\widehat{b}_{k,l}$ and $\widehat{C}_{k,l}$ as $A_{k,l}$, $b_{k,l}$ and $C_{k,l}$ respectively. We now rewrite the equations the *FDPE* Algorithm as a unique network recursion as follows:

$$\boldsymbol{\zeta}_{i+1}^e = \bar{\mathcal{L}}\left(\boldsymbol{\zeta}_0^0 - \mu\left(\boldsymbol{\mathcal{G}}_0^0\boldsymbol{\zeta}_0^0 - \boldsymbol{t}_0^0\right)\right), \qquad \text{for } i=0 \text{ and } e=0 \quad (2.78a)$$

$$\boldsymbol{\zeta}_{i+1}^e = \bar{\mathcal{L}}\left(2\boldsymbol{\zeta}_i^e - \boldsymbol{\zeta}_{i-1}^e - \mu\left(\boldsymbol{\mathcal{G}}_i^e\boldsymbol{\zeta}_i^e - \boldsymbol{\mathcal{G}}_{i-1}^e\boldsymbol{\zeta}_{i-1}^e - \boldsymbol{t}_i^0 + \boldsymbol{t}_{i-1}^0\right)\right), \qquad \text{for } i>0 \text{ and } e=0 \quad (2.78b)$$

$$\boldsymbol{\zeta}_{i+1}^e = \bar{\mathcal{L}}\left(2\boldsymbol{\zeta}_i^e - \boldsymbol{\zeta}_{i-1}^e - \mu\left(\boldsymbol{\mathcal{T}}(\boldsymbol{\zeta}_i^e) - \boldsymbol{\mathcal{G}}_{J-1}^0\boldsymbol{\zeta}_{J-1}^0 - p + \boldsymbol{t}_{J-1}^0\right)\right), \qquad \text{for } i=0 \text{ and } e=1 \quad (2.78c)$$

$$\boldsymbol{\zeta}_{i+1}^e = \bar{\mathcal{L}}\left(2\boldsymbol{\zeta}_i^e - \boldsymbol{\zeta}_{i-1}^e - \mu\left(\boldsymbol{\mathcal{T}}(\boldsymbol{\zeta}_i^e) - \boldsymbol{\mathcal{T}}(\boldsymbol{\zeta}_{i-1}^e)\right)\right), \qquad \text{else} \quad (2.78d)$$

where we defined υ, μ, $\bar{L}$ and $\bar{\mathcal{L}}$ are defined in the main document and we further define:

$$\boldsymbol{\zeta}_{k,i}^e \triangleq \begin{bmatrix} \boldsymbol{\theta}_{k,i}^e \\ \frac{1}{\sqrt{\upsilon}}\boldsymbol{\omega}_{k,i}^e \end{bmatrix}, \qquad \boldsymbol{\phi}_{k,i}^e \triangleq \begin{bmatrix} \boldsymbol{\phi}_{k,i}^{\theta,e} \\ \frac{1}{\sqrt{\upsilon}}\boldsymbol{\phi}_{k,i}^{\omega,e} \end{bmatrix}, \qquad \boldsymbol{\psi}_{k,i}^e \triangleq \begin{bmatrix} \boldsymbol{\psi}_{k,i}^{\theta,e} \\ \frac{1}{\sqrt{\upsilon}}\boldsymbol{\psi}_{k,i}^{\omega,e} \end{bmatrix} \quad (2.79a)$$

$$G_k \triangleq \tau_k \begin{bmatrix} \eta U_k & -\sqrt{\upsilon}A_k^T \\ \sqrt{\upsilon}A_k & \upsilon C_k \end{bmatrix}, \qquad G \triangleq \sum_{k=1}^K G_K, \qquad p_k \triangleq \tau_k \begin{bmatrix} \eta U_k \theta_{\mathrm{p}} \\ \sqrt{\upsilon} b_k \end{bmatrix} \quad (2.79b)$$

$$\boldsymbol{t}_{k,i}^e \triangleq \tau_k \begin{bmatrix} U_{k,\boldsymbol{\sigma}_k^e(i)}\theta_{\mathrm{p}} \\ \sqrt{\upsilon} b_{k,\boldsymbol{\sigma}_k^e(i)} \end{bmatrix}, \qquad \boldsymbol{G}_{k,i}^e \triangleq \tau_k \begin{bmatrix} \eta U_{k,\boldsymbol{\sigma}_k^e(i)} & -\sqrt{\upsilon}A_{k,\boldsymbol{\sigma}_k^e(i)}^T \\ \sqrt{\upsilon}A_{k,\boldsymbol{\sigma}_k^e(i)} & \upsilon C_{k,\boldsymbol{\sigma}_k^e(i)} \end{bmatrix} \quad (2.79c)$$

$$\boldsymbol{T}(\boldsymbol{\zeta}_{k,i}^e) \triangleq \boldsymbol{G}_{k,i}^e(\boldsymbol{\zeta}_{k,i}^e - \boldsymbol{\zeta}_{k,0}^e) + \frac{1}{J}\sum_{n=1}^J \boldsymbol{G}_{k,n}^{e-1}\boldsymbol{\zeta}_{k,n}^{e-1} \quad (2.79d)$$

$$\boldsymbol{\zeta}_i^e \triangleq \mathrm{col}\{\boldsymbol{\zeta}_{1,i}^e, \ldots, \boldsymbol{\zeta}_{K,i}^e\}, \qquad \boldsymbol{\phi}_i^e \triangleq \mathrm{col}\{\boldsymbol{\phi}_{1,i}^e, \ldots, \boldsymbol{\phi}_{K,i}^e\}, \qquad \boldsymbol{\psi}_i^e \triangleq \mathrm{col}\{\boldsymbol{\psi}_{1,i}^e, \ldots, \boldsymbol{\psi}_{K,i}^e\} \quad (2.79e)$$

$$\mathcal{G} \triangleq \mathrm{diag}\{G_1, \cdots, G_K\}, \qquad p \triangleq \mathrm{col}\{p_1, p_2, \cdots, p_K\} \quad (2.79f)$$

$$\boldsymbol{\mathcal{G}}_i^e \triangleq \mathrm{diag}\{\boldsymbol{G}_{1,i}^e, \cdots, \boldsymbol{G}_{K,i}^e\}, \qquad \boldsymbol{\mathcal{T}}(\boldsymbol{\zeta}_i^e) \triangleq \mathrm{col}\{\boldsymbol{T}(\boldsymbol{\zeta}_{1,i}^e), \cdots, \boldsymbol{T}(\boldsymbol{\zeta}_{K,i}^e)\} \quad (2.79g)$$

$$\boldsymbol{t}_i^e \triangleq \mathrm{col}\{\boldsymbol{t}_{1,i}^e, \boldsymbol{t}_{2,i}^e, \cdots, \boldsymbol{t}_{K,i}^e\} \quad (2.79h)$$

we further clarify that the notation $A_{k,\boldsymbol{\sigma}_k^e(i)}$ refers to the k-th agent's estimate of A using all the samples from the $\boldsymbol{\sigma}_k^e(i)$-th mini-batch (if no mini-batches are being used, $A_{k,\boldsymbol{\sigma}_k^e(i)}$ is

just a sample estimate, otherwise it's the empirical average using the samples corresponding to the $\boldsymbol{\sigma}_k^e(i)$-th mini-batch). There are two reasons why we are getting four different update equations in (2.78). In the first place, during the very first iteration (i.e., $e = 0$ and $i = 0$) there's no variance reduction or correction step, which accounts for (2.78a). In the rest of the iterations within the first epoch (i.e., (2.78b)) there is a correction step, however there is still no variance reduction. Update equation (2.78c) reflects the fact that during the first iterate of the second epoch the old gradients in the correction step don't have variance reduction (yet the new ones do). Equation (2.78d) is for the rest of the recursions where both the correction step and variance reduction are present. Note that (2.78) describes a second order recursion. From here we take steps to rewrite the update equations as a first order recursion driven by gradient noise.

Lemma 2.8. *If $\boldsymbol{\mathcal{Y}}_0^0$ is initialized to $\boldsymbol{\mathcal{Y}}_0^0 = 0$, then recursion* (2.78) *is equivalent to:*

$$\boldsymbol{\zeta}_{i+1}^e = \begin{cases} \bar{\mathcal{L}}\left(\boldsymbol{\zeta}_i^e - \mu\left(\boldsymbol{\mathcal{G}}\boldsymbol{\zeta}_i^e - \boldsymbol{t}_i^e\right)\right) - \mathcal{V}\boldsymbol{\mathcal{Y}}_i^e, & \textit{for } e = 0 \\ \bar{\mathcal{L}}\left(\boldsymbol{\zeta}_i^e - \mu\left(\boldsymbol{\mathcal{T}}(\boldsymbol{\zeta}_i^e) - p\right)\right) - \mathcal{V}\boldsymbol{\mathcal{Y}}_i^e, & \textit{else} \end{cases} \tag{2.80a}$$

$$\boldsymbol{\mathcal{Y}}_{i+1}^e = \boldsymbol{\mathcal{Y}}_i^e + \mathcal{V}\boldsymbol{\zeta}_{i+1}^e \tag{2.80b}$$

Proof. See subsection 2.B.1. □

Note that (2.80) is analogous to equation (2.65) from the main document, with the difference that in this case the gradients are subject to gradient noise. The first line in (2.80a) accounts only for a finite number of updates. Therefore, to analyze the convergence properties of the algorithm we only need to focus on the second line of (2.80a). In other words, for the purpose of proving convergence we will only consider updates for $e > 0$.

Now we define the following error quantities and gradient noise:

$$\widetilde{\boldsymbol{\zeta}}_{i+1}^e = \zeta^o - \boldsymbol{\zeta}_{i+1}^e \tag{2.81}$$

$$\widetilde{\boldsymbol{\mathcal{Y}}}_{i+1}^e = \mathcal{Y}^o - \boldsymbol{\mathcal{Y}}_{i+1}^e \tag{2.82}$$

$$\boldsymbol{s}(\boldsymbol{\zeta}_i^e) = \boldsymbol{\mathcal{T}}(\zeta_i^e) - \mathcal{G}\boldsymbol{\zeta}_i^e \tag{2.83}$$

Following steps (2.65) through (2.68) from the main document we get:

$$\begin{bmatrix} \widetilde{\boldsymbol{\zeta}}_{i+1}^{e} \\ \widetilde{\boldsymbol{\mathcal{Y}}}_{i+1}^{e} \end{bmatrix} = \begin{bmatrix} \bar{\mathcal{L}}\,(I-\mu\mathcal{G}) & -K\mathcal{V} \\ \mathcal{V}\bar{\mathcal{L}}\,(I-\mu\mathcal{G}) & \bar{\mathcal{L}} \end{bmatrix} \begin{bmatrix} \widetilde{\boldsymbol{\zeta}}_{i}^{e} \\ \widetilde{\boldsymbol{\mathcal{Y}}}_{i}^{e} \end{bmatrix} + \mu \begin{bmatrix} \bar{\mathcal{L}}\boldsymbol{s}(\boldsymbol{\zeta}_{i}^{e}) \\ \mathcal{V}\bar{\mathcal{L}}\boldsymbol{s}(\boldsymbol{\zeta}_{i}^{e}) \end{bmatrix} \tag{2.84}$$

Note that (2.84) has the same form as (2.68) from the main document plus a term due to the gradient noise. Since the driving matrix in (2.84) has a structure that makes it difficult to calculate its eigenstructure. We circumvent this issue by doing a coordinate transformation as the following lemma indicates.

Lemma 2.9. *Through a coordinate transformation, recursion* (2.84) *can be transformed to obtain the following recursion:*

$$\begin{aligned} \begin{bmatrix} \bar{\boldsymbol{x}}_{i+1}^{e} \\ \hat{\boldsymbol{x}}_{i+1}^{e} \end{bmatrix} = & \begin{bmatrix} I_{2M}-\mu GK^{-1} & -K^{-1/2}\mu\mathcal{I}^{T}\mathcal{G}\mathcal{H}_{u} \\ -\mu K^{-1/2}(\mathcal{H}_{l}^{T}+\mathcal{H}_{r}^{T}\mathcal{V})\bar{\mathcal{L}}\mathcal{G}\mathcal{I} & \mathcal{D}_{1}-\mu(\mathcal{H}_{l}^{T}+\mathcal{H}_{r}^{T}\mathcal{V})\bar{\mathcal{L}}\mathcal{G}\mathcal{H}_{u} \end{bmatrix} \begin{bmatrix} \bar{\boldsymbol{x}}_{i}^{e} \\ \hat{\boldsymbol{x}}_{i}^{e} \end{bmatrix} \\ & + \mu \begin{bmatrix} \mathcal{I}^{T}K^{-1/2} \\ (\mathcal{H}_{l}^{T}+\mathcal{H}_{r}^{T}\mathcal{V})\bar{\mathcal{L}} \end{bmatrix} \boldsymbol{s}(\boldsymbol{\zeta}_{i}^{e}) \end{aligned} \tag{2.85}$$

where $\mathcal{H}_l, \mathcal{H}_r, \mathcal{H}_u, \mathcal{H}_d \in \mathbb{R}^{2KM\times 4KM-4M}$ *are some constant matrices and* $\mathcal{D}_1$ *is a diagonal matrix with* $\|\mathcal{D}_1\|_2^2 = \lambda_2(\bar{L}) < 1$. *And also we have:*

$$\bar{\boldsymbol{x}}_{i}^{e} \triangleq K^{-1/2}\mathcal{I}^{T}\widetilde{\boldsymbol{\zeta}}_{i}^{e} \tag{2.86a}$$

$$\hat{\boldsymbol{x}}_{i}^{e} \triangleq \mathcal{H}_{l}^{T}\widetilde{\boldsymbol{\zeta}}_{i}^{e} + \mathcal{H}_{r}^{T}\widetilde{\boldsymbol{\mathcal{Y}}}_{i}^{e} \tag{2.86b}$$

$$\mathcal{I} \triangleq \mathbb{1}_{K}\otimes I_{2M} \tag{2.86c}$$

Proof. See subsection 2.B.2. □

Due to Theorem 2.1 from [77], if Assumption 4 and the following condition are satisfied:

$$\upsilon\lambda_{\min}(C) > \eta\lambda_{\max}(U) + 2\sqrt{\upsilon\lambda_{\min}(C)\lambda_{\max}(AC^{-1}A^{T})} \tag{2.87}$$

then matrix G is diagonalizable with strictly positive eigenvalues. Hence, we can write

$G = Z\Lambda_G Z^{-1}$. Therefore, defining $\check{\boldsymbol{x}}_i = Z^{-1}\bar{\boldsymbol{x}}_i$ we can transform (2.85) into:

$$\begin{bmatrix} \check{\boldsymbol{x}}^e_{i+1} \\ \hat{\boldsymbol{x}}^e_{i+1} \end{bmatrix} = \begin{bmatrix} I_{2M} - \mu K^{-1}\Lambda_G & -\mu K^{-\frac{1}{2}} Z^{-1}\mathcal{I}^T\mathcal{G}\mathcal{H}_u \\ \frac{-\mu}{\sqrt{K}}(\mathcal{H}_l^T + \mathcal{H}_r^T\mathcal{V})\bar{\mathcal{L}}\mathcal{G}\mathcal{I}Z & \mathcal{D}_1 - \mu(\mathcal{H}_l^T + \mathcal{H}_r^T\mathcal{V})\bar{\mathcal{L}}\mathcal{G}\mathcal{H}_u \end{bmatrix} \begin{bmatrix} \check{\boldsymbol{x}}^e_i \\ \hat{\boldsymbol{x}}^e_i \end{bmatrix} + \mu \begin{bmatrix} Z^{-1}\mathcal{I}^T K^{-1/2} \\ (\mathcal{H}_l^T + \mathcal{H}_r^T\mathcal{V})\bar{\mathcal{L}} \end{bmatrix} \boldsymbol{s}(\boldsymbol{\zeta}^e_i) \tag{2.88}$$

Lemma 2.10. *If $\mu \leq K/\rho(\Lambda_G)$ then the following inequality holds:*

$$\begin{aligned} \begin{bmatrix} \|\bar{\boldsymbol{x}}^e_{i+1}\|^2 \\ \|\hat{\boldsymbol{x}}^e_{i+1}\|^2 \end{bmatrix} \preceq & \begin{bmatrix} 1 - \mu a_1 K^{-1} & \mu a_2 \\ \mu^2 a_6 & \sqrt{\lambda_2(\bar{L})} + \mu^2 a_5 \end{bmatrix} \begin{bmatrix} \|\bar{\boldsymbol{x}}^e_i\|^2 \\ \|\hat{\boldsymbol{x}}^e_i\|^2 \end{bmatrix} \\ & + \mu \begin{bmatrix} a_3 & \|\mathcal{H}_u\|^2 a_3 \\ \mu a_7 & \mu\|\mathcal{H}_u\|^2 a_7 \end{bmatrix} \begin{bmatrix} \|\bar{\boldsymbol{x}}^e_i - \bar{\boldsymbol{x}}^e_0\|^2 \\ \|\hat{\boldsymbol{x}}^e_i - \hat{\boldsymbol{x}}^e_0\|^2 \end{bmatrix} \\ & + \mu \begin{bmatrix} a_4 & \|\mathcal{H}_u\|^2 a_4 \\ \mu a_8 & \mu\|\mathcal{H}_u\|^2 a_8 \end{bmatrix} \frac{1}{J} \sum_{n=1}^{J} \begin{bmatrix} \|\bar{\boldsymbol{x}}^{e-1}_J - \bar{\boldsymbol{x}}^{e-1}_{n-1}\|^2 \\ \|\hat{\boldsymbol{x}}^{e-1}_J - \hat{\boldsymbol{x}}^{e-1}_{n-1}\|^2 \end{bmatrix} \end{aligned} \tag{2.89}$$

where a_1 is the smallest eigenvalue of G and a_2, a_3, a_4, a_5, a_6, a_7 and a_8 are positive constants.

Proof. See Appendix 2.B.3. □

Note that (2.89) is still not sufficient to prove convergence since we have no bounds for the evolution of the norms of differences (for instance $\|\bar{\boldsymbol{x}}^e_i - \bar{\boldsymbol{x}}^e_0\|$). The following lemma solves this inconvenience.

Lemma 2.11. *There exists a μ_0 such that if the step-size μ satisfies $\mu < \mu_0$ then it holds:*

$$\begin{bmatrix} \|\bar{\boldsymbol{x}}^{e+1}_0\|^2 \\ \|\hat{\boldsymbol{x}}^{e+1}_0\|^2 \\ w^e \end{bmatrix} \preceq \begin{bmatrix} M_1^J + \mu f_1 J M_2 M_4 (I - M_1)^{-1} & \mu J \begin{bmatrix} a_4 + \mu^2 f_2 a_3 \\ \mu a_8 + \mu^3 f_2 a_7 \end{bmatrix} \\ f_1 \begin{bmatrix} 1 & \|\mathcal{H}_u\|^2 \end{bmatrix} M_4 (I - M_1)^{-1} & \mu^2 f_2 \end{bmatrix} \begin{bmatrix} \|\bar{\boldsymbol{x}}^e_0\|^2 \\ \|\hat{\boldsymbol{x}}^e_0\|^2 \\ w^{e-1} \end{bmatrix} \tag{2.90}$$

where f_1 and f_2 are positive scalars and we also defined:

$$w^e = \frac{1}{N}\sum_{n=0}^{J-1}\left(\|\bar{\boldsymbol{x}}_J^e - \bar{\boldsymbol{x}}_j^e\|^2 + \|\mathcal{H}_u\|^2\|\hat{\boldsymbol{x}}_J^e - \hat{\boldsymbol{x}}_j^e\|^2\right) \tag{2.91a}$$

$$M_1 = \begin{bmatrix} 1-\mu K^{-1}a_1 & \mu a_2 \\ \mu^2 a_6 & \sqrt{\lambda_2(\bar{L})}+\mu^2 a_5 \end{bmatrix} \tag{2.91b}$$

$$M_2 = \begin{bmatrix} a_3 & \|\mathcal{H}_u\|^2 a_3 \\ \mu a_7 & \mu\|\mathcal{H}_u\|^2 a_7 \end{bmatrix} \tag{2.91c}$$

$$M_4 = \begin{bmatrix} 3\mu^2 K^{-2}\|\Lambda_G\|^2 & 3\mu^2 K^{-1}\|Z^{-1}\mathcal{I}^T\mathcal{G}\mathcal{H}_u\|^2 \\ \mu^2 a_6 & 1-\sqrt{\lambda_2(\bar{L})}+\mu^2 a_5 \end{bmatrix} \tag{2.91d}$$

Proof. See Appendix 2.B.5. □

Theorem 2.3. *If the step-size μ is small enough then the iterates* $\begin{bmatrix} \|\bar{\boldsymbol{x}}_0^{e+1}\|^2 & \|\hat{\boldsymbol{x}}_0^{e+1}\|^2 & w^e \end{bmatrix}$ *and $\boldsymbol{\omega}_{k,0}^e$ generated by (2.90) converge linearly to* $\begin{bmatrix} 0 & 0 & 0 \end{bmatrix}$*, with a rate upper bounded by* $\max[\rho(I-\mu K^{-1}\Lambda_G)+\mathcal{O}(\mu^2), \sqrt{\lambda_2(\bar{L})}+\mathcal{O}(\mu)]$.

Proof. See Appendix 2.B.7. □

Finally definitions (2.86a), (2.81), (2.82) and the definitions of $\boldsymbol{\zeta}_{k,i}^e$ and $\boldsymbol{\zeta}_i^e$ in (2.79) combined with Theorem 2.3 imply that the iterates $\boldsymbol{\theta}_{k,0}^e$ and $\boldsymbol{\omega}_{k,0}^e$ generated by *Fast Diffusion for Policy Evaluation* converge linearly to the saddle point of problem (2.38) for every agent k; which completes the proof.

2.B.1 Proof of Lemma 2.8.

We initialize $\boldsymbol{\zeta}_0^0$ in (2.80) to the same value used in (2.78) and $\boldsymbol{\mathcal{Y}}_0^0 = 0$, hence for $i = 0$ and $e = 0$ the claim is trivially true. Replacing the initial conditions in (2.78) and (2.80) for $e = 0$ and $i = 1$ we get in both cases the same update, which is:

$$\boldsymbol{\zeta}_1^0 = \bar{\mathcal{L}}\left(\boldsymbol{\zeta}_0^0 - \mu\mathcal{Q}\left(\boldsymbol{\mathcal{G}}\boldsymbol{\zeta}_0^0 - \boldsymbol{t}_0^0\right)\right) \tag{2.92}$$

which proves the claim for $e = 0$ and $i = 1$. Now we prove the claim for $e = 0$ and $i > 1$

$$\begin{aligned}
\boldsymbol{\zeta}_{i+1}^e &= \bar{\mathcal{L}}\left(\boldsymbol{\zeta}_i^e - \mu\mathcal{Q}\left(\boldsymbol{\mathcal{G}}\boldsymbol{\zeta}_i^e - \boldsymbol{t}_i^e\right)\right) - K\mathcal{V}\boldsymbol{\mathcal{Y}}_i^e \\
&= \bar{\mathcal{L}}\left(\boldsymbol{\zeta}_i^e - \mu\mathcal{Q}\left(\boldsymbol{\mathcal{G}}\boldsymbol{\zeta}_i^e - \boldsymbol{t}_i^e\right)\right) - K\mathcal{V}\boldsymbol{\mathcal{Y}}_i^e + \boldsymbol{\zeta}_i^e - \boldsymbol{\zeta}_i^e \\
&= \bar{\mathcal{L}}\left(\boldsymbol{\zeta}_i^e - \boldsymbol{\zeta}_{i-1}^e - \mu\mathcal{Q}\left(\boldsymbol{\mathcal{G}}\boldsymbol{\zeta}_i^e - \boldsymbol{\mathcal{G}}\boldsymbol{\zeta}_{i-1}^e - \boldsymbol{t}_i^e + \boldsymbol{t}_{i-1}^e\right)\right) - K\mathcal{V}(\boldsymbol{\mathcal{Y}}_i^e - \boldsymbol{\mathcal{Y}}_{i-1}^e) + \boldsymbol{\zeta}_i^e \\
&= \bar{\mathcal{L}}\left(2\boldsymbol{\zeta}_i^e - \boldsymbol{\zeta}_{i-1}^e - \mu\mathcal{Q}\left(\boldsymbol{\mathcal{G}}\boldsymbol{\zeta}_i^e - \boldsymbol{\mathcal{G}}\boldsymbol{\zeta}_{i-1}^e - \boldsymbol{t}_i^e + \boldsymbol{t}_{i-1}^e\right)\right) - K\mathcal{V}(\boldsymbol{\mathcal{Y}}_i^e - \boldsymbol{\mathcal{Y}}_{i-1}^e) + (I - \bar{\mathcal{L}})\boldsymbol{\zeta}_i^e \\
&\overset{(a)}{=} \bar{\mathcal{L}}\left(2\boldsymbol{\zeta}_i^e - \boldsymbol{\zeta}_{i-1}^e - \mu\mathcal{Q}\left(\boldsymbol{\mathcal{G}}\boldsymbol{\zeta}_i^e - \boldsymbol{\mathcal{G}}\boldsymbol{\zeta}_{i-1}^e - \boldsymbol{t}_i^e + \boldsymbol{t}_{i-1}^e\right)\right) - K\mathcal{V}(\boldsymbol{\mathcal{Y}}_i^e - \boldsymbol{\mathcal{Y}}_{i-1}^e) + K\mathcal{V}^2\boldsymbol{\zeta}_i^e \\
&\overset{(b)}{=} \bar{\mathcal{L}}\left(2\boldsymbol{\zeta}_i^e - \boldsymbol{\zeta}_{i-1}^e - \mu\mathcal{Q}\left(\boldsymbol{\mathcal{G}}\boldsymbol{\zeta}_i^e - \boldsymbol{\mathcal{G}}\boldsymbol{\zeta}_{i-1}^e - \boldsymbol{t}_i^e + \boldsymbol{t}_{i-1}^e\right)\right) \qquad (2.93)
\end{aligned}$$

where in (a) we used the definition of $\mathcal{V}$ and in (b) we used (2.80b). For $\boldsymbol{\zeta}_1^1$ we have

$$\begin{aligned}
\boldsymbol{\zeta}_1^1 &= \bar{\mathcal{L}}\left(\boldsymbol{\zeta}_0^1 - \mu\mathcal{Q}\left(\boldsymbol{\mathcal{T}}(\boldsymbol{\zeta}_0^1) - p\right)\right) - K\mathcal{V}\boldsymbol{\mathcal{Y}}_0^1 \\
&= \bar{\mathcal{L}}\left(\boldsymbol{\zeta}_0^1 - \mu\mathcal{Q}\left(\boldsymbol{\mathcal{T}}(\boldsymbol{\zeta}_0^1) - p\right)\right) - K\mathcal{V}\boldsymbol{\mathcal{Y}}_0^1 + \boldsymbol{\zeta}_0^1 - \boldsymbol{\zeta}_0^1 \\
&= \bar{\mathcal{L}}\left(\boldsymbol{\zeta}_0^1 - \boldsymbol{\zeta}_{J-1}^0 - \mu\mathcal{Q}\left(\boldsymbol{\mathcal{T}}(\boldsymbol{\zeta}_0^1) - \boldsymbol{\mathcal{G}}_{J-1}^0\boldsymbol{\zeta}_{J-1}^0 - p + \boldsymbol{t}_{J-1}^0\right)\right) - K\mathcal{V}(\boldsymbol{\mathcal{Y}}_0^1 - \boldsymbol{\mathcal{Y}}_{J-1}^0) + \boldsymbol{\zeta}_0^1 \\
&= \bar{\mathcal{L}}\left(2\boldsymbol{\zeta}_0^1 - \boldsymbol{\zeta}_{J-1}^0 - \mu\mathcal{Q}\left(\boldsymbol{\mathcal{T}}(\boldsymbol{\zeta}_0^1) - \boldsymbol{\mathcal{G}}_{J-1}^0\boldsymbol{\zeta}_{J-1}^0 - p + \boldsymbol{t}_{J-1}^0\right)\right) - K\mathcal{V}(\boldsymbol{\mathcal{Y}}_0^1 - \boldsymbol{\mathcal{Y}}_{J-1}^0) + (I - \bar{\mathcal{L}})\boldsymbol{\zeta}_0^1 \\
&= \bar{\mathcal{L}}\left(2\boldsymbol{\zeta}_0^1 - \boldsymbol{\zeta}_{J-1}^0 - \mu\mathcal{Q}\left(\boldsymbol{\mathcal{T}}(\boldsymbol{\zeta}_0^1) - \boldsymbol{\mathcal{G}}_{J-1}^0\boldsymbol{\zeta}_{J-1}^0 - p + \boldsymbol{t}_{J-1}^0\right)\right) \qquad (2.94)
\end{aligned}$$

Finally, for $e \geq 1$ we get

$$\begin{aligned}
\boldsymbol{\zeta}_{i+1}^e &= \bar{\mathcal{L}}\left(\boldsymbol{\zeta}_i^e - \mu\mathcal{Q}\left(\boldsymbol{\mathcal{T}}(\boldsymbol{\zeta}_i^e) - p\right)\right) - K\mathcal{V}\boldsymbol{\mathcal{Y}}_i^e \\
&= \bar{\mathcal{L}}\left(\boldsymbol{\zeta}_i^e - \mu\mathcal{Q}\left(\boldsymbol{\mathcal{T}}(\boldsymbol{\zeta}_i^e) - p\right)\right) - K\mathcal{V}\boldsymbol{\mathcal{Y}}_i^e + \boldsymbol{\zeta}_i^e - \boldsymbol{\zeta}_i^e \\
&= \bar{\mathcal{L}}\left(\boldsymbol{\zeta}_i^e - \boldsymbol{\zeta}_{i-1}^e - \mu\mathcal{Q}\left(\boldsymbol{\mathcal{T}}(\boldsymbol{\zeta}_i^e) - \boldsymbol{\mathcal{T}}(\boldsymbol{\zeta}_{i-1}^e)\right)\right) - K\mathcal{V}(\boldsymbol{\mathcal{Y}}_i^e - \boldsymbol{\mathcal{Y}}_{i-1}^e) + \boldsymbol{\zeta}_i^e \\
&= \bar{\mathcal{L}}\left(2\boldsymbol{\zeta}_i^e - \boldsymbol{\zeta}_{i-1}^e - \mu\mathcal{Q}\left(\boldsymbol{\mathcal{T}}(\boldsymbol{\zeta}_i^e) - \boldsymbol{\mathcal{T}}(\boldsymbol{\zeta}_{i-1}^e)\right)\right) - K\mathcal{V}(\boldsymbol{\mathcal{Y}}_i^e - \boldsymbol{\mathcal{Y}}_{i-1}^e) + (I - \bar{\mathcal{L}})\boldsymbol{\zeta}_i^e \\
&= \bar{\mathcal{L}}\left(2\boldsymbol{\zeta}_i^e - \boldsymbol{\zeta}_{i-1}^e - \mu\mathcal{Q}\left(\boldsymbol{\mathcal{T}}(\boldsymbol{\zeta}_i^e) - \boldsymbol{\mathcal{T}}(\boldsymbol{\zeta}_{i-1}^e)\right)\right) - K\mathcal{V}(\boldsymbol{\mathcal{Y}}_i^e - \boldsymbol{\mathcal{Y}}_{i-1}^e) + K\mathcal{V}^2\boldsymbol{\zeta}_i^e \\
&= \bar{\mathcal{L}}\left(2\boldsymbol{\zeta}_i^e - \boldsymbol{\zeta}_{i-1}^e - \mu\mathcal{Q}\left(\boldsymbol{\mathcal{T}}(\boldsymbol{\zeta}_i^e) - \boldsymbol{\mathcal{T}}(\boldsymbol{\zeta}_{i-1}^e)\right)\right) \qquad (2.95)
\end{aligned}$$

Note that (2.93), (2.94) and (2.95) coincide with (2.78). This completes the proof.

2.B.2 Proof of Lemma 2.9

This proof is a simple extension from the one presented in Appendix F of the main document. Therefore we use the same matrix decomposition defined in the document to get:

$$\mathcal{H}^{-1}\begin{bmatrix}\widetilde{\boldsymbol{\zeta}}^e_{i+1}\\ \widetilde{\boldsymbol{\mathcal{Y}}}^e_{i+1}\end{bmatrix}=\Bigg(\mathcal{D}-\mu\mathcal{H}^{-1}\underbrace{\begin{bmatrix}\bar{\mathcal{L}}\mathcal{G} & 0_{2KM\times 2KM}\\ \mathcal{V}\bar{\mathcal{L}}\mathcal{G} & 0_{2KM\times 2KM}\end{bmatrix}}_{\triangleq\mathcal{T}}\mathcal{H}\Bigg)\mathcal{H}^{-1}\begin{bmatrix}\widetilde{\boldsymbol{\zeta}}^e_{i}\\ \widetilde{\boldsymbol{\mathcal{Y}}}^e_{i}\end{bmatrix}+\mu\mathcal{H}^{-1}\underbrace{\begin{bmatrix}\bar{\mathcal{L}}\\ \mathcal{V}\bar{\mathcal{L}}\end{bmatrix}}_{\triangleq\mathcal{P}}\boldsymbol{s}(\boldsymbol{\zeta}^e_i) \tag{2.96}$$

We now define the new coordinates:

$$\mathcal{H}^{-1}\begin{bmatrix}\widetilde{\boldsymbol{\zeta}}^e_{i+1}\\ \widetilde{\boldsymbol{\mathcal{Y}}}^e_{i+1}\end{bmatrix}=\begin{bmatrix}\bar{\boldsymbol{x}}^e_{i+1}\triangleq K^{-1/2}\mathcal{I}^T\widetilde{\boldsymbol{\zeta}}^e_{i+1}\\ \tilde{\boldsymbol{x}}^e_{i+1}\triangleq K^{-1/2}\mathcal{I}^T\widetilde{\boldsymbol{\mathcal{Y}}}^e_{i+1}\\ \hat{\boldsymbol{x}}^e_{i+1}\triangleq \mathcal{H}_l^T\widetilde{\boldsymbol{\zeta}}^e_{i+1}+\mathcal{H}_r^T\widetilde{\boldsymbol{\mathcal{Y}}}^e_{i+1}\end{bmatrix} \tag{2.97}$$

Expanding $\mathcal{H}^{-1}\mathcal{P}$ and following exactly the steps from Appendix F in the main document we get the desired relation.

$$\begin{aligned}\begin{bmatrix}\bar{\boldsymbol{x}}^e_{i+1}\\ \hat{\boldsymbol{x}}^e_{i+1}\end{bmatrix}=&\begin{bmatrix}I_{2M}-\mu GK^{-1} & -\mu\mathcal{I}^T\mathcal{G}\mathcal{H}_uK^{-1/2}\\ -\mu K^{-1/2}(\mathcal{H}_l^T+\mathcal{H}_r^T\mathcal{V})\bar{\mathcal{L}}\mathcal{G}\mathcal{I} & \mathcal{D}_1-\mu(\mathcal{H}_l^T+\mathcal{H}_r^T\mathcal{V})\bar{\mathcal{L}}\mathcal{G}\mathcal{H}_u\end{bmatrix}\begin{bmatrix}\bar{\boldsymbol{x}}^e_{i}\\ \hat{\boldsymbol{x}}^e_{i}\end{bmatrix}\\ &+\mu\begin{bmatrix}\mathcal{I}^TK^{-1/2}\\ (\mathcal{H}_l^T+\mathcal{H}_r^T\mathcal{V})\bar{\mathcal{L}}\end{bmatrix}\boldsymbol{s}(\boldsymbol{\zeta}^e_i)\end{aligned} \tag{2.98}$$

2.B.3 Proof of Lemma 2.10

We start by stating the following useful Lemma.

Lemma 2.12. *The following bounds hold:*

$$\begin{aligned}\left\|\mathcal{I}^T \boldsymbol{s}(\boldsymbol{\zeta}_i^e)\right\|^2 \leq\ & 2K \max_{k,n}\left(\|\widetilde{G}_{k,n}\|^2\right)\left\|\bar{\boldsymbol{x}}_0^e - \bar{\boldsymbol{x}}_i^e\right\|^2 + 2K \max_{k,n}\left(\left\|G_{k,n}\right\|^2\right) J^{-1}\sum_{n=1}^{J}\left\|\bar{\boldsymbol{x}}_J^{e-1} - \bar{\boldsymbol{x}}_{n-1}^{e-1}\right\|^2 \\ & + 2K\|\mathcal{H}_u\|^2 \max_{k,n}\left(\|\widetilde{G}_{k,n}\|^2\right)\|\hat{\boldsymbol{x}}_0^e - \hat{\boldsymbol{x}}_i^e\|^2 \\ & + 2K\|\mathcal{H}_u\|^2 J^{-1}\max_{k,n}\left(\left\|G_{k,n}\right\|^2\right)\sum_{n=1}^{J}\left\|\hat{\boldsymbol{x}}_J^{e-1} - \hat{\boldsymbol{x}}_{n-1}^{e-1}\right\|^2 \end{aligned} \tag{2.99}$$

$$\begin{aligned}\left\|(\mathcal{H}_l^T + \mathcal{H}_r^T\mathcal{V})\bar{\mathcal{L}}\boldsymbol{s}(\boldsymbol{\zeta}_i^e)\right\|^2 \leq\ & 2\left\|(\mathcal{H}_l^T + \mathcal{H}_r^T\mathcal{V})\bar{\mathcal{L}}\right\|^2 \max_{k,n}\left(\left\|\widetilde{G}_{k,n}\right\|^2\right) \\ & \cdot\left(\left\|\bar{\boldsymbol{x}}_0^e - \bar{\boldsymbol{x}}_i^e\right\|^2 + \|\mathcal{H}_u\|^2\|\hat{\boldsymbol{x}}_0^e - \hat{\boldsymbol{x}}_i^e\|^2\right) \\ & + 2\left\|(\mathcal{H}_l^T + \mathcal{H}_r^T\mathcal{V})\bar{\mathcal{L}}\right\|^2 \max_{k,n}\left(\left\|G_{k,n}\right\|^2\right)\frac{1}{J}\sum_{n=1}^{J}\left(\left\|\bar{\boldsymbol{x}}_0^e - \bar{\boldsymbol{x}}_i^e\right\|^2 + \|\mathcal{H}_u\|^2\|\hat{\boldsymbol{x}}_{n-1}^{e-1} - \hat{\boldsymbol{x}}_J^{e-1}\|^2\right)\end{aligned} \tag{2.100}$$

where $\widetilde{G}_{k,n} = \big(G - G_{k,n}\big)$.

Proof. See Appendix 2.B.4. □

Now we expand the top recursion of (2.85) and take the squared norm on both sides of the equation to get:

$$\|\check{\boldsymbol{x}}_{i+1}^e\|^2 = \left\|\left(I_{2M} - \mu\Lambda_G K^{-1}\right)\check{\boldsymbol{x}}_i^e - \mu Z^{-1}\mathcal{I}^T\mathcal{G}\mathcal{H}_u K^{-1/2}\hat{\boldsymbol{x}}_i^e + \mu Z^{-1}\mathcal{I}^T K^{-\frac{1}{2}}\boldsymbol{s}(\boldsymbol{\zeta}_i^e)\right\|^2 \tag{2.101}$$

$$\begin{aligned}\|\check{\boldsymbol{x}}_{i+1}^e\|^2 = & \left\|\frac{t}{t}\left(I_{2M} - \mu\Lambda_G K^{-1}\right)\check{\boldsymbol{x}}_i^e + \frac{(1-t)/2}{(1-t)/2}\big(-\mu Z^{-1}\mathcal{I}^T\mathcal{G}\mathcal{H}_u K^{-\frac{1}{2}}\hat{\boldsymbol{x}}_i^e\big)\right. \\ & \left. + \frac{(1-t)/2}{(1-t)/2}\mu Z^{-1}\mathcal{I}^T K^{-\frac{1}{2}}\boldsymbol{s}(\boldsymbol{\zeta}_i^e)\right\|^2\end{aligned} \tag{2.102}$$

$$\begin{aligned}\|\check{\boldsymbol{x}}_{i+1}^e\|^2 \overset{(a)}{\leq}\ & t^{-1}\left\|\big(I_{2M} - \mu\Lambda_G K^{-1}\big)\check{\boldsymbol{x}}_i^e\right\|^2 + 2(1-t)^{-1}\mu^2 K^{-1}\left\|Z^{-1}\mathcal{I}^T\mathcal{G}\mathcal{H}_u\hat{\boldsymbol{x}}_i^e\right\|^2 \\ & + 2(1-t)^{-1}\mu^2 K^{-1}\left\|Z^{-1}\mathcal{I}^T\boldsymbol{s}(\boldsymbol{\zeta}_i^e)\right\|^2\end{aligned} \tag{2.103}$$

$$\begin{aligned}\|\check{\boldsymbol{x}}_{i+1}^e\|^2 \overset{(b)}{\leq}\ & t^{-1}\left\|I_{2M} - \mu\Lambda_G K^{-1}\right\|^2\|\check{\boldsymbol{x}}_i^e\|^2 \\ & + 2(1-t)^{-1}\mu^2 K^{-1}\left\|Z^{-1}\mathcal{I}^T\mathcal{G}\mathcal{H}_u\hat{\boldsymbol{x}}_i^e\right\|^2 + 2(1-t)^{-1}\mu^2 K^{-1}\left\|Z^{-1}\mathcal{I}^T\boldsymbol{s}(\boldsymbol{\zeta}_i^e)\right\|^2\end{aligned} \tag{2.104}$$

where $t \in (0,1)$. In (a) we used Jensen's inequality and in (b) we used Schwarz inequality.

If $\mu < K/\rho(\Lambda_G)$ then we can choose $t = \|I_{2M} - \mu\Lambda_G K^{-1}\| = 1 - \mu a_1 K^{-1} < 1$, where a_1 is the minimum eigenvalue of G:

$$a_1 = \Lambda_{G,\min} \tag{2.105}$$

Hence, we can write:

$$\begin{aligned}
\|\check{\boldsymbol{x}}_{i+1}^e\|^2 &\le (1-\mu a_1 K^{-1})\|\check{\boldsymbol{x}}_i^e\|^2 + 4\mu a_1^{-1}\big\|Z^{-1}\mathcal{I}^T\mathcal{G}\mathcal{H}_u\hat{\boldsymbol{x}}_i^e\big\|^2 + \mu 4a_1^{-1}\big\|Z^{-1}\mathcal{I}^T\boldsymbol{s}(\boldsymbol{\zeta}_i^e)\big\|^2 \qquad (2.106)\\
&\le (1-\mu a_1 K^{-1})\|\check{\boldsymbol{x}}_i^e\|^2 + \mu\underbrace{4a_1^{-1}\|Z^{-1}\mathcal{I}^T\mathcal{G}\mathcal{H}_u\|^2}_{\triangleq a_2}\|\hat{\boldsymbol{x}}_i^e\|^2 + \mu 4a_1^{-1}\|Z^{-1}\|^2\big\|\mathcal{I}^T\boldsymbol{s}(\boldsymbol{\zeta}_i^e)\big\|^2\\
&\overset{(c)}{\le} (1-\mu a_1 K^{-1})\|\check{\boldsymbol{x}}_i^e\|^2 + \mu a_2\|\hat{\boldsymbol{x}}_i^e\|^2 + \mu\underbrace{8Ka_1^{-1}\|Z^{-1}\|^2\max_{k,n}\big(\|\widetilde{G}_{k,n}\|^2\big)}_{\triangleq a_3}\big\|\check{\boldsymbol{x}}_0^e - \check{\boldsymbol{x}}_i^e\big\|^2\\
&+ \mu\underbrace{8Ka_1^{-1}\|Z^{-1}\|^2\max_{k,n}\big(\big\|G_{k,n}\big\|^2\big)}_{\triangleq a_4}J^{-1}\sum_{n=1}^{J}\big\|\check{\boldsymbol{x}}_J^{e-1} - \check{\boldsymbol{x}}_{n-1}^{e-1}\big\|^2\\
&+ \mu 8Ka_1^{-1}\|Z^{-1}\|^2\|\mathcal{H}_u\|^2\max_{k,n}\big(\|\widetilde{G}_{k,n}\|^2\big)\|\hat{\boldsymbol{x}}_0^e - \hat{\boldsymbol{x}}_i^e\|^2\\
&+ \mu 8Ka_1^{-1}\|Z^{-1}\|^2\|\mathcal{H}_u\|^2\max_{k,n}\big(\big\|G_{k,n}\big\|^2\big)J^{-1}\sum_{n=1}^{J}\big\|\hat{\boldsymbol{x}}_J^{e-1} - \hat{\boldsymbol{x}}_{n-1}^{e-1}\big\|^2\\
&= (1-\mu a_1 K^{-1})\|\check{\boldsymbol{x}}_i^e\|^2 + \mu a_2\|\hat{\boldsymbol{x}}_i^e\|^2 + \mu a_3\Big(\big\|\check{\boldsymbol{x}}_0^e - \check{\boldsymbol{x}}_i^e\big\|^2 + \|\mathcal{H}_u\|^2\|\hat{\boldsymbol{x}}_0^e - \hat{\boldsymbol{x}}_i^e\|^2\Big)\\
&+ \mu a_4 J^{-1}\sum_{n=1}^{J}\Big(\big\|\check{\boldsymbol{x}}_J^{e-1} - \check{\boldsymbol{x}}_{n-1}^{e-1}\big\|^2 + \|\mathcal{H}_u\|^2\big\|\hat{\boldsymbol{x}}_J^{e-1} - \hat{\boldsymbol{x}}_{n-1}^{e-1}\big\|^2\Big) \qquad (2.107)
\end{aligned}$$

where in (c) we used (2.99). Note that the inequality obtained constitutes the top inequality of (2.89). We now repeat the procedure for the bottom recursion of (2.89):

$$\begin{aligned}
\|\hat{\boldsymbol{x}}_{i+1}^e\|^2 &= \big\| - \mu K^{-\frac{1}{2}}(\mathcal{H}_l^T + \mathcal{H}_r^T\mathcal{V})\bar{\mathcal{L}}\mathcal{G}\mathcal{I}Z\check{\boldsymbol{x}}_i^e + \big(\mathcal{D}_1 - \mu(\mathcal{H}_l^T + \mathcal{H}_r^T\mathcal{V})\bar{\mathcal{L}}\mathcal{G}\mathcal{H}_u\big)\hat{\boldsymbol{x}}_i^e\\
&\quad + \mu(\mathcal{H}_l^T + \mathcal{H}_r^T\mathcal{V})\bar{\mathcal{L}}\boldsymbol{s}(\boldsymbol{\zeta}_i^e)\big\|^2\\
&= \Big\| - \frac{(1-t)/3}{(1-t)/3}\mu K^{-\frac{1}{2}}(\mathcal{H}_l^T + \mathcal{H}_r^T\mathcal{V})\bar{\mathcal{L}}\mathcal{G}\mathcal{I}Z\check{\boldsymbol{x}}_i^e + \frac{t}{t}\mathcal{D}_1\hat{\boldsymbol{x}}_i^e\\
&\quad - \frac{(1-t)/3}{(1-t)/3}\big(\mu(\mathcal{H}_l^T + \mathcal{H}_r^T\mathcal{V})\bar{\mathcal{L}}\mathcal{G}\mathcal{H}_u\big)\hat{\boldsymbol{x}}_i^e
\end{aligned}$$

$$
\begin{aligned}
&+\frac{(1-t)/3}{(1-t)/3}\mu(\mathcal{H}_l^T+\mathcal{H}_r^T\mathcal{V})\bar{\mathcal{L}}\boldsymbol{s}(\boldsymbol{\zeta}_i^e)\Big\|^2\\
&\le 3(1-t)^{-1}\|\mu K^{-\frac{1}{2}}(\mathcal{H}_l^T+\mathcal{H}_r^T\mathcal{V})\bar{\mathcal{L}}\mathcal{G}\mathcal{I}Z\check{\boldsymbol{x}}_i^e\|^2+t^{-1}\|\mathcal{D}_1\hat{\boldsymbol{x}}_i^e\|^2\\
&+3(1-t)^{-1}\|\mu(\mathcal{H}_l^T+\mathcal{H}_r^T\mathcal{V})\bar{\mathcal{L}}\mathcal{G}\mathcal{H}_u\hat{\boldsymbol{x}}_i^e\|^2+3(1-t)^{-1}\|\mu(\mathcal{H}_l^T+\mathcal{H}_r^T\mathcal{V})\bar{\mathcal{L}}\boldsymbol{s}(\boldsymbol{\zeta}_i^e)\|^2\\
&\le 3(1-t)^{-1}\mu^2K^{-1}\|(\mathcal{H}_l^T+\mathcal{H}_r^T\mathcal{V})\bar{\mathcal{L}}\mathcal{G}\mathcal{I}Z\|^2\|\check{\boldsymbol{x}}_i^e\|^2+t^{-1}\|\mathcal{D}_1\|^2\|\hat{\boldsymbol{x}}_i^e\|^2\\
&+3(1-t)^{-1}\mu^2\|(\mathcal{H}_l^T+\mathcal{H}_r^T\mathcal{V})\bar{\mathcal{L}}\mathcal{G}\mathcal{H}_u\|^2\|\hat{\boldsymbol{x}}_i^e\|^2+3(1-t)^{-1}\mu^2\|(\mathcal{H}_l^T+\mathcal{H}_r^T\mathcal{V})\bar{\mathcal{L}}\boldsymbol{s}(\boldsymbol{\zeta}_i^e)\|^2\\
&\overset{(f)}{\le}\sqrt{\lambda_2(\bar{L})}\|\hat{\boldsymbol{x}}_i^e\|^2+\mu^2\underbrace{\frac{3\|(\mathcal{H}_l^T+\mathcal{H}_r^T\mathcal{V})\bar{\mathcal{L}}\mathcal{G}\mathcal{H}_u\|^2}{\left(1-\sqrt{\lambda_2(\bar{L})}\right)}}_{\triangleq a_5}\|\hat{\boldsymbol{x}}_i^e\|^2\\
&+\mu^2\underbrace{\frac{3K^{-1}\|(\mathcal{H}_l^T+\mathcal{H}_r^T\mathcal{V})\bar{\mathcal{L}}\mathcal{G}\mathcal{I}Z\|^2}{1-\sqrt{\lambda_2(\bar{L})}}}_{\triangleq a_6}\|\check{\boldsymbol{x}}_i^e\|^2\\
&+\mu^2\underbrace{\frac{6\left\|(\mathcal{H}_l^T+\mathcal{H}_r^T\mathcal{V})\bar{\mathcal{L}}\right\|^2\max_{k,n}\left(\left\|\widetilde{G}_{k,n}\right\|^2\right)}{\left(1-\sqrt{\lambda_2(\bar{L})}\right)}}_{\triangleq a_7}\left(\left\|\check{\boldsymbol{x}}_0^e-\check{\boldsymbol{x}}_i^e\right\|^2+\|\mathcal{H}_u\|^2\|(\hat{\boldsymbol{x}}_0^e-\hat{\boldsymbol{x}}_i^e)\|^2\right)\\
&+\mu^2\underbrace{\frac{6\left\|(\mathcal{H}_l^T+\mathcal{H}_r^T\mathcal{V})\bar{\mathcal{L}}\right\|^2\max_{k,n}\left(\left\|G_{k,n}\right\|^2\right)}{\left(1-\sqrt{\lambda_2(\bar{L})}\right)}}_{\triangleq a_8}J^{-1}\\
&\cdot\sum_{n=1}^{J}\left(\left\|\check{\boldsymbol{x}}_0^e-\check{\boldsymbol{x}}_i^e\right\|^2+\|\mathcal{H}_u\|^2\|(\hat{\boldsymbol{x}}_{n-1}^{e-1}-\hat{\boldsymbol{x}}_J^{e-1})\|^2\right) \qquad (2.108)
\end{aligned}
$$

where in (f) we chose $t=\|\mathcal{D}_1\|=\sqrt{\lambda_2(\bar{L})}$ and used (2.100). Noting that the inequality obtained constitutes the bottom inequality of (2.89) completes the proof.

2.B.4 Proof of Lemma 2.12

Now we proceed to bound the gradient noise related terms.

$$\|\mathcal{I}^T \boldsymbol{s}(\boldsymbol{\zeta}_i^e)\|^2 = \left\|\mathcal{I}^T \left(\boldsymbol{\mathcal{T}}(\zeta_i^e) - \mathcal{G}\boldsymbol{\zeta}_i^e\right)\right\|^2 = \left\|\sum_{k=1}^K \boldsymbol{T}(\zeta_{k,i}^e) - G_k\zeta_{k,i}^e\right\|^2 \overset{(a)}{\leq} \sum_{k=1}^K K\|\widehat{\boldsymbol{T}}(\zeta_{k,i}^e) - G_k\zeta_{k,i}^e\|^2 \tag{2.109}$$

where in (a) we used Jensen's inequality. Next we analyze the individual terms $\left\|\boldsymbol{T}(\zeta_{k,i}^e) - G_k\zeta_{k,i}^e\right\|^2$:

$$\begin{aligned}
\left\|\boldsymbol{T}(\boldsymbol{\zeta}_{k,i}^e) - G_k\boldsymbol{\zeta}_{k,i}^e\right\|^2 &= \left\|\boldsymbol{G}_{k,i}^e(\boldsymbol{\zeta}_{k,i}^e - \boldsymbol{\zeta}_{k,0}^e) + \frac{1}{J}\sum_{n=1}^J \boldsymbol{G}_{k,n}^{e-1}\boldsymbol{\zeta}_{k,n-1}^{e-1} - G_k\boldsymbol{\zeta}_{k,i}^e\right\|^2 \\
&\overset{(b)}{=} \left\|\boldsymbol{G}_{k,i}^e(\boldsymbol{\zeta}_{k,i}^e - \boldsymbol{\zeta}_{k,0}^e) + G_k\left(\boldsymbol{\zeta}_{k,0}^e - \boldsymbol{\zeta}_{k,\hat{N}}^{e-1}\right) + \frac{1}{J}\sum_{n=1}^J \boldsymbol{G}_{k,n}^{e-1}\boldsymbol{\zeta}_{k,n-1}^{e-1} - G_k\boldsymbol{\zeta}_{k,i}^e\right\|^2 \\
&= \left\|\boldsymbol{G}_{k,i}^e(\boldsymbol{\zeta}_{k,i}^e - \boldsymbol{\zeta}_{k,0}^e) + G_k\left(\boldsymbol{\zeta}_{k,0}^e - \boldsymbol{\zeta}_{k,i}^e\right) + \frac{1}{J}\sum_{n=1}^J \boldsymbol{G}_{k,n}^{e-1}\left(\boldsymbol{\zeta}_{k,n-1}^{e-1} - \boldsymbol{\zeta}_{k,J}^{e-1}\right)\right\|^2 \\
&\overset{(c)}{=} \left\|\left(G_k - \boldsymbol{G}_{k,i}^e\right)\left(\boldsymbol{\zeta}_{k,0}^e - \boldsymbol{\zeta}_{k,i}^e\right) + \frac{1}{J}\sum_{n=1}^J \boldsymbol{G}_{k,n}^{e-1}\left(\boldsymbol{\zeta}_{k,n-1}^{e-1} - \boldsymbol{\zeta}_{k,J}^{e-1}\right)\right\|^2 \\
&\overset{(d)}{\leq} 2\left\|\widetilde{\boldsymbol{G}}_{k,i}^e(\boldsymbol{\zeta}_{k,0}^e - \boldsymbol{\zeta}_{k,i}^e)\right\|^2 + \frac{2}{J}\sum_{n=1}^J \left\|\boldsymbol{G}_{k,n}^{e-1}\left(\boldsymbol{\zeta}_{k,n-1}^{e-1} - \boldsymbol{\zeta}_{k,J}^{e-1}\right)\right\|^2 \\
&\overset{(e)}{\leq} 2\left\|\widetilde{\boldsymbol{G}}_{k,i}^e\right\|^2\left\|\boldsymbol{\zeta}_{k,0}^e - \boldsymbol{\zeta}_{k,i}^e\right\|^2 + \frac{2}{J}\sum_{n=1}^J \left\|\boldsymbol{G}_{k,n}^{e-1}\right\|^2\left\|\boldsymbol{\zeta}_{k,n-1}^{e-1} - \boldsymbol{\zeta}_{k,J}^{e-1}\right\|^2 \\
&\leq 2\max_n\left(\left\|\widetilde{G}_{k,n}\right\|^2\right)\left\|\boldsymbol{\zeta}_{k,0}^e - \boldsymbol{\zeta}_{k,i}^e\right\|^2 + \max_n\left(\left\|G_{k,n}\right\|^2\right)\frac{2}{J}\sum_{n=1}^J \left\|\boldsymbol{\zeta}_{k,n-1}^{e-1} - \boldsymbol{\zeta}_{k,J}^{e-1}\right\|^2
\end{aligned} \tag{2.110}$$

where in (b) we used the fact that $\boldsymbol{\zeta}_{k,0}^e = \boldsymbol{\zeta}_{k,\hat{N}}^{e-1}$, in (c) we defined $G_k - \boldsymbol{G}_{k,i}^e = \widetilde{\boldsymbol{G}}_{k,i}^e$, in (d) we used Jensen's inequality and in (e) we used Schwarz inequality. Combining (2.109) and

(2.110) we get:

$$\begin{aligned}\|\mathcal{I}^T \boldsymbol{s}(\boldsymbol{\zeta}_i^e)\|^2 \leq \sum_{k=1}^{K} K\Bigg(& 2\max_n \left(\left\|\widetilde{G}_{k,n}\right\|^2\right) \left\|\boldsymbol{\zeta}_{k,0}^e - \boldsymbol{\zeta}_{k,i}^e\right\|^2 \\ &+ \max_n \left(\left\|G_{k,n}\right\|^2\right) \frac{2}{J} \sum_{n=1}^{J} \left\|\boldsymbol{\zeta}_{k,n-1}^{e-1} - \boldsymbol{\zeta}_{k,J}^{e-1}\right\|^2\Bigg) \\ &\overset{(a)}{\leq} 2K \max_{k,n} \left(\|\widetilde{G}_{k,n}\|^2\right) \left\|\boldsymbol{\zeta}_0^e - \boldsymbol{\zeta}_i^e\right\|^2 + \frac{2K}{J} \max_{k,n} \left(\left\|G_{k,n}\right\|^2\right) \sum_{n=1}^{J} \left\|\boldsymbol{\zeta}_{n-1}^{e-1} - \boldsymbol{\zeta}_J^{e-1}\right\|^2 \\ &= 2K \max_{k,n} \left(\|\widetilde{G}_{k,n}\|^2\right) \left\|\widetilde{\boldsymbol{\zeta}}_i^e - \widetilde{\boldsymbol{\zeta}}_0^e\right\|^2 + \frac{2K}{J} \max_{k,n} \left(\left\|G_{k,n}\right\|^2\right) \sum_{n=1}^{J} \left\|\widetilde{\boldsymbol{\zeta}}_J^{e-1} - \widetilde{\boldsymbol{\zeta}}_{n-1}^{e-1}\right\|^2\end{aligned} \tag{2.111}$$

where in (a) we used Hölder's inequality. Now we need an equation relating $\left\|\widetilde{\boldsymbol{\zeta}}_i^e - \widetilde{\boldsymbol{\zeta}}_0^e\right\|$ with $\left\|\bar{\boldsymbol{x}}_0^e - \bar{\boldsymbol{x}}_i^e\right\|$ and $\|\hat{\boldsymbol{x}}_0^e - \hat{\boldsymbol{x}}_i^e\|$ which we get as follows:

$$\begin{aligned}\left\|\widetilde{\boldsymbol{\zeta}}_i^e - \widetilde{\boldsymbol{\zeta}}_0^e\right\|^2 &\overset{(a)}{=} \left\|K^{-1/2}\mathcal{I}(\bar{\boldsymbol{x}}_0^e - \bar{\boldsymbol{x}}_i^e) + \mathcal{H}_u(\hat{\boldsymbol{x}}_0^e - \hat{\boldsymbol{x}}_i^e)\right\|^2 \overset{(b)}{=} K^{-1}\left\|\mathcal{I}(\bar{\boldsymbol{x}}_0^e - \bar{\boldsymbol{x}}_i^e)\right\|^2 + \left\|\mathcal{H}_u(\hat{\boldsymbol{x}}_0^e - \hat{\boldsymbol{x}}_i^e)\right\|^2 \\ &\overset{(c)}{\leq} \left\|\bar{\boldsymbol{x}}_0^e - \bar{\boldsymbol{x}}_i^e\right\|^2 + \|\mathcal{H}_u\|^2 \|(\hat{\boldsymbol{x}}_0^e - \hat{\boldsymbol{x}}_i^e)\|^2\end{aligned} \tag{2.112}$$

where in (a) we used (2.97), in (b) we used $\mathcal{I}^T\mathcal{H}_u = 0$, in (c) we used the fact that the maximum eigenvalue of $\mathcal{I}^T\mathcal{I}$ is K. Combining (2.111) and (2.112) we finally get:

$$\begin{aligned}\|\mathcal{I}^T \boldsymbol{s}(\boldsymbol{\zeta}_i^e)\|^2 \leq\ & 2K \max_{k,n} \left(\|\widetilde{G}_{k,n}\|^2\right) \left\|\bar{\boldsymbol{x}}_0^e - \bar{\boldsymbol{x}}_i^e\right\|^2 + 2K \max_{k,n} \left(\left\|G_{k,n}\right\|^2\right) J^{-1} \sum_{n=1}^{J} \left\|\bar{\boldsymbol{x}}_J^{e-1} - \bar{\boldsymbol{x}}_{n-1}^{e-1}\right\|^2 \\ &+ 2K\|\mathcal{H}_u\|^2 \max_{k,n} \left(\|\widetilde{G}_{k,n}\|^2\right) \|\hat{\boldsymbol{x}}_0^e - \hat{\boldsymbol{x}}_i^e\|^2 \\ &+ 2K\|\mathcal{H}_u\|^2 J^{-1} \max_{k,n} \left(\left\|G_{k,n}\right\|^2\right) \sum_{n=1}^{J} \left\|\hat{\boldsymbol{x}}_J^{e-1} - \hat{\boldsymbol{x}}_{n-1}^{e-1}\right\|^2\end{aligned} \tag{2.113}$$

which is (2.99). We now proceed to bound the other noise term.

$$
\begin{aligned}
&\|(\mathcal{H}_l^T + \mathcal{H}_r^T\mathcal{V})\bar{\mathcal{L}}\boldsymbol{s}(\boldsymbol{\zeta}_i^e)\|^2 = \left\|(\mathcal{H}_l^T + \mathcal{H}_r^T\mathcal{V})\bar{\mathcal{L}}\left(\mathcal{T}(\boldsymbol{\zeta}_i^e) - \mathcal{G}\boldsymbol{\zeta}_i^e\right)\right\|^2 \\
&\leq \left\|(\mathcal{H}_l^T + \mathcal{H}_r^T\mathcal{V})\bar{\mathcal{L}}\right\|^2 \|\mathcal{T}(\boldsymbol{\zeta}_i^e) - \mathcal{G}\boldsymbol{\zeta}_i^e\|^2 \\
&= \left\|(\mathcal{H}_l^T + \mathcal{H}_r^T\mathcal{V})\bar{\mathcal{L}}\right\|^2 \left\| \begin{bmatrix} \boldsymbol{T}(\zeta_{1,i}^e) - G_1\zeta_{1,i}^e \\ \vdots \\ \boldsymbol{T}(\zeta_{K,i}^e) - G_K\zeta_{K,i}^e \end{bmatrix} \right\|^2 \\
&= \left\|(\mathcal{H}_l^T + \mathcal{H}_r^T\mathcal{V})\bar{\mathcal{L}}\right\|^2 \sum_{k=1}^{K} \left\|\boldsymbol{T}(\zeta_{k,i}^e) - G_k\zeta_{k,i}^e\right\|^2 \\
&\overset{(a)}{\leq} 2\left\|(\mathcal{H}_l^T + \mathcal{H}_r^T\mathcal{V})\bar{\mathcal{L}}\right\|^2 \sum_{k=1}^{K} \Bigg(\max_n \left(\left\|\widetilde{G}_{k,n}\right\|^2\right) \left\|\boldsymbol{\zeta}_{k,0}^e - \boldsymbol{\zeta}_{k,i}^e\right\|^2 \\
&\qquad\qquad + \max_n \left(\left\|G_{k,n}\right\|^2\right) \frac{1}{J} \sum_{n=1}^{J} \left\|\boldsymbol{\zeta}_{k,n-1}^{e-1} - \boldsymbol{\zeta}_{k,J}^{e-1}\right\|^2 \Bigg) \\
&\overset{(b)}{\leq} 2\left\|(\mathcal{H}_l^T + \mathcal{H}_r^T\mathcal{V})\bar{\mathcal{L}}\right\|^2 \left(\max_{k,n} \left(\left\|\widetilde{G}_{k,n}\right\|^2\right) \left\|\boldsymbol{\zeta}_0^e - \boldsymbol{\zeta}_i^e\right\|^2 + \max_{k,n} \left(\left\|G_{k,n}\right\|^2\right) \frac{1}{J} \sum_{n=1}^{J} \left\|\boldsymbol{\zeta}_{n-1}^{e-1} - \boldsymbol{\zeta}_J^{e-1}\right\|^2 \right) \\
&\overset{(c)}{\leq} 2\left\|(\mathcal{H}_l^T + \mathcal{H}_r^T\mathcal{V})\bar{\mathcal{L}}\right\|^2 \Bigg[\max_{k,n} \left(\left\|\widetilde{G}_{k,n}\right\|^2\right) \left(\left\|\bar{\boldsymbol{x}}_0^e - \bar{\boldsymbol{x}}_i^e\right\|^2 + \|\mathcal{H}_u\|^2 \|(\hat{\boldsymbol{x}}_0^e - \hat{\boldsymbol{x}}_i^e)\|^2 \right) \\
&+ \max_{k,n} \left(\left\|G_{k,n}\right\|^2\right) \frac{1}{J} \sum_{n=1}^{J} \left(\left\|\bar{\boldsymbol{x}}_0^e - \bar{\boldsymbol{x}}_i^e\right\|^2 + \|\mathcal{H}_u\|^2 \|(\hat{\boldsymbol{x}}_{n-1}^{e-1} - \hat{\boldsymbol{x}}_J^{e-1})\|^2 \right) \Bigg] \qquad (2.114)
\end{aligned}
$$

where in (a) we used (2.110), in (b) we used Hölder's inequality and in (c) we used (2.112). Note that the bound obtained is (2.100).

2.B.5 Proof Lemma 2.11

We start by introducing the definitions

$$y_j^e \triangleq \begin{bmatrix} \|\bar{\boldsymbol{x}}_j^e\|^2 \\ \|\hat{\boldsymbol{x}}_j^e\|^2 \end{bmatrix} \tag{2.115a}$$

$$z_j^e \triangleq \begin{bmatrix} \|\bar{\boldsymbol{x}}_j^e - \bar{\boldsymbol{x}}_0^e\|^2 \\ \|\hat{\boldsymbol{x}}_j^e - \hat{\boldsymbol{x}}_0^e\|^2 \end{bmatrix} \tag{2.115b}$$

$$w_j^e \triangleq \begin{bmatrix} \|\bar{\boldsymbol{x}}_J^e - \bar{\boldsymbol{x}}_j^e\|^2 \\ \|\hat{\boldsymbol{x}}_J^e - \hat{\boldsymbol{x}}_j^e\|^2 \end{bmatrix} \tag{2.115c}$$

$$M_1 \triangleq \begin{bmatrix} 1-\mu a_1 K^{-1} & \mu a_2 \\ \mu^2 a_6 & \sqrt{\lambda_2(\bar{L})} + \mu^2 a_5 \end{bmatrix} \tag{2.115d}$$

$$M_2 \triangleq \begin{bmatrix} a_3 & \|\mathcal{H}_u\|^2 a_3 \\ \mu a_7 & \mu\|\mathcal{H}_u\|^2 a_7 \end{bmatrix} = \begin{bmatrix} a_3 \\ \mu a_7 \end{bmatrix} \begin{bmatrix} 1 \\ \|\mathcal{H}_u\|^2 \end{bmatrix}^T \tag{2.115e}$$

$$M_3 \triangleq \begin{bmatrix} a_4 & \|\mathcal{H}_u\|^2 a_4 \\ \mu a_8 & \mu\|\mathcal{H}_u\|^2 a_8 \end{bmatrix} = \begin{bmatrix} a_4 \\ \mu a_8 \end{bmatrix} \begin{bmatrix} 1 \\ \|\mathcal{H}_u\|^2 \end{bmatrix}^T \tag{2.115f}$$

2.B.5.1 Bound for y_j^e

We iterate equation (2.89) to get

$$\begin{aligned} y_j^e &\preceq M_1 \left(M_1 y_{j-2}^e + \mu M_2 z_{j-2}^e + \mu \frac{1}{J} \sum_{n=0}^{J-1} M_3 w_n^{e-1} \right) + \mu M_2 z_{j-1}^e + \mu M_3 \frac{1}{J} \sum_{n=0}^{J-1} w_n^{e-1} \\ &= M_1^2 y_{j-2}^e + \mu \sum_{n=1}^{2} M_1^{n-1} M_2 z_{j-n}^e + \mu \sum_{k=0}^{1} M_1^k M_3 \frac{1}{J} \sum_{n=0}^{J-1} w_n^{e-1} \\ &\preceq M_1^j y_0^e + \mu \sum_{n=1}^{j} M_1^{n-1} M_2 z_{j-n}^e + \mu \sum_{k=0}^{j-1} M_1^k M_3 \frac{1}{J} \sum_{n=0}^{J-1} w_n^{e-1} \end{aligned} \tag{2.116}$$

Now we impose the following conditions on μ:

$$(1-\mu K^{-1}a_1)a_3+\mu^2 a_2 a_7 \leq a_3 \rightarrow \mu \leq \frac{a_1 a_3}{K a_2 a_7} \tag{2.117a}$$

$$\sqrt{\lambda_2(\bar{L})}a_7+\mu a_3 a_6+\mu^2 a_5 a_7 \leq a_7 \tag{2.117b}$$

$$(1-\mu K^{-1}a_1)a_4+\mu^2 a_2 a_8 \leq a_4 \rightarrow \mu \leq \frac{a_1 a_4}{K a_2 a_8} \tag{2.117c}$$

$$\sqrt{\lambda_2(\bar{L})}a_8+\mu a_4 a_6+\mu^2 a_5 a_8 \leq a_8 \tag{2.117d}$$

Notice that since constants a_1 through a_8 are all strictly positive, there's always a step-size μ small enough such that the above conditions are satisfied. These conditions imply that the following entry-wise matrix inequalities hold:

$$\begin{aligned} M_1 M_2 &= \begin{bmatrix} 1-\mu K^{-1}a_1 & \mu a_2 \\ \mu^2 a_6 & \left(\sqrt{\lambda_2(\bar{L})}+\mu^2 a_5\right) \end{bmatrix} \begin{bmatrix} a_3 \\ \mu a_7 \end{bmatrix} \begin{bmatrix} 1 \\ \|\mathcal{H}_u\|^2 \end{bmatrix}^T \\ &= \begin{bmatrix} (1-\mu K^{-1}a_1)a_3+\mu^2 a_2 a_7 \\ \mu^2 a_3 a_6+\left(\sqrt{\lambda_2(\bar{L})}+\mu^2 a_5\right)\mu a_7 \end{bmatrix} \begin{bmatrix} 1 \\ \|\mathcal{H}_u\|^2 \end{bmatrix}^T \preceq \begin{bmatrix} a_3 \\ \mu a_7 \end{bmatrix} \begin{bmatrix} 1 \\ \|\mathcal{H}_u\|^2 \end{bmatrix}^T = M_2 \end{aligned} \tag{2.118}$$

$$\begin{aligned} M_1 M_3 &= \begin{bmatrix} 1-\mu K^{-1}a_1 & \mu a_2 \\ \mu^2 a_6 & \left(\sqrt{\lambda_2(\bar{L})}+\mu^2 a_5\right) \end{bmatrix} \begin{bmatrix} a_4 \\ \mu a_8 \end{bmatrix} \begin{bmatrix} 1 \\ K \end{bmatrix}^T \\ &= \begin{bmatrix} (1-\mu K^{-1}a_1)a_4+\mu^2 a_2 a_8 \\ \mu^2 a_4 a_6+\left(\sqrt{\lambda_2(\bar{L})}+\mu^2 a_5\right)\mu a_8 \end{bmatrix} \begin{bmatrix} 1 \\ \|\mathcal{H}_u\|^2 \end{bmatrix}^T \preceq \begin{bmatrix} a_4 \\ \mu a_8 \end{bmatrix} \begin{bmatrix} 1 \\ \|\mathcal{H}_u\|^2 \end{bmatrix}^T = M_3 \end{aligned} \tag{2.119}$$

and hence we can write:

$$y_j^e \preceq M_1^j y_0^e + \mu M_2 \sum_{n=1}^{j-1} z_n^e + \mu j M_3 \frac{1}{J} \sum_{n=0}^{J-1} w_n^{e-1} \tag{2.120}$$

2.B.5.2 Bound for z_j^e

We can proceed to get a similar inequality for z_j^e. For this we start by using the top equation of (2.88):

$$\check{\boldsymbol{x}}_{i+1}^e - \check{\boldsymbol{x}}_i^e = -\mu\Lambda_G K^{-1}\check{\boldsymbol{x}}_i^e - \mu Z^{-1}\mathcal{I}^T\mathcal{G}\mathcal{H}_u K^{-1/2}\hat{\boldsymbol{x}}_i^e + \mu Z^{-1}\mathcal{I}^T K^{-1/2}\boldsymbol{s}(\boldsymbol{\zeta}_i^e) \tag{2.121}$$

Using the above equation we can write the following expression for $\check{\boldsymbol{x}}_{i+1}^e - \check{\boldsymbol{x}}_0^e$:

$$\check{\boldsymbol{x}}_{i+1}^e - \check{\boldsymbol{x}}_0^e = \sum_{j=0}^{i}\check{\boldsymbol{x}}_{j+1}^e - \check{\boldsymbol{x}}_j^e = \mu\sum_{j=0}^{i} -\Lambda_G K^{-1}\check{\boldsymbol{x}}_j^e - Z^{-1}\mathcal{I}^T\mathcal{G}\mathcal{H}_u K^{-1/2}\hat{\boldsymbol{x}}_j^e + Z^{-1}\mathcal{I}^T K^{-1/2}\boldsymbol{s}(\boldsymbol{\zeta}_j^e) \tag{2.122}$$

Taking squared norm on both sides we get:

$$\begin{aligned}
\|\check{\boldsymbol{x}}_{i+1}^e - \check{\boldsymbol{x}}_0^e\|^2 &= \mu^2\left\|\sum_{j=0}^{i} -\Lambda_G K^{-1}\check{\boldsymbol{x}}_j^e - Z^{-1}\mathcal{I}^T\mathcal{G}\mathcal{H}_u K^{-1/2}\hat{\boldsymbol{x}}_j^e + Z^{-1}\mathcal{I}^T K^{-1/2}\boldsymbol{s}(\boldsymbol{\zeta}_j^e)\right\|^2 \\
&\overset{(a)}{\leq} 3\mu^2(i+1)\sum_{j=0}^{i}\Big(K^{-2}\|\Lambda_G\check{\boldsymbol{x}}_j^e\|^2 + K^{-1}\|Z^{-1}\mathcal{I}^T\mathcal{G}\mathcal{H}_u\|^2\|\hat{\boldsymbol{x}}_j^e\|^2 \\
&\quad + K^{-1}\|Z^{-1}\mathcal{I}^T\boldsymbol{s}(\boldsymbol{\zeta}_j^e)\|^2\Big) \\
&\overset{(b)}{\leq} 3\mu^2(i+1)\sum_{j=0}^{i}\Big(K^{-2}\|\Lambda_G\|^2\|\check{\boldsymbol{x}}_j^e\|^2 + K^{-1}\|Z^{-1}\mathcal{I}^T\mathcal{G}\mathcal{H}_u\|^2\|\hat{\boldsymbol{x}}_j^e\|^2 \\
&\quad + K^{-1}\|Z^{-1}\|^2\|\mathcal{I}^T\boldsymbol{s}(\boldsymbol{\zeta}_j^e)\|^2\Big) \\
&\overset{(c)}{\leq} 3\mu^2(i+1)\Bigg(\sum_{j=0}^{i}K^{-2}\|\Lambda_G\|^2\|\check{\boldsymbol{x}}_j^e\|^2 + K^{-1}\|Z^{-1}\mathcal{I}^T\mathcal{G}\mathcal{H}_u\|^2\|\hat{\boldsymbol{x}}_j^e\|^2 \\
&+ 2\|Z^{-1}\|^2\max_{k,n}\left(\|\widetilde{G}_{k,n}\|^2\right)\left\|\check{\boldsymbol{x}}_0^e - \check{\boldsymbol{x}}_j^e\right\|^2 \\
&+ 2\|Z^{-1}\|^2\max_{k,n}\left(\left\|G_{k,n}\right\|^2\right)J^{-1}\sum_{n=1}^{J}\left\|\check{\boldsymbol{x}}_J^{e-1} - \check{\boldsymbol{x}}_{n-1}^{e-1}\right\|^2 \\
&+ 2\|Z^{-1}\|^2\|\mathcal{H}_u\|^2\max_{k,n}\left(\|\widetilde{G}_{k,n}\|^2\right)\left\|\hat{\boldsymbol{x}}_0^e - \hat{\boldsymbol{x}}_j^e\right\|^2
\end{aligned}$$

$$+ 2\|Z^{-1}\|^2\|\mathcal{H}_u\|^2 \max_{k,n}\left(\left\|G_{k,n}\right\|^2\right) J^{-1}\sum_{n=1}^{J}\left\|\hat{\boldsymbol{x}}_J^{e-1} - \hat{\boldsymbol{x}}_{n-1}^{e-1}\right\|^2\Bigg) \tag{2.123}$$

where in (a) we used the Jensen's inequality, in (b) Cauchy-Schawrtz inequality and in (c) we used (2.99). We now proceed in a similar fashion to get an inequality for $\|\hat{\boldsymbol{x}}_{i+1}^e - \hat{\boldsymbol{x}}_0^e\|^2$.

$$\begin{aligned}\hat{\boldsymbol{x}}_{i+1}^e - \hat{\boldsymbol{x}}_i^e = &-\mu K^{-1/2}(\mathcal{H}_l^T + \mathcal{H}_r^T\mathcal{V})\bar{\mathcal{L}}\mathcal{G}\mathcal{I}Z\check{\boldsymbol{x}}_i^e + [\mathcal{D}_1 - I_{4M(K-1)} - \mu(\mathcal{H}_l^T + \mathcal{H}_r^T\mathcal{V})\bar{\mathcal{L}}\mathcal{G}\mathcal{H}_u]\hat{\boldsymbol{x}}_i^e \\ &+ \mu(\mathcal{H}_l^T + \mathcal{H}_r^T\mathcal{V})\bar{\mathcal{L}}\boldsymbol{s}(\boldsymbol{\zeta}_i^e)\end{aligned} \tag{2.124}$$

Similarly, as we did before, we get

$$\begin{aligned}\|\hat{\boldsymbol{x}}_{i+1}^e - \hat{\boldsymbol{x}}_0^e\|^2 = &\Bigg\| \sum_{j=0}^{i} -\mu K^{-\frac{1}{2}}(\mathcal{H}_l^T + \mathcal{H}_r^T\mathcal{V})\bar{\mathcal{L}}\mathcal{G}\mathcal{I}Z\check{\boldsymbol{x}}_j^e + [\mathcal{D}_1 - I - \mu(\mathcal{H}_l^T + \mathcal{H}_r^T\mathcal{V})\bar{\mathcal{L}}\mathcal{G}\mathcal{H}_u]\hat{\boldsymbol{x}}_j^e \\ &+ \mu(\mathcal{H}_l^T + \mathcal{H}_r^T\mathcal{V})\bar{\mathcal{L}}\boldsymbol{s}(\boldsymbol{\zeta}_i^e)\Bigg\|^2 \\ &\overset{(a)}{=} (i+1)\sum_{j=0}^{i}\Big\| - \mu K^{-\frac{1}{2}}(\mathcal{H}_l^T + \mathcal{H}_r^T\mathcal{V})\bar{\mathcal{L}}\mathcal{G}\mathcal{I}Z\check{\boldsymbol{x}}_j^e + (\mathcal{D}_1 - I)\hat{\boldsymbol{x}}_j^e \\ &- \mu(\mathcal{H}_l^T + \mathcal{H}_r^T\mathcal{V})\bar{\mathcal{L}}\mathcal{G}\mathcal{H}_u\hat{\boldsymbol{x}}_j^e + \mu(\mathcal{H}_l^T + \mathcal{H}_r^T\mathcal{V})\bar{\mathcal{L}}\boldsymbol{s}(\boldsymbol{\zeta}_i^e)\Big\|^2 \\ &\overset{(b)}{\leq} (i+1)\left(1 - \sqrt{\lambda_2(\bar{L})} + \mu^2 a_5\right)\sum_{j=0}^{i}\|\hat{\boldsymbol{x}}_j^e\|^2 + \mu^2(i+1)a_6\sum_{j=0}^{i}\|\check{\boldsymbol{x}}_j^e\|^2 \\ &+ \mu^2(i+1)a_7\sum_{j=0}^{i}\left(\left\|\check{\boldsymbol{x}}_0^e - \check{\boldsymbol{x}}_j^e\right\|^2 + \|\mathcal{H}_u\|^2\|(\hat{\boldsymbol{x}}_0^e - \hat{\boldsymbol{x}}_j^e)\|^2\right) \\ &+ \mu^2 a_8(i+1)^2 J^{-1}\sum_{n=1}^{J}\left(\left\|\check{\boldsymbol{x}}_0^e - \check{\boldsymbol{x}}_n^e\right\|^2 + \|\mathcal{H}_u\|^2\|(\hat{\boldsymbol{x}}_{n-1}^{e-1} - \hat{\boldsymbol{x}}_J^{e-1})\|^2\right)\end{aligned} \tag{2.125}$$

where in (a) we used Jensen's inequality and in (b) we reused the calculation of the norm in (2.108). Now combining (2.123) and (2.125) in matrix form we get the following inequality

$$
\begin{aligned}
z_{i+1}^e \preceq (i+1) & \underbrace{\begin{bmatrix} 3\mu^2 K^{-2}\|\Lambda_G\|^2 & 3\mu^2 K^{-1}\|Z^{-1}\mathcal{I}^T\mathcal{G}\mathcal{H}_u\|^2 \\ \mu^2 a_6 & 1-\sqrt{\lambda_2(\bar{L})}+\mu^2 a_5 \end{bmatrix}}_{\triangleq M_4} \sum_{j=0}^{i} y_j^e \\
& + (i+1)\mu^2 \underbrace{\begin{bmatrix} 6\|Z^{-1}\|^2 \max_{k,n}\left(\|G_{k,n}\|^2\right) \\ a_7 \end{bmatrix} \begin{bmatrix} 1 \\ \|\mathcal{H}_u\|^2 \end{bmatrix}^T}_{\triangleq M_5} \sum_{j=0}^{i} z_j^e \\
& + (i+1)^2\mu^2 \underbrace{\begin{bmatrix} 6\|Z^{-1}\|^2 \max_{k,n}\left(\left\|G_{k,n}\right\|^2\right) \\ a_8 \end{bmatrix} \begin{bmatrix} 1 \\ \|\mathcal{H}_u\|^2 \end{bmatrix}^T}_{\triangleq M_6} J^{-1} \sum_{n=0}^{J-1} w_n^{e-1} \qquad (2.126)
\end{aligned}
$$

Combining (2.120) and (2.126) we get:

$$
\begin{aligned}
z_{i+1}^e &\preceq (i+1)M_4 \sum_{j=0}^{i}\left(M_1^j y_0^e + \mu M_2 \sum_{n=0}^{j-1} z_n^e + \mu j M_3 \frac{1}{J}\sum_{n=0}^{J-1} w_n^{e-1}\right) + (i+1)\mu^2 M_5 \sum_{j=0}^{i} z_j^e \\
&+ (i+1)^2\mu^2 M_6 \frac{1}{J}\sum_{n=0}^{J-1} w_n^{e-1} \\
&= (i+1)M_4 \sum_{j=0}^{i} M_1^j y_0^e + (i+1)\mu M_4 M_2 \sum_{j=1}^{i}\sum_{n=0}^{j-1} b_n^e + (i+1)\mu^2 M_5 \sum_{j=0}^{i} z_j^e \\
&+ (i+1)\mu\left(\frac{(i+1)i}{2} M_4 M_3 + (i+1)\mu M_6\right)\frac{1}{J}\sum_{n=0}^{J-1} w_n^{e-1} \\
&= (i+1)M_4 \sum_{j=0}^{i} M_1^j y_0^e + (i+1)\mu \sum_{j=0}^{i}((i-j)M_4 M_2 + \mu M_5) z_j^e \\
&+ (i+1)^2\mu\left(\frac{i}{2} M_4 M_3 + \mu M_6\right)\frac{1}{J}\sum_{n=0}^{J-1} w_n^{e-1} \\
&\preceq (i+1)M_4 \sum_{j=0}^{i} M_1^j y_0^e + (i+1)\mu(iM_4 M_2 + \mu M_5)\sum_{j=0}^{i} z_j^e \\
&+ (i+1)^2\mu\left(\frac{i}{2} M_4 M_3 + \mu M_6\right)\frac{1}{J}\sum_{n=0}^{J-1} w_n^{e-1}
\end{aligned}
$$

$$\preceq (i+1)M_4(I-M_1)^{-1}y_0^e + (i+1)\mu(iM_4M_2+\mu M_5)\sum_{j=0}^{i} z_j^e$$
$$+ (i+1)^2\mu\left(\frac{i}{2}M_4M_3+\mu M_6\right)\frac{1}{J}\sum_{n=0}^{J-1} w_n^{e-1} \tag{2.127}$$

Summing across all z_i^e terms in an epoch we get:

$$\frac{1}{J}\sum_{i=1}^{J} z_i^e \preceq \frac{J+1}{2}M_4(I-M_1)^{-1}y_0^e + \mu(J+1)\frac{J}{2}\left(\mu M_5 + JM_4M_2\right)\frac{1}{J}\sum_{i=1}^{J} z_i^e$$
$$+ \mu\frac{J+1}{2}J\left(\frac{J}{2}M_4M_3+\mu M_6\right)\frac{1}{J}\sum_{n=0}^{J-1} w_n^{e-1} \tag{2.128}$$

2.B.6 Bound for w_j^e

We now proceed to obtain a recursion for w_i^e. Similarly, as we did in (2.126) but summing from i to $J-1$, we write:

$$\begin{aligned}
w_i^e &\preceq (J-i)M_4\sum_{j=i}^{J-1} y_j^e + (J-i)\mu^2M_5\sum_{j=i}^{J-1} z_i^e + (J-i)^2\mu^2M_6J^{-1}\sum_{n=1}^{J} w_{n-1}^{e-1} \\
&\preceq (J-i)M_4\sum_{j=i}^{J-1}\left(M_1^j y_0^e + \mu M_2\sum_{n=0}^{j-1} z_n^e + \mu j M_3\frac{1}{J}\sum_{n=1}^{J} w_{n-1}^{e-1}\right) \\
&+ (J-i)\mu^2M_5\sum_{j=i}^{J-1} z_j^e + (J-i)^2\mu^2M_6J^{-1}\sum_{n=1}^{J} w_{n-1}^{e-1} \\
&= (J-i)M_4\sum_{j=i}^{J-1} M_1^j y_0^e + (J-i)\mu M_4M_2\sum_{j=i}^{J-1}\sum_{n=0}^{j-1} z_n^e + (J-i)\mu^2M_5\sum_{j=i}^{J-1} z_j^e \\
&+ (J-i)^2\left(\mu^2M_6+\mu\frac{(J-1+i)}{2}M_4M_3\right)J^{-1}\sum_{n=1}^{J} w_{n-1}^{e-1} \\
&= (J-i)M_4\sum_{j=i}^{J-1} M_1^j y_0^e + (J-i)\mu M_4M_2\sum_{j=1}^{J-2}\min(J-i,J-j-1)z_j^e + (J-i)\mu^2M_5\sum_{j=i}^{J-1} z_j^e \\
&+ (J-i)^2\left(\mu^2M_6+\mu\frac{(J-1+i)}{2}M_4M_3\right)J^{-1}\sum_{n=1}^{J} w_{n-1}^{e-1} \\
&\preceq (J-i)M_4\sum_{j=i}^{J-1} M_1^j y_0^e + (J-i)^2\mu M_4M_2\sum_{j=1}^{J-2} z_j^e + (J-i)\mu^2M_5\sum_{j=i}^{J-1} z_j^e
\end{aligned}$$

$$
\begin{aligned}
&+ \mu(J-i)^2 \left(\mu M_6 + \frac{(J-1+i)}{2} M_4 M_3\right) J^{-1} \sum_{n=1}^{J} w_{n-1}^{e-1} \\
&\preceq (J-i) M_4 (I - M_1)^{-1} y_0^e + (J-i)^2 \mu M_4 M_2 \sum_{j=1}^{J-2} z_j^e + (J-i)\mu^2 M_5 \sum_{j=i}^{J-1} z_j^e \\
&+ \mu(J-i)^2 \left(\mu M_6 + \frac{(J-1+i)}{2} M_4 M_3\right) J^{-1} \sum_{n=1}^{J} w_{n-1}^{e-1} \qquad (2.129)
\end{aligned}
$$

Summing we finally get:

$$
\begin{aligned}
\frac{1}{J}\sum_{i=0}^{J-1} w_i^e &\preceq J^{-1} \sum_{i=0}^{J-1} (J-i) M_4 (I-M_1)^{-1} y_0^e + \frac{1}{J}\sum_{i=0}^{J-1} (J-i)^2 \mu M_4 M_2 \sum_{j=1}^{J-2} z_j^e \\
&+ \frac{1}{J}\sum_{i=0}^{J-1} (J-i)\mu^2 M_5 \sum_{j=i}^{J-1} z_j^e \\
&+ J^{-1} \sum_{i=0}^{J-1} \mu (J-i)^2 \left(\mu M_6 + \frac{(J-1+i)}{2} M_4 M_3\right) \frac{1}{J} \sum_{n=1}^{J} w_{n-1}^{e-1} \\
&= J^{-1} \sum_{i=1}^{J} i M_4 (I-M_1)^{-1} y_0^e + \sum_{i=1}^{J} i^2 \mu M_4 M_2 \frac{1}{J} \sum_{j=1}^{J} z_j^e + \sum_{i=0}^{J-1} (J-i)\mu^2 M_5 \frac{1}{J} \sum_{j=i}^{J-1} z_j^e \\
&+ J^{-1} \mu \sum_{i=1}^{J} i^2 \left(\mu M_6 + (J-1) M_4 M_3\right) \frac{1}{J} \sum_{n=1}^{J} w_{n-1}^{e-1} \\
&\preceq J^{-1} \sum_{i=1}^{J} i M_4 (I-M_1)^{-1} y_0^e + \sum_{i=1}^{J} i^2 \mu M_4 M_2 \frac{1}{J} \sum_{j=1}^{J} z_j^e + \sum_{i=1}^{J} i \mu^2 M_5 \frac{1}{J} \sum_{j=1}^{J-1} z_j^e \\
&+ J^{-1} \mu \sum_{i=1}^{J} i^2 \left(\mu M_6 + (J-1) M_4 M_3\right) \frac{1}{J} \sum_{n=1}^{J} w_{n-1}^{e-1} \\
&\preceq \frac{J+1}{2} M_4 (I-M_1)^{-1} y_0^e + \mu (J+1) \frac{J}{2} \big(\mu M_5 + J M_4 M_2\big) \frac{1}{J} \sum_{j=1}^{J} z_j^e \\
&+ \mu (J+1) \frac{J}{2} \left(\mu M_6 + (J-1) M_4 M_3\right) \frac{1}{J} \sum_{n=1}^{J} w_{n-1}^{e-1} \qquad (2.130)
\end{aligned}
$$

At this point we have recursions for y_0^{e+1}, $J^{-1}\sum_{i=1}^{J} z_i^e$ and $J^{-1}\sum_{i=0}^{J-1} w_i^e$ which we rewrite

for convenience.

$$y_0^{e+1} \preceq M_1^J y_0^e + \mu J M_2 \frac{1}{J}\sum_{i=1}^{J} z_i^e + \mu J M_3 \frac{1}{J}\sum_{i=0}^{J-1} w_i^{e-1} \tag{2.131}$$

$$\begin{aligned}\frac{1}{J}\sum_{i=1}^{J} z_i^e \preceq{}& \frac{J+1}{2} M_4 (I - M_1)^{-1} y_0^e + \mu(J+1)\frac{J}{2}\big(\mu M_5 + J M_4 M_2\big)\frac{1}{J}\sum_{i=1}^{J} z_i^e \\ &+ \mu(J+1)\frac{J}{2}\big(\mu M_6 + J M_4 M_3\big)\frac{1}{J}\sum_{i=0}^{J-1} w_i^{e-1}\end{aligned} \tag{2.132}$$

$$\begin{aligned}\frac{1}{J}\sum_{i=0}^{J-1} w_i^e \preceq{}& \frac{J+1}{2} M_4 (I - M_1)^{-1} y_0^e + \mu(J+1)\frac{J}{2}\big(\mu M_5 + J M_4 M_2\big)\frac{1}{J}\sum_{i=1}^{J} z_i^e \\ &+ \mu(J+1)\frac{J}{2}\big(\mu M_6 + J M_4 M_3\big)\frac{1}{J}\sum_{i=0}^{J-1} w_i^{e-1}\end{aligned} \tag{2.133}$$

We now make the following definitions:

$$y^e \triangleq y_0^e \in \mathbb{R}^2 \tag{2.134a}$$

$$z^e \triangleq \begin{bmatrix} 1 & \|\mathcal{H}_u\|^2 \end{bmatrix} J^{-1} \sum_{n=1}^{J} z_n^e \in \mathbb{R} \tag{2.134b}$$

$$w^e \triangleq \begin{bmatrix} 1 & \|\mathcal{H}_u\|^2 \end{bmatrix} J^{-1} \sum_{n=0}^{J-1} w_n^e \in \mathbb{R} \tag{2.134c}$$

Noting that

$$\mu M_5 + J M_4 M_2 = \mu \underbrace{\begin{bmatrix} 6\|Z^{-1}\|^2 \max_{k,n}(\|G_{k,n}\|^2) \\ +3\mu J\left(K^{-2}\|\Lambda_G\|^2 a_3 + \mu a_7 K^{-1}\|Z^{-1}\mathcal{I}^T\mathcal{G}\mathcal{H}_u\|^2\right) \\ a_7 + J\left(\mu a_3 a_6 + a_7\left(1 - \sqrt{\lambda_2(\bar{\bar{L}})} + \mu^2 a_5\right)\right) \end{bmatrix}}_{\triangleq \begin{bmatrix} t_1 \\ t_2 \end{bmatrix}} \begin{bmatrix} 1 \\ \|\mathcal{H}_u\|^2 \end{bmatrix}^T \tag{2.135}$$

$$\mu M_6 + JM_4M_3 = \mu \underbrace{\begin{bmatrix} 6\|Z^{-1}\|^2 \max_{k,n}(\|G_{k,n}\|^2) \\ +3\mu J\left(K^{-2}\|\Lambda_G\|^2 a_4 + \mu a_8 K^{-1}\|Z^{-1}\mathcal{I}^T\mathcal{G}\mathcal{H}_u\|^2\right) \\ \\ a_8 + J\left(\mu a_4 a_6 + a_8\left(1-\sqrt{\lambda_2(\bar{\bar{L}})}+\mu^2 a_5\right)\right) \end{bmatrix}}_{\triangleq \begin{bmatrix} t_3 \\ t_4 \end{bmatrix}} \begin{bmatrix} 1 \\ \|\mathcal{H}_u\|^2 \end{bmatrix}^T \tag{2.136}$$

using (2.115e) and (2.115f) and multiplying both sides of (2.132) and (2.133) by $\begin{bmatrix} 1 & \|\mathcal{H}_u\|^2 \end{bmatrix}$ we get:

$$y^{e+1} \preceq M_1^J y^e + \mu J \begin{bmatrix} a_3 \\ \mu a_7 \end{bmatrix} z^e + \mu J \begin{bmatrix} a_4 \\ \mu a_8 \end{bmatrix} w^{e-1} \tag{2.137}$$

$$\begin{aligned} z^e \leq \frac{J+1}{2} \begin{bmatrix} 1 \\ \|\mathcal{H}_u\|^2 \end{bmatrix}^T & M_4(I-M_1)^{-1}y^e \\ & + \mu^2(J+1)\frac{J}{2}(t_1 + \|\mathcal{H}_u\|^2 t_2)z^e + \mu^2(J+1)\frac{J}{2}(t_3 + \|\mathcal{H}_u\|^2 t_4)w^{e-1} \end{aligned} \tag{2.138}$$

$$\begin{aligned} w^e \leq \frac{J+1}{2} \begin{bmatrix} 1 \\ \|\mathcal{H}_u\|^2 \end{bmatrix}^T & M_4(I-M_1)^{-1}y^e + \mu^2(J+1)\frac{J}{2}(t_1 + \|\mathcal{H}_u\|^2 t_2)z^e \\ & + \mu^2(J+1)\frac{J}{2}(t_3 + \|\mathcal{H}_u\|^2 t_4)w^{e-1} \end{aligned} \tag{2.139}$$

we now impose the following constraint on μ:

$$\mu < \sqrt{\frac{2}{(J+1)J(t_1 + \|\mathcal{H}_u\|^2 t_2)}} \tag{2.140}$$

which implies that $(1 - \mu^2 K^{-1}(J+1)J(t_1 + \|\mathcal{H}_u\|^2 t_2)) > 0$ and hence we can derive the

following inequality for z^e:

$$\underbrace{\left(1-\mu^2 2^{-1}(J+1)J(t_1+\|\mathcal{H}_u\|^2 t_2)\right)}_{\triangleq d_1} z^e \leq 2^{-1}(J+1)\Big(\begin{bmatrix} 1 & \|\mathcal{H}_u\|^2 \end{bmatrix} M_4(I-M_1)^{-1}y^e$$
$$+\mu^2 J(t_3+\|\mathcal{H}_u\|^2 t_4)w^e\Big) \tag{2.141}$$
$$z^e \leq \frac{2^{-1}(J+1)}{d_1}\Big(\begin{bmatrix} 1 & \|\mathcal{H}_u\|^2 \end{bmatrix} M_4(I-M_1)^{-1}y^e + \mu^2 J(t_3+\|\mathcal{H}_u\|^2 t_4)w^{e-1}\Big) \tag{2.142}$$

Replacing (2.142) into (2.137) and (2.139) we get

$$\begin{aligned} y^{e+1} &\preceq \left(M_1^J + \mu(2d_1)^{-1}(J+1)JM_2M_4(I-M_1)^{-1}\right)y^e \\ &\quad + \mu J\left(\begin{bmatrix} a_4 \\ \mu a_8 \end{bmatrix} + \mu^2 J(2d_1)^{-1}(J+1)(t_3+\|\mathcal{H}_u\|^2 t_4)\begin{bmatrix} a_3 \\ \mu a_7 \end{bmatrix}\right)w^{e-1} \qquad (2.143) \\ w^e &\leq \frac{J+1}{2}\left(1+\mu^2(2d_1)^{-1}(J+1)J(t_1+\|\mathcal{H}_u\|^2 t_2)\right)\begin{bmatrix} 1 \\ \|\mathcal{H}_u\|^2 \end{bmatrix}^T M_4(I-M_1)^{-1}y^e \\ &\quad + \mu^2(J+1)\frac{J}{2}(t_3+\|\mathcal{H}_u\|^2 t_4)\left(1+\mu^2(2d_1)^{-1}(J+1)J(t_1+\|\mathcal{H}_u\|^2 t_2)\right)w^{e-1} \\ &= \frac{J+1}{2d_1}\left(\begin{bmatrix} 1 \\ \|\mathcal{H}_u\|^2 \end{bmatrix}^T M_4(I-M_1)^{-1}y^e + \mu^2 J(t_3+\|\mathcal{H}_u\|^2 t_4)w^{e-1}\right) \qquad (2.144) \end{aligned}$$

Or written in matrix form:

$$\begin{bmatrix} y^{e+1} \\ w^e \end{bmatrix} \preceq \begin{bmatrix} M_1^J + \mu f_1 JM_2M_4(I-M_1)^{-1} & \mu J\begin{bmatrix} a_4+\mu^2 f_2 a_3 \\ \mu a_8+\mu^3 f_2 a_7 \end{bmatrix} \\ f_1\begin{bmatrix} 1 & \|\mathcal{H}_u\|^2 \end{bmatrix} M_4(I-M_1)^{-1} & \mu^2 f_2 \end{bmatrix}\begin{bmatrix} y^e \\ w^{e-1} \end{bmatrix} \tag{2.145}$$

where we defined:

$$f_1 \triangleq (2d_1)^{-1}(J+1) \tag{2.146a}$$
$$f_2 \triangleq (2d_1)^{-1}(J+1)J(t_3+\|\mathcal{H}_u\|^2 t_4) \tag{2.146b}$$

which completes the proof.

2.B.7 Proof Theorem 2.3

We start by multiplying both sides of (2.145) by $\begin{bmatrix} 1 & 1 & \mu\epsilon \end{bmatrix}$ as follows

$$\begin{bmatrix} 1 \\ 1 \\ \mu\epsilon \end{bmatrix}^T \begin{bmatrix} \|\check{\boldsymbol{x}}_0^{e+1}\|^2 \\ \|\hat{\boldsymbol{x}}_0^{e+1}\|^2 \\ w^e \end{bmatrix} \leq \begin{bmatrix} b_1 & b_2 & \mu\epsilon b_3 \end{bmatrix} \begin{bmatrix} \|\check{\boldsymbol{x}}_0^{e}\|^2 \\ \|\hat{\boldsymbol{x}}_0^{e}\|^2 \\ w^{e-1} \end{bmatrix} \leq \max(b_1, b_2, b_3) \begin{bmatrix} 1 \\ 1 \\ \mu\epsilon \end{bmatrix}^T \begin{bmatrix} \|\check{\boldsymbol{x}}_0^{e}\|^2 \\ \|\hat{\boldsymbol{x}}_0^{e}\|^2 \\ w^{e-1} \end{bmatrix} \tag{2.147}$$

where ϵ is positive constant which we will define later and b_1, b_2 and b_3 are:

$$\begin{aligned} \begin{bmatrix} b_1 & b_2 \end{bmatrix} &= \begin{bmatrix} 1 & 1 \end{bmatrix} M_1^J + \mu f_1 \left(J \begin{bmatrix} 1 & 1 \end{bmatrix} M_2 + \epsilon \begin{bmatrix} 1 & \|\mathcal{H}_u\|^2 \end{bmatrix} \right) M_4 (I - M_1)^{-1} \\ &= \begin{bmatrix} 1 & 1 \end{bmatrix} M_1^J + \mu f_1 \left(J(a_3 + \mu a_7) + \epsilon \right) \begin{bmatrix} 1 & \|\mathcal{H}_u\|^2 \end{bmatrix} M_4 (I - M_1)^{-1} \end{aligned} \tag{2.148}$$

$$b_3 = \epsilon^{-1} J(a_4 + \mu^2 f_2 a_3) + \epsilon^{-1} J(\mu a_8 + \mu^3 f_2 a_7) + \mu^2 f_2 = \epsilon^{-1} J(a_4 + \mu a_8) + \mathcal{O}(\mu^2) \tag{2.149}$$

Note that if we constrain the step-size μ to be small enough so that $1 - \mu K^{-1} a_1 + \mu^2 a_6 < 1$ and $\sqrt{\lambda_2(\bar{L})} + \mu^2 a_5 + \mu a_2 < 1$, we get that $\begin{bmatrix} 1 & 1 \end{bmatrix} M_1^J \preceq \begin{bmatrix} 1 & 1 \end{bmatrix} M_1$ and hence we can further write:

$$\begin{bmatrix} b_1 & b_2 \end{bmatrix} \preceq \begin{bmatrix} 1 & 1 \end{bmatrix} M_1 + \mu f_1 \left(J(a_3 + \mu a_7) + \epsilon \right) \begin{bmatrix} 1 & \|\mathcal{H}_u\|^2 \end{bmatrix} M_4 (I - M_1)^{-1} \tag{2.150}$$

The objective now is to prove that $\max(b_1, b_2, b_3) < 1$ and hence the algorithm converges linearly to the minimizer. We now proceed to obtain expressions for b_1 and b_2. For this, we

expand the terms in (2.150). For this we expand $\begin{bmatrix} 1 & \|\mathcal{H}_u\|^2 \end{bmatrix} M_4(I-M_1)^{-1}$:

$$
\begin{aligned}
(I-M_1)^{-1} &= \begin{bmatrix} \mu K^{-1}a_1 & -\mu a_2 \\ -\mu^2 a_6 & 1-\sqrt{\lambda_2(\bar{L})}-\mu^2 a_5 \end{bmatrix}^{-1} \\
&= \underbrace{\frac{1}{a_1K^{-1}\left(1-\sqrt{\lambda_2(\bar{L})}-\mu^2a_5\right)-\mu^2a_2a_6}}_{\triangleq d_2} \begin{bmatrix} \frac{1-\sqrt{\lambda_2(\bar{L})}}{\mu}-\mu a_5 & a_2 \\ \mu a_6 & K^{-1}a_1 \end{bmatrix}
\end{aligned} \tag{2.151}
$$

$$
\begin{aligned}
&\begin{bmatrix} 1 \\ \|\mathcal{H}_u\|^2 \end{bmatrix}^T M_4(I-M_1)^{-1} \\
&= \frac{1}{d_2}\begin{bmatrix} 3\mu^2K^{-2}\|\Lambda_G\|^2+\mu^2a_6\|\mathcal{H}_u\|^2 \\ 3\mu^2K^{-1}\|Z^{-1}\mathcal{I}^T\mathcal{G}\mathcal{H}_u\|^2+\|\mathcal{H}_u\|^2\left(1-\sqrt{\lambda_2(\bar{L})}+\mu^2a_5\right) \end{bmatrix}^T \\
&\cdot \begin{bmatrix} \frac{1-\sqrt{\lambda_2(\bar{L})}}{\mu}-\mu a_5 & a_2 \\ \mu a_6 & K^{-1}a_1 \end{bmatrix} \\
&= \frac{1}{d_2}\begin{bmatrix} \mu\left(1-\sqrt{\lambda_2(\bar{L})}\right)(3K^{-2}\|\Lambda_G\|^2+2a_6\|\mathcal{H}_u\|^2) \\ +3\mu^3\left(a_6K^{-1}\|Z^{-1}\mathcal{I}^T\mathcal{G}\mathcal{H}_u\|^2-K^{-2}\|\Lambda_G\|^2a_5\right) \\ \\ K^{-1}\|\mathcal{H}_u\|^2\left(1-\sqrt{\lambda_2(\bar{L})}\right)a_1+\mu^2(3K^{-2}\|\Lambda_G\|^2+a_6\|\mathcal{H}_u\|^2)a_2 \\ +\mu^2\left(K^{-1}\|Z^{-1}\mathcal{I}^T\mathcal{G}\mathcal{H}_u\|^2+a_5\|\mathcal{H}_u\|^2\right)K^{-1}a_1 \end{bmatrix}^T
\end{aligned} \tag{2.152}
$$

Now using (2.150) and (2.152) and setting $\epsilon = \frac{J(a_4+\mu a_8)}{1-\mu K^{-1}a_1+\mu^2a_6}$ we can write the following expressions for b_1, b_2 and b_3:

$$b_1 = 1-\mu K^{-1}a_1+\mathcal{O}(\mu^2) \tag{2.153a}$$

$$b_2 = \sqrt{\lambda_2(\bar{L})}+\mathcal{O}(\mu) \tag{2.153b}$$

$$b_3 = 1-\mu K^{-1}a_1+\mathcal{O}(\mu^2) \tag{2.153c}$$

From (2.153a) and (2.153c) it is clear that for small enough step-size μ, $b_1 < 1$ and $b_3 < 1$. Also since $\lim_{\mu\to 0} b_2 = \sqrt{\lambda_2(\bar{L})}$, then it is clear that there is a step-size μ small enough such that $b_2 < 1$. Like we said before since $\max(b_1, b_2, b_3) < 1$ this implies that $\begin{bmatrix} \|\check{\boldsymbol{x}}_0^{e+1}\|^2 & \|\hat{\boldsymbol{x}}_0^{e+1}\|^2 & w^e \end{bmatrix}$ converges linearly to $\begin{bmatrix} 0 & 0 & 0 \end{bmatrix}$ with a rate bounded by $\max[1-\mu K^{-1}a_1+\mathcal{O}(\mu^2), \sqrt{\lambda_2(\bar{L})}+\mathcal{O}(\mu)]$. Remembering the definition of a_1 in (2.105) we can equivalently bound the convergence rate by $\max[\rho(I-\mu K^{-1}\Lambda_G)+\mathcal{O}(\mu^2), \sqrt{\lambda_2(\bar{L})}+\mathcal{O}(\mu)]$.

2.C Proof of Lemma 2.1

To simplify the notation we refer to $\theta^o(H,\lambda)$ as θ^o. Using (2.19) and defining $R = A^{-1}CA^{-T}U$, we can write $\theta^o = (I+\eta R)^{-1}(\eta R\theta_{\mathrm{p}} + A^{-1}b)$. Using the Jordan decomposition $R = J_R\Lambda_R J_R^{-1}$ we get:

$$\theta^o = J_R(I+\eta\Lambda_R)^{-1}J_R^{-1}A^{-1}b + J_R(I+\eta^{-1}\Lambda_R^{-1})^{-1}J_R^{-1}\theta_{\mathrm{p}} \tag{2.154}$$

Note that all the eigenvalues of $(I+\eta\Lambda_R)^{-1}$ and $(I+\eta^{-1}\Lambda_R^{-1})^{-1}$ have the forms $(1+\eta\Lambda_{R,i})^{-1}$ and $\eta\Lambda_{R,i}/(1+\eta\Lambda_{R,i})$, respectively (where $\Lambda_{R,i}$ is the i-th eigenvalue). Therefore, to be able to obtain an approximation of the bias that depends only on the design parameters, we make the following approximation:

$$\theta^o \approx (1+\kappa_1\eta)^{-1}A^{-1}b + \kappa_1\eta(1+\kappa_1\eta)^{-1}\theta_{\mathrm{p}} \tag{2.155}$$

for some constant κ_1. Note that expression (2.155) is an equality for $\eta = 0$, since the regularization constant is typically small in practice and (2.155) is a useful approximation. We proceed to calculate the bias of $\theta^\bullet = A^{-1}b$ with respect to $\theta^\star$. Using (2.8) and (2.13) we

can write:

$$v^{\pi} = \Gamma_2 r^{\pi} + \rho_1 \Gamma_1 v^{\pi} = \sum_{n=0}^{\infty} (\rho_1 \Gamma_1)^n \Gamma_2 r^{\pi} \tag{2.156}$$

$$X\theta^{\bullet} = \Pi \left(\Gamma_2 r^{\pi} + \rho_1 \Gamma_1 X\theta^{\bullet} \right) = \Pi \sum_{n=0}^{\infty} (\rho_1 \Gamma_1 \Pi)^n \Gamma_2 r^{\pi} \tag{2.157}$$

$$X\theta^{\star} = \Pi v^{\pi} = \Pi \sum_{n=0}^{\infty} (\rho_1 \Gamma_1)^n \Gamma_2 r^{\pi} \tag{2.158}$$

Combining the expressions from above we get:

$$X(\theta^{\bullet} - \theta^{\star}) = \Pi \sum_{n=0}^{\infty} \left((\rho_1 \Gamma_1 \Pi)^n \Gamma_2 r^{\pi} - v^{\pi} \right) \tag{2.159}$$

$$= \Pi \sum_{n=1}^{\infty} \left((\rho_1 \Gamma_1 \Pi)^n - (\rho_1 \Gamma_1)^n \right) \Gamma_2 r^{\pi} \tag{2.160}$$

Note that if $v^{\pi} = \Pi v^{\pi} = X\theta^{\star}$, then (2.156) can be rewritten as:

$$v^{\pi} = \sum_{n=0}^{\infty} (\rho_1 \Gamma_1 \Pi)^n \Gamma_2 r^{\pi} \tag{2.161}$$

Combining (2.161) with (2.159) we get that if $v^{\pi} = \Pi v^{\pi}$, then $\breve{\theta} - \theta^{\star} = 0$. Therefore, we can write (2.160) as:

$$X(\theta^{\bullet} - \theta^{\star}) = \mathbb{I}(v^{\pi} \neq \Pi v^{\pi}) \Pi \sum_{n=1}^{\infty} \rho_1^n \left((\Gamma_1 \Pi)^n - \Gamma_1^n \right) \Gamma_2 r^{\pi} \tag{2.162}$$

where $\mathbb{I}$ is the indicator function. We now approximate the right stochastic matrix P^{π} by it's steady state limit (given by $(P^{\pi})^{\infty} = \mathbb{1} p^T$, where $\mathbb{1}$ is the all ones vector and p is the vector with the steady state distribution induced by the transition matrix P^{π}). Consequently, we

get:

$$\Gamma_1 \approx \mathbb{1}p^T \tag{2.163}$$

$$\Gamma_2 r^\pi \approx \frac{1-(\gamma\lambda)^H}{1-\gamma\lambda}\mathbb{1}p^T r^\pi = \left(\frac{1-\rho_1}{1-\gamma}\right)\mathbb{1}p^T r^\pi \tag{2.164}$$

$$\epsilon = p^T \Pi \mathbb{1} \tag{2.165}$$

$$\sum_{n=1}^{\infty} \rho_1^n \left((\Gamma_1\Pi)^n - \Gamma_1^n\right)\Gamma_2 r^\pi \approx \sum_{n=1}^{\infty} \rho_1^n \left((\mathbb{1}p^T\Pi)^n - \mathbb{1}p^T\right)\Gamma_2 r^\pi \tag{2.166}$$

$$= \mathbb{1}\left(\frac{1-(\gamma\lambda)^H}{1-\gamma\lambda}\right) p^T r^\pi \sum_{n=1}^{\infty} \rho_1^n \left(\epsilon^n - 1\right) \tag{2.167}$$

We note that the above approximations become tighter as $\gamma \to 1$, $\lambda \to 1$, $H \to \infty$ or when P^π has a fast mixing time. Since $p^T\mathbb{1} = 1$ and Π is a projection matrix (and therefore its eigenvalues are either 0 or 1) we get that $|\epsilon| \leq 1$ and therefore we can write:

$$\sum_{n=1}^{\infty} \rho_1^n \left(\epsilon^n - 1\right) = \frac{\rho_1\epsilon}{1-\rho_1\epsilon} - \frac{\rho_1}{1-\rho_1} = \frac{\rho_1(\epsilon-1)}{(1-\rho_1\epsilon)(1-\rho_1)} \tag{2.168}$$

$$\sum_{n=1}^{\infty} \rho_1^n \left((\Gamma_1\Pi)^n - \Gamma_1^n\right)\Gamma_2 r^\pi \approx \mathbb{1}p^T r^\pi = \mathcal{O}\left(\frac{\rho_1(\epsilon-1)}{(1-\rho_1\epsilon)(1-\gamma)}\right) \tag{2.169}$$

Therefore, combining (2.155), (2.162) and (2.169) and grouping constants we can finally write:

$$\|\theta^o - \theta^\star\|^2 \approx \left(\mathbb{I}\big(v^\pi \neq \Pi v^\pi\big)\frac{\kappa_2\rho_1}{(1+\kappa_1\eta)(\kappa_3-\rho_1)} + \frac{\kappa_1\eta\|\theta_{\mathrm{p}} - \theta^\star\|}{1+\kappa_1\eta}\right)^2 \tag{2.170}$$

2.D Proof of Lemma 2.2

Since the regularization term of (2.155) is not subject to variance, to calculate the variance in the estimate of $\widehat{\theta^o}$ we need to calculate the variance of $\widehat{\theta^\bullet} = \widehat{A}^{-1}\hat{b}$ as follows:

$$\mathbb{E}\,\|\widehat{\theta^\bullet} - \theta^\bullet\|^2 = \mathbb{E}\,\|\widehat{A}^{-1}\hat{b} - \theta^\bullet\|^2 = \mathbb{E}\,\left\|\widehat{A}^{-1}\big(\hat{b} - \widehat{A}\theta^\bullet\big)\right\|^2 \tag{2.171}$$

where we are using the squared Euclidean norm. Due to the Rayleigh-Ritz' Theorem we have $\sigma_{\min}(\widehat{A}^{-1}) \leq \|\widehat{A}^{-1}\| \leq \sigma_{\max}(\widehat{A}^{-1})$ and since $\sigma_{\max}(\widehat{A}) = \sigma_{\min}(\widehat{A}^{-1})$ and $\sigma_{\max}(\widehat{A}^{-1}) = \sigma_{\min}(\widehat{A})$ we can write:

$$\mathbb{E}\,\sigma^2_{\max}(\widehat{A})\left\|\hat{b} - \widehat{A}\theta^\bullet\right\|^2 \leq \mathbb{E}\,\|\widehat{\theta^\bullet} - \theta^\bullet\|^2 \leq \mathbb{E}\,\sigma^2_{\min}(\widehat{A})\left\|\hat{b} - \widehat{A}\theta^\bullet\right\|^2 \tag{2.172}$$

Now using Proposition 9.6.8 of [78] and the fact that $\widehat{A} = \widehat{C} - X^T D\widehat{\Gamma}_1 X$ we can write:

$$\sigma_{\min}(\widehat{A}) = \lambda_{\min}(\widehat{C}) \pm \rho_1\sigma_{\max}(\widehat{E}) = \lambda_{\min}(\widehat{C}) + \mathcal{O}(\rho_1) \tag{2.173}$$

$$\sigma_{\max}(\widehat{A}) = \lambda_{\max}(\widehat{C}) \pm \rho_1\sigma_{\max}(\widehat{E}) = \lambda_{\max}(\widehat{C}) + \mathcal{O}(\rho_1) \tag{2.174}$$

$$\widehat{E} = X^T D\widehat{\Gamma}_1 X \tag{2.175}$$

Using the above results and the intermediate value theorem we can write:

$$\mathbb{E}\,\|\widehat{\theta^\bullet} - \theta^\bullet\|^2 = (\epsilon_2 + \mathcal{O}(\rho_1))^2\left\|\hat{b} - \widehat{A}\theta^\bullet\right\|^2 \tag{2.176}$$

for some constant ϵ_2. For $\mathbb{E}\left\|\hat{b} - \widehat{A}\theta^\bullet\right\|^2$ we can write:

$$\mathbb{E}\left\|\hat{b} - \widehat{A}\theta^\bullet\right\|^2 = \mathbb{E}\left\|\sum_{t=1}^{N-H}\frac{\hat{b}_t - \widehat{A}_t\theta^\bullet}{N-H}\right\|^2 \overset{(b)}{=} \frac{\mathbb{E}\,\|\hat{b}_t - \widehat{A}_t\theta^\bullet\|^2}{N-H} \tag{2.177}$$

where in (b) we assumed that the total amount of data collected is significantly larger than the mixing rate of the Markov Chain, and therefore the terms $\hat{b}_t - \widehat{A}_t\theta^\bullet$ corresponding to different times can be considered independent of each other. Note that this is a standard assumption (similar to the one stated in Assumption 2.4) in the sense that in practice it demands that enough data is gathered so that the effect of the policy on every state can be

accurately estimated. To approximate $\mathbb{E}\left\|\hat{b}_t - \widehat{A}_t\theta^\bullet\right\|^2$ we proceed as follows:

$$\begin{aligned}\mathbb{E}\left\|\hat{b}_t - \widehat{A}_t\theta^\bullet\right\|^2 &= \mathbb{E}\left\|x_t\right\|^2\Bigg((\gamma\lambda)^{H-1}(r_{t+H-1} + \gamma x_{t+H}^T\theta^\bullet) + \sum_{n=0}^{H-2}(\gamma\lambda)^n\left(r_{t+n} + \gamma(1-\lambda)x_{t+n+1}^T\theta^\bullet\right) \\ &\qquad - x_t^T\theta^\bullet\Bigg)^2 \\ &\overset{(a)}{=} \mathbb{E}\left\|x_t\right\|^2\left(\sum_{n=0}^{H-1}(\lambda)^n\left(x_{t+n}^T\theta^\bullet - r_{t+n} - \gamma x_{t+n+1}^T\theta^\bullet\right)\right)^2 \\ &\overset{(b)}{\approx} \mathbb{E}\left\|x_t\right\|^2\left(\sum_{n=0}^{H-1}(\gamma\lambda)^n\left(v(s_{t+n}) - r_{t+n} - \gamma v(s_{t+n+1})\right)\right)^2 \\ &\overset{(c)}{=} \mathbb{E}\left\|x_t\right\|^2\sum_{n=0}^{H-1}(\gamma\lambda)^{2n}\left(v(s_{t+n}) - r_{t+n} - \gamma v(s_{t+n+1})\right)^2 \end{aligned} \tag{2.178}$$

where in (a) we simply reorganized the terms, in (b) we used $x_t^T\theta^\bullet \approx v(s_t)$ and in (c) we used the fact that due to the Markov property each term $v(s_{t+n}) - r_{t+n} - \gamma v(s_{t+n+1})$ is conditionally independent from all previous terms (conditioned on s_{t+n}) and that by definition $\mathbb{E}\left[v(s_{t+n}) - r_{t+n} - \gamma v(s_{t+n+1})|s_{t+n}\right] = 0$. Now we lower and upper bound (2.178) as follows:

$$m\sum_{n=0}^{H-1}(\gamma\lambda)^{2n} \leq \mathbb{E}\left\|\hat{b}_t - \widehat{A}_t\theta^\bullet\right\|^2 \leq M\sum_{n=0}^{H-1}(\gamma\lambda)^{2n} \tag{2.179}$$

$$M = \max_{t,n}\left\|x_t\right\|^2\left(v(s_{t+n}) - r_{t+n} - \gamma v(s_{t+n+1})\right)^2 \tag{2.180}$$

$$m = \min_{t,n}\left\|x_t\right\|^2\left(v(s_{t+n}) - r_{t+n} - \gamma v(s_{t+n+1})\right)^2 \tag{2.181}$$

Combining the above result with the intermediate value theorem we get:

$$\mathbb{E}\left\|\hat{b}_t - \widehat{A}_t\theta^\bullet\right\|^2 \approx \epsilon_3\left(\frac{1-(\gamma\lambda)^{2H}}{1-(\gamma\lambda)^2}\right) \tag{2.182}$$

for some constant ϵ_3. Finally, combining (2.155), (2.182), (2.176) and (2.177) and assuming $N >> H$ (which in practice should be the case) we get:

$$\mathbb{E}\left\|\widehat{\theta^o} - \theta^o\right\|^2 \approx \frac{\kappa_4}{(1+\kappa_1\eta)^2(N-H)}\left(\frac{1-(\gamma\lambda)^{2H}}{1-(\gamma\lambda)^2}\right) \tag{2.183}$$

where $\kappa_4 = \epsilon_2\epsilon_3$.

2.E Proof Lemma 2.3

To write A, b and C as expectations we use definitions (2.16) and (2.9), and then expand the matrix operations.

$$\begin{aligned}
A &= X^T D(I-\rho_1\Gamma_1)X \\
&= X^T D\Big(I - \gamma(1-\lambda)P^\pi \textstyle\sum_{n=0}^{H-1}(\gamma\lambda P^\pi)^n + (\gamma\lambda P^\pi)^H\Big)X \\
&= \sum_{s_t\in\mathcal{S}} d^\phi(s_t)x_{s_t}\Big(x_{s_t} - \gamma(1-\lambda)\sum_{n=0}^{H-1}(\gamma\lambda)^n \sum_{s_{t+1+n}\in\mathcal{S}} p^\pi_{s_t,s_{t+1+n}} x_{s_{t+1+n}} + (\gamma\lambda)^H \sum_{s_{t+H}\in\mathcal{S}} p^\pi_{s_t,s_{t+H}} x_{s_{t+H}}\Big)^T \\
&= \mathbb{E}_{d^\phi,\mathcal{P},\pi}\left[\boldsymbol{x}_t\Big(\boldsymbol{x}_t - \gamma(1-\lambda)\sum_{n=0}^{H-1}(\gamma\lambda)^n\boldsymbol{x}_{t+n+1} - (\gamma\lambda)^H\boldsymbol{x}_{t+H}\Big)^T\right] \qquad (2.184) \\
b &= X^T D\sum_{n=0}^{H-1}(\gamma\lambda P^\pi)^n r^\pi \\
&= \sum_{s_t\in\mathcal{S}} d^\phi(s_t)x_{s_t}\sum_{a\in\mathcal{L}}\sum_{s_{t+1}\in\mathcal{S}}\pi(a|s_t)\mathcal{P}(s_{t+1}|s_t,a)r(s_t,a,s_{t+1}) \\
&+ \sum_{s_t\in\mathcal{S}} d^\phi(s_t)x_{s_t}\sum_{n=1}^{H-1}(\gamma\lambda)^n p^\pi_{s_t,s_{t+n}}\sum_{s_{t+n}\in\mathcal{S}}\sum_{a\in\mathcal{L}}\sum_{s_{t+n+1}\in\mathcal{S}}\pi(a|s_{t+n})\mathcal{P}(s_{t+n+1}|s_{t+n},a)r(s_{t+n},a,s_{t+n+1}) \\
&= \mathbb{E}_{d^\phi,\mathcal{P},\pi}\Big(\boldsymbol{x}_t\sum_{n=0}^{H-1}(\gamma\lambda)^n\boldsymbol{r}_{t+n}\Big) \qquad (2.185) \\
C &= X^T DX = \sum_{s_t\in\mathcal{S}} d^\phi(s_t)x_{s_t}x_{s_t} = \mathbb{E}_{d^\phi}\left[\boldsymbol{x}_t\boldsymbol{x}_t^T\right] \qquad (2.186)
\end{aligned}$$

2.F Proof of Lemma 2.4

We start with (2.23a):

$$
\begin{aligned}
A &= \mathbb{E}_{d^{\phi},\mathcal{P},\pi}\Bigg[\boldsymbol{x}_t\Big(\boldsymbol{x}_t - \gamma(1-\lambda)\sum_{h=0}^{H-1}(\gamma\lambda)^h\boldsymbol{x}_{t+h+1} - (\gamma\lambda)^H\boldsymbol{x}_{t+H}\Big)^T\Bigg] \\
&\overset{(a)}{=} (1-\lambda)\sum_{h=0}^{H-1}\lambda^h\mathbb{E}_{d^{\phi},\mathcal{P},\pi}\Big(\boldsymbol{x}_t\big(\boldsymbol{x}_t - \gamma^{h+1}\boldsymbol{x}_{t+h+1}\big)^T\Big) + \lambda^H\mathbb{E}_{d^{\phi},\mathcal{P},\pi}\left(\boldsymbol{x}_t\left(\boldsymbol{x}_t - \gamma^H\boldsymbol{x}_{t+H}\right)^T\right) \\
&\overset{(b)}{=} (1-\lambda)\sum_{h=0}^{H-1}\lambda^h\mathbb{E}_{d^{\phi},\mathcal{P},\phi,f_r}\Big[\boldsymbol{\xi}_{t,t+h+1}\boldsymbol{x}_t\big(\boldsymbol{x}_t - \gamma^{h+1}\boldsymbol{x}_{t+h+1}\big)^T\Big] \\
&+ \lambda^H\mathbb{E}_{d^{\phi},\mathcal{P},\phi,f_r}\Big[\boldsymbol{\xi}_{t,t+H}\boldsymbol{x}_t\big(\boldsymbol{x}_t - \gamma^H\boldsymbol{x}_{t+H}\big)^T\Big] \\
&\overset{(c)}{=} \mathbb{E}_{d^{\phi},\mathcal{P},\phi,f_r}\Bigg[\boldsymbol{x}_t\Bigg(\Big((1-\lambda)\textstyle\sum_{h=0}^{H-1}\lambda^h\boldsymbol{\xi}_{t,t+h+1} + \lambda^H\boldsymbol{\xi}_{t,t+H}\Big)\boldsymbol{x}_t \\
&- \gamma(1-\lambda)\sum_{h=0}^{H-1}(\gamma\lambda)^h\boldsymbol{\xi}_{t,t+h+1}\boldsymbol{x}_{t+h+1} - (\gamma\lambda)^H\boldsymbol{\xi}_{t,t+H}\boldsymbol{x}_{t+H}\Bigg)^T\Bigg]
\end{aligned}
\tag{2.187}
$$

where in (a) and (c) we simply rearranged terms. And in (b) we introduced the importance sampling weights corresponding to the trajectory that started at some state s_t and took h steps before arriving at some other state s_{t+h}, which are given by:

$$
\xi_{t,t+h} = \prod_{j=t}^{t+h-1}\pi(a_j|s_j)/\phi(a_j|s_j) \tag{2.188}
$$

Hence, by removing the expectation in (2.187), we can get the following estimate of A using a single H-step trajectory:

$$
\widehat{A}_n = x_n\big(\rho_{n,0}^H x_n - \gamma(1-\lambda)\sum_{h=0}^{H-1}(\gamma\lambda)^h\xi_{n,n+h+1}x_{n+h+1} - (\gamma\lambda)^H\xi_{n,n+H}x_{n+H}\big)^T \tag{2.189}
$$

where we defined:

$$
\rho_{t,n}^H = \left((1-\lambda)\textstyle\sum_{h=n}^{H-1}\lambda^{h-n}\boldsymbol{\xi}_{t,t+h+1} + \lambda^{H-n}\boldsymbol{\xi}_{t,t+H}\right) \tag{2.190}
$$

Following similar same steps to estimate b, we get:

$$\widehat{b}_n = x_n \sum_{h=0}^{H-1} (\gamma\lambda)^h \rho_{n,h}^H r_{n+h} \tag{2.191}$$

The estimate for C does not require importance sampling because its expectation only depends on d^ϕ. Hence, the sample estimate is given by $\widehat{C}_n = x_n x_n^T$ which completes the proof. □

2.G Proof of Lemma 2.5

From Remark 2.6 we know that (2.66b) implies that the consensus constraints are satisfied by ζ^o and hence for some θ^o and ω^o:

$$\zeta^o = \mathbb{1}_K \otimes [\theta^{oT}, \upsilon^{-1/2}\omega^{oT}]^T \tag{2.192}$$

Multiplying (2.66a) by $\mathcal{I}^T \triangleq \mathbb{1}_K^T \otimes I_{2M}$ we get the following condition:

$$0 = \mathcal{I}^T \left(\bar{\mathcal{L}}\left(\mathcal{G}\zeta^o - p\right) + \mu^{-1}\mathcal{V}\mathcal{Y}^o\right) \overset{(a)}{=} \mathcal{I}^T\left(\mathcal{G}\zeta^o - p\right) \tag{2.193}$$

where in (a) we used $\mathcal{I}^T\bar{\mathcal{L}} = \mathcal{I}^T$ and $\mathcal{I}^T\mathcal{V} = 0$. Combining (2.192) and (2.193) we get that θ^o and ω^o must satisfy:

$$\begin{bmatrix} \eta\widehat{U} & -\widehat{A}^T \\ \widehat{A} & \widehat{C} \end{bmatrix} \begin{bmatrix} \theta^o \\ \omega^o \end{bmatrix} = \begin{bmatrix} \eta\widehat{U}\theta_{\mathrm{p}} \\ b \end{bmatrix} \tag{2.194}$$

since the left hand-side matrix is invertible (because $\widehat{A}$ and $\widehat{C}$ are invertible) ζ^o is unique. Left multiplying by the inverse of such matrix we get (2.37). Therefore, we conclude that $\zeta^o = \mathbb{1}_K \otimes [\widehat{\theta}^{oT}\widehat{\omega}^{oT}]^T$. The fact that $\mathcal{I}^T\left(\mathcal{G}\zeta^o - p\right) = 0$ implies that $\mathcal{G}\zeta^o - p$ lies in the range space of $\mathcal{V}$ (because it is orthogonal to $\mathcal{I}$ which lies in the null space of $\mathcal{V}$), which in turn implies that a vector $\mathcal{Y}$ that satisfies (2.66a) exists. We conclude the proof by showing that there is unique $\mathcal{Y}^o$ that satisfies (2.66a) and lies in the range space of $\mathcal{V}$. We do this by

contradiction. Assume that there are two fixed points $(\zeta^o, \mathcal{Y}_1 = \mathcal{V}a_1)$ and $(\zeta^o, \mathcal{Y}_2 = \mathcal{V}a_2)$ such that $\mathcal{Y}_1 \neq \mathcal{Y}_2$, applying both to (2.66a) and subtracting the resulting equations we get:

$$\mathcal{V}^2(a_1 - a_2) = 0 \iff \mathcal{V}(a_1 - a_2) = 0 \implies \mathcal{Y}_1 = \mathcal{Y}_2 \tag{2.195}$$

which is a contradiction. This concludes the proof.

2.H Proof of Lemma 2.6

We start by doing the following matrix decomposition:

$$\begin{aligned}
\begin{bmatrix} \bar{L} & -V \\ V\bar{L} & \bar{L} \end{bmatrix} &\overset{(a)}{=} \begin{bmatrix} H(\frac{I_K+\Lambda}{2})H^T & -H\left(\frac{I_K-\Lambda}{2}\right)^{\frac{1}{2}}H^T \\ H\left(\frac{I_K+\Lambda}{2}\right)\left(\frac{I_K-\Lambda}{2}\right)^{\frac{1}{2}}H^T & H(\frac{I_K+\Lambda}{2})H^T \end{bmatrix} \\
&= \begin{bmatrix} H & 0 \\ 0 & H \end{bmatrix} \begin{bmatrix} \frac{I_K+\Lambda}{2} & -\left(\frac{I_K-\Lambda}{2}\right)^{\frac{1}{2}} \\ \left(\frac{I_K+\Lambda}{2}\right)\left(\frac{I_K-\Lambda}{2}\right)^{\frac{1}{2}} & \frac{I_K+\Lambda}{2} \end{bmatrix} \begin{bmatrix} H & 0 \\ 0 & H \end{bmatrix}^T \qquad (2.196) \\
&\overset{(b)}{=} H_e \mathrm{diag}\{E_k\}_{k=1}^K H_e^T \qquad (2.197)
\end{aligned}$$

where in (a) we used Remark 2.4 and the definition of V (2.39) and in (b) we rearranged the order of the eigenvectors and their corresponding eigenvalues through permutations to get the following matrices:

$$E_k = \begin{bmatrix} \frac{1+\lambda_k}{2} & -\left(\frac{1-\lambda_k}{2}\right)^{\frac{1}{2}} \\ \left(\frac{1+\lambda_k}{2}\right)\left(\frac{1-\lambda_k}{2}\right)^{\frac{1}{2}} & \frac{1+\lambda_k}{2} \end{bmatrix} \tag{2.198}$$

$$H_e = \begin{bmatrix} K^{-\frac{1}{2}}\mathbb{1}_K & 0 & h_2 & 0 & \cdots & h_K & 0 \\ 0 & K^{-\frac{1}{2}}\mathbb{1}_K & 0 & h_2 & \cdots & 0 & h_K \end{bmatrix} \tag{2.199}$$

where λ_k is the k-th eigenvalue of L and h_k its corresponding eigenvector. Note that $E_1 = I$. Moreover, the matrices E_k have two distinct eigenvalues given by $(1 + \lambda_k \pm \sqrt{1 - \lambda_k^2})/2$ and therefore we can diagonalize E_k using its Jordan Canonical Form as $D_k = Z_k^{-1} E_k Z_k$ for some Z_k. Therefore, defining $Z = \mathrm{diag}\{Z_k\}_{k=1}^K$ (where $Z_1 = I$) we arrive at the following

diagonalization:

$$\begin{bmatrix} \bar{L} & -V \\ V\bar{L} & \bar{L} \end{bmatrix} = H_e Z \text{diag}\{I, D_2, \cdots, D_K\} Z^{-1} H_e^T \tag{2.200}$$

We can extend this diagonalization to the network-wide matrix:

$$\begin{bmatrix} \bar{\mathcal{L}} & -\mathcal{V} \\ \mathcal{V}\bar{\mathcal{L}} & \bar{\mathcal{L}} \end{bmatrix} = \begin{bmatrix} \bar{L} & -V \\ V\bar{L} & \bar{L} \end{bmatrix} \otimes I = \underbrace{(H_e Z \otimes I)}_{\triangleq \mathcal{H}} \mathcal{D} \underbrace{(Z^{-1} H_e^T \otimes I)}_{=\mathcal{H}^{-1}} \tag{2.201}$$

$$\mathcal{D} = \text{diag}\{I, \mathcal{D}_1\}, \mathcal{D}_1 = \text{diag}\{D_k\}_{k=2}^K \otimes I \tag{2.202}$$

Expanding $\mathcal{H}$ and $\mathcal{H}^{-1}$ we get:

$$\mathcal{H} = \begin{bmatrix} K^{-\frac{1}{2}}\mathcal{I} & 0 & \mathcal{H}_u \\ 0 & K^{-\frac{1}{2}}\mathcal{I} & \mathcal{H}_d \end{bmatrix}, \mathcal{H}^{-1} = \begin{bmatrix} K^{-\frac{1}{2}}\mathcal{I} & 0 & \mathcal{H}_l \\ 0 & K^{-\frac{1}{2}}\mathcal{I} & \mathcal{H}_r \end{bmatrix}^T \tag{2.203}$$

where $\mathcal{H}_l, \mathcal{H}_r, \mathcal{H}_u, \mathcal{H}_d \in \mathbb{R}^{2KM \times 4KM-4M}$ are some constant matrices. Now we use this decomposition to transform recursion (2.68):

$$\underbrace{\mathcal{H}^{-1} \begin{bmatrix} \widetilde{\zeta}_{i+1} \\ \widetilde{\mathcal{Y}}_{i+1} \end{bmatrix}}_{\triangleq [\bar{x}_{i+1}^T, \tilde{x}_{i+1}^T, \hat{x}_{i+1}^T]^T} = \Big(\mathcal{D} - \mu \mathcal{H}^{-1} \underbrace{\begin{bmatrix} \bar{\mathcal{L}}\mathcal{G} & 0 \\ \mathcal{V}\bar{\mathcal{L}}\mathcal{G} & 0 \end{bmatrix}}_{\triangleq \mathcal{T}} \mathcal{H} \Big) \mathcal{H}^{-1} \begin{bmatrix} \widetilde{\zeta}_i \\ \widetilde{\mathcal{Y}}_i \end{bmatrix} \tag{2.204}$$

where $\bar{x}_i = K^{\frac{1}{2}} \mathcal{I}^T \widetilde{\zeta}_i$ and $\hat{x}_i = \mathcal{H}_l^T \widetilde{\zeta}_i + \mathcal{H}_r^T \widetilde{\mathcal{Y}}_i$. Using $\mathcal{H}$ we can further write $\widetilde{\zeta}_i = K^{\frac{1}{2}} \mathcal{I} \bar{x}_i + \mathcal{H}_u \hat{x}_i$ and $\widetilde{\mathcal{Y}}_i = \mathcal{H}_d \hat{x}_i$, from which we get the following bounds:

$$\|\widetilde{\zeta}_i\|^2 \leq \|\bar{x}_i\|^2 + \|\mathcal{H}_u\|^2 \|\hat{x}_i\|^2, \quad \|\widetilde{\mathcal{Y}}_i\|^2 \leq \|\mathcal{H}_d\|^2 \|\hat{x}_i\|^2 \tag{2.205}$$

Expanding $\mathcal{H}^{-1}\mathcal{T}\mathcal{H}$ we get:

$$\mathcal{H}^{-1}\mathcal{T}\mathcal{H} = \begin{bmatrix} K^{-1}\mathcal{I}^T\bar{\mathcal{L}}\mathcal{G}\mathcal{I} & 0 & K^{-\frac{1}{2}}\mathcal{I}^T\bar{\mathcal{L}}\mathcal{G}\mathcal{H}_u \\ K^{-\frac{1}{2}}\mathcal{I}^T\mathcal{V}\bar{\mathcal{L}}\mathcal{G}\mathcal{I} & 0 & K^{-\frac{1}{2}}\mathcal{I}^T\mathcal{V}\bar{\mathcal{L}}\mathcal{G}\mathcal{H}_u \\ K^{-\frac{1}{2}}(\mathcal{H}_l^T + \mathcal{H}_r^T\mathcal{V})\bar{\mathcal{L}}\mathcal{G}\mathcal{I} & 0 & (\mathcal{H}_l^T + \mathcal{H}_r^T\mathcal{V})\bar{\mathcal{L}}\mathcal{G}\mathcal{H}_u \end{bmatrix} \tag{2.206}$$

Since $\mathcal{I}^T\mathcal{V} = 0$ all elements in the mid row in the above equation are equal to zero and therefore $\tilde{x}_i = \tilde{x}_0$. Also since all the elements in the mid column are equal to zero, it turns out that $\bar{x}_i$ and $\widehat{x}_i$ are independent of $\tilde{x}_0$. Hence, we can disregard variable $\tilde{x}_i$ and arrive at the desired recursion, which completes the proof.

2.I Proof of Lemma 2.7

We start by expanding the top recursion of (2.69) and take the squared norm on both sides of the equation to get:

$$\begin{aligned} \|\check{x}_{i+1}\|^2 &= \| \left(I_{2M} - \mu\Lambda_G K^{-1}\right)\check{x}_i - \mu K^{-\frac{1}{2}} Z^{-1}\mathcal{I}^T\mathcal{G}\mathcal{H}_u\hat{x}_i\|^2 \\ &= \left\| \frac{t}{t}\left(I_{2M} - \mu\Lambda_G K^{-1}\right)\check{x}_i - \frac{(1-t)\mu Z^{-1}\mathcal{I}^T\mathcal{G}\mathcal{H}_u}{(1-t)\sqrt{K}}\hat{x}_i \right\|^2 \\ &\overset{(a)}{\leq} t^{-1}\left\|\left(I_{2M} - \mu\Lambda_G K^{-1}\right)\check{x}_i\right\|^2 + \mu^2\frac{\|Z^{-1}\mathcal{I}^T\mathcal{G}\mathcal{H}_u\hat{x}_i\|^2}{(1-t)K} \\ &\overset{(b)}{\leq} t^{-1}\left\|I_{2M} - \mu\Lambda_G K^{-1}\right\|^2\|\check{x}_i\|^2 + \frac{\mu^2\|Z^{-1}\mathcal{I}^T\mathcal{G}\mathcal{H}_u\|^2}{(1-t)K}\|\hat{x}_i^e\|^2 \end{aligned} \tag{2.207}$$

where we defined a scalar $t \in (0,1)$, in (a) we used Jensen's inequality and in (b) we used the Cauchy-Schwarz inequality. If $\mu < K/\rho(\Lambda_G)$ (where $\rho(\Lambda_G)$ is the spectral radius of Λ_G) then $\left\|I_{2M} - \mu\Lambda_G K^{-1}\right\| < 1$ and hence setting $t = \left\|I_{2M} - \mu\Lambda_G K^{-1}\right\|$ we get:

$$\|\check{x}_{i+1}\|^2 \leq \rho(I_{2M} - \mu K^{-1}\Lambda_G)\|\check{x}_i\|^2 + \mu a_2\|\hat{x}_i\|^2 \tag{2.208}$$

where we defined $a_2 = \|Z^{-1}\mathcal{I}^T\mathcal{G}\mathcal{H}_u\|^2/\lambda_{\min}(\Lambda_G)$. We now repeat the procedure for the bottom recursion of (2.69):

$$\begin{aligned}\|\hat{x}_{i+1}\|^2 &\overset{(d)}{\leq} \frac{2\mu^2}{(1-t_2)K}\|(\mathcal{H}_l^T+\mathcal{H}_r^T\mathcal{V})\bar{\mathcal{L}}\mathcal{G}\mathcal{I}Z\|^2\|\breve{x}_i^e\|^2 + \frac{\|\mathcal{D}_1\|^2}{t_2}\|\hat{x}_i^e\|^2 \\ &\quad + 2(1-t_2)^{-1}\mu^2\|(\mathcal{H}_l^T+\mathcal{H}_r^T\mathcal{V})\bar{\mathcal{L}}\mathcal{G}\mathcal{H}_u\|^2\|\hat{x}_i^e\|^2 \\ &\overset{(f)}{\leq} \mu^2 a_3\|\breve{x}_i^e\|^2 + \sqrt{\lambda_2(\bar{L})}\|\hat{x}_i^e\|^2 + \mu^2 a_4\|\hat{x}_i^e\|^2 \end{aligned} \tag{2.209}$$

where in (d) we used Jensen's and the Cauchy-Schwarz inequalities and we also introduced $t_2 \in (0,1)$ and in (f) we chose $t_2 = \|\mathcal{D}_1\| = \sqrt{\lambda_2(\bar{L})}$. We further defined:

$$a_3 = \frac{2\|(\mathcal{H}_l^T+\mathcal{H}_r^T\mathcal{V})\bar{\mathcal{L}}\mathcal{G}\mathcal{I}Z\|^2}{1-\sqrt{\lambda_2(\bar{L})}} \tag{2.210}$$

$$a_4 = \frac{2\|(\mathcal{H}_l^T+\mathcal{H}_r^T\mathcal{V})\bar{\mathcal{L}}\mathcal{G}\mathcal{H}_u\|^2}{1-\sqrt{\lambda_2(\bar{L})}} \tag{2.211}$$

Writing (2.208) and (2.209) in matrix form completes the proof.

CHAPTER 3

Distributed Optimal Policy Learning in MARL

In the previous chapter we addressed the policy evaluation problem in the MARL setting. In this chapter we consider the problem of finding the optimal team policy in a fully distributed manner in the same MARL setting. The situation under consideration is one in which all agents form a team (to solve a common objective) and make decisions independently of one another based on their local information with the aim of maximizing the team's performance. The contribution of this chapter is the introduction of *Diffusion Team Policy Optimization (DTPO)*, a fully distributed algorithm capable of converging to the same team policy as its centralized counterpart. DTPO belongs to the family of Soft RL algorithms we briefly discussed in Section 1.2.5 of Chapter 1.

3.1 Related Works

There is a considerable body of work in MARL [36,61,79–95]. However, most of these works only consider the case where the reward is global and observed by all agents (i.e. they do not consider the case where agents observe local rewards). To the best of our knowledge, the only work that considers the case with local rewards is [61]. In [61] the authors extend the policy gradient theorem [14] to the MARL case we study and derive to actor-critic algorithms. These schemes work on-policy and therefore are not data inefficient. More critically, the algorithms rely on a naive factorization of the team's policy and therefore are only guaranteed to converge to Nash equilibrium policies that can be highly suboptimal in environments where coordination is necessary, (we illustrate this in section 3.4). *DTPO* works off-policy and is therefore more sample efficient than the actor-critic algorithms from [61]. Furthermore,

DTPO does not rely on such naive factorization and hence does not converge to suboptimal Nash equilibria. In appendix 3.A we provide a brief introduction to Nash equilibria along with a discussion of why such equilibria can be undesirable in the context of cooperative MARL.

3.2 Problem Setting

We address the cooperative MARL problem we introduce in section 1.3.1 of Chapter 1. We recall that in this scenario we consider a team of K agents that form a network. The network is represented by a graph in which the edges represent the communication links. Agent k communicates only with a subset of the agents in the network which we refer to as the neighborhood of k (denoted as $\mathcal{N}_k$). The topology of the network is determined by some combination matrix L whose kn-th entry (denoted by ℓ_{kn}) is a scalar with which agent n weights information incoming from agent k (note that $\ell_{kn} \neq 0 \iff k \in \mathcal{N}_n$). We make the following assumption about the network structure (which will be used in Lemma 3.2).

Assumption 3.1. *We assume that the network is strongly-connected. This implies that there is at least one path from any node to any other node and that at least one node has a self-loop (i.e., at least one agent uses its own information). We further assume that the combination matrix L is symmetric and doubly-stochastic.*

The agents interact with an environment and with each other, we model their behavior as a MA-MDP, which is defined by the tuple $(K,\mathcal{S},\mathcal{A}^k,\mathcal{P},r^k)$. The goal of all agents is to maximize the aggregated return:

$$J(\pi) = \sum_{t=0}^{\infty} \frac{\gamma^t}{K} \left(\sum_{k=1}^{K} \mathbb{E}_{\pi,\mathcal{P},d,n} \left[\boldsymbol{r}^k(\boldsymbol{s_t}, \bar{\boldsymbol{a}}_{\boldsymbol{t}}, \boldsymbol{s_{t+1}}) \right] \right) \tag{3.1}$$

Accordingly, the team's value function is given by:

$$v^{\pi}(s) = \mathbb{E}_{\pi} \left[\frac{1}{K} \sum_{k=1}^{K} r^k(s, \bar{\boldsymbol{a}}) + \gamma \mathbb{E}_{\boldsymbol{s'} \sim \mathcal{P}} v^{\pi}(\boldsymbol{s'}) \right] \tag{3.2}$$

where we defined $r^k(s,\bar{a}) = \mathbb{E}_{\mathcal{P},n}\boldsymbol{r}^k(s,\bar{a},\boldsymbol{s'})$. The team's optimal value function and optimal policies[1] satisfy:

$$\pi^\dagger(\bar{a}|s) = \arg\max_{\pi(\bar{a}|s)} \mathbb{E}_{\bar{\boldsymbol{a}}\sim\pi}\left[\frac{1}{K}\sum_{k=1}^{K} r^k(s,\bar{\boldsymbol{a}}) + \gamma\mathbb{E}_{\boldsymbol{s'}\sim\mathcal{P}} v^\dagger(\boldsymbol{s'})\right] \tag{3.3a}$$

$$v^\dagger(s) = \max_{\pi(\bar{a}|s)} \mathbb{E}_{\bar{\boldsymbol{a}}\sim\pi}\left[\frac{1}{K}\sum_{k=1}^{K} r^k(s,\bar{\boldsymbol{a}}) + \gamma\mathbb{E}_{\boldsymbol{s'}\sim\mathcal{P}} v^\dagger(\boldsymbol{s'})\right] \tag{3.3b}$$

The max operator in equations (3.3) is inconvenient because it is non-differentiable and in this chapter we are interested in deriving gradient algorithms. To circumvent this issue we use a differentiable approximation to the max and define the quasi-optimal value function as:

$$v^\star_\lambda(s) = \lambda(s)\log\left\{\sum_{\bar{a}}\exp\left(\frac{K^{-1}\sum_{k=1}^{K} r^k(s,\bar{a}) + \gamma\mathbb{E}_{\boldsymbol{s'}\sim\mathcal{P}} v^\star_\lambda(\boldsymbol{s'})}{\lambda(s)}\right)\right\} \tag{3.4}$$

where $\lambda(s) > 0$ is a temperature parameter that controls the accuracy of the approximation. Note that $v^\star_\lambda(s)$ in (3.4) can equivalently be defined using the conjugate of the log-sum-exp function as [96]:

$$v^\star_\lambda(s) = \max_{\pi(\bar{a}|s)} \mathbb{E}_{\bar{\boldsymbol{a}}\sim\pi}\left[\sum_{k=1}^{K}\frac{r^k(s,\bar{\boldsymbol{a}})}{K} - \lambda(s)\log\pi(\bar{\boldsymbol{a}}|s) + \gamma\mathbb{E}_{\boldsymbol{s'}\sim\mathcal{P}} v^\star_\lambda(\boldsymbol{s'})\right] \tag{3.5}$$

The max in (3.5) can easily be solved by differentiating and equating to zero. Doing so gives the policy corresponding to (3.5), which we refer to as the quasi-optimal policy:

$$\pi^\star_\lambda(\bar{a}|s) = \frac{\exp\left[\lambda(s)^{-1}\left(K^{-1}\sum_{k=1}^{K} r^k(s,\bar{a}) + \gamma\mathbb{E}_{\boldsymbol{s'}\sim\mathcal{P}} v^\star_\lambda(\boldsymbol{s'})\right)\right]}{\sum_{\bar{a}}\exp\left[\lambda(s)^{-1}\left(K^{-1}\sum_{k=1}^{K} r^k(s,\bar{a}) + \gamma\mathbb{E}_{\boldsymbol{s'}\sim\mathcal{P}} v^\star_\lambda(\boldsymbol{s'})\right)\right]} \tag{3.6}$$

Remark 3.1. *Note that* $\lim_{\lambda(s)\to 0}\pi^\star_\lambda(\bar{a}|s) = \pi^\bullet(\bar{a}|s)$ *and* $\lim_{\lambda(s)\to\infty}\pi^\star_\lambda(\bar{a}|s) = 1/A$.

We defined $\lambda(s)$ as a function of the state because it provides an effective way of trading

[1]Note that an MDP may have many optimal policies; at least one of which is deterministic and chooses an optimal action with probability one. Without loss of generality, we assume that there is a unique optimal policy.

exploration and exploitation [97]. The parameter $\lambda(s)$ can be initialized at high values to encourage exploration (since for high values of $\lambda(s)$ expression (3.6) tends to the uniform distribution) in unexplored states and slowly decays as those states are explored. As a consequence of the fact that $\pi^{\star}_{\lambda}(\bar{a}|s)$ is a continuous function of $\lambda(s)$, a corollary of remark 3.1 is that for small enough $\lambda(s)$ it holds that:

$$\arg\max_{\bar{a}} \pi^{\star}_{\lambda}(\bar{a}|s) = \arg\max_{\bar{a}} \pi^{\dagger}(\bar{a}|s) \tag{3.7}$$

Since our main goal is to obtain the optimal policy $\pi^{\dagger}(\bar{a}|s)$, expression 3.7 is very important. This is because the theorem guarantees that for small enough $\lambda(s)$, an optimal action at every state can be extracted from $\pi^{\star}_{\lambda}(\bar{a}|s)$ (and hence an optimal policy $\pi^{\dagger}(\bar{a}|s)$ can be extracted). The following lemma is an extension of [98] to the multi-agent setting:

Lemma 3.1. *$v^{\star}_{\lambda}(s)$ and $\pi^{\star}_{\lambda}(a|s)$ are the only pair that satisfy the following consistency equation:*

$$v(s) - \gamma \mathbb{E}_{\boldsymbol{s'} \sim \mathcal{P}} v(\boldsymbol{s'}) = \frac{1}{K} \sum_{k=1}^{K} r^{k}(s,a) - \lambda(s) \log \pi(a|s) \tag{3.8}$$

Using (3.8) we can write the following quadratic optimization problem (whose solution is $\pi^{\star}_{\lambda}(a|s)$):

$$\min_{\pi, v} \left(\frac{1}{K} \sum_{k=1}^{K} r^{k}(s,\bar{a}) - \lambda(s) \log \pi(\bar{a}|s) + \gamma \mathbb{E}_{\boldsymbol{s'} \sim \mathcal{P}} v(\boldsymbol{s'}) - v(s) \right)^{2} \tag{3.9}$$

Unfortunately (3.9) cannot be used to learn $\pi^{\star}_{\lambda}(\bar{a}|s)$ because $r^{k}(s,\bar{a})$ and the transition kernel $\mathcal{P}$ are unknown and we want to derive a stochastic algorithm that learns from interactions with the environment. Before we proceed to propose an alternative cost function that relies on samples we note one key feature of (1.2), which is that there is no expectation taken with respect to the policy of any of the agents (the only expectation is taken with respect to $\mathcal{P}$, which depends on the MDP) and therefore transitions obtained following *any* policy are useful to learn $\pi^{\star}_{\lambda}(\bar{a}|s)$ (this is what allows us to derive an off-policy algorithm). With

this in consideration we propose the following cost:

$$\min_{\pi,v} \frac{1}{2}\mathbb{E}_{(\boldsymbol{s},\bar{\boldsymbol{a}})\sim\psi}\left(\frac{1}{K}\sum_{k=1}^{K} r^k(\boldsymbol{s},\bar{\boldsymbol{a}}) - \lambda(\boldsymbol{s})\log\pi(\bar{\boldsymbol{a}}|\boldsymbol{s}) + \gamma\mathbb{E}_{\boldsymbol{s'}\sim\mathcal{P}} v(\boldsymbol{s'}) - v(\boldsymbol{s})\right)^2 \tag{3.10}$$

where the minimization variables π and v refer to $\pi(\bar{a}|s)$ and $v(s)$ for all states and actions and ψ is some distribution under which $(s, \bar{a})$ pairs are sampled. Distribution ψ is determined by two factors: the individual behavior policies of the agents under which data is collected and the experience selection strategy set for the replay buffers [99] of each of the agents. Replay buffers store past data collected by the agents which is later sampled by the learning algorithm and have been shown to be essential to achieve stable and fast learning [99], [5], [100]. The experience selection strategy refers to the strategy that decides what data samples are stored in the buffer (not all data can be stored due to the finite memory availability of the buffer), and the strategy with which data is sampled from the buffer for training. A uniformly sampled First In First Out (FIFO) buffer is a popular experience selection strategy. An important characteristic of ψ is that the behavior policies used by the agents for data collection are potentially independent of one another, the only requirement for these policies is that every possible transition in the MA-MDP has a positive probability of being sampled. This is a key feature for multi-agent learning because it means that data collection can be done in a fully distributed manner without any communication or synchronization requirements among the decision makers. Also different agents can have different replay buffer sizes with different experience selection strategies.

3.3 Algorithm Derivation

We now proceed to derive the algorithm. To derive a distributed algorithm in which each agent uses only local information, we let $v^k(s)$ and $\pi^k(\bar{a}|s)$ denotes the local copies of $v(s)$

and $\pi(\bar{a}|s)$ available at k and instead solve the equivalent problem:

$$\min_{\pi^k, v^k} \frac{1}{2}\mathbb{E}_{\psi}\left[\left(\sum_{k=1}^{K}\mathbb{E}_{\mathcal{P},n}\boldsymbol{r}^k(\boldsymbol{s},\bar{\boldsymbol{a}},\boldsymbol{s'}) + \gamma\mathbb{E}_{\mathcal{P}}v^k(\boldsymbol{s'}) - \lambda(\boldsymbol{s})\log\pi^k(\bar{\boldsymbol{a}}|\boldsymbol{s}) - v^k(\boldsymbol{s})\right)^2\right]$$
$$\text{s.t.} \qquad \pi^1 = \cdots = \pi^K \qquad v^1 = \cdots = v^K \tag{3.11}$$

We are interested in deriving a stochastic gradient algorithm that relies on samples. Problem (3.11) is not suitable for this because the gradient with respect to the variables that parameterize the value function is a product of expectations, and therefore a stochastic approximation of the gradient obtained with samples by removing such expectations would be biased. Notice that this is the same difficulty we had in the previous chapter, and we go around it in the same way, using the conjugate of the quadratic function. We thus obtain the following saddle-point formulation:

$$\min_{\pi^k, v^k} \mathbb{E}_{\psi}\left[\sum_{k=1}^{K}\max_{\rho}\rho(\boldsymbol{s},\bar{\boldsymbol{a}})\big(\mathbb{E}_{\mathcal{P},n}\boldsymbol{r}^k(\boldsymbol{s},\bar{\boldsymbol{a}},\boldsymbol{s'}) + \gamma\mathbb{E}_{\mathcal{P}}v^k(\boldsymbol{s'}) - \lambda(\boldsymbol{s})\log\pi^k(\bar{\boldsymbol{a}}|\boldsymbol{s}) - v^k(\boldsymbol{s})\big) - \frac{1}{2}\rho(\boldsymbol{s},\bar{\boldsymbol{a}})^2\right]$$
$$\text{s.t.} \qquad \pi^1 = \cdots = \pi^K \qquad v^1 = \cdots = v^K \tag{3.12}$$

Remark 3.2. *The order of the expectation and* max *operator in* (3.12) *can be interchanged.*

Proof. The lemma can easily be proved by solving for both cases and checking that both solutions are equal. □

Making use of remark 3.2 results in the following optimization problem:

$$\min_{\pi^k, v^k}\max_{\rho}\mathbb{E}_{\psi,\mathcal{P},n}\left[\sum_{k=1}^{K}\rho(\boldsymbol{s},\bar{\boldsymbol{a}})\big(\boldsymbol{r}^k(\boldsymbol{s},\bar{\boldsymbol{a}},\boldsymbol{s'}) + \gamma v^k(\boldsymbol{s'}) - \lambda(\boldsymbol{s})\log\pi^k(\bar{\boldsymbol{a}}|\boldsymbol{s}) - v^k(\boldsymbol{s})\big) - \frac{1}{2}\rho(\boldsymbol{s},\bar{\boldsymbol{a}})^2\right]$$
$$\text{s.t.} \qquad \pi^1 = \cdots = \pi^K \qquad v^1 = \cdots = v^K \tag{3.13}$$

Finally creating local copies ρ^k we can write (3.13) in the following equivalent form:

$$\min_{\pi^k, v^k} \max_{\rho^k} \sum_{k=1}^{K} \mathbb{E}_{\psi,\mathcal{P},n} \Big[\rho^k(\boldsymbol{s}, \bar{\boldsymbol{a}}) \big(\boldsymbol{r}^k(\boldsymbol{s}, \bar{\boldsymbol{a}}, \boldsymbol{s'}) + \gamma v^k(\boldsymbol{s'}) - \lambda(\boldsymbol{s}) \log \pi^k(\bar{\boldsymbol{a}}|\boldsymbol{s}) - v^k(\boldsymbol{s}) \big) - \frac{1}{2K} \rho^k(\boldsymbol{s}, \bar{\boldsymbol{a}})^2 \Big]$$

$$\text{s.t.} \qquad \pi^1 = \cdots = \pi^K \qquad v^1 = \cdots = v^K \qquad \rho^1 = \cdots = \rho^K \tag{3.14}$$

Note that up to this point no approximations have been made and hence problems (3.14) and (3.10) are equivalent in the sense that the value of the optimizing variables π^k and π are the same. A similar saddle-point problem to (3.13) appears in [59], however, [59] deals with a single-agent scenario and has a constant λ (i.e. $\lambda(s) = \lambda$).

Lemma 3.2. *We introduce the eigenvalue decomposition:*

$$0.5(I - L) = U \Sigma U^T \tag{3.15}$$

and define:

$$B \triangleq U \Sigma^{1/2} U^T \tag{3.16}$$

$$\bar{v}(s) \triangleq [v^1(s), \cdots, v^K(s)]^T \tag{3.17}$$

$$\bar{\rho}(s) \triangleq [\rho^1(s), \cdots, \rho^K(s)]^T \tag{3.18}$$

$$\bar{\pi}(\bar{a}|s) \triangleq [\pi^1(\bar{a}|s), \cdots, \pi^K(\bar{a}|s)]^T \tag{3.19}$$

where U is an orthogonal matrix and $\Sigma^{1/2}$ is the element-wise square root of Σ, which is a diagonal matrix with non-negative entries, then under Assumption 3.1 it holds that [70]:

$$B\bar{v}(s) = 0 \iff v^1(s) = \cdots = v^K(s) \tag{3.21}$$

$$B\bar{\rho}(s) = 0 \iff \rho^1(s) = \cdots = \rho^K(s) \tag{3.22}$$

$$B \log \bar{\pi}(\bar{a}|s) = 0 \iff \log \pi^1(\bar{a}|s) = \cdots = \log \pi^K(\bar{a}|s) \tag{3.23}$$

Finally, we approximate (3.13) by the following penalized formulation (for convenience

we write it in vector form):

$$\min_{\pi^k, v^k} \max_{\rho^k} S(\pi^k, v^k, \rho^k) + \eta_v \|B\bar{v}(\boldsymbol{s})\|^2 + \eta_\pi \|B \log \bar{\pi}(\bar{\boldsymbol{a}}|\boldsymbol{s})\|^2 - \eta_\rho \|B\bar{\rho}(\boldsymbol{s}, \bar{\boldsymbol{a}})\|^2$$
$$S(\pi^k, v^k, \rho^k) = \mathbb{E}\left(\bar{\rho}(\boldsymbol{s}, \bar{\boldsymbol{a}})^T \big(\bar{r}(\boldsymbol{s}, \bar{\boldsymbol{a}}, \boldsymbol{s'}) + \gamma \bar{v}(\boldsymbol{s'}) - \lambda(\boldsymbol{s}) \log \bar{\pi}(\bar{\boldsymbol{a}}|\boldsymbol{s}) - \bar{v}(\boldsymbol{s})\big) - \frac{\|\bar{\rho}(\boldsymbol{s}, \bar{\boldsymbol{a}})\|^2}{2K}\right) \tag{3.24}$$

where η_π, η_ρ and η_v are non-negative constants, the matrix B, $\bar{v}(s)$, $\bar{\rho}(s, s)$ and $\bar{\pi}(a|s)$ are defined in Lemma 3.2. Using $2B^2 = I - L$ (from Lemma 3.2), choosing $\eta_\pi = \mu_\pi^{-1}$, $\eta_\rho = \mu_\rho^{-1}$ and $\eta_v = \mu_v^{-1}$ (where μ_π, μ_ρ and μ_v are two step-sizes that will be used in the update equations) and applying stochastic gradient descent (and ascent) updates (incremental updates to π and v), we get our proposed *DTPO* algorithm. In algorithm 3.1 we show the tabular version of the algorithm.

Algorithm 3.1 *Diffusion Team Policy Optimization* at node k (tabular version)

Initialize: $(v^{k,0}, \log \pi^{k,0}, \rho^{k,0})$, $\lambda(s)$ for all s and en empty replay buffer $\mathcal{R}_k$.
for $e = 0, 1, 2 \ldots$: **do**
  Sample T transitions $(s, \bar{a}, r^k, s')$ by following some behavior policy and store them in $\mathcal{R}_k$.
  Anneal $\lambda(s)$.
  for $i = 0, 1, 2 \ldots$: **do**
    Sample a transition $(s, \bar{a}, r^k, s')$ from the replay buffer.
    $\phi_{\rho,k}^{e+1}(s, \bar{a}) = \rho^{k,e}(s, \bar{a}) + \mu_\rho \left(r^k(s, \bar{a}, s') + \gamma v^k(s') - v^k(s) - \lambda(s) \log \pi^k(\bar{a}|s) - \frac{\rho^k(s, \bar{a})}{K}\right)$
    $\phi_{\pi,k}^{e+1}(\bar{a}|s) = \log \pi^{k,e}(\bar{a}|s) + \mu_\pi \lambda(s) \rho^{k,e+1}(s, \bar{a})$
    $\phi_{v,k}^{e+1}(s) = v^{k,e}(s) + \mu_v \rho^{k,e+1}(s, \bar{a})(1 - \gamma \mathbb{I}(s' = s))$
    $\rho^{k,e+1}(s, a) = \sum_{n \in \mathcal{N}_k} \ell_{nk} \phi_{\rho,n}^{e+1}(s, \bar{a})$
    $\log \pi^{k,e+1}(\bar{a}|s) = \sum_{n \in \mathcal{N}_k} \ell_{nk} \phi_{\pi,n}^{e+1}(\bar{a}|s)$
    $v^{k,e+1}(s) = \sum_{n \in \mathcal{N}_k} \ell_{nk} \phi_{v,n}^{e+1}(s)$
  end for
end for

In practical applications, it is often necessary to parameterize functions $v^k(s)$, $\rho^k(s, \bar{a})$ and $\pi^k(\bar{a}|s)$. Furthermore, if this is the case then $\lambda(s)$ would also require to be parameterized.

The question is then how to train the parameters of λ to guarantee a steady annealing. One simple way is to make $\lambda(s)$ independent of s (i.e. $\lambda(s) = \lambda$), which is what we use in this chapter. We will analyze a more sophisticated approach in chapter 5. Parameterizing $v^k(s)$, $\log \pi^k(\bar{a}|s)$ and $\rho^k(s, \bar{a})$ with ω^k, θ^k and ζ^k, respectively, function $S(\theta^k, v^k, \rho^k)$ becomes:

$$S(\theta^k, \omega^k, \zeta^k) = \mathbb{E}\left(\bar{\rho}(\boldsymbol{s}, \bar{\boldsymbol{a}}; \zeta)^T \big(\bar{\boldsymbol{r}}(\boldsymbol{s}, \bar{\boldsymbol{a}}, \boldsymbol{s'}) + \gamma \bar{v}(\boldsymbol{s'}; \omega) - \lambda(\boldsymbol{s}) \log \bar{\pi}(\bar{\boldsymbol{a}}|\boldsymbol{s}; \theta) - \bar{v}(\boldsymbol{s}; \omega)\big) - \frac{\|\bar{\rho}(\boldsymbol{s}, \bar{\boldsymbol{a}}; \zeta)\|^2}{2K}\right) \tag{3.26}$$

where $\bar{\rho}(s, \bar{a}; \zeta) \stackrel{\Delta}{=} [\rho^1(s, \bar{a}; \zeta^k), \cdots, \rho^K(s, \bar{a}; \zeta^K)]^T$ and similarly for $\bar{v}(\boldsymbol{s'}; \omega)$ and $\bar{\pi}(\bar{\boldsymbol{a}}|\boldsymbol{s}; \theta)$. In algorithm 3.2 we list the version of *DTPO* with function approximation. We note that in both listings of *DTPO* we used a constant step-size and a single transition for each update (i.e. a mini-batch with only one sample), however this can be replaced with decaying step-sizes or other schemes such as *Adam* [101] and mini-batches with more than one sample.

Algorithm 3.2 *DTPO* with function approximation at node k

Initialize: $(\omega^{k,0}, \theta^{k,0}, \zeta^{k,0})$, λ and en empty replay buffer $\mathcal{R}_k$.
for $e = 0, 1, 2 \ldots$: **do**
  Sample T transitions $(s, \bar{a}, r^k, s')$ by following some behavior policy and store them in $\mathcal{R}_k$.
  Anneal λ.
  for $i = 0, 1, 2 \ldots$: **do**
    Sample a transition $(s, \bar{a}, r^k, s')$ from the replay buffer.

$$\begin{aligned}
\phi_{\zeta,k}^{e+1} &= \zeta^{k,e} + \mu_\rho \big(r^k(s, \bar{a}, s') + \gamma v^k(s'; \omega^k) - v^k(s; \omega^k) - \lambda \log \pi^k(\bar{a}|s; \theta^k) \\
&\qquad - K^{-1} \rho^k(s, \bar{a}; \zeta^k)\big) \nabla_{\zeta^k} \rho^{k,e+1}(s, \bar{a}; \zeta^k) \\
\phi_{\theta,k}^{e+1} &= \theta^{k,e} + \mu_\pi \lambda \rho^{k,e+1}(s, \bar{a}; \zeta^k) \nabla_{\theta^k} \log \pi^{k,e+1}(\bar{a}|s; \zeta^k) \\
\phi_{\omega,k}^{e+1} &= \omega^{k,e} + \mu_v \rho^{k,e+1}(s, \bar{a}; \zeta^k) \nabla_{\omega^k} \left(v^{k,e+1}(s; \omega^k) - \gamma v^{k,e+1}(s'; \omega^k)\right) \\
\zeta^{k,e+1} &= \sum_{n \in \mathcal{N}_k} \ell_{nk} \phi_{\zeta,n}^{e+1} \\
\theta^{k,e+1} &= \sum_{n \in \mathcal{N}_k} \ell_{nk} \phi_{\theta,n}^{e+1} \\
\omega^{k,e+1} &= \sum_{n \in \mathcal{N}_k} \ell_{nk} \phi_{\omega,n}^{e+1}
\end{aligned}$$

  end for
end for

3.4 Experiments

In this section we test the performance of DTPO and compare with [61, Algorithm 1], which we refer to as *Distributed Actor-critic* (DAC). Note that DAC is the only algorithm mentioned in the introduction suitable for this problem.

We consider a simple yet challenging team game. In this game, there are two agents (dog and monkey) who want to collect food (steak and banana) as fast as possible with as little movements as possible; to achieve this objective, they need to safely cross a river, which can only be done by collaboration. Figure 1 (a) illustrates the situation. Note that there are two buttons which when pressed a bridge to safely cross the river appears. In the optimal strategy, the dog presses the southern button for the monkey to cross the bridge which in turn presses the northern button so that the dog can cross (see Figure 1 (b)). Finally, each of the agents walks towards their respective food. At every time step, each of the agents has five possible actions: move north, east, south, west or stay (if an agent moves against an edge it just stays in place). When an agent falls into the river or collects its food it spawns back in the lower left corner. The reward structure for each of the agents is as follows: -1 is obtained at every time step, -0.5 every time the agent moves in any direction, -100 if it falls in the river and +50 if it collects its food (these rewards are additive, so if an agent moves and falls into the river it collects a rewards equal to -101.5). There are a total of 21 positions and 5 actions for each agent, hence the total state space for the team is $(21 \times 5)^2 = 11025$. The discount factor is set to $\gamma = 0.97$.

We implemented a game with only 2 agents because we wanted a game that could be implemented in tabular form (note that in this game, the state space grows exponentially with the number of agents, hence function approximation becomes necessary to reduce the dimensionality of the state space) so that the optimal policy would be attainable by the learning agents and could be calculated exactly to monitor the progress of the algorithm. Note that even with two agents, this game is extremely challenging to solve for fully distributed algorithms due to the fact that there is a highly suboptimal (yet easily attainable) Nash equilibrium, which is for both agents to stay still at all times in the lower left corner

(where they start the episode). Therefore, algorithms that only guarantee convergence to equilibrium point are extremely likely to converge to this very poor strategy (which is what happens with DAC).[2]

For the DAC implementation we used $\beta_\omega = 10^{-3}$ and $\beta_\theta = 10^{-4}$. The hyper-parameters of the DTPO implementation are as follows. We set $\mu_\rho = 1$, $\mu_V = 1$, $\mu_\pi = 0.01$ and $\lambda(s) = 20$ for every state and subtracted 0.05 every time the state was visited until $\lambda(s) = 0.5$. Since there are two agents, all combination weights are given by $\ell_{nk} = 0.5$. The constants T and I in Algorithm 1 were set to 1200 and 1 respectively. In our experience selection strategy transitions were included in the replay buffer if the state-action was selected for the first time and the replay buffers of all the agents were big enough to hold all these transitions.

The results of DTPO and DAC are shown in Figures 1 (c) and (d). The former shows the performance of the greedy policy with respect to the learned one for each of the algorithms for 12 time-steps (enough to collect the food). Figure 1 (d) shows the mean square difference between (3.6) (for $\lambda(s) = 0.5$ for all states) and $\pi_1(a|s)$. Note that DTPO learns the optimal strategy while DAC converges to the suboptimal Nash equilibrium. A video showing the evolution of the policy during learning is available online[3].

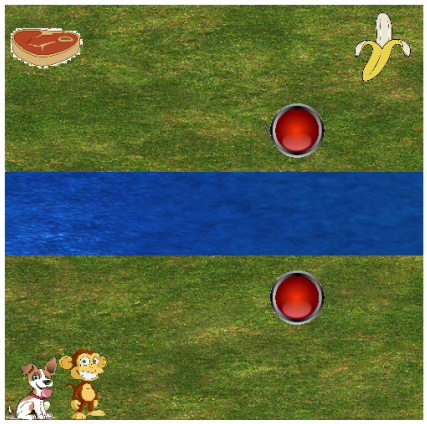
3.5 Summary

In this chapter we introduced the DTPO algorithm. A fully decentralized MARL algorithm for optimal policy learning. The algorithm is of the off-policy kind. We identify two important limitations with the *DTPO* algorithm.

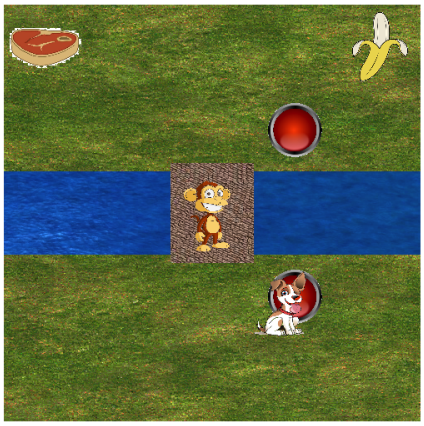
1) Note that since it is an off-policy algorithm, any behavior policy can be used to collect data (as long as it allocates strictly positive probability to all actions). This does

[2]We clarify that both agents staying still constitute a Nash equilibrium because if any of the two agents decides to modify its policy unilaterally it will be worst off. This is due to the fact that if any agent moves, while the other one stays still, it will never be able to cross to the other side and and hence, will only incur in the negative rewards associated with movement. Therefore, this unilateral deviation would result in lower cumulative rewards compared to staying still.

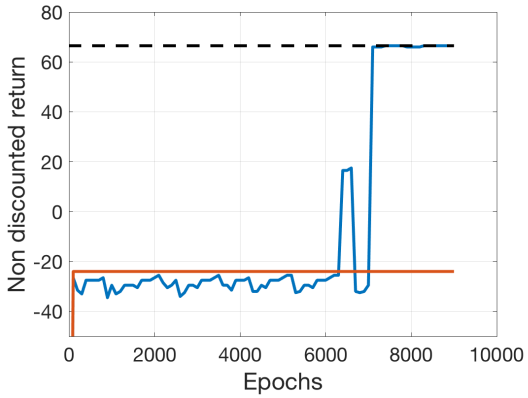
[3]Video available at EPFL's Adaptive Systems Lab website https://asl.epfl.ch/conferences/.



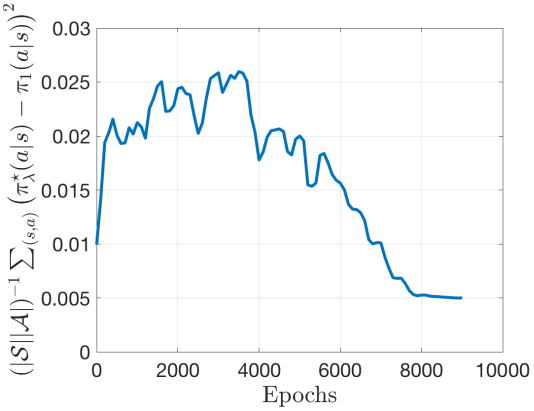
(a) Start state



(b)



(c)



(d)

Figure 3.1: In (c), the dashed line is the performance of the optimal policy, blue is DTPO and red is DAC.

not mean though, that all behavior policies are efficient at exploring the state space. For example a policy which selects actions uniformly satisfies the requirement but is obviously efficient. Consider how many tries it would take for agents following such policy to collect the food in the dog-monkey game. Therefore to efficiently explore the state-space of the game, the behavior policy has to actively search for states it hasn't visited before (this is typically referred to as *deep exploration*). Such behavior policy would need to have two main characteristics: i- it should have some mechanism which encourages discovery of new state; and ii- it should act more deterministically in states where it is confident of what the optimal action is and act more exploratory in states where it has more uncertainty about the optimal action. In the implementation of *DTPO* in this chapter we used the learned policy as the behavior policy. And we achieved the two previous conditions by: i- initializing the value

function for all states at values higher than $\max_s v(s)$ (therefore when it bootstraps from the value function of novel states the estimate is very high); and ii- slowly annealed $\lambda(s)$ as $\lambda(s) = \max(0.5; 20 - 0.05 \times \text{visitation\_count})$. However, this approach is a heuristic that is only useful for tabular implementations. The question then raises, is there a more principled way of obtaining a behavior policy that is efficient at performing exploration and that can be used with function approximation? In chapter 5 we address this question and provide such a behavior policy.

2) The other limitation is that every agent has to estimate team quantities. More specifically, each agent has to consider the global state and the actions of all agents. Note that this very quickly becomes computationally intractable since it grows exponentially with the number of agents. For instance, in the dog-monkey game the team's state space is given by $(21 \times 5)^2 = 11025$, if there were three agents then the team's state space would be $(21 \times 5)^3 = 1157625$. The question is then: is there a way where each agent learns only its policy π^k conditioned on its observation of the state o_s^k (i.e. $\pi^k(a^k|o_s^k)$) while still learning optimal team policies? In the next chapter we address this issue and provide one such algorithm.

3.A Nash Equilibria

In stochastic games, agents' strategies constitute Nash equilibria when no agent can benefit by unilaterally modifying its strategy. In cooperative MARL all agents optimize the same cost function and, hence, for this particular case it is equivalent to saying that agents' policies constitute Nash equilibria when the resulting joint team policy cannot be improved by modifying any single agent's policy unilaterally. Mathematically, agents' policies $\pi^k_{\text{nash}} \in [1, \cdots, K]$ constitute a Nash equilibrium if

$$J(\pi^1_{\text{nash}}, \cdots, \pi^K_{\text{nash}}) \geq \max_{\pi^k \neq \pi^k_{\text{nash}}} J(\pi^k, \pi^{-k}_{\text{nash}}), \quad \forall k \in [1, \cdots, K] \tag{3.28}$$

The equilibrium is said to be *strict* if the inequality in (3.28) is strict, otherwise it is said to be *weak*. We clarify that team optimum policies constitute, by definition, a Nash equilibrium. However, team policies that constitute a Nash equilibrium can have poor performance. We show this with a simple example.

Consider the matrix game with two homogeneous agents, each of which has two actions ($\mathcal{A} = \{\alpha; \beta\}$) and the following reward structure:

Table 3.1: Reward structure

		Agent 2	
		α	β
Agent 1	α	-5	-10
	β	-10	1

For this game the optimal team policy, which obtains the $+1$ reward, is given by:

$$\pi^{\dagger}(a^1, a^2) = \begin{cases} 1, & \text{if } \ a^1 = \beta \wedge a^2 = \beta \\ 0, & \text{otherwise} \end{cases} \tag{3.29}$$

Team policy $\pi^{\dagger}(a^1, a^2)$ can be written in terms of the individual policies $\pi^{\dagger,1}(a) = \pi^{\dagger,2}(a) = \mathbb{I}(a = \beta)$, where $\mathbb{I}$ is the indicator function. Indeed, policies $\pi^{\dagger,1}$ and $\pi^{\dagger,2}$ constitute a Nash equilibrium. Note, however, that the following sub-optimal team policy that obtains the -5 reward (which results from the individual policies $\pi^1(a) = \pi^2(a) = \mathbb{I}(a = \alpha)$) also constitutes a Nash equilibrium:

$$\pi(a^1, a^2) = \begin{cases} 1, & \text{if } \ a^1 = \alpha \wedge a^2 = \alpha \\ 0, & \text{otherwise} \end{cases} \tag{3.30}$$

CHAPTER 4

Logical Team Q-learning

In this chapter we address the factorization issue we mentioned in the previous chapter. This problem is fundamental in MARL and is orthogonal to that of distributed rewards. Since in this chapter we are concerned with addressing the factorization issue, we will consider the simpler problem were the reward function is global. We consider two scenarios.

In the first scenario, the global state and all actions are visible to all agents. It is well known that in this scenario the team can be regarded as one single agent where the aggregate action consists of the joint actions by all agents [36]. The fundamental drawback of this approach is that the joint action space grows exponentially in the number of agents and the problem quickly becomes intractable [85,86]. Another important inconvenience with this approach is that it cannot cope with a changing number of agents (for example if the system is trained with 4 agents, it cannot be executed by a team of 5 agents; we expand on this point in a later section). One well-known and popular approach to solve these issues, is to consider each agent as an independent learner (IL) [12]. However, this approach has a number of issues. First, from the point of view of each IL, the environment is non-stationary (due to the changing policies of the other agents), which jeopardizes convergence. And second, replay buffers cannot be used due to the changing nature of the environment and therefore even in cases where this approach might work, the data efficiency of the algorithm is negatively affected. Ideally, it is desirable to derive an algorithm with the following features: i) it learns individual policies (and is therefore scalable), ii) local actions chosen greedily with respect to these individual policies result in an optimal team action iii) can be combined with NNs, iv) works off-policy and can leverage replay buffers (for data efficiency), v) and enjoys theoretical guarantees to team optimal policies at least in the dynamic programming

scenario. Indeed, the main contribution of this chapter is the introduction of *Logical Team Q-learning (LTQL)*, an algorithm that has all these properties. We start in the dynamic programing setting and derive equations that characterize the desired solution. We use these equations to define the *Factored Team Optimality Bellman Operator* and provide a Theorem that characterizes the convergence properties of this operator. A stochastic approximation of the dynamic programming setting is used to obtain the tabular and deep versions of our algorithm. For the single agent setting, these steps reduce to: the Bellman optimality equation, the Bellman optimality operator (and the theorem which states the linear convergence of repeated application of this operator) and Q-learning (in its tabular form and DQN).

In the second scenario, we consider the centralized training and decentralized execution paradigm. During execution, agents only have access to observations which we assume provide enough information to play an optimal team policy. An example of this case would be a soccer team in which the attackers have the ball and see each other but do not see the goalkeeper or the defenders of their own team (arguably this information is enough to play optimally and score a goal). The techniques we develop for the previous scenario can be applied to this case without modification.

4.1 Related Works

Some of the earliest works on MARL are [12, 13]. Independent Q-learning (IQL) was first studied by [12] where it was identified that IQL learners in a MARL setting may fail to converge due to the non-stationarity of the perceived environment. Later [13] compared the performance of IQL and joint action learners (JAL) where all agents learn the Q-values for all the joint actions, and identified the problem of coordination during decentralized execution when multiple optimal policies are available. A proof of convergence for JALs was provided by [36]. Recently, [94] did an experimental study of ILs using DQNS in the Atari game Pong. All these mentioned approaches cannot use experience replay due to the non-stationarity of the perceived environment. Following *Hyper Q-learning* [81], [95] addressed this issue to some extent using *fingerprints* as proxys to model other agents' strategies.

The *Distributed Q-learning (DistQ)* algorithm was introduced in [79], which in the tabular setting has guaranteed convergence to an optimal policy for deterministic MDPs. However, this algorithm performs very poorly in stochastic scenarios and becomes divergent when combined with function approximation. Later *Hysteretic Q-learning (HystQ)* was introduced in [83] to improve these two limitations. *HystQ* is based on a heuristic and can be thought of as a generalization of *DistQ*. These works also consider the scenario where agents cannot perceive the actions of other agents. They are related to *LTQL* (from this work) in that they can be considered approximations to our algorithm in the scenario where agents do not have information about other agents' actions. Recently [93] introduced *Dec-HDRQNs* for multi-task MARL, which combines *HystQ* with Recurrent NNs and experience replay (which they recognize is important to achieve high sample efficiency) through the use of *Concurrent Experience Replay Trajectories*.

OAB was introduced by [82], the first algorithm that converges to an optimal Nash equilibrium with probability one in any team Markov game. *OAB* considers the team scenario where agents observe the full state and joint actions. The main disadvantage of this algorithm is that it requires estimation of the transition kernel and rewards for the joint action state space and also relies on keeping count of state-action visitation, which makes it impractical for MDPs of even moderate size and cannot be combined with function approximators.

The idea of factoring the joint Q-function to handle the scalability issue was first introduced in [84–86]. These papers have the disadvantage that they require coordination graphs that specify how agents affect each other (the graphs require significant domain knowledge). The main shortcoming of these papers is the factoring model they use, in particular they model the optimal Q-function (which depends on the joint actions) as a sum of K local Q-functions (where K is the number of agents, and each Q-function considers only the action of its corresponding agent). The main issue with this factorization model is that the optimal Q-function cannot always be factored in this way, in fact, the tasks for which this model does not hold are typically the ones that require a high degree of coordination, which happen to be the tasks where one is most interested in applying specific MARL approaches as opposed to ILs. Moreover, even if the Q-function can be accurately modeled in this way, there is

no guarantee that if individual agents select their optimum strategies by maximizing their local Q-functions the resulting joint action maximizes the global Q-function. The approach we introduce in this chapter also considers learning factored Q-functions. However, the fundamental difference is that the factored relations we estimate always exist and the joint action that results from maximizing these individual Q-functions is optimal. *VDN* [91] and *QMIX* [92] are two recent *deep* methods that also factorize the optimal Q-function assuming additivity and monotonicity, respectively. This factoring is their main limitation since many MARL problems of interest do not satisfy any of these two assumptions. Indeed, [90] showed that these methods are unable to solve a simple matrix game. Furthermore, the individual policies cannot be used for prediction, since the individual Q values are not estimates of the return. To improve on the representation limitation due to the factoring assumption, [90] introduced *QTRAN* which factors the Q-function in a more general manner and therefore allows for a wider applicability. The main issue with *QTRAN* is that although it can approximate a wider class of Q-functions than *VDN* and *QMIX*, the algorithm resorts to other approximations, which degrade its performance in complex environments (see [89]). Recently, actor-critic strategies have been explored [87, 88]. However, these methods have the inconvenience that they are on-policy and therefore do not enjoy the data efficiency that off-policy methods can achieve. This is of significant importance in practical MARL settings since the state-action space is very large.

4.2 Problem Setting

As in the previous chapter we consider a situation where multiple agents form a team and interact with an environment and with each other. However, in this chapter we consider a slightly different formulation in which the reward function is global and furthermore we introduce the concept of type of agent and individual observations of the global state. We model our current setting as a Team Markov Decision Process (TMDP),[1] which we define by

[1]Prior works use definitions such as Dec-POMDP [102], Multi-agent MDPs (MAMDP) [79] or Team Markov Games [82]. However, these definitions are different from ours, which is why we opted for the alternative name of TMDP. In particular, TMDPs include the notion of *types* of agents.

the tuple $(\mathcal{S},\mathcal{T},K,o^\tau,\mathcal{A}^\tau,\mathcal{P},r)$. Here, $\mathcal{S}$ is a set of global states shared by all agents; $\mathcal{T}$ is the set of types of agents; K is the total amount of agents, each of type $\tau_k \in \mathcal{T}$; $o^\tau : \mathcal{S} \to \mathcal{O}^\tau$ is the observation function for agents of type $\tau \in \mathcal{T}$, whose output lies in some set of observations $\mathcal{O}^\tau$;[2] $\mathcal{A}^\tau$ is the set of actions available to agents of type τ; $\mathcal{P}(s'|s,\bar{a})$ specifies the probability of transitioning to state $s' \in \mathcal{S}$ from state $s \in \mathcal{S}$ having taken joint actions $a^k \in \mathcal{A}^{\tau_k}$; and $r : \mathcal{S} \times \bar{\mathcal{A}} \times \mathcal{S} \to \mathbb{R}$ is a global reward function, which can be a random variable following some distribution $n_{s,\bar{a},s'}(r)$. The goal of the team is to maximize the team's return:

$$J(\pi) = \sum_{t=0}^{\infty} \gamma^t \mathbb{E}_{\pi,\mathcal{P},d,n}\left[\boldsymbol{r}(\boldsymbol{s_t}, \bar{\boldsymbol{a}}_{\boldsymbol{t}}, \boldsymbol{s_{t+1}})\right] \tag{4.1}$$

Accordingly, for our current setting, the team's optimal state-action value function ($q^\dagger$) and optimal policy ($\pi^\dagger$) are given by:

$$\pi^\dagger(\bar{a}|s) = \arg\max_{\pi(\bar{a}|s)} \mathbb{E}_{\bar{\boldsymbol{a}}\sim\pi,\boldsymbol{s'}\sim\mathcal{P},\boldsymbol{r}\sim n}\left[\boldsymbol{r}(s,\bar{\boldsymbol{a}},\boldsymbol{s'}) + \gamma \max_{\bar{a}'} q^\dagger(\boldsymbol{s'},\bar{a}')\right] \tag{4.2a}$$

$$q^\dagger(s,\bar{a}) = \mathbb{E}_{\boldsymbol{s'}\sim\mathcal{P},\boldsymbol{r}\sim n}\left[\boldsymbol{r}(s,\bar{a},\boldsymbol{s'}) + \gamma \max_{\bar{a}'} q^\dagger(\boldsymbol{s'},\bar{a}')\right] \tag{4.2b}$$

As already mentioned, a team problem of this form can be addressed with any single-agent algorithm. The fundamental inconvenience with this approach is that the joint action space scales exponentially with the number of agents, more specifically $|\bar{\mathcal{A}}| = \prod_{k=1}^{K} |\mathcal{A}^{\tau_k}|$. Another problem with this approach is that the learned Q-function cannot be executed in a decentralized manner using the agents' observations. Furthermore, the learned quantities (value functions or policies) are useless if the number of agents changes. However, if factored policies are learned, then these could be executed by teams with different number of agents (as long as the extra agents are of the same "type" as the agents used for learning. In section 4.4 we provide one example of this scenario). For these reasons, in this chapter we concern ourselves with learning factored quantities.

Assumption 4.1. *We assume that if for two states s_1 and s_2 we have $o^{\tau_k}_{s_1} = o^{\tau_k}_{s_2}$, then $q^\dagger(s_1,a^k,a^{-k})|_{a^{-k}\sim\pi^\dagger(a^k,a^{-k}|s_1)} = q^\dagger(s_2,a^k,a^{-k})|_{a^{-k}\sim\pi^\dagger(a^k,a^{-k}|s_2)}$.*

[2] In other words, $o^{\tau_k}(s) = o^{\tau_k}_s$ is agent's k description of the global state s from its own perspective.

Assumption 4.2. *Agents of the same type are assumed to be homogeneous. Mathematically, if two agents n and k are homogeneous, then for every state s_1 there is another equivalent state s_2 such that:*

$$\left(o^{\ell}_{s_1} = o^{\ell}_{s_2} \forall \ell \neq (n,k)\right) \wedge \left(o^{k}_{s_2} = o^{n}_{s_1}\right) \wedge \left(o^{k}_{s_1} = o^{n}_{s_2}\right) \rightarrow q^{\pi}(s_1, \bar{a})|_{\substack{a^k=b \\ a^n=c}} = q^{\pi}(s_2, \bar{a})|_{\substack{a^k=c \\ a^n=b}} \quad (4.3)$$

In simple terms assumption 4.1 means that even though observations are not full descriptions of the state, they provide enough information to know the effect of individual actions assuming everybody else in the team acts optimally (intuitively this is a reasonable requirement if the agents are expected to be able to play a team optimum strategy using only their partial observations). Assumption 4.2 means that if two agents of the same type are swapped (while other agents remain unchanged), then the value functions of the corresponding states are equal independently of the policy being executed by the team (as long as the agents swap their corresponding policies as well).

4.3 Algorithm Derivation

4.3.1 Factored Bellman Relations and Dynamic Programming

Similarly to the way that relations 4.2 are used to derive Q-learning in the single agent setting, the goal of this section is to derive relations in the dynamic programming setting from which we can derive a MARL algorithm. The following two lemmas take the first steps in this direction.

Lemma 4.1. *All TMDPs that satisfy assumptions 4.1 and 4.2 have, for each deterministic*

team optimal policy, $|\mathcal{T}|$ *factored functions* $q^{\tau,\star} : \mathcal{O}^\tau \times \mathcal{A}^\tau \to \mathbb{R}$ *such that:*

$$\max_{\bar{a}} q^\dagger(s,\bar{a}) = \max_{a^1} q^{\tau_1,\star}(o_s^{\tau_1}, a^1) = \cdots = \max_{a^K} q^{\tau_K,\star}(o_s^{\tau_K}, a^K) \tag{4.4a}$$

$$\max_{\bar{a}} q^\dagger(s,\bar{a}) = q^\dagger\Big(s, \arg\max_{a^1} q^{\tau_1,\star}(o_s^{\tau_1}, a^1), \cdots, \arg\max_{a^K} q^{\tau_K,\star}(o_s^{\tau_K}, a^K)\Big) \tag{4.4b}$$

$$q^{\tau_k,\star}(o_s^{\tau_k}, a^k) = \mathbb{E}_{\mathcal{P},n}\big[\boldsymbol{r}(s, a^k, a^{-k}, \boldsymbol{s'}) + \gamma \max_{a',k} q^{\tau_k,\star}(\boldsymbol{o_{s'}^{\tau_k}}, a'^{,k})\big]\big|_{a^n = \arg\max_{a^n} q^{\tau_n,\star}(o_s^{\tau_n}, a^n) \forall n \neq k} \tag{4.4c}$$

Proof. See appendix 4.A. □

A simple interpretation of equation (4.4c) is that $q^{\tau_k,\star}(o_s^{\tau_k}, a^k)$ is the expected return starting from state s when agent k takes action a^k while the rest of the team acts in an optimal manner.

Lemma 4.2. *All TMDPs that satisfy assumptions 4.1 and 4.2 have at least one deterministic team optimal policy that can be factored into* $|\mathcal{T}|$ *deterministic policies* $\pi^{\tau,\star}(a|o)$*, where* $a \in \mathcal{A}^\tau$ *and* $o \in \mathcal{O}^\tau$*. Such factored deterministic policies can be obtained as follows:*

$$\pi^{\tau,\star}(a|o) = \begin{cases} 1, & \text{if} \quad a = \arg\max_a q^{\tau,\star}(o,a) \\ 0, & \text{else} \end{cases} \tag{4.5}$$

Proof. The proof simply follows from equations (4.4a) and (4.4b). □

Lemmas 4.2 and 4.2 are important because they show that if the agents learn factored functions that satisfy (4.4) and act greedily with respect to their corresponding $q^{\tau_k,\star}$, then the resulting team policy is guaranteed to be optimal and hence they are not subject to the coordination problem identified in [79][3] (we show this in section 4.4.1). Therefore, an algorithm that learns $q^{\tau_k,\star}$ would satisfy the first two of the five desired properties that were enumerated in the introduction. As a sanity check, note that for the case where there is only one agent, equation (4.4c) simplifies to the Bellman optimality equation. Furthermore,

[3]This problem arises in situations in which the TMDP has multiple deterministic team optimal policies and the agents learn factored functions of the form $\max_{a^{-k}} q^\dagger(s,a^k,a^{-k})$ (we remark that these are not the same as $q^{\tau_k,\star}(o_s^{\tau_k}, a^k)$).

Lemma 4.2 can be seen as an extension to the TMDP case of the well known result that states that every MDP has at least one deterministic optimal policy [22]. Although in the single agent case the Bellman optimality equation can be used to obtain $q^\dagger$ (applying repeatedly the operator of the same name), we cannot do the same with (4.4c). The fundamental reason for this is that the $|\mathcal{T}|$ functions $q^{\tau,\star}$ are not the only functions that satisfy relation (4.4c).

Remark 4.1. ***Sub-optimal Nash fixed points:*** *There may exist $\mathcal{T}$ functions q^τ such that* (4.4c) *is satisfied but* (4.4b) *is not.*

Proof. We prove this with an example. See appendix 4.B. □

Note that Remark 4.1 implies that relation (4.4c) is not sufficient to derive a learning algorithm capable of obtaining a team optimal policy because it can find sub-optimal Nash equilibria. To avoid this inconvenience, it is necessary to find another relation that is only satisfied by $q^\star$. We can obtain one such relation combining (4.4a) and (4.4c):

$$\max_{a^k} q^{\tau_k,\star}(o_s^{\tau_k}, a^k) = \max_{\bar{a}} \mathbb{E}_{\mathcal{P},n}\big[\boldsymbol{r}(s,\bar{a},\boldsymbol{s'}) + \gamma \max_{a',k} q^{\tau_k,\star}(\boldsymbol{o}_{\boldsymbol{s'}}^{\boldsymbol{k}}, a'^{,k})\big] \tag{4.6}$$

The sub-optimal Nash fixed points mentioned in Remark 4.1 do not satisfy relation (4.6) since by definition the right hand side is equal to $\max_{\bar{a}} q^\dagger(s,\bar{a})$. Intuitively, equation (4.6) is not satisfied by these suboptimal strategies because the $\max_{\bar{a}}$ considers all possible team actions (while Nash equilibria only consider unilateral deviations).

Definition 4.1. *Combining equations* (4.4c) *and* (4.6)*, we define the Factored Team Optimality Bellman operator $\mathcal{B}^\psi$ as follows:*

$$\mathcal{B}^\psi q^{\tau_k}(o_s^{\tau_k}, a^k) = \big[\mathbb{I}(c_1)\mathcal{B}_{a^{-k}} q^{\tau_k}(o_s^{\tau_k}, a^k) + \mathbb{I}(\bar{c}_1)\max\big\{q^{\tau_k}(o_s^{\tau_k}, a^k), \mathcal{B}_{a^{-k}} q^{\tau_k}(o_s^{\tau_k}, a^k)\big\}\big]\big|_{a^{-k}\sim\psi} \tag{4.7}$$

$$c_1 \triangleq a^n == \arg\max_{a^n} q^{\tau_n}(o_s^{\tau_n}, a^n) \quad \forall n \neq k \tag{4.8}$$

$$\mathcal{B}_{a^{-k}} q^{\tau_k}(o_s^{\tau_k}, a^k) = \mathbb{E}_{\mathcal{P},n}\big(\boldsymbol{r}(s, a^k, a^{-k}, \boldsymbol{s'}) + \gamma \max_{a'} q^{\tau_k}(\boldsymbol{o}_{\boldsymbol{s'}}^{\boldsymbol{\tau_k}}, a')\big)\big|_{a^{-k}} \tag{4.9}$$

where $\mathbb{I}$ is the indicator function, c_1 is a Boolean variable, $\bar{c}_1$ is the logical not operator applied on c_1, and ψ is some distribution that assigns strictly positive probability to every a^{-k}. Note

that operator $\mathcal{B}^{\psi}$ is stochastic: every time it is applied to q^{τ_k}, a^{-k} is sampled according to ψ.

A simple interpretation of operator $\mathcal{B}^{\psi}$ is the following. Consider a basketball game, in which player α has the ball and passes the ball to teammate β. If β gets distracted, misses the ball and the opposing team ends up scoring, should α learn from this experience and modify its policy to not pass the ball? The answer is no, since the poor outcome was player β's fault. In plain English, from the point of view of some player k, what the first term of (4.7) means is *"I will only learn from experiences in which my teammates acted according to what I think is the optimal team strategy"*. It is easy to see why this kind of stubborn rationale cannot escape Nash equilibria (i.e., agents do not learn when the team deviates from its current best strategy, which obviously is a necessary condition to learn better strategies). The interpretation of the full operator $\mathcal{B}^{\psi}$ is *"I will learn from experiences in which: a) my teammates acted according to what I think is the optimal team strategy; or b) my teammates deviated from what I believe is the optimal strategy and the outcome of such deviation was better than I expected if they had acted according to what I thought was optimal"*, which arguably is what a logical player would do (this is the origin of the algorithm's name).

Theorem 4.1. *Repeated application of the operator $\mathcal{B}^{\psi}$ to any initial $|\mathcal{T}|$ q^{τ}-functions converge to set $\mathcal{S}$ with probability one. Mathematically:*

$$\mathbb{P}\big(\lim_{N\to\infty}(\mathcal{B}^{\psi})^{N} q^{\tau_k}(o_s^{\tau_k}, a^k) \in \mathcal{S}\big) = 1 \tag{4.10}$$

$$\mathcal{S} = \Big\{ q^{\tau_k} \Big| q^{\tau_k,\star}(o_s^{\tau_k}, a^k) \leq q^{\tau}(o_s^{\tau_k}, a^k) \leq \max_{a^{-k}} q^{\dagger}(s, a^k, a^{-k}) \;\; \forall (\tau_k, o_s^{\tau_k}, a^k) \in (\mathcal{T}, \mathcal{O}^{\tau_k}, \mathcal{A}^{\tau_k}) \Big\} \tag{4.11}$$

The mean convergence rate is exponential with constant lower bounded by γ^p, where p is the lowest probability assigned to any a^{-k} by ψ (i.e. $p = \arg\min_{a^{-k}} \mathbb{P}_{\psi}(a^{-k})$).

Proof. See appendix 4.C. □

As a sanity check, notice that in the single agent case operator $\mathcal{B}^{\psi}$ reduces to the Bellman optimality operator and Theorem 4.1 reduces to the well known result that repeated application of the Bellman optimality operator to any initial Q-function converges at an exponential rate (with constant γ) to $q^{\dagger}$.

4.3.2 Reinforcement Learning Setting

In this section we present *LTQL* (see algorithm 4.1), which we obtain as a stochastic approximation to operator $\mathcal{B}^{\psi}$. Note that the algorithm utilizes two q estimates for each type τ, a biased one parameterized by θ^{τ} (which we denote $q_{\theta^{\tau}}$) and an unbiased one parameterized by ω^{τ} (which we denote $q_{\omega^{\tau}}$). We clarify that in the listing of algorithm 4.1 we used a constant step-size, however this can be replaced with decaying step-sizes or other schemes such as *AdaGrad* [103] and *Adam* [101]. Note that the target of the unbiased network is used to calculate the target values for both functions; this prevents the bias in the estimates $q_{\theta^{\tau}}$ (which arises due to the c_2 condition) from propagating through bootstrapping. The target parameters of the biased estimates (θ^{τ}_{T}) are used solely to evaluate condition c_1. We have found that this stabilizes the training of the networks, as opposed to just using θ^{τ}. Hyperparameter α weights samples that satisfy condition c_2 ($(r + \max_a q_{\theta^{\tau_k}_{T}}(o^{\tau_k}_{s'}, a) > q_{\theta^{\tau_k}}(o^{\tau_k}_{s}, a^k))$ differently from those who satisfy c_1. Intuitively, since the purpose of condition c_2 is to escape Nash equilibria, α should be chosen as small as possible as long as the algorithm doesn't get stuck in such equilibria. As we remarked in the introduction, *LTQL* reduces to *DQN* for the case where there is a unique agent. In appendix 4.F we include the tabular version of the algorithm along with a brief discussion.

Note that *LTQL* works off-policy and there is no necessity of synchronization for exploration. Therefore, it can be implemented in a fully decentralized manner as long as all agents have access to all observations (and therefore to the full state) and actions of other agents (so that they can evaluate c_1). Interestingly, if condition c_1 ($a^n = \arg\max_{a^n} q_{\theta^{\tau_k}_{T}}(o^{\tau_n}_{s}, a^n)\ \forall n \neq k$) was omitted (to eliminate the requirement that agents have access to all this information), the resulting algorithm is exactly *DistQ* [79]. However, as the proof of theorem 4.1 indicates, the resulting algorithm would only converge in situations where it could be guaranteed that during learning overestimation of the q values is not possible (i.e., the tabular setting applied to deterministic MDPs; this remark was already made in [79]). In the case where this condition could not be guaranteed (i.e., when using function approximation and/or stochastic MDPs), some mechanism to decrease overestimated q values would be necessary, as this is

Algorithm 4.1 *Logical Team Q-Learning*

Initialize: an empty replay buffer $\mathcal{R}$, parameters θ^τ and ω^τ and their corresponding targets θ_T^τ and ω_T^τ for all types $\tau \in \mathcal{T}$.
for iterations $e = 0, \dots, E$ **do**
  Sample T transitions $(o_s^{\tau_1}, \cdots, o_s^{\tau_K}, \bar{a}, r, o_{s'}^{\tau_1}, \cdots, o_{s'}^{\tau_K})$ by following some behavior policy which guarantees all joint actions are sampled with non-zero probability and store them in $\mathcal{R}$.
  for iterations $i = 0, \dots, I$ **do**
    Sample a mini-batch of B transitions $(o_s^{\tau_1}, \cdots, o_s^{\tau_K}, \bar{a}, r, o_{s'}^{\tau_1}, \cdots, o_{s'}^{\tau_K})$ from $\mathcal{R}$.
    Set $\Delta_{\theta^\tau} = 0$ and $\Delta_{\omega^\tau} = 0$ for all types τ.
    for each transition of the mini-batch $b = 1, \cdots, B$ and each agent $k = 1, \cdots, K$ **do**
      if $a^n = \arg\max_{a^n} q_{\theta_T^{\tau_k}}(o_s^{\tau_n}, a^n)\ \forall n \neq k$ **then**
        $\Delta_{\theta^{\tau_k}} = \Delta_{\theta^{\tau_k}} + \big(r + \max_a q_{\omega_T^{\tau_k}}(o_{s'}^{\tau_k}, a) - q_{\theta^{\tau_k}}(o_s^{\tau_k}, a^k)\big)\nabla_{\theta^{\tau_k}} q_{\theta^{\tau_k}}(o_s^{\tau_k}, a^k)$
        $\Delta_{\omega^{\tau_k}} = \Delta_{\omega^{\tau_k}} + \big(r + \max_a q_{\omega_T^{\tau_k}}(o_{s'}^{\tau_k}, a) - q_{\omega^{\tau_k}}(o_s^{\tau_k}, a^k)\big)\nabla_{\omega^{\tau_k}} q_{\omega^{\tau_k}}(o_s^{\tau_k}, a^k)$
      else if $\big(r + \max_a q_{\theta_T^{\tau_k}}(o_{s'}^{\tau_k}, a) > q_{\theta^{\tau_k}}(o_s^{\tau_k}, a^k)\big)$ **then**
        $\Delta_{\theta^{\tau_k}} = \Delta_{\theta^{\tau_k}} + \alpha\big(r + \max_a q_{\omega_T^{\tau_k}}(o_{s'}^{\tau_k}, a) - q_{\theta^{\tau_k}}(o_s^{\tau_k}, a^k)\big)\nabla_{\theta^{\tau_k}} q_{\theta^{\tau_k}}(o_s^{\tau_k}, a^k)$
      end if
    end for
    $\theta^\tau = \theta^\tau + \mu\Delta_{\theta^\tau} \qquad \omega^\tau = \omega^\tau + \mu\Delta_{\omega^\tau}$
  end for
  Update targets $\theta_T^\tau = \theta^\tau$ and $\omega_T^\tau = \omega^\tau$.
end for

the main tasks of updates due to c_1. One possible way to do this would be to use all transitions to update the q estimates but use a smaller step-size for the ones that do not satisfy c_2. Notice that the resulting algorithm would be exactly *HystQ* [83].

4.4 Experiments

4.4.1 Matrix Game

The first experiment is a simple matrix game (table 4.4.1 shows the payoff structure) with multiple team optimum policies to evaluate the resilience of the algorithm to the coordination issue mentioned in section 4.3.1.

In this case, we implemented *LTQL* and *DistQ* in tabular form (we do not include *HystQ* because in deterministic environments with tabular representation this algorithm is domi-

		Agent 2		
		a_1	a_2	a_3
Agent 1	b_1	0	2	0
	b_2	0	1	2

Table 4.1: Payoff matrix

nated by *DistQ*) and we also implemented *Qmix* (note that this algorithm cannot be implemented in tabular form due to the use of the mixing network). In all cases we used uniform exploratory policies ($\epsilon = 1$) and we did not use replay buffer. *DistQ* converges to (4.12), which clearly shows why *DistQ* has a coordination issue. However, *LTQL* converges to either of the two possible solutions shown in (4.13) (depending on the seed) for which individual greedy policies result in team optimal policies. *Qmix* converges to (4.14). Note that *Qmix* fails at identifying an optimum team policy and the resulting joint Q-function q_{jt} obtained using the mixing network also fails at predicting the rewards. The full q_{jt} is shown in table 4.4.1.

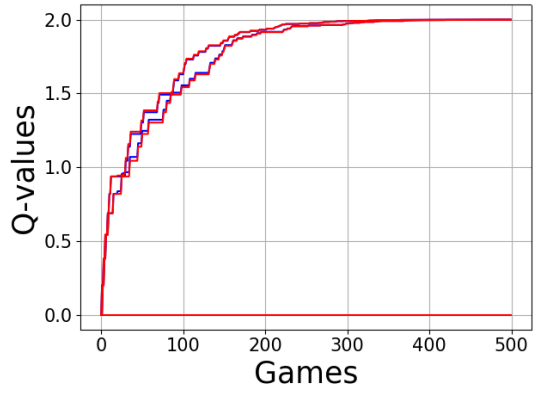
$$q^1(a^1) = \max_{a^2} q^\dagger(a^1, a^2) = [2, 2] \quad q^2(a^2) = \max_{a^1} q^\dagger(a^1, a^2) = [0, 2, 2] \tag{4.12}$$

$$q^{1,\star}(a^1) = [2, 1] \quad q^{2,\star}(a^2) = [0, 2, 0] \qquad \textbf{or} \qquad q^{1,\star}(a^1) = [0, 2] \quad q^{2,\star}(a^2) = [0, 1, 2] \tag{4.13}$$

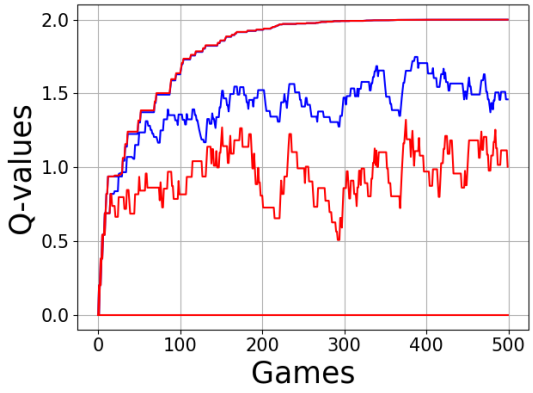
$$q^1(a^1) = [-0.75, 1.09] \quad q^2(a^2) = [-3.49, 1.83, 0.62] \tag{4.14}$$

In figure 4.1 we show the convergence curves for *Qmix* 4.1(f), *DistQ* 4.1(a) and *Logical Team Q-learning*. Figures 4.1(b) and 4.1(c) correspond to the deterministic (algorithm 4.2) and general version (algorithm 4.3), respectively. Figures 4.1(d) and 4.1(e) also show curves for *LTQL* but use different seeds and provide evidence to our claim that this algorithm can converge to either of the two options in (4.13). One interesting fact to note is that the suboptimal values of q_B^τ (in figures 4.1(b) and 4.1(d)) do not converge while the same values do converge in the case of q_U^τ. This is a direct consequence of theorem 4.1. As the theorem suggests, only the optimal values of the estimates generated by *LTQL* converge to $q^{\tau,\star}$, and suboptimal values lie in the region between $q^{\tau_k,\star}(o_s^{\tau_k}, a^k)$ and $\max_{a^{-k}} q^\dagger(s, a^k, a^{-k})$. Due to the bias generated by condition c_2, the values corresponding to sub-optimal actions converge

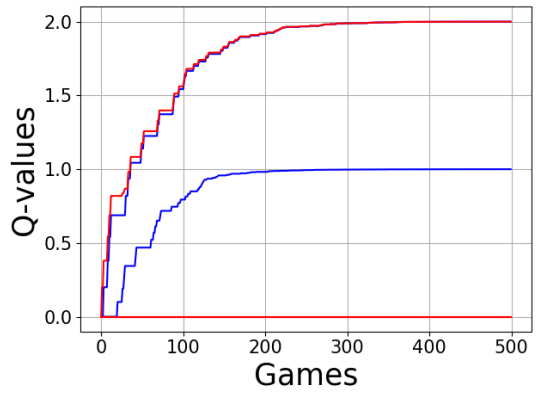
to values in between $q^{\tau,\star}$ and $q^{\dagger}$. All the values q_U^{τ} converge because these estimates are not affected by this bias (because q_U^{τ} is only updated through c_1, see algorithm 4.3).



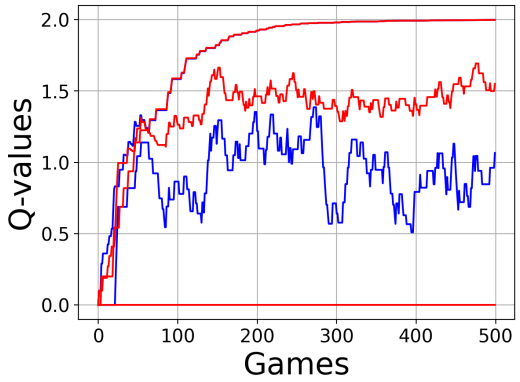
(a) DistQ



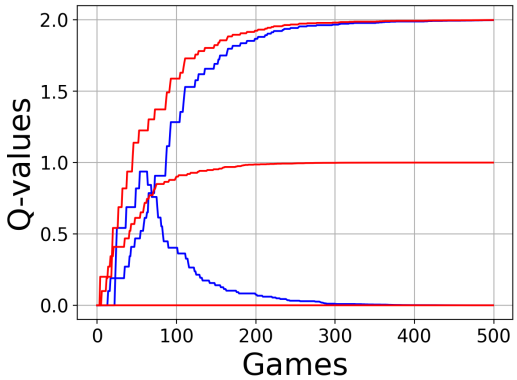
(b) q_B^{τ} (seed=0)



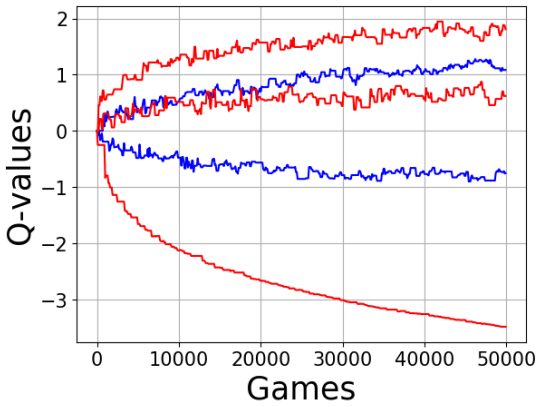
(c) q_U^{τ} (seed=0)



(d) q_B^{τ} (seed=1)



(e) q_U^{τ} (seed=1)



(f) *Qmix*

Figure 4.1: Matrix game. In all figures the red curves correspond to the three actions of agent 2, while the blue curves correspond to the two actions from agent 1.

Below we show the joint q values generated by *Qmix*'s mixing network.

		Agent 2		
		a_1 (-3.49)	a_2 (1.83)	a_3 (0.62)
Agent 1	b_1 (-0.74)	-4.78×10^{-2}	1.17	6.86×10^{-1}
	b_2 (1.09)	1.51×10^{-3}	1.57	1.09

Table 4.2: *Qmix* full results

We now specify the hyperparameters. For *Logical Team Q-learning* and *DistQ* we used a step-size equal to 0.1. The α parameter for *LTQL* is equal to 1. The mixing network in *Qmix* has 2 hidden layers with 5 units each, the nonlinearity used was the *ELu* and the step-size used was 0.05 (we had to make it smaller than the others to make the SGD optimizer converge). We finally remark that due to the more complex structure of *Qmix* (compared to the other two algorithms) we had to train this algorithm with 100 times more games (notice the x-axis in figure 4.1).

4.4.2 Stochastic Finite TMDP

In this experiment we use a tabular representation in a stochastic episodic TMDP. The environment is a linear grid with 4 positions and 2 agents. At the beginning of the episode, the agents are initialized in the far right. Agent 1 cannot move and has 2 actions (*push button* or *not push*), while agent 2 has 3 actions (*stay*, *move left* or *move right*). If agent 2 is located in the far left and chooses to *stay* while agent 2 chooses *push*, the team receives a $+10$ reward. If the button is pushed while agent 2 is moving left the team receives a -30 reward. This negative reward is also obtained if agent 2 stays still in the leftmost position and agent 1 does not push the button. All rewards are subject to additive Gaussian noise with mean 0 and standard deviation equal to 1. Furthermore if agent 2 tries to move beyond an edge (left or right), it stays in place and the team receives a Gaussian reward with 0 mean and standard deviation equal to 3. The TMDP finishes after 5 timesteps or if the team gets the $+10$ reward (whichever happens first). We ran the simulation 5 times with different seeds. The observation corresponding to agent 1 is a vector with two binary elements: the

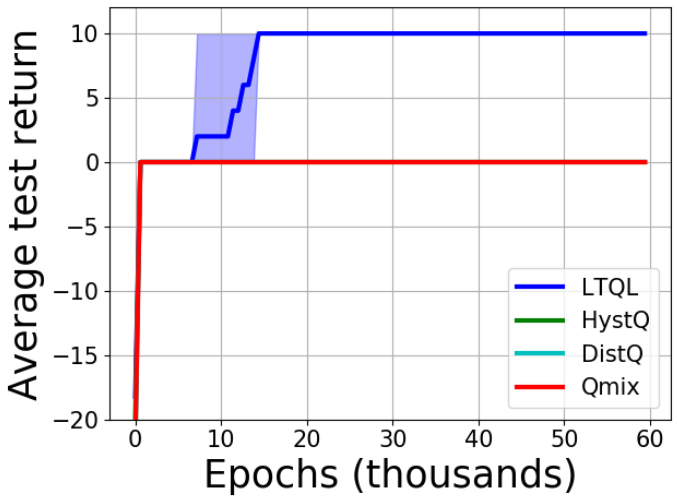


Figure 4.2: The dark curves show the mean over all seeds while the shaded regions show the min and max limits over the seeds.

first one indicates whether or not agent 2 is in the leftmost position, and the second element indicates whether or not there is enough time for agent 2 to reach the leftmost position. The observation corresponding to agent 2 is a vector with two elements: the first one is the number of the position it occupies and the second one is the same as agent 1 (whether there is enough time to reach the leftmost position). Note that these observations are not full descriptions of the state, however they do satisfy assumption 4.1.

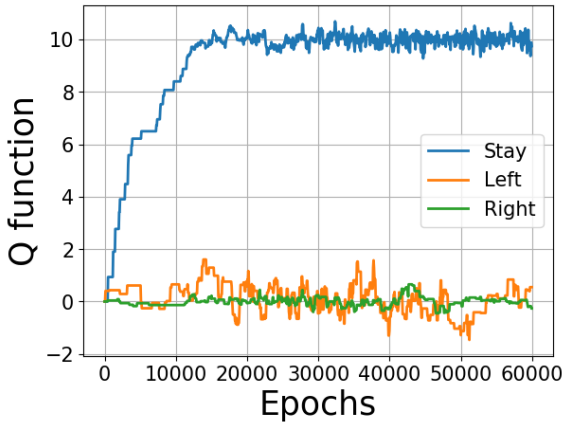
All algorithms are implemented in an on-line manner with no replay buffer. ϵ-greedy exploration with a decaying schedule is used in all cases ($\epsilon = \max[0.05, 1 - epoch/5 \times 10^4]$). The learning rate used is $\mu = 10^{-1}$ and the smaller second rate for *HystQ* is $\mu_{\text{small}} = 5.10^{-2}$, in the case of *Qmix* we used $\mu = 10^{-3}$ to guarantee stability. The α parameter for *LTQL* is equal to 1.

Figure 4.4.2 shows the average test return[4] (without the added noise) of *LTQL*, *HystQ*, *DistQ* and *Qmix*. As can be seen, *LTQL* is the only algorithm capable of learning the optimal team policy.

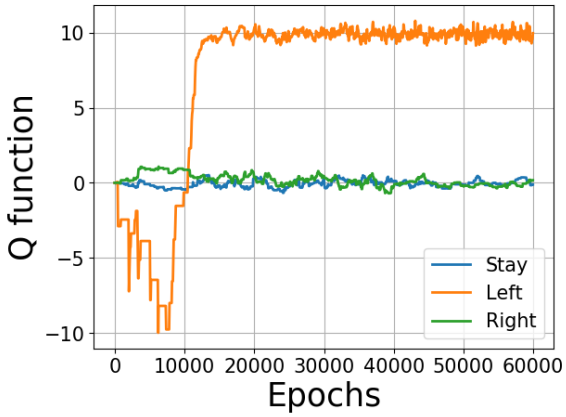
Figures 4.3 show the estimated q-values for *Logical Team Q-learning* corresponding to 4 different observations at the 4 positions. Note that in all figures the optimum action has the

[4]The average test return is the return following a greedy policy averaged over 50 games.

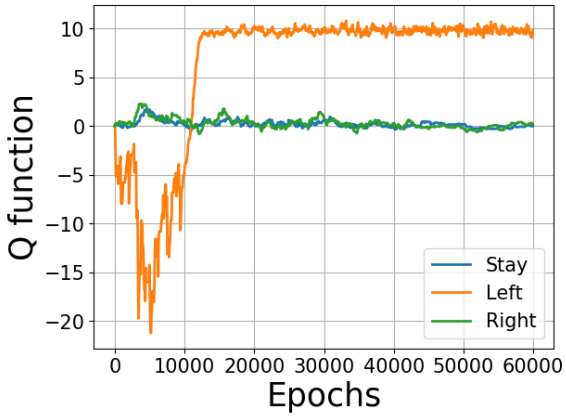
highest value and correctly estimates the return corresponding to the optimal team policy (+10).



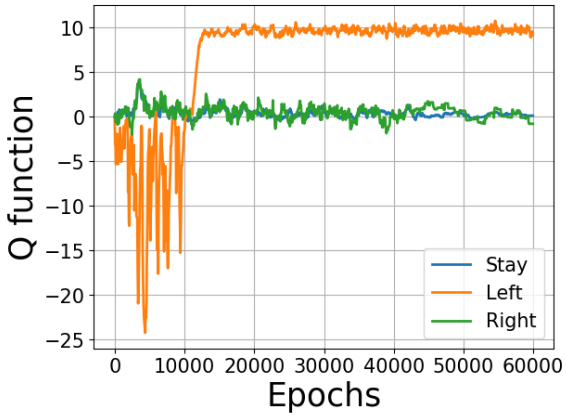
(a) Leftmost position $t = 3$



(b) Slot adjacent to leftmost $t = 2$



(c) Slot adjacent to rightmost $t = 1$

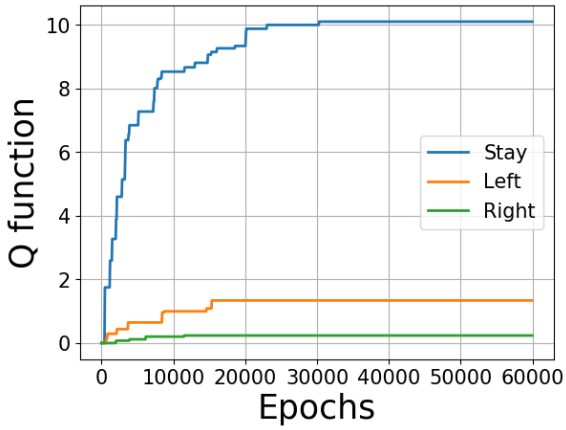


(d) Rightmost position $t = 0$

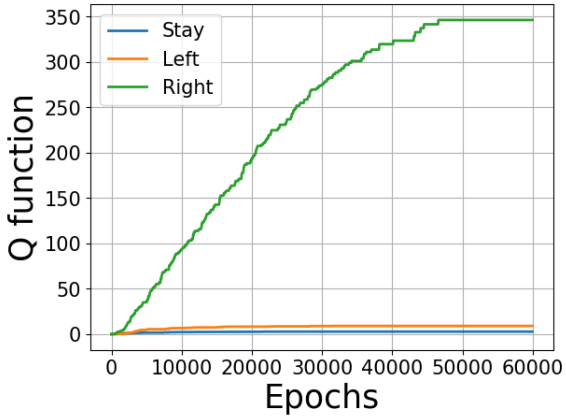
Figure 4.3: Learning curves for agent 2 of *Logical Team Q-learning* for a random seed.

Figures 4.4 show the learning curves for *DistQ*. Note that the reason that this algorithm cannot solve this environment is that it severely overestimates the value of choosing to move to the right whilst on the rightmost position. It is well known that this is a consequence of the fact that *DistQ* only performs updates that increase the estimates of the Q-values combined with the stochastic reward received when agent 2 "stumbles" against the right edge.

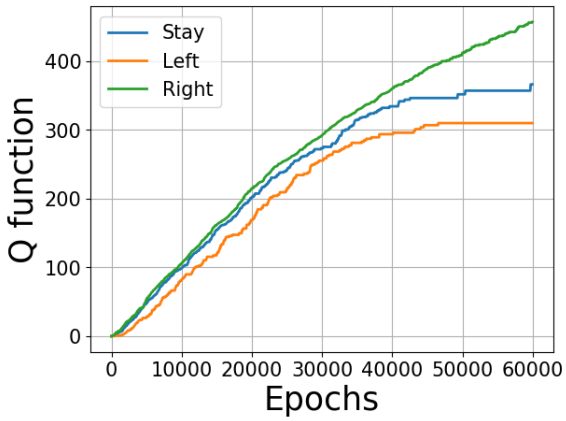
Figures 4.5 show the learning curves for *HystQ*. This algorithm cannot solve this environment because it has two issues and the way to solve one makes the other worse. More



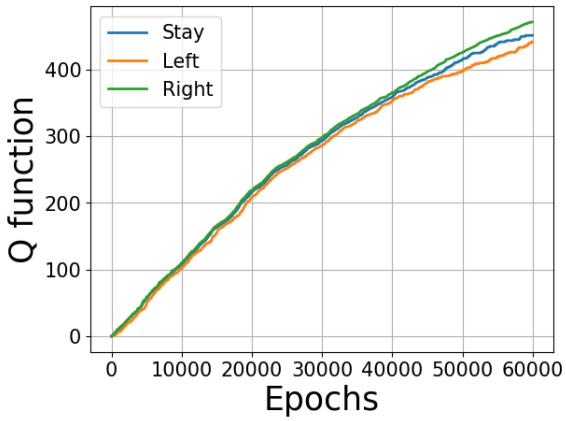
(a) Leftmost position $t = 3$



(b) Slot adjacent to leftmost $t = 2$



(c) Slot adjacent to rightmost $t = 1$

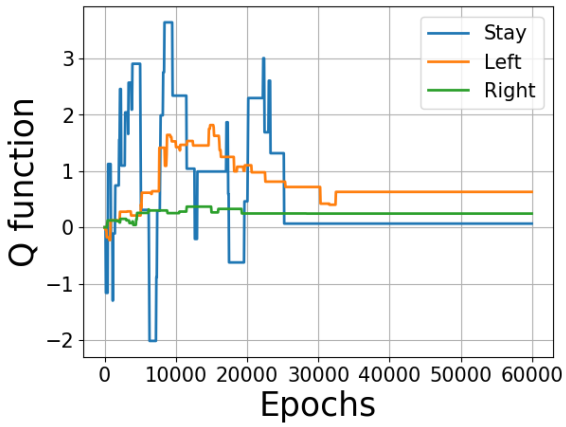


(d) Rightmost position $t = 0$

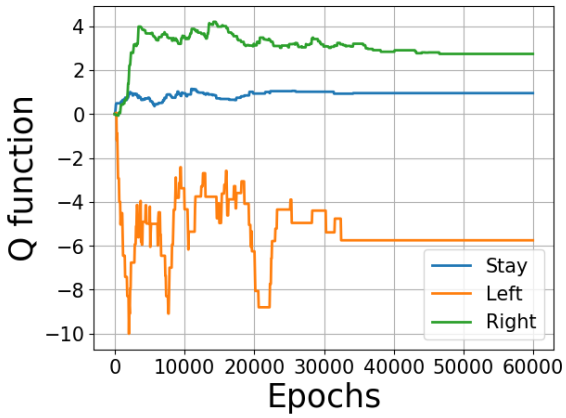
Figure 4.4: Learning curves for agent 2 of *DistQ* for a random seed.

specifically, one can be solved by increasing the smaller step-size, while the other needs to decrease it. The first issue is the same one that affects *DistQ*, i.e., the overestimation of the *move right* action in the rightmost position. Note that this can be ameliorated by increasing the small step-size. The second issue is the penalty incurred due to moving to the left when agent 1 presses the button. This can be ameliorated by decreasing the small step-size. The fact that there is no intermediate value for the small step-size to solve both issues is the reason that this algorithm cannot solve this environment.

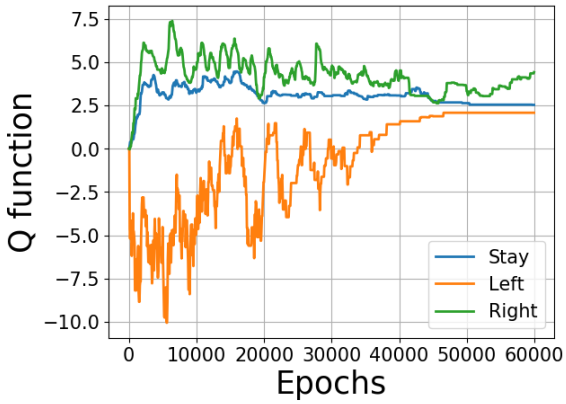
Figures 4.6 show the learning curves for *Qmix*. The architecture used is as follows: we used tabular representation for the individual q functions, and for the mixing and hyper-networks we used the architecture specified in [89]. More specifically, the mixing network is



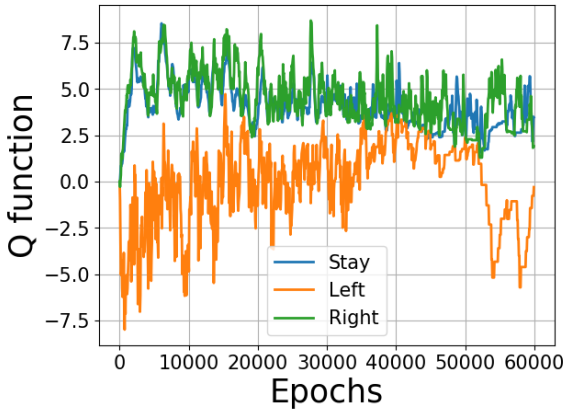
(a) Leftmost position $t = 3$



(b) Slot adjacent to leftmost $t = 2$



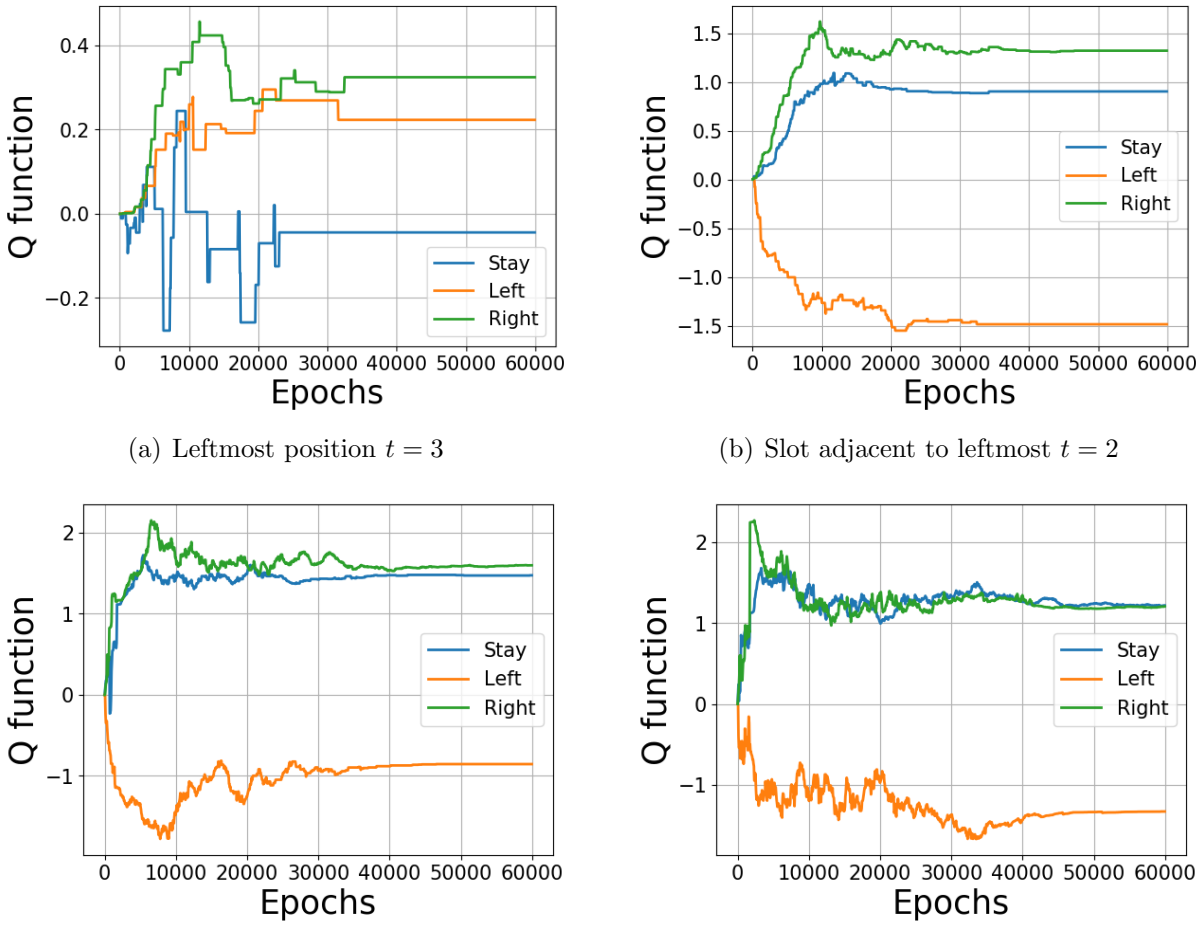
(c) Slot adjacent to rightmost $t = 1$



(d) Rightmost position $t = 0$

Figure 4.5: Learning curves for agent 2 of *HystQ* for a random seed.

composed of two hidden layers (with 10 units each) with *ELu* nonlinearities in the first layer while the second layer is linear. The hypernetworks that output the weights of the mixing network consist of two layers with *ReLU* nonlinearities followed by an activation function that takes the absolute value to ensure that the mixing network weights are non-negative. The bias of the first mixing layer is produced by a network with a unique linear layer and the other bias is produced by a two layer hypernetwork with a *ReLU* nonlinearity. All hypernetwork layers are fully connected and have 5 units.



(a) Leftmost position $t = 3$

(b) Slot adjacent to leftmost $t = 2$

(c) Slot adjacent to rightmost $t = 1$

(d) Rightmost position $t = 0$

Figure 4.6: Learning curves for agent 2 of *Qmix*.

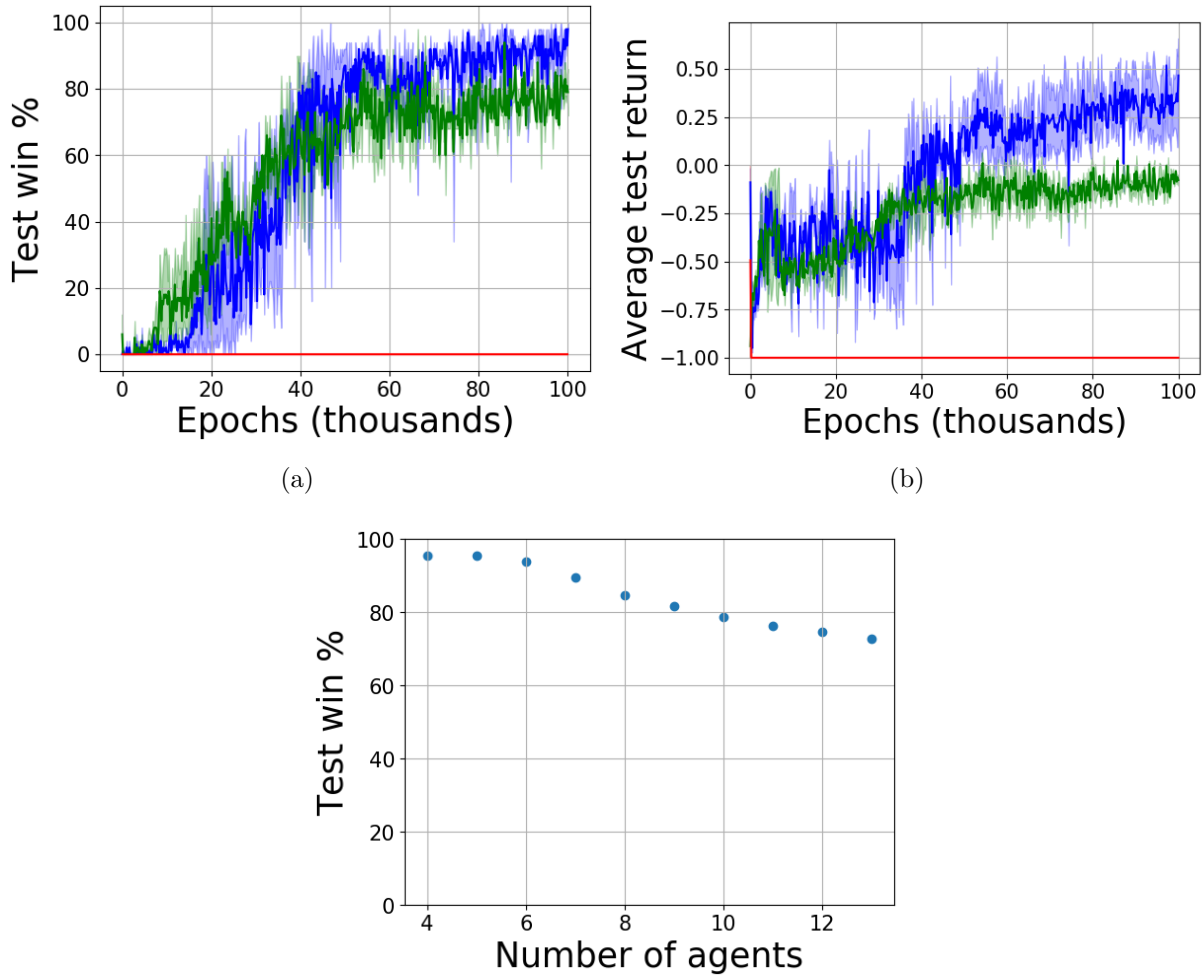
4.4.3 Cowboy Bull Game

In this experiment we use a more complex environment. The TMDP is a challenging predator-prey type game, in which 4 (homogeneous) cowboys try to catch a bull (see figure 4.4.3). The position of all players is a continuous variable (and hence the state space is continuous). The space is unbounded and the bull can move 20% faster than the cowboys. The bull follows a fixed stochastic policy, which is handcrafted to mimic natural behavior and evade capture. In appendix 4.G we specify the bull's policy. Due to the unbounded space and the fact that the bull moves faster than the cowboys, it cannot be captured unless all agents develop a coordinated strategy (the bull can only be caught if the agents first

Figure 4.7: Cowboy bull game

surround it and then close in evenly). The task is episodic and ends after 75 timesteps or when the bull is caught. Each agent has 5 actions (the four moves plus *stay*). When the bull is caught a +1 reward is obtained and the team also receives a small penalty ($-1/(4 \times 75)$) for every agent that moves. Note that due to the reward structure there is a very easily attainable Nash equilibrium, which is for every agent to stay still (since in this way they do not incur in the penalties associated with movement). In this game, since all agents are homogeneous, only one Q-function is learned whose input is the agent's observation and the output are the Q-values corresponding to the 5 possible actions. Figure 4.8(a) shows the test win percentage[5] and figure 4.8(b) shows the average test return for *LTQL*, *HystQ* and *Qmix*. The best performing algorithm is *LTQL*. *HystQ* learns a policy that catches the bull 80% of the times, although it fails at obtaining returns higher than zero. We believe that the poor performance of *Qmix* in this task is a consequence of its limited representation capacity due to its monotonic factoring model. As we mentioned in the introduction, we can test the learned policy on teams with different number of agents, figure 4.8(c) shows the results. The policy scores above 70% for teams of all sizes bigger than 4. Note that the policy can be improved for any particular team size by further training if necessary.

[5]Percentage of games, out of 50, in which the team succeeds to catch the bull following a greedy policy.



(a) (b)

(c)

Figure 4.8: In (a) and (b) the blue, green and red curves correspond to *LTQL*, *HystQ* and *Qmix*, respectively. The dark curves show the mean over all seeds while the shaded regions show the min and max limits over the seeds.

We now specify the hyperparameters for *Logical Team Q-learning*. All NN's have two hidden layers with 50 units and ReLu nonlinearities. However, for each Q-network, instead of having one network with 5 outputs (one for each action), we have 5 networks each with 1 output. At every epoch the agent collects data data by playing 32 full games and then performs 50 gradient backpropagation steps. Half of the 32 games are played greedily and the other half use a Boltzmann policy with temperature b_T with decays according to the following schedule $b_T = \max[0.05, 0.5 \times (1 - epoch/15 \times 10^3)]$. We use this behavior policy to ensure that there are sufficient transitions that satisfy condition c_1 and that also there

are transitions that satisfy c_2. The target networks are updated every 50 backprop steps. The capacity of the replay buffer is 2.10^5 transitions, the mini-batch size is 1024, we use a discount factor equal to 0.99 and optimize the networks using the Adam optimizer with initial step-size 10^{-5} and α parameter is 1.

The hyperparameters of the *HystQ* implementation are the same as those of *LTQL*, the ratio of the two step-sizes used by *HystQ* is 0.1.

The architecture used by *Qmix* is the one suggested in [89] with the exception that, for fairness, the individual Q-networks used the same architecture as the ones used by the other algorithms (i.e., 5 networks with a unique output as opposed to 1 network with 5 outputs). All hidden layers of the hypernetworks as well the mixing network have 10 units. In this case we did 5 backprop iterations per epoch and the target network update period is 15. We use a batch size of 256, a discount factor equal to 0.98 and optimize the networks with the Adam optimizer with initial step-size 10^{-6}. In this case the behavior policy is always Boltzmann with the following annealing schedule for the temperature parameter $b_T = \max[0.005, 0.05 \times (1 - epoch/25 \times 10^3)]$.

The batch size, Boltzmann temperature value, learning step-size and target update period were chosen by grid search.

4.5 Summary

In this chapter we have introduced theoretical groundwork for factored cooperative MARL. We also introduced *LTQL*, which has the 5 desirable properties mentioned in the introduction. Furthermore, it does not impose constraints on the learned individual Q-functions and hence it can solve environments where previous algorithms, which are considered to be state of the art such as *Qmix* [89], fail. The algorithm fits in the centralized training and decentralized execution paradigm. It can also be implemented in a fully distributed manner in situations where all agents have access to each others' observations and actions.

4.A Proof of Lemma 4.1

We start by rewriting equation (4.2b) for convenience:

$$q^{\dagger}(s,\bar{a}) = \mathbb{E}\left[\boldsymbol{r}(s,\bar{a},\boldsymbol{s'}) + \gamma \max_{\bar{a}'} q^{\dagger}(\boldsymbol{s'},\bar{a}')\right] \tag{4.15}$$

Now assume that we have some deterministic team optimal policy $\pi^{\dagger}(\bar{a}|s)$. We define $q^{k,*}(s,a^k)$ as follows:

$$q^{k,*}(s,a^k) = q^{\dagger}(s,a^k,a^{-k})|_{a^{-k}\sim\pi^{\dagger}(a^k,a^{-k}|s)} \tag{4.16}$$

In simple terms $q^{k,*}(s,a^k)$ is the q-value if agent k takes action a^k while the rest of the agents act optimally (note that this is not the same as $\max_{a^{-k}} q^{\dagger}(s,a^k,a^{-k})$). Due to assumption 4.1, $q^{k,*}(s,a^k)$ can be written as a function of the observations. Therefore, we define $q^{k,\bullet}$ as:

$$q^{k,\bullet}(o_s^{\tau_k},a^k) = q^{k,*}(s,a^k) = q^{\dagger}(s,a^k,a^{-k})|_{a^{-k}\sim\pi^{\dagger}(a^k,a^{-k}|s)} \tag{4.17}$$

Note that by construction we get:

$$\max_{a^k} q^{k,\bullet}(o_s^{\tau_k},a^k) = \max_{\bar{a}} q^{\dagger}(s,\bar{a}) \;\; \forall k \tag{4.18}$$

$$\arg\max_{a^k} q^{k,\bullet}(o_s^{\tau_k},a^k) = a^k \sim \pi^{\dagger}(a^k,a^{-k}|s) \;\; \forall k \tag{4.19}$$

$$\max_{\bar{a}} q^{\dagger}(s,\bar{a}) = q^{\dagger}\big(s, \arg\max_{a^1} q^{1,\bullet}(o_s^{\tau_1},a^1), \cdots, \arg\max_{a^K} q^{\tau,\bullet}(o_s^{\tau_K},a^K)\big) \;\; \forall k \tag{4.20}$$

Combining (4.15), (4.17) and (4.19) we get the following relation for $q^{k,\bullet}$:

$$q^{k,\bullet}(o_s^{\tau_k},a^k) = q^{\dagger}(s,a^k,a^{-k})|_{a^{-k}\sim\pi^{\dagger}(a^k,a^{-k}|s)} \;\; \forall k \tag{4.21}$$

$$q^{k,\bullet}(o_s^{\tau_k},a^k) = q^{k,\dagger}(s,a^k,a^{-k})|_{a^n=\arg\max_{a^n} q^{n,\bullet}(o_s^{\tau_n},a^n)\forall n\neq k} \;\; \forall k \tag{4.22}$$

$$= \mathbb{E}\left[\boldsymbol{r}(s,a^k,a^{-k},\boldsymbol{s'}) + \gamma \max_{a',k} q^{k,\bullet}(\boldsymbol{o_{s'}^{\tau_k}},a'^{,k})\right]\big|_{a^n \arg\max_{a^n} q^{n,\bullet}(o_s^{\tau_n},a^n)\forall n\neq k} \;\; \forall k \tag{4.23}$$

Note that (4.18), (4.20) and (4.23) satisfy equations (4.4a), (4.4b) and (4.4c), respectively. However, functions $q^{k,\bullet}$ are defined on a per agent basis (which means that there are K such functions) while functions $q^{\tau_k,\star}$ depend on the type (and hence there are $|\mathcal{T}|$ such functions). Therefore, we still have to prove that functions $q^{k,\bullet}$ corresponding to agents of the same type are equal. From assumption 4.2, it follows that choosing q^{π} in (4.3) to be $q^{\dagger}(s, a^k, a^{-k})|_{a^{-k}\sim\pi^{\dagger}(a^k,a^{-k}|s)}$ we get:

$$q^{\dagger}(s_1, a^k, a^{-k})\Big|_{\substack{a^k=b \\ a^{-k}\sim\pi^{\dagger}(a^k,a^{-k}|s_1)}} = q^{\dagger}(s_2, a^n, a^{-n})|_{\substack{a^n=b \\ a^{-n}\sim\pi^{\dagger}(a^n,a^{-n}|s_2)}} \quad \forall b \in \mathcal{A}^{\tau} \tag{4.24}$$

$$o^k_{s_1} = o^n_{s_2} \tag{4.25}$$

where s_1 and s_2 are two equivalent states that swap agents k and n such that $\tau_k = \tau_n = \tau$ for some type τ. Since (4.24) holds for any $b \in \mathcal{A}^{\tau}$ and for all equivalent states, combining (4.24), (4.25) and (4.17) and setting $o = o^{\tau_k}_{s_1} = o^{\tau_n}_{s_2}$ we get that:

$$q^{\tau,\star}(o, a) = q^{k,\bullet}(o, a) = q^{n,\bullet}(o, a) \;\; \forall (k, o, a)\big|\tau_k = \tau, o \in \mathcal{O}^{\tau}, a \in \mathcal{A}^{\tau} \tag{4.26}$$

which completes the proof.

4.B Proof of remark 4.1

Consider the matrix game with two homogeneous agents, each of which has two actions ($\mathcal{A} = \{\alpha; \beta\}$) and the following reward structure:

Table 4.3: Reward structure

		Agent 2	
		α	β
Agent 1	α	0	-1
	β	-1	1

For this case $q^{\dagger}$, $q^{\star}$, $\pi^{\dagger}$ and $\pi^{\star}$ are given by:

Table 4.4: $q^\dagger(a^1, a^2)$

	α	β
α	0	-1
β	-1	1

Table 4.5: $\pi^\dagger(a^1, a^2)$

	α	β
α	0	0
β	0	1

Table 4.6: $q^\star(a)$

α	β
-1	1

Table 4.7: $\pi^\star(a)$

α	β
0	1

Notice that as expected, $q^\star$ satisfies (4.4c). However, note that (4.4c) is also satisfied by the following q function which is different from $q^\star$.

$$q(a = \alpha) = 0, \quad q(a = \beta) = -1 \tag{4.27}$$

Notice further that the team policy obtained by choosing actions in a greedy fashion with respect to q constitutes a Nash equilibrium.

4.C Proof of Theorem 4.1

We start defining the following auxiliary constants and operators:

$$q_{\max} = r_{\max}(1-\gamma)^{-1} \tag{4.28}$$

$$q_{\min} = r_{\min}(1-\gamma)^{-1} \tag{4.29}$$

$$\mathcal{B}_E q^{\tau_k}(o_s^{\tau_k}, a^k) = \mathbb{E}_{\mathcal{P},n}\big(\boldsymbol{r}(s, a^k, a^{-k}, \boldsymbol{s'}) + \gamma \max_{a'} q^{\tau_k}(\boldsymbol{o}_{\boldsymbol{s'}}^{\boldsymbol{\tau_k}}, a')\big)\big|_{a^n = \arg\max_{a^n} q^{\tau_n}(o_s^{\tau_n}, a^n)\ \forall n \neq k} \tag{4.30}$$

$$\mathcal{B}_I q^{\tau_k}(o_s^{\tau_k}, a^k) = \max\Big\{q^{\tau_k}(o_s^{\tau_k}, a^k), \max_{a^{-k}} \mathbb{E}_{\mathcal{P},n}\big(\boldsymbol{r}(s, a^k, a^{-k}, \boldsymbol{s'}) + \gamma \max_{a'} q^{\tau_k}(\boldsymbol{o}_{\boldsymbol{s'}}^{\boldsymbol{\tau_k}}, a')\big)\Big\} \tag{4.31}$$

$$\mathcal{B}_U^\psi q^{\tau_k}(o_s^{\tau_k}, a^k) = \begin{cases} \mathcal{B}_E q^{\tau_k}(o_s^{\tau_k}, a^k) & \text{with probability } p \\ \mathcal{B}_I q^{\tau_k}(o_s^{\tau_k}, a^k) & \text{else} \end{cases} \tag{4.32}$$

$$\begin{aligned} \mathcal{B}_L^\psi q^{\tau_k}(o_s^{\tau_k}, a^k) = \big[&\mathbb{I}(c_1)\mathcal{B}_E q^{\tau_k}(o_s^{\tau_k}, a^k) + \mathbb{I}(\bar{c}_1 \wedge c_2)\mathcal{B}_I q^{\tau_k}(o_s^{\tau_k}, a^k) \\ &+ \mathbb{I}(\bar{c}_1 \wedge \bar{c}_2) q^{\tau_k}(o_s^{\tau_k}, a^k)\big]\big|_{a^{-k} \sim \psi} \end{aligned} \tag{4.33}$$

$$c_1 \triangleq \left(a^n == \arg\max_{a^n} q^{\tau_n}(o_s^{\tau_n}, a^n) \ \ \forall n \neq k\right) \tag{4.34}$$

$$c_2 \triangleq \left(a^{-k} == \arg\max_{a^{-k}} \mathcal{B}_{a^{-k}} q^{\tau_k}(o_s^{\tau_k}, a^k)\right) \tag{4.35}$$

where $r_{\max}$ is the maximum mean reward, $r_{\min}$ is the minimum mean reward and p is the minimum probability assigned to any a^{-k} by the discrete distribution ψ.

Lemma 4.3. *Repeated application of operator $\mathcal{B}_U^{\psi}$ to any $q_U^{\tau_k}(o_s^{\tau_k}, a^k) \geq q_{max}$ converges to set $\mathcal{S}_U = \left\{q^{\tau_k} | q^{\tau_k}(o_s^{\tau_k}, a^k) \leq \max_{a^{-k}} q^{\dagger}(s, a^k, a^{-k}) \ \forall o_s^{\tau_k} \in \mathcal{O}^{\tau_k} \wedge a^k \in \mathcal{A}^{\tau_k}\right\}$ with probability one. The mean rate of convergence is exponential with Lipschitz constant γ^p.*

Proof. See appendix 4.D. □

Lemma 4.4. *Repeated application of operator $\mathcal{B}_L^{\psi}$ to any $q_L^{\tau_k}(o_s^{\tau_k}, a^k) \leq q_{min}$ converges to set $\mathcal{S}_L = \left\{q^{\tau_k} | q^{\tau_k}(o_s^{\tau_k}, a^k) \geq q^{\tau_k,\star}(o_s^{\tau_k}, a^k) \ \forall o_s^{\tau_k} \in \mathcal{O}^{\tau_k} \wedge a^k \in \mathcal{A}^{\tau_k}\right\}$ with probability one. The mean rate of convergence is exponential with Lipschitz constant γ^p.*

Proof. See appendix 4.E. □

Now assume we have an initial Q-function and define $q_L(o_s^{\tau_k}, a^k) = \min\{q_{\min}, q(o_s^{\tau_k}, a^k)\}$ and $q_U(o_s^{\tau_k}, a^k) = \max\{q_{\max}, q(o_s^{\tau_k}, a^k)\}$. We conclude the proof by noting that $\mathcal{B}_L^{\psi} q_L(o_s^{\tau_k}, a^k) \leq \mathcal{B}^{\psi} q(o_s^{\tau_k}, a^k) \leq \mathcal{B}_U^{\psi} q_U(o_s^{\tau_k}, a^k)$ and making use of lemmas (4.3) and (4.4) and the sandwich theorem.

4.D Proof of Lemma 4.3

We start defining $q_U^{\tau_k}(o_s^{\tau_k}, a^k) = q_U, \forall(\tau_k, o_s^{\tau_k}, a^k)$, where $q_U \geq q_{\max}$. The first part of the proof consists in showing that any sequence of the form $\mathcal{B}_I^{K_n} \mathcal{B}_E \cdots \mathcal{B}_I^{K_1} \mathcal{B}_E \mathcal{B}_I^{K_0} q_U^{\tau_k}(o_s^{\tau_k}, a^k)$ is equal to $\mathcal{B}_I^n q_U^{\tau_k}(o_s^{\tau_k}, a^k)$ where $K_n \in \mathbb{N}$ and $n \in \mathbb{N}$ is the number of times that operator $\mathcal{B}_I$ is applied in the aforementioned sequence. Applying operator $\mathcal{B}_I$ to $q_U^{\tau_k}(o_s^{\tau_k}, a^k)$ we get:

$$\mathcal{B}_I q_U^{\tau_k}(o_s^{\tau_k}, a^k) = \max\left\{q_U, \max_{a^{-k}} \mathbb{E}_{\mathcal{P},n}\big(\boldsymbol{r}(s, a^k, a^{-k}, \boldsymbol{s'}) + \gamma q_U\big)\right\} = q_U \tag{4.36}$$

Therefore, $\mathcal{B}_I^{K_0} q_U^{\tau_k}(o_s^{\tau_k}, a^k) = q_U^{\tau_k}(o_s^{\tau_k}, a^k)$ for any $K_0 \in \mathbb{N}$. Applying operator $\mathcal{B}_E$ we get:

$$\begin{aligned}
\mathcal{B}_E q_U^{\tau_k}(o_s^{\tau_k}, a^k) &= \mathbb{E}_{\boldsymbol{s'}\sim\mathcal{P}}\big(r(s, a^k, a^{-k}) + \gamma q_U\big)\big|_{a^n = \arg\max_{a^n} q_U \quad \forall n \neq k} \\
&\leq \max_{a^{-k}} r(s, a^k, a^{-k}) + \gamma q_U \qquad (4.37) \\
\mathcal{B}_E^2 q_U^{\tau_k}(o_s^{\tau_k}, a^k) &\leq \mathbb{E}_{\boldsymbol{s'}\sim\mathcal{P}}\big(r(s, a^k, a^{-k}) + \gamma \max_{\bar{a}'} r(\boldsymbol{s'}, \bar{a}') + \gamma^2 q_U\big)\big|_{a^n = \arg\max_{a^n} \mathcal{B}_E q_U^{\tau_n}(o_s^{\tau_n}, a^n) \quad \forall n \neq k} \\
&\leq \max_{a^{-k}} \mathbb{E}_{\boldsymbol{s'}\sim\mathcal{P}}\left(r(s, a^k, a^{-k}) + \gamma \max_{\bar{a}'} r(\boldsymbol{s'}, \bar{a}') + \gamma^2 q_U\right) \qquad (4.38) \\
\mathcal{B}_E^{K_1} q_U^{\tau_k}(o_s^{\tau_k}, a^k) &\leq \max_{a_0^{-k}, \bar{a}_1, \cdots, \bar{a}_{K_1-1}} \mathbb{E}\left(\sum_{i=0}^{K_1-1} \gamma^i r(\boldsymbol{s}_i, a_i^k, a_i^{-k}) | \boldsymbol{s}_0 = s\right) + \gamma^{K_1} q_U \qquad (4.39)
\end{aligned}$$

where to simplify notation we defined $\mathbb{E}_{\mathcal{P},n}\boldsymbol{r}(s, a^k, a^{-k}, \boldsymbol{s'}) = r(s, a^k, a^{-k})$. Further application of $\mathcal{B}_I$ we get:

$$\begin{aligned}
\mathcal{B}_I \mathcal{B}_E^{K_1} q_U^{\tau_k}(o_s^{\tau_k}, a^k) \leq \max\Bigg\{ & \max_{a_0^{-k}, \bar{a}_1, \cdots, \bar{a}_{K_1-1}} \mathbb{E}\left(\sum_{i=0}^{K_1-1} \gamma^i r(\boldsymbol{s}_i, a_i^k, a_i^{-k}) | \boldsymbol{s}_0 = s\right) + \gamma^{K_1} q_U, \\
& \max_{a_0^{-k}, \bar{a}_1, \cdots, \bar{a}_{K_1}} \mathbb{E}\left(\sum_{i=0}^{K_1} \gamma^i r(\boldsymbol{s}_i, a_i^k, a_i^{-k}) | \boldsymbol{s}_0 = s\right) + \gamma^{K_1+1} q_U \Bigg\} \\
= & \max_{a_0^{-k}, \bar{a}_1, \cdots, \bar{a}_{K_1-1}} \mathbb{E}\left(\sum_{i=0}^{K_1-1} \gamma^i r(\boldsymbol{s}_i, a_i^k, a_i^{-k}) | \boldsymbol{s}_0 = s\right) + \gamma^{K_1} q_U \qquad (4.40)
\end{aligned}$$

Therefore, we conclude that $\mathcal{B}_I^{K_2} \mathcal{B}_E^{K_1} \mathcal{B}_I^{K_0} q_U^{\tau_k}(o_s^{\tau_k}, a^k) = \mathcal{B}_E^{K_1} q_U^{\tau_k}(o_s^{\tau_k}, a^k)$. More generally, we can write:

$$\begin{aligned}
\mathcal{B}_I^{K_n} \cdots \mathcal{B}_E^{K_1} \mathcal{B}_I^{K_0} q_U^{\tau_k}(o_s^{\tau_k}, a^k) &= \mathcal{B}_E^n q_U^{\tau_k}(o_s^{\tau_k}, a^k) \\
&\leq \max_{a_0^k, \bar{a}_1, \cdots, \bar{a}_{n-1}} \mathbb{E}\left(\sum_{i=0}^{n-1} \gamma^i r(\boldsymbol{s}_i, a_i^k, a_i^{-k}) | \boldsymbol{s}_0 = s\right) + \gamma^n q_U \\
&\leq \max_{a^{-k}} q^\dagger(s, a^k, a^{-k}) + \gamma^n \big(q_U - \min_s \max_{\bar{a}} q^\dagger(s, \bar{a})\big) \qquad (4.41)
\end{aligned}$$

where n is the number of times that operator $\mathcal{B}_E$ is applied. Therefore, we get:

$$(\mathcal{B}_U^\psi)^N q_U^{\tau_k}(o_s^{\tau_k}, a^k) \leq \max_{a^{-k}} q^\dagger(s, a^k, a^{-k}) + \gamma^n \big(q_U - \min_s \max_{\bar{a}} q^\dagger(s, \bar{a})\big) \qquad (4.42)$$

where $n \in \mathbb{N}$ is defined as before (i.e, it is the amount of times that $\mathcal{B}_E$ was selected out of N). Now we proceed to show that the probability that n is finite in any infinite sequence is zero. The probability that operator $\mathcal{B}_E$ is selected n times in a sequence of N (which we denote as $\mathbb{P}(n; N)$) can be bounded as follows:

$$\mathbb{P}(n; N) \overset{(b)}{\leq} \binom{N}{n} P^n (1-P)^{N-n} = \left(\frac{1}{n!} \prod_{k=N-n+1}^{N} k \right) P^n (1-P)^{N-n} \leq \frac{N^n}{n!} P^n (1-P)^{N-n} \tag{4.43}$$

$$\lim_{N \to \infty} \mathbb{P}(n; N) = 0 \tag{4.44}$$

where in (b) we upper bounded the probability of $\mathcal{B}_E$ with P, which is the highest probability assigned to any a^{-k} by ψ (i.e. $P = \max_{a^{-k}} \mathbb{P}_\psi(a^{-k})$) and then calculated the probability using the binomial distribution[6]. Since $\lim_{N \to \infty} \mathbb{P}(n; N) = 0$, it follows that the probability of n being finite in an infinite sequence is also 0 (because it is the finite sum of probabilities which tend to 0). Therefore we can conclude:

$$\lim_{N \to \infty} (\mathcal{B}_U^\psi)^N q_U^{\tau_k}(o_s^{\tau_k}, a^k) \overset{\text{w.p. } 1}{\leq} \max_{a^{-k}} q^\dagger(s, a^k, a^{-k}) \tag{4.45}$$

We conclude the proof by lower bounding the mean convergence rate as follows:

$$\mathbb{E}\,(\gamma^{\boldsymbol{n}}) \overset{(c)}{\geq} \gamma^{\mathbb{E}\,\boldsymbol{n}} \overset{(d)}{\geq} (\gamma^p)^N \tag{4.46}$$

where in (c) we used Jensen's inequality and in (d) we lower bounded the probability $\mathbb{P}(n; N)$ using a binomial distribution with probability p, where p is the lowest probability assigned to any a^{-k} by ψ (i.e. $p = \min_{a^{-k}} \mathbb{P}_\psi(a^{-k})$).

[6] n follows a binomial distribution because the sampled $a^{-k} \sim \psi$ in successive applications of operator $\mathcal{B}_E$ are independent of each other.

4.E Proof of Lemma 4.4

The first part of the proof consists in showing that any sequence of the form $\mathcal{B}_I \cdots \mathcal{B}_E^{K_2} \mathcal{B}_I^{K_1} \mathcal{B}_E^{K_0} q_L^{\tau_k}(o_s^{\tau_k}, a^k)$ is lower bounded by the sequence $\mathcal{B}_I^n q_L^{\tau_k}(o_s^{\tau_k}, a^k)$ where $K_k \in \mathbb{N}$ and $n \in \mathbb{N}$ is the number of times that operator $\mathcal{B}_I$ is applied in the aforementioned sequence. We start defining $q_L^{\tau_k}(o_s^{\tau_k}, a^k) = q_L$ where $q_L \leq q_{\min}$. If we apply $\mathcal{B}_E$ to $q_L(o_s^{\tau_k}, a^k)$ we get:

$$\mathcal{B}_E q_L^{\tau_k}(o_s^{\tau_k}, a^k) \geq q_L = q_L^{\tau_k}(o_s^{\tau_k}, a^k) \tag{4.47}$$

$$\mathcal{B}_E^{K_0} q_L^{\tau_k}(o_s^{\tau_k}, a^k) \geq q_L^{\tau_k}(o_s^{\tau_k}, a^k) \tag{4.48}$$

Therefore, we get $\mathcal{B}_I \mathcal{B}_E^{K_0} q_L^{\tau_k}(o_s^{\tau_k}, a^k) \geq \mathcal{B}_I q_L^{\tau_k}(o_s^{\tau_k}, a^k)$. More specifically, we can write:

$$\mathcal{B}_I q_L^{\tau_k}(o_s^{\tau_k}, a^k) = \max\left\{q_L, \max_{a^{-k}} r(s, a^k, a^{-k}) + \gamma q_L\right\} = \max_{a^{-k}} r(s, a^k, a^{-k}) + \gamma q_L \tag{4.49}$$

$$\mathcal{B}_I^{K_1} q_L^{\tau_k}(o_s^{\tau_k}, a^k) = \max_{a_0^{-k}, \bar{a}_1, \cdots, \bar{a}_{K_1-1}} \mathbb{E}\left(\sum_{i=0}^{K_1-1} \gamma^i r(\boldsymbol{s}_i, a_i^k, a_i^{-k}) | \boldsymbol{s}_0 = s\right) + \gamma^{K_1} q_L \tag{4.50}$$

where like in the previous section we defined $\mathbb{E}_{\mathcal{P},n} \boldsymbol{r}(s, a^k, a^{-k}, \boldsymbol{s'}) = r(s, a^k, a^{-k})$. Applying $\mathcal{B}_E$ we get:

$$\mathcal{B}_E \mathcal{B}_I^{K_1} q_L^{\tau_k}(o_s^{\tau_k}, a^k) = \max_{\bar{a}_1, \cdots, \bar{a}_{K_1}} \mathbb{E}\left(\sum_{i=0}^{K_1} \gamma^i r(\boldsymbol{s}_i, a_i^k, a_i^{-k}) | \boldsymbol{s}_0 = s\right)\Bigg|_{\substack{a_0^k = a^k \\ a_0^n = \arg\max_{a^n} \mathcal{B}_I^{K_1} q_L^{\tau_n}(o_s^{\tau_n}, a^n) \forall n \neq k}} + \gamma^{K_1+1} q_L \tag{4.51}$$

$$\mathcal{B}_E^2 \mathcal{B}_I^{K_1} q_L^{\tau_k}(o_s^{\tau_k}, a^k) = \max_{\bar{a}_1, \cdots, \bar{a}_{K_1+1}} \mathbb{E}\left(\sum_{i=0}^{K_1+1} \gamma^i r(\boldsymbol{s}_i, a_i^k, a_i^{-k}) | \boldsymbol{s}_0 = s\right)\Bigg|_{\substack{a_0^k = a^k \\ a_0^n = \arg\max_{a^n} \mathcal{B}_E \mathcal{B}_I^{K_1} q_L^{\tau_n}(o_s^{\tau_n}, a^n) \forall n \neq k}} + \gamma^{K_1+2} q_L \tag{4.52}$$

Expanding $\arg\max_{a^n} \mathcal{B}_E \mathcal{B}_I^{K_1} q_L^{\tau_n}(o_s^{\tau_n}, a^n) \ \forall n \neq k$ we get:

$$\begin{aligned}
&\arg\max_{a^n} \mathcal{B}_E \mathcal{B}_I^{K_1} q_L^{\tau_n}(o_s^{\tau_n}, a^n) = \\
&\arg\max_{a_0^n} \max_{\bar{a}_1,\cdots,\bar{a}_{K_1-1}} \mathbb{E}\left(\sum_{i=0}^{K_1} \gamma^n r(\boldsymbol{s}_i, a_i^n, a_i^{-n}) | \boldsymbol{s}_0 = s\right)\Big|_{a_0^\ell = \arg\max_{a^\ell} \mathcal{B}_I^{K_1} q_L^{\tau_\ell}(o_s^{\tau_\ell}, a^\ell) \ \forall \ell \neq n} \\
&\overset{(a)}{=} \arg\max_{a_0^n} \max_{a_0^{-n}, \bar{a}_1,\cdots,\bar{a}_{K_1-1}} \mathbb{E}\left(\sum_{i=0}^{K_1} \gamma^n r(\boldsymbol{s}_i, a_i^n, a_i^{-n}) | \boldsymbol{s}_0 = s\right) = \arg\max_{a^n} \mathcal{B}_I^{K_1} q_L^{\tau_n}(o_s^{\tau_n}, a^n) \quad (4.53)
\end{aligned}$$

where in (a) we used equation (4.50). Combining (4.53) with (4.52) we get :

$$\begin{aligned}
\mathcal{B}_E^2 \mathcal{B}_I^{K_1} q_L^{\tau_k}(o_s^{\tau_k}, a^k) &= \max_{\bar{a}_1,\cdots,\bar{a}_{K_1+1}} \mathbb{E}\left(\sum_{i=0}^{K_1+1} \gamma^i r(\boldsymbol{s}_i, a_i^k, a_i^{-k}) | \boldsymbol{s}_0 = s\right)\Big|_{a_0^n = \arg\max_{a^n} \mathcal{B}_I^{K_1} q_L^{\tau_n}(o_s^{\tau_n}, a^n) \ \forall n \neq k} + \gamma^{K_1+2} q_L \\
&\geq \mathcal{B}_E \mathcal{B}_I^{K_1} q_L^{\tau_k}(o_s^{\tau_k}, a^k) \quad (4.54)
\end{aligned}$$

Therefore, it follows:

$$\begin{aligned}
\mathcal{B}_E^{K_2} \mathcal{B}_I^{K_1} q_L^{\tau_k}(o_s^{\tau_k}, a^k) &\geq \mathcal{B}_E \mathcal{B}_I^{K_1} q_L^{\tau_k}(o_s^{\tau_k}, a^k) \quad (4.55) \\
&\geq \max_{\bar{a}_1,\cdots,\bar{a}_{K_1-1}} \mathbb{E}\left(\sum_{i=0}^{K_1-1} \gamma^i r(\boldsymbol{s}_i, a_i^k, a_i^{-k}) | \boldsymbol{s}_0 = s\right)\Big|_{a_0^n = \arg\max_{a^n} \mathcal{B}_I^{K_1} q_L^{\tau_n}(o_s^{\tau_n}, a^n) \ \forall n \neq k} + \gamma^{K_1} q_L \quad (4.56)
\end{aligned}$$

Applying $\mathcal{B}_I$ to (4.56) we get:

$$\mathcal{B}_I \mathcal{B}_E^{K_2} \mathcal{B}_I^{K_1} q_L^{\tau_k}(o_s^{\tau_k}, a^k) \geq \max_{\bar{a}_0,\cdots,\bar{a}_{K_1}} \mathbb{E}\left(\sum_{i=0}^{K_1} \gamma^i r(\boldsymbol{s}_i, a_i^k, a_i^{-k}) | \boldsymbol{s}_0 = s\right) + \gamma^{K_1+1} q_L = \mathcal{B}_I^{K_1+1} q_L^{\tau_k}(o_s^{\tau_k}, a^k) \quad (4.57)$$

Therefore, we conclude that any sequence of the form $\mathcal{B}_I \cdots \mathcal{B}_E^{K_2} \mathcal{B}_I^{K_1} \mathcal{B}_E^{K_0} q_L^{\tau_k}(o_s^{\tau_k}, a^k)$ is lower bounded by the sequence $\mathcal{B}_I^n q_L^{\tau_k}(o_s^{\tau_k}, a^k)$ where n is the number of times that operator $\mathcal{B}_I$ is applied in the aforementioned sequence. Furthermore, from equation (4.50) we see that as $n \to \infty$ $\mathcal{B}_I^n q_L^{\tau_k}(o_s^{\tau_k}, a^k)$ converges to $\max_{a^{-k}} q^\dagger(s, a^k, a^{-k})$ at an exponential rate with Lipschitz

constant γ:

$$\lim_{n\to\infty} \mathcal{B}_I^n q_L^{\tau_k}(o_s^{\tau_k}, a^k) = \lim_{n\to\infty} \max_{a^{-k}} q^\dagger(s, a^k, a^{-k}) + \gamma^n \big(q_L - \max_{\bar{a}} q^\dagger(\boldsymbol{s}_{K_1+1}, \bar{a})\big) = \max_{a^{-k}} q^\dagger(s, a^k, a^{-k}) \tag{4.58}$$

A corollary of the previous statement is that any sequence of the form $\mathcal{B}_E \cdots \mathcal{B}_E^{K_2} \mathcal{B}_I^{K_1} \mathcal{B}_E^{K_0} q_L^{\tau_k}(o_s^{\tau_k}, a^k)$ is lower bounded by the sequence $\mathcal{B}_E \mathcal{B}_I^n q_L^{\tau_k}(o_s^{\tau_k}, a^k)$ (note that this sequence ends with the application of $\mathcal{B}_E$ as opposed to $\mathcal{B}_I$, which is the case for the sequence studied in the previous paragraph). In this case the sequence still converges at the same rate to $q^\star(o_s^{\tau_k}, a^k)$ as opposed to $\max_{a^{-k}} q^\dagger(s, a^k, a^{-k})$. This can easily be seen as follows:

$$\begin{aligned} \lim_{n\to\infty} \mathcal{B}_E \mathcal{B}_I^n q_L^{\tau_k}(o_s^{\tau_k}, a^k) &= \lim_{n\to\infty} \mathcal{B}_E \Big[\max_{a^{-k}} q^\dagger(s, a^k, a^{-k}) + \gamma^n \big(q_L - \max_{\bar{a}} q^\dagger(\boldsymbol{s}_{K_1+1}, \bar{a})\big) \Big] \\ &= \mathcal{B}_E \max_{a^{-k}} q^\dagger(s, a^k, a^{-k}) = q^{\tau_k, \star}(o_s^{\tau_k}, a^k) \end{aligned} \tag{4.59}$$

Since by definition $q^{\tau_k,\star}(o_s^{\tau_k}, a^k) \leq \max_{a^{-k}} q^\dagger(s, a^k, a^{-k})$ and operator $\mathcal{B}_L^\psi$ applies $\mathcal{B}_I$ or $\mathcal{B}_E$ at random (and therefore a sequence of composed of several applications of $\mathcal{B}_L^\psi$ can end with either $\mathcal{B}_I$ or $\mathcal{B}_E$) we get:

$$(\mathcal{B}_L^\psi)^N q_L^{\tau_k}(o_s^{\tau_k}, a^k) \geq \mathcal{B}_E \Big[\max_{a^{-k}} q^\dagger(s, a^k, a^{-k}) + \gamma^n \big(q_L - \max_{\bar{a}} q^\dagger(\boldsymbol{s}_{K_1+1}, \bar{a})\big) \Big] \tag{4.60}$$

$$= q^{\tau_k,\star}(o_s^{\tau_k}, a^k) + \mathcal{O}(\gamma^n) \tag{4.61}$$

where $n \in \mathbb{N}$ is defined as before (i.e, it is the amount of times that $\mathcal{B}_I$ was selected out of N). The same arguments we used in the previous section to prove the convergence $w.p.1$ and lower bound the rate of convergence apply here without modification. Therefore, we conclude:

$$\lim_{N\to\infty} (\mathcal{B}_L^\psi)^N q_L(o_s^{\tau_k}, a^k) \overset{\text{w.p. } 1}{\geq} q^{\tau_k,\star}(o_s^{\tau_k}, a^k) \tag{4.62}$$

$$\mathbb{E}(\gamma^{\boldsymbol{n}}) \geq \gamma^{\mathbb{E}\boldsymbol{n}} \geq \gamma^{pN} \tag{4.63}$$

Algorithm 4.2 Tabular *Logical Team Q-Learning* for deterministic TMDPs

Initialize: an empty replay buffer $\mathcal{R}$ and estimates $\widehat{q}^{\tau_k}$.
for iterations $e = 0, \dots, E$ **do**
  Sample T transitions $(o_s^{\tau_1}, \cdots, o_s^{\tau_K}, \bar{a}, r, o_{s'}^{\tau_1}, \cdots, o_{s'}^{\tau_K})$ by following some behavior policy which guarantees all joint actions are sampled with non-zero probability and store them in $\mathcal{R}$.
  for iterations $i = 0, \dots, I$ **do**
    Sample a transition $(o_s^{\tau_1}, \cdots, o_s^{\tau_K}, \bar{a}, r, o_{s'}^{\tau_1}, \cdots, o_{s'}^{\tau_K})$ from $\mathcal{R}$.
    for agent $k = 1, \cdots, K$ **do**
      if $\big(a^n = \arg\max_{a^n} \widehat{q}^{\tau_n}(o_s^{\tau_n}, a^n)\ \forall n \neq k\big)$ **then**
        $\widehat{q}^{\tau_k}(o_s^{\tau_k}, a^k) = \widehat{q}^{\tau_k}(o_s^{\tau_k}, a^k) + \mu\big(r + \max_a \widehat{q}^{\tau_k}(o_{s'}^{\tau_k}, a) - \widehat{q}^{\tau_k}(o_s^{\tau_k}, a^k)\big)$
      else if $\big(r + \max_a \widehat{q}^{\tau_k}(o_{s'}^{\tau_k}, a) > \widehat{q}^{\tau_k}(o_s^{\tau_k}, a^k)\big)$ **then**
        $\widehat{q}^{\tau_k}(o_s^{\tau_k}, a^k) = \widehat{q}^{\tau_k}(o_s^{\tau_k}, a^k) + \mu\alpha\big(r + \max_a \widehat{q}^{\tau_k}(o_{s'}^{\tau_k}, a) - \widehat{q}^{\tau_k}(o_s^{\tau_k}, a^k)\big)$
      end if
    end for
  end for
end for

4.F Tabular *Logical Team Q-Learning*

In the case where the TMDP is deterministic it holds:

$$\mathbb{E}\left(\boldsymbol{r}(s, \bar{a}, \boldsymbol{s'}) + \gamma \max_{a'} q^{\tau_k}(\boldsymbol{o}_{\boldsymbol{s'}}^{\boldsymbol{k}}, a')\right) = r(s, \bar{a}, s') + \gamma \max_{a'} q^{\tau_k}(o_{s'}^k, a') \tag{4.64}$$

For this particular case, the tabular version of *LTQL* is given by algorithm 4.2.

If algorithm 4.2 were applied to a stochastic TMDP, due to condition c_2 (given by $r + \max_a \widehat{q}^{\tau_k}(o_{s'}^{\tau_k}, a) > \widehat{q}^{\tau_k}(o_s^{\tau_k}, a^k)$), it would be subject to bias, which would propagate through bootstrapping and hence could compromise its ability to find optimal team policies. This can be solved by having a second unbiased estimate q_U that is updated only when c_1 is satisfied and use this unbiased estimate to bootstrap. The resulting algorithm is shown in algorithm 4.3.

Algorithm 4.3 Tabular *Logical Team Q-Learning*

Initialize: an empty replay buffer $\mathcal{R}$ and estimates $\widehat{q}_B^{\tau_k}$ and $\widehat{q}_U^{\tau_k}$.
for iterations $e = 0, \dots, E$ **do**
  Sample T transitions $(o_s^{\tau_1}, \cdots, o_s^{\tau_K}, \bar{a}, r, o_{s'}^{\tau_1}, \cdots, o_{s'}^{\tau_K})$ by following some behavior policy and store them in $\mathcal{R}$.
  for iterations $i = 0, \dots, I$ **do**
    Sample a transition $(o_s^{\tau_1}, \cdots, o_s^{\tau_K}, \bar{a}, r, o_{s'}^{\tau_1}, \cdots, o_{s'}^{\tau_K})$ from $\mathcal{R}$.
    for agent $k = 1, \cdots, K$ **do**
      if $a^n = \arg\max_{a^n} \widehat{q}_B^{\tau_n}(o_s^{\tau_n}, a^n)\ \forall n \neq k$ **then**
        $\widehat{q}_B^{\tau_k}(o_s^{\tau_k}, a^k) = \widehat{q}_B^{\tau_k}(o_s^{\tau_k}, a^k) + \mu\big(r + \max_a \widehat{q}_U^{\tau_k}(o_{s'}^{\tau_k}, a) - \widehat{q}_B^{\tau_k}(o_s^{\tau_k}, a^k)\big)$
        $\widehat{q}_U^{\tau_k}(o_s^{\tau_k}, a^k) = \widehat{q}_U^{\tau_k}(o_s^{\tau_k}, a^k) + \mu\big(r + \max_a \widehat{q}_U^{\tau_k}(o_{s'}^{\tau_k}, a) - \widehat{q}_U^{\tau_k}(o_s^{\tau_k}, a^k)\big)$
      end if
      if $\big(r + \max_a \widehat{q}_U^{\tau_k}(o_{s'}^{\tau_k}, a) > \widehat{q}_B^{\tau_k}(o_s^{\tau_k}, a^k)\big)$ **then**
        $\widehat{q}_B^{\tau_k}(o_s^{\tau_k}, a^k) = \widehat{q}_B^{\tau_k}(o_s^{\tau_k}, a^k) + \mu\alpha\big(r + \max_a \widehat{q}_U^{\tau_k}(o_{s'}^{\tau_k}, a) - \widehat{q}_B^{\tau_k}(o_s^{\tau_k}, a^k)\big)$
      end if
    end for
  end for
end for

4.G Bull's policy

The bull's policy is given by the pseudocode shown in algorithm 4.4.

Algorithm 4.4 Bull's policy.

if distance to all predators ¿ 10 **then**
  Natural foraging behavior: Stay still with 90% probability, otherwise make a small move in a random direction.
else
  if the maximum angle formed by two predators is ¿ 108^o **then**
    There's a hole to escape: Escape through the direction in between these two predators.
  else if distance to farthest predator - distance to closest predator ¿ 5 **then**
    There's no hole, but one predator is much closer than the others so run in the direction opposite to this predator.
  else
    No way out (scared): Stay still with 70% probability, otherwise make a fast move in a random direction.
  end if
end if

CHAPTER 5

ISL: A Novel Approach for Deep Exploration

In section 3.5 of chapter 3 we identified two main limitations with the DTPO algorithm, the problem of efficient exploration and the problem of learning factored policies. In this chapter we address the problem of exploration. The problem of deep exploration is still an open question in single-agent RL, hence in this chapter we consider the single agent case.

The DTPO algorithm we derived in Chapter 3 is based in the maximum-entropy RL framework. Several single-agent algorithms have been introduced based on the maximum entropy framework [28–34]. In maximum-entropy RL, the traditional RL objective is augmented with the entropy of the learned policy, which is weighted by a temperature parameter. The main benefit of the maximum RL framework is that it allows to derive algorithms that are more sample efficient (because they operate off-policy) than policy gradients algorithms like A3C [19], TRPO [20] and PPO [21] and exhibit improved stability over algorithms based on Q-learning. Furthermore, it has been demonstrated empirically that algorithms based on the maximum entropy framework have improved exploration capabilities [29]. Although these recent algorithms have shown state of the art performance in several tasks, they are not without shortcomings. In the first place, augmenting the original RL objective with the entropy bonuses biases the solution of the optimization problem. This could be solved by slowly annealing the temperature parameter, but it is unclear what is the best way to do so and how this change affects convergence speed. The second, and more important, shortcoming is that although these algorithms have improved exploration capabilities, they are still unable to perform deep exploration (i.e., they tend to perform poorly in environments with very sparse reward structures).

These difficulties suggest that there may not be sufficient theoretical justification for

the use of the policy's entropy as the regularizer for the RL objective, which in turn raises the question of whether another regularizer could help address the two aforementioned challenges. In this chapter we explore this possibility and connect the problem of deep exploration with that of maximum entropy RL. Traditionally, the problem of deriving deep exploration strategies has been treated separately from that of deriving the learning equations [104–108]. In this chapter we show that both the learning equations and the exploration strategy can be derived in tandem as the solution to a well-posed optimization problem whose minimization leads to the optimal value function. We do this by augmenting the objective function with a novel regularizer. Instead of using the policy's entropy as the regularizer (as maximum-entropy RL does), we use a function which depends on the uncertainties over the agent's estimates of the q values.

The contribution of this chapter is the introduction of an alternative approach to address *deep exploration* and the introduction of an algorithm that we refer to as the *Information Seeking Learner* (ISL). This algorithm has a similar form to soft RL algorithms like SAC and SBEED, but the fundamental difference is that it explicitly estimates the uncertainties of its q estimates and uses these uncertainties to drive deep exploration.

5.1 Related Works

ISL is related to recent work on maximum entropy algorithms [28–33]. All these algorithms augment the traditional RL objective with a term that aims to maximize the entropy of the learned policy, which is weighted by a temperature parameter. The consequence of using this augmented objective is two-fold. First, it allows to derive off-policy algorithms with improved stability compared to algorithms based on Q-learning. Second, it improves the exploration properties of the algorithms. However, using this augmented objective has two main drawbacks. First, the policy to which these algorithms converge is biased away from the true optimal policy. This point can be handled by annealing the temperature parameter but this can slow down convergence and compromise exploration. Furthermore, it is unclear what the optimal schedule to perform annealing is and how it affects the conditioning of

the optimization problem. Secondly, although exploration is improved, algorithms derived from this modified cost are not efficient at performing deep exploration. The reason for this is that a unique temperature parameter is used for all states. In order to perform deep exploration it is necessary to have a scheme that allows agents to learn policies that exploit more in states where the agent has high confidence in the optimal action (in order to be able to reach further novel states with high probability) and act in a more exploratory manner in unknown states. In chapter 3 we explored this possibility in a tabular setting making the temperature parameter state dependent and annealing it progressively as states are explored; however, the annealing schedule is determined heuristically and it is unclear what would be an effective way to do so when using function approximation. The relation between ISL and these works is that we also augment the RL objective with an entropy based regularizer and as a consequence we derive a an algorithm that relies on a soft backup. Under our scheme, agents converge to the true optimal policy without the need for annealing any parameter and, moreover, an exploration strategy is derived that is capable of performing deep exploration.

The work we present in this chapter is also related to works on deep exploration strategies. One line of research is based on posterior sampling for RL [105, 106, 108]. The main idea in these works is to sample the value function from a posterior distribution over such function. The common point between these works and ISL is that deep exploration is driven by some measure of the uncertainty on the estimated value function. One difference between these works and ours is that they rely on Posterior Sampling while our work does not. Another fundamental difference is that these works treat the exploration problem separately from the derivation of the learning rules. In other words, they use heuristics based on randomised value functions to drive exploration but use off-the-shelf learning rules like Q-learning. In our approach, we use the uncertainty over the learned value function not only to drive exploration but to formulate a new objective from which we derive a new learning rule. Another line of work relies on pseudo-counts [109, 110], the fundamental limitation of this approach is that it only performs good if the generalization of the density model is aligned with the task objective [108]. Another line of research relies on intrinsic rewards (IRs) [104, 111–113], these rewards are based on some heuristic based proxy that reflects the novelty of a state. ISL can

also be seen as utilizing IRs, however there are two main differences with the previous works. Firstly, we use IRs that depend on the uncertainty of the q function. Secondly, we never actually compute IRs, but rather we rely on a different backup to update the q estimates (this point is clarified in section 5.3).

5.2 Problem Setting

We consider the problem of policy optimization within the traditional RL framework. We model our setting as an MDP defined by ($\mathcal{S}$,$\mathcal{A}$,$\mathcal{P}$,r), where $\mathcal{S}$ is a set of states, $\mathcal{A}$ is a set of actions of size $A = |\mathcal{A}|$, $\mathcal{P}(s'|s,a)$ specifies the probability of transitioning to state $s' \in \mathcal{S}$ from state $s \in \mathcal{S}$ having taken action $a \in \mathcal{A}$, and $r : \mathcal{S} \times \mathcal{A} \times \mathcal{S} \to \mathbb{R}$ is the reward mapping $r(s,a,s')$, which can be a random variable with distribution $n_{s,a,s'}(r)$.

Assumption 5.1. $r_{\min} \leq \mathbb{E}_n \boldsymbol{r}(s,a,s') \leq r_{\max}$, $\forall (s,a,s')$.

We further recall that the optimal policy is given by:

$$\pi^\dagger(a|s) = \arg\max_\pi \mathbb{E}_{\mathcal{P},\pi,n}\left(\sum_{t=0}^{\infty} \gamma^t \boldsymbol{r}(\boldsymbol{s}_t, \boldsymbol{a}_t, \boldsymbol{s}_{t+1}) \big| \boldsymbol{s}_0 = s \right) \tag{5.1}$$

which can also be rewritten as a function of the optimal state-action value function as follows:

$$q^\dagger(s,a) = r(s,a) + \gamma \mathbb{E}_{\mathcal{P}} \max_{a'} q^\dagger(\boldsymbol{s}', a') \tag{5.2a}$$

$$\pi^\dagger(a|s) = \arg\max_\pi \sum_a \pi(a|s) q^\dagger(s,a) \tag{5.2b}$$

where as usual we defined $r(s,a) = \mathbb{E}_{\mathcal{P},n} \boldsymbol{r}(s,a,\boldsymbol{s}')$. Equations (5.2) are useful to derive algorithms for planning problems (i.e., problems in which the reward function and transition kernel are known) but may be unfit to derive RL algorithms because they mask the fact that the agent relies on estimated quantities (which are subject to uncertainty). Hence, in this work, we modify (5.1) to reflect the fact that an RL agent is constrained by the uncertainty of its estimates. Intuitively, we change the goal of the agent to not just maximize the discounted

cumulative rewards but also to collect information about the MDP in order to minimize the uncertainty of its estimated quantities. For this purpose, we assume that at any point in time the agent has some estimate of the optimal value function denoted by $\widehat{q}(s,a)$, which is subject to some error $\widetilde{q}(s,a) = q^{\dagger}(s,a) - \widehat{q}(s,a)$. We model the unknown quantities $\widetilde{q}(s,a)$ as random variables. More specifically, for each state-action pair we assume $\widetilde{\boldsymbol{q}}(s,a)$ follows a uniform probability distribution with zero mean $\widetilde{\boldsymbol{q}}(s,a) \sim \mathrm{U}(0, \ell(s,a))$ such that:

$$\widetilde{q}(s,a) \in [\widehat{q}(s,a) - \ell(s,a); \widehat{q}(s,a) + \ell(s,a)] \tag{5.3}$$

We will refer to the probability density function of $\widetilde{\boldsymbol{q}}(s,a)$ as $d_{(s,a)}(\widetilde{q})$. We assume zero mean uniform distributions for the following reasons. First, if the mean were different than zero, it could be used to improve the estimate $\widehat{q}(s,a)$ as $\widehat{q}(s,a) \leftarrow \widehat{q}(s,a) + \mathbb{E}\,\widetilde{\boldsymbol{q}}(s,a)$ resulting in a new estimate whose corresponding error would be zero mean. Secondly, under assumption 1, we know that for any infinitely discounted MDP, a symmetric bound for the error of the value function exists in the form $-\ell(s,a) < \widetilde{\boldsymbol{q}}(s,a) < \ell(s,a)$.[1] Moreover, typically there is no prior information about the error distribution between these bounds and therefore a non-informative uniform distribution becomes appropriate.

We further define the state uncertainty distribution $u_s^{\pi}(\widetilde{q})$ (which is given by a mixture of the state-action error distributions) and the Maximum-Uncertainty-Entropy policy $\pi^{\bullet}$ (which at every state chooses the action whose corresponding $\widetilde{\boldsymbol{q}}(s,a)$ has the greatest uncertainty):

$$u_s^{\pi}(\widetilde{q}) = \mathbb{E}_{\pi} d_{(s,\boldsymbol{a})}(\widetilde{q}), \qquad \pi^{\bullet}(a|s) = \begin{cases} 1, \text{ if } \ \ell(s,a) = \max_a \ell(s,a) \\ 0, \text{ else} \end{cases} \tag{5.4}$$

We thus define the goal of our reinforcement learning agent to be:

$$\pi^{\star}(a|s) = \arg\max_{\pi} \mathbb{E}_{\mathcal{P},\pi,n}\left(\sum_{t=0}^{\infty} \gamma^t \big[\boldsymbol{r}(\boldsymbol{s}_t, \boldsymbol{a}_t, \boldsymbol{s}_{t+1}) - \kappa D_{KL}(u^{\pi}_{\boldsymbol{s}_t}(\widetilde{q}) || u^{\bullet}_{\boldsymbol{s}_t}(\widetilde{q})) \big] \Big| \boldsymbol{s}_0 = s \right) \tag{5.5}$$

[1] This is due to the fact that the value functions are lower and upper bounded by $r_{\min}/(1-\gamma)$ and $r_{\max}/(1-\gamma)$.

where κ is a positive regularization parameter, D_{KL} is the Kullback-Leibler divergence, and $u^{\bullet}_{\boldsymbol{s}_t}(\widetilde{q})$ is the state uncertainty distribution corresponding to $\pi^{\bullet}$. In this work we refer to $\pi^{\star}(a|s)$ as the *uncertainty constrained* optimal policy (or *uc*-optimal policy). Under this new objective, we redefine the value functions as:

$$q^{\pi}(s,a) = r(s,a) + \gamma \mathbb{E}_{\mathcal{P}} v^{\pi}(\boldsymbol{s}') \tag{5.6a}$$

$$\begin{aligned} v^{\pi}(s) &= \mathbb{E}\left(\sum_{t=0}^{\infty} \gamma^t \left[\boldsymbol{r}(\boldsymbol{s}_t, \boldsymbol{a}_t, \boldsymbol{s}_{t+1}) - \kappa D_{KL}(u^{\pi}_{\boldsymbol{s}_t}(\widetilde{q})) \right] \big| \boldsymbol{s}_0 = s \right) \\ &= \mathbb{E}\left(r(s,a) - \kappa D_{KL}(u^{\pi}_{s}(\widetilde{q})) + \gamma v^{\pi}(\boldsymbol{s}') \right) \end{aligned} \tag{5.6b}$$

where, with a little abuse of notation, we defined $D_{KL}(u^{\pi}_{s}(\widetilde{q})) \triangleq D_{KL}(u^{\pi}_{s}(\widetilde{q}) || u^{\bullet}_{s}(\widetilde{q}))$. Using (5.6) we can rewrite (5.5) as:

$$\pi^{\star} = \arg\max_{\pi} \sum_{a} \pi(a|s) \widetilde{q}(s,a) - \kappa D_{KL}(u^{\pi}_{s}(\widetilde{q})) \tag{5.7}$$

Note that the exploration-exploitation trade-off becomes explicit in (5.7). To maximize the first term of the summation in (5.7), the agent has to exploit its knowledge of $\widehat{q}$, while to maximize the second term, the agent's policy needs to match $\pi^{\bullet}$, which is a policy that seeks to maximize the information gathered through exploration.

5.3 Algorithm Derivation

Since the argument being maximized in (5.7) is differentiable with respect to $\pi(a|s)$ we can obtain a closed-form expression for $\pi^{\star}$. Before presenting such closed-form expression we introduce the following useful remark, lemmas and definitions.

Remark 5.1. *In the interest of simplifying equations and notation we clarify that from now on we will use $\ell_i(s) \triangleq \ell(s, a_i)$ and even $\ell_i \triangleq \ell(s, a_i)$ when the state s is clear from the context.*

Lemma 5.1. *Assuming the actions are ordered such that $\ell_i > \ell_j \iff i > j$, the KL*

divergence term in (5.7) *for a given state* s *is given by:*

$$\sum_{n=1}^{A}(\ell_n - \ell_{n-1})\left(\sum_{k=n}^{A}\frac{\pi(a_k|s)}{\ell_k}\right)\log\left[\sum_{b=n}^{A}\frac{\pi(b|s)\ell_A}{\ell_b}\right] \tag{5.8}$$

Proof. The proof is included in appendix 5.A. □

Definition 5.1. ***Pareto dominated action****: For a certain state* s *we say that an action* a_j *is Pareto dominated by action* a_i *if* $\widehat{q}(s,a_j) \leq \widehat{q}(s,a_i)$ *and* $\ell(s,a_j) < \ell(s,a_i)$*.*

Definition 5.2. ***Mixed Pareto dominated action****: For a certain state* s *we say that an action* a_k *is mixed Pareto dominated if there exist two actions* a_i *and* a_j *that satisfy:*

$$\ell(s,a_i) > \ell(s,a_k) > \ell(s,a_j), \qquad 1 < \frac{(\ell_i - \ell_k)\,\ell_j\widehat{q}(s,a_j) + (\ell_k - \ell_j)\,\ell_i\widehat{q}(s,a_i)}{(\ell_i - \ell_j)\,\ell_k\widehat{q}(s,a_k)} \tag{5.9}$$

Definition 5.3. ***Pareto optimal action****: We define an action* a *as Pareto optimal if it is not Pareto dominated or mixed Pareto dominated.*

Intuitively (mixed) Pareto dominated actions are actions that the agent should not choose because there is another action (or group of actions) that is better in terms of both exploration and exploitation.

Lemma 5.2. *For all Pareto dominated and mixed Pareto dominated actions it holds that* $\pi^\star(a|s) = 0$*.*

Proof. The proof is included in appendix 5.B. □

The statement of Lemma 5.2 is intuitive since choosing a Pareto dominated action lowers the expected cumulative reward and the information gained, relative to choosing the action that dominates it. Also note that Lemma 5.2 implies that for all Pareto optimal actions it must be the case that if $q(s,a_i) < q(s,a_j)$ then $\ell(s,a_i) > \ell(s,a_j)$. We now introduce the state dependent set of actions $\mathcal{E}_s$ with cardinality $|\mathcal{E}_s|$, which is formed by all the Pareto optimal actions corresponding to state s. Furthermore, we introduce the ordering functions $\sigma_s(k) : [1, |\mathcal{E}_s|] \to [1, A]$, which for every state provide an ordering amongst the Pareto

optimal actions from lowest uncertainty to highest (i.e., $\ell(s,\sigma_s(i)) > \ell(s,\sigma_s(j)) \iff i > j$). For instance, $\sigma_s(1)$ provides the index of the action at state s that has the lowest uncertainty amongst the actions contained in $\mathcal{E}_s$.

Theorem 5.1. *$\pi^\star(a|s)$ is given by:*

$$\pi^\star(a|s) = \begin{cases} \frac{\ell_{\sigma(j)}(p_j(s)-p_{j+1}(s))}{|\mathcal{E}_s|}, & \text{if } a = \sigma_s(j) \text{ for some } j \in [1, |\mathcal{E}_s|] \\ \sum_{j=1}^{} (\ell_{\sigma(j)}-\ell_{\sigma(j-1)})p_j(s) & \\ 0, & \text{otherwise} \end{cases} \tag{5.10a}$$

$$p_j(s) = \exp\left[\frac{\ell_{\sigma(j)}(s)\widehat{q}(s,\sigma(j)) - \ell_{\sigma(j-1)}(s)\widehat{q}(s,\sigma(j-1))}{\kappa\left(\ell_{\sigma(j)}(s) - \ell_{\sigma(j-1)}(s)\right)}\right] \tag{5.10b}$$

where to simplify notation we defined $\ell_{\sigma(j)}(s) = \ell_{\sigma_s(j)}(s)$ and $\widehat{q}(s,\sigma(j)) = \widehat{q}(s,\sigma_s(j))$ and we also set $p_{|\mathcal{E}_s|+1}(s) = 0$, $l_0(s) = 0$.

Proof. The proof is included in appendix 5.C. □

Note that as expected, $\pi^\star(a|s)$ is always strictly positive for Pareto optimal actions.

Lemma 5.3. *The value function corresponding to policy $\pi^\star(a|s)$ is given by:*

$$v^\star(s) = \kappa \log\left[\sum_{j=1}^{|\mathcal{E}_s|} \frac{\ell_{\sigma(j)}(s) - \ell_{\sigma(j-1)}(s)}{\ell_{\max}(s)} p_j(s)\right], \quad q^\star(s,a) = r(s,a) + \gamma \mathbb{E}\, v^\star(\boldsymbol{s'}) \tag{5.11}$$

where $\ell_{\max}(s) = \max_a \ell(s,a)$.

Proof. The proof follows by combining (5.10) with (5.6b) and (5.6a). □

Remark 5.2. *$\pi^\star(a|s)$ satisfies the following conditions:*

$$\lim_{\kappa\to 0^+} \pi^\star(a|s) = \pi^\dagger(a|s), \qquad \lim_{\ell_A \to \ell_{A-1}^+ \cdots \to \ell_1^+} \pi^\star(a|s) = \pi^\dagger(a|s) \tag{5.12}$$

The first condition is expected since when the relative entropy term is eliminated, (5.1) and (5.5) become equivalent. The second condition reflects the fact that when the uncertainty

is equal for all actions, the distributions $u_s^\pi(\widetilde{q})$ and $u_s^\bullet(\widetilde{q})$ become equal regardless of π and therefore $D_{KL}(u_s^\pi(\widetilde{q})) = 0$ and hence (5.1) and (5.5) become equivalent. The second condition of (5.12) is of fundamental importance because it guarantees that as the uncertainties over $\widehat{q}$ diminish (and therefore converge to some required threshold $\ell(s,a) \to \ell_{\min}$), policy $\pi^\star$ tends to the desired policy $\pi^\dagger$ (note that annealing of κ is not necessary for this convergence of $\pi^\star$ towards $\pi^\dagger$). Note that policy (5.10) has the previously discussed qualities that policies induced by maximum entropy RL do not. Namely, as learning progresses, the effect of the regularizer also diminishes, and hence so does the bias of $\pi^\star$ with respect to $\pi^\dagger$, without the need for annealing κ. Furthermore, the effect of the regularizer diminishes over time on a per state basis, allowing for a high degree of exploration in states where the agent has high uncertainty and high exploitation in states were the state has high certainty over its estimates.

5.3.1 Uncertainty Constrained Value Iteration

In a dynamic programing setting $q^\star(s,a)$ can be found by iteratively applying to any vector $q(s,a)$ the operator $\mathcal{T}^\ell$ defined by:

$$\mathcal{T}^\ell q(s,a) = r(s,a) + \gamma\kappa\mathbb{E}\log\left[\sum_{j=1}^{|\mathcal{E}_s|}\frac{\ell_{\sigma(j)}(\boldsymbol{s'}) - \ell_{\sigma(j-1)}(\boldsymbol{s'})}{\ell_{\max}(\boldsymbol{s'})}p_j(\boldsymbol{s'})\right] \tag{5.13}$$

where we make the ℓ explicit to highlight the fact that $q^\star(s,a)$ is a function of the uncertainties.

Lemma 5.4. ***ℓ-Policy Evaluation****: For any mapping $q^0 : \mathcal{S}\times\mathcal{A} :\to \mathbb{R}$, the sequence $q^{n+1} = \mathcal{T}^\ell q^n$ converges to $q^\star$.*

Proof. The proof follows by noting that $\mathcal{T}^\ell$ is a contraction mapping and applying Banach's fixed point theorem. See appendix 5.D. □

Note that Lemma 5.4 provides an algorithm to learn $q^\star$, however the ultimate goal of the RL agent is to learn $q^\dagger$. To accomplish this goal a mechanism to estimate ℓ is necessary.

5.3.2 Uncertainty Estimation

Recalling that $\widetilde{q}(s,a) = q^\dagger(s,a) - \widehat{q}(s,a)$ we can write:

$$\widetilde{q}(s,a) = r(s,a) + \gamma \mathbb{E}_{\boldsymbol{s'}} v^\dagger(\boldsymbol{s'}) - \widehat{q}(s,a) = \mathbb{E}\,\boldsymbol{\delta}(s,a,\boldsymbol{s'}) + \gamma \mathbb{E}_{\mathcal{P}} \widetilde{v}(\boldsymbol{s'}) \tag{5.14}$$

where $\boldsymbol{\delta}(s,a,\boldsymbol{s'}) = \boldsymbol{r}(s,a,\boldsymbol{s'}) + \gamma \widehat{v}(\boldsymbol{s'}) - \widehat{q}(s,a)$ and $v^\dagger(s) = \widehat{v}(s) + \widetilde{v}(s)$. Furthermore $\widehat{v}(s)$ is the estimate obtained using (5.11) and $\widehat{q}(s,a)$. We can now bound $\widetilde{v}(s')$ as follows:

$$v^\dagger(s) - \widehat{v}(s) = \max_a q^\dagger(s,a) - \widehat{v}(s) \overset{(a)}{\leq} \max_a q^\dagger(s,a) - \max_a \widehat{q}(s,a) \overset{(b)}{\leq} \max_a \ell(s,a) \tag{5.15}$$

where (a) follows from Jensen's inequality applied to $v^\star(s)$ and in (b) we applied (5.3). Combining (5.14) and (5.15) we get $|\widetilde{q}(s,a)| \leq |\mathbb{E}\,\boldsymbol{\delta}(s,a,\boldsymbol{s'})| + \gamma \mathbb{E} \max_a \ell(\boldsymbol{s'},a)$. Therefore updating $\ell(s,a)$ as:

$$\ell(s,a) \leftarrow |\mathbb{E}\,\boldsymbol{\delta}(s,a,\boldsymbol{s'})| + \gamma \mathbb{E} \max_a \ell(\boldsymbol{s'},a) \tag{5.16}$$

guarantees that condition (5.3) is satisfied.

Lemma 5.5. ***Uncertainty Constrained Policy Evaluation****: For any mapping $q^0 : \mathcal{S} \times \mathcal{A} :\to \mathbb{R}$, repeated application of ℓ-Policy Evaluation and* (5.16) *converges to $q^\dagger$.*

Proof. See Appendix 5.E. □

5.3.3 Information Seeking Learner

We now proceed to derive a practical approximation to Uncertainty Constrained Policy Evaluation. We start by noting that (5.16) is not adequate to design a stochastic algorithm because the $|\mathbb{E}\,\boldsymbol{\delta}(s,a,\boldsymbol{s'})|$ term has an expectation inside the absolute value operator. Therefore a stochastic approximation of the form $|\delta(s_t,a_t,s_{t+1})|$ would be biased (since the sample approximation $|\delta(s_t,a_t,s_{t+1})|$ approximates $\mathbb{E}|\boldsymbol{\delta}(s,a,\boldsymbol{s'})|$ instead of $|\mathbb{E}\,\boldsymbol{\delta}(s,a,\boldsymbol{s'})|$). In the particular case where the MDP is deterministic this is not an issue because $\boldsymbol{\delta}(s,a,\boldsymbol{s'})$ is a

deterministic quantity and therefore can be calculated with any sample transition (s, a, r, s'). But in the general case, we can have an estimator $\widehat{\rho}(s,a)$ of $\mathbb{E}\,\boldsymbol{\delta}(s,a,\boldsymbol{s'})$, which then can be used to estimate $|\mathbb{E}\,\boldsymbol{\delta}(s,a,\boldsymbol{s'})|$ as $|\rho(s,a)|$. With this consideration and equations (5.13) and (5.16) we can define the update equations for the tabular version of the algorithm we present in this chapter as:

$$q(s_t,a_t) = q(s_t,a_t) + \mu_q\left(\delta(s_t,a_t,s_{t+1})\right) \tag{5.17a}$$

$$\rho(s_t,a_t) = \rho(s_t,a_t) + \mu_\rho\left(\delta(s_t,a_t,s_{t+1}) - \rho(s_t,a_t)\right) \tag{5.17b}$$

$$\ell(s_t,a_t) = \ell(s_t,a_t) + \mu_\ell\big((1-\eta_1)|\delta(s_t,a_t,s_{t+1})| + \eta_1|\rho(s_t,a_t)| + \gamma\ell_{\max}(s_{t+1}) - \ell(s_t,a_t)\big) \tag{5.17c}$$

$$\delta(s_t,a_t,s_{t+1}) = r(s_t,a_t,s_{t+1}) + \gamma\widehat{v}(s_{t+1}) - \widehat{q}(s_t,a_t) \tag{5.17d}$$

where η_1 is a tunable hyperparameter. In cases where the MDP is deterministic then $\eta_1 = 0$. For stochastic games η_1 closer to 1 becomes more convenient. As we mentioned before we want to derive an approximation to Uncertainty Constrained Policy Evaluation suitable for practical applications, which typically require that the q, ρ and ℓ functions are parameterized using expressive function approximators such as neural networks. In this work we use the parameters ω, θ and ν to parameterize q, ρ and ℓ, respectively. To extend (5.17) to this general case, ω, θ and ν can be trained to minimize (5.18).

$$J_\rho(\theta) = 2^{-1}\mathbb{E}_{(\boldsymbol{s},\boldsymbol{a})\sim\psi}\left[\delta(\boldsymbol{s},\boldsymbol{a};\omega) - \rho(\boldsymbol{s},\boldsymbol{a};\theta)\right]^2 \tag{5.18a}$$

$$J_q(\omega) = 2^{-1}\mathbb{E}_{(\boldsymbol{s},\boldsymbol{a})\sim\psi}\big[(q_T(\boldsymbol{s},\boldsymbol{a};\widetilde{\omega}) - \hat{q}(\boldsymbol{s},\boldsymbol{a};\omega)) \cdot \big((1-\eta_2)(q_T(\boldsymbol{s},\boldsymbol{a};\widetilde{\omega}) - \hat{q}(\boldsymbol{s},\boldsymbol{a};\omega)) + \eta_2\rho(\boldsymbol{s},\boldsymbol{a};\widetilde{\theta})\big)\big] \tag{5.18b}$$

$$J_\ell(\nu) = 2^{-1}\mathbb{E}_{(\boldsymbol{s},\boldsymbol{a})\sim\psi}[\ell_T(\boldsymbol{s},\boldsymbol{a};\widetilde{\nu}) - \ell(\boldsymbol{s},\boldsymbol{a};\nu)]^2 \tag{5.18c}$$

$$q_T(\boldsymbol{s},\boldsymbol{a};\widetilde{\omega}) = r(\boldsymbol{s},\boldsymbol{a}) + \kappa\mathbb{E}\log\left[\sum_{j=1}^{|\mathcal{E}_s|}\frac{(\ell_{\sigma(j)}(\boldsymbol{s'}) - \ell_{\sigma(j-1)}(\boldsymbol{s'}))}{\ell_{\max}(\boldsymbol{s'})}p_j(\boldsymbol{s'})\right] \tag{5.18d}$$

$$\ell_T(s,a;\widetilde{\nu}) = (1-\eta_1)|\delta(s,a)| + \eta_1|\rho(s,a)| + \gamma\mathbb{E}_{\boldsymbol{s'}}\ell_{\max}(\boldsymbol{s'};\widetilde{\nu}) \tag{5.18e}$$

Algorithm 5.1 Information Seeking Learner (ISL)

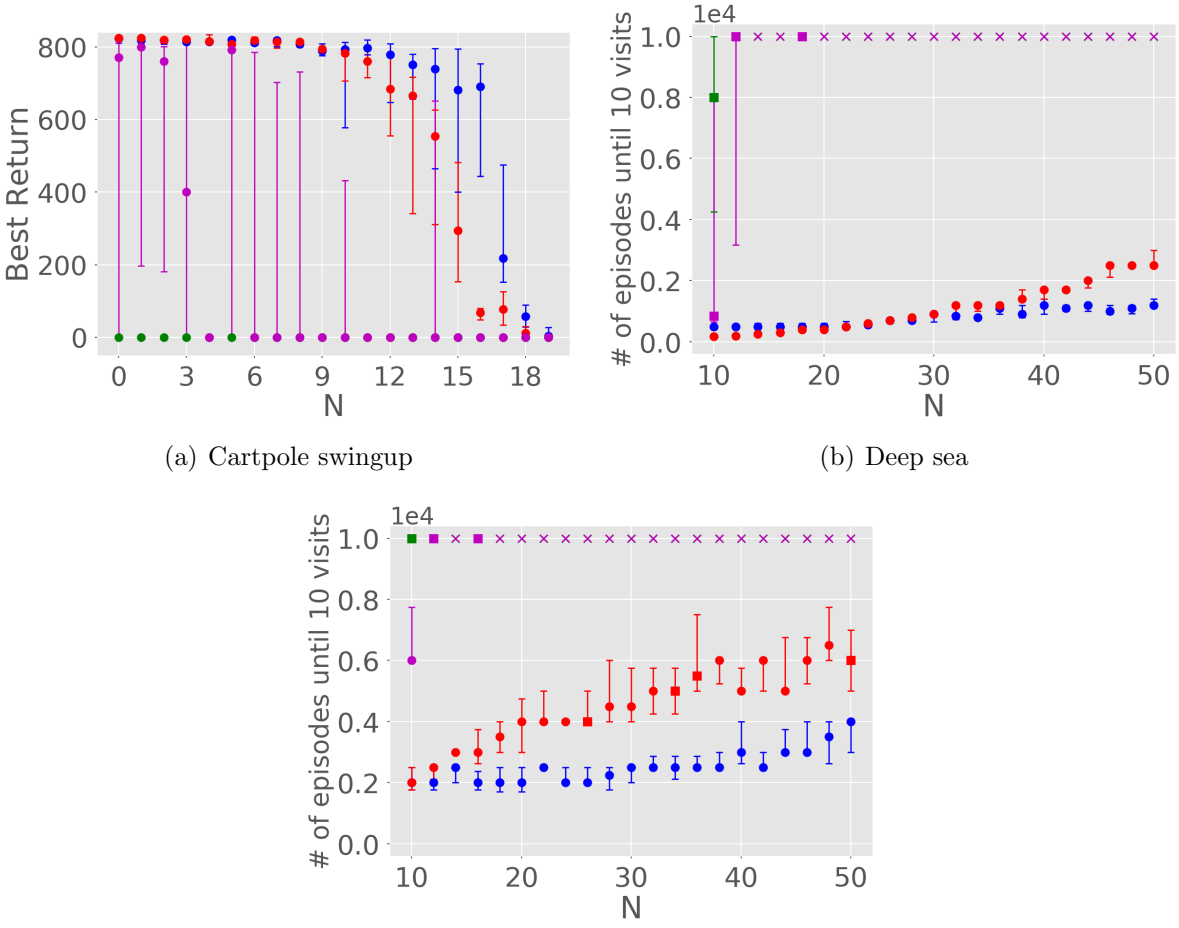
Initialize: counter=0, ω, θ and ν randomly, and an empty replay buffer $\mathcal{R}$.
for iterations $k = 0, \dots, K$ **do**
  for environment transitions $t = 0, \dots, T$ **do**
    Sample transitions (s, a, r, s') by following policy (5.10) and store them in $\mathcal{R}$.
  end for
  for iterations $i = 0, \dots, I$ **do**
    Sample a minibatch from $\mathcal{R}$ and compute stochastic gradients.
    $\omega \leftarrow \omega - \mu_\omega \widehat{\nabla}_\omega J_q(\omega)$
    $\theta \leftarrow \theta - \mu_\theta \widehat{\nabla}_\theta J_\rho(\theta)$
    $\nu \leftarrow \nu - \mu_\nu \widehat{\nabla}_\nu J_\ell(\nu)$
    counter += 1
    if counter **mod** targetUpdatePeriod **then**
      $\widetilde{\nu} \leftarrow \nu$
      $\widetilde{\omega} \leftarrow \omega$
      counter = 0
    end if
  end for
end for

where ψ is the distribution according to which the $(\boldsymbol{s}, \boldsymbol{a})$ pairs are sampled, and $\widetilde{\omega}$ and $\widetilde{\nu}$ are used to denote the parameters of the target networks corresponding to q and ℓ, respectively. Note also that we have added another tunable parameter η_2. In the tabular case this is not necessary, but in the case with neural networks we observed empirically that using the ρ network to train q helps stabilize training and improves performance. This is similar to the SAC algorithm where a network is used to estimate the value function $v(s)$ even though apparently it is not necessary [31]. The resulting algorithm, which we refer to as *Information Seeking Learner*, is listed in algorithm 5.1.

5.4 Experiments

The goal of the experiments in this chapter is to evaluate the capacity of ISL to perform deep exploration. We test our algorithm in the three deep exploration environments provided by the *bsuite* benchmark [114] (i.e., Cartpole Swingup, Deep Sea and Deep Sea Stochastic) and

compare it against SBEED, UBE [106] and Bootstrap DQN with prior networks (BSP)[2] (which provides state of the art results in deep exploration tasks). All three environments have the common quality that exploration is discouraged (due to negative rewards) and positive rewards are only obtained by the agent in states that are hard to reach. We used NNs as function approximators in all cases.



(a) Cartpole swingup

(b) Deep sea

(c) Deep sea stochastic

Figure 5.1: Blue, red, purple and green curves correspond to ISL, BSP, UBE and SBEED, respectively. In all cases we ran 10 experiments with different seeds, the plots show the median and first and third quartiles. In figures 5.1(b) and 5.1(c) we used dots are markers when the goal was accomplished (at least 10 visits where made to the desired state) for all seeds, square markers denote that the goal was accomplished for some seeds and the cross markers denote failure for all seeds.

[2]We use the implementation provided by [114].

5.4.1 Sparse Cartpole Swingup

This is the classical cartpole swingup task with the added difficulty that positive rewards are only provided when the pole is 'almost stabilized'. The action space is {*left, stay, right*} and the state space is continuous and given by $s_t = (\cos(\theta_t), \sin(\theta_t), \dot{\theta}_t, x_t, \dot{x}_t)$, where θ is the angle of the pole and x is the position of the cart. The feature that makes this task a challenging exploration task is the reward structure; every move is penalized with a -0.1 reward and a $+1$ is only observed when the cart is 'almost' centered and the pole is 'almost' upright and stabilized. More specifically, a $+1$ reward is obtained when $cos(\theta) > N/20$, $|\dot{\theta}| < 1$ and $|x| < 1 - N/20$, where N parameterizes the difficulty of the environment. This is an episodic task where each episode ends when the cart moves too far away from the center ($|x| > 3$) or at 10^3 time-steps, whichever occurs first. We ran each algorithm for 10^3 episodes for ten random seeds. In this benchmark the performance measure is the best return attained during training. The results for N from 0 to 19 are shown in figures 5.1(a). As expected SBEED fails in this task for all values of N due to the lack of a mechanism to encourage deep exploration. ISL and BSP find close to optimal policies for all values of N up to 10 approximately. However, for higher values of N ISL outperforms BSP by a significant margin. For $N = 19$ all algorithms fail, however ISL is the only algorithm who can obtain some positive rewards. Implementation details can be found in appendix 5.F.

5.4.2 Deep Sea and Deep Sea Stochastic

Deep Sea is an episodic grid-world type game which consists of an $N \times N$ grid with N^2 states. The observation encodes the agent's row and column as a one-hot vector $s_t \in \{0, 1\}^{N \times N}$. The environment has two possible actions $\{a_1, a_2\}$ and a mask $M \sim \text{Ber}(0.5)^{N \times N}$. The mask maps for every state each action to {*left, right*}. The agent always starts at the top-left corner and at every step deterministically descends one level and further moves left or right (depending on the chosen action). Every time the agent moves right it gets a $-0.01/N$ reward, except for the bottom-right state in which case it gets a $+1$, while left moves always receive 0 reward. The game ends after N time steps and we ran each algorithm for 10^4 episodes. The optimal

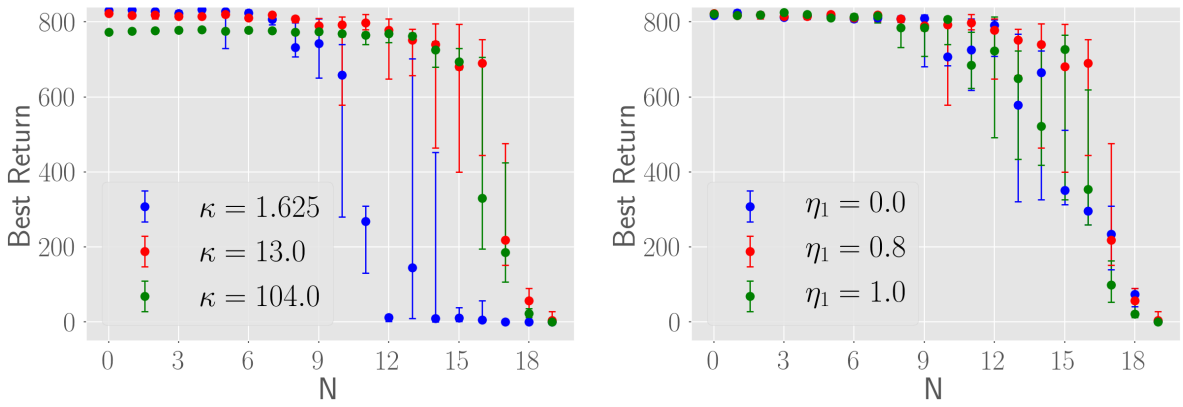
strategy of the environment is to always move right in which case the total return would be 0.99. Note that the parameter N parameterizes the difficulty of the game. An important point about this environment is that policies that do not encourage deep exploration take $\mathcal{O}(2^N)$ episodes to learn the optimal policy [115], while for policies that explore optimally it takes at best $\mathcal{O}(N)$ episodes (because the state-action space is of size $\mathcal{O}(N^2)$ and at every episode N state-action pairs are sampled). The Deep Sea Stochastic environment has the added complexity that transitions and rewards are stochastic. In particular, the reward of the last step of the episode is corrupted with additive Gaussian noise with zero mean and variance equal to 1 and further, agents transition to the right only with $1 - 1/N$ probability. In these benchmarks the performance measure is the amount of episodes played before the agent visits the goal state for the 10-th time. Hyperparameters were tuned for each of the cases, figures 5.1(b) and 5.1(c) show the results. Again SBEED fails at these tasks as expected. In the deterministic case ISL and BSP perform similarly. Note also that while ISL shows linear complexity for all values of N, BSP shows linear complexity only for low values of N. We clarify that the complexity of BSP could probably be improved for larger values of N by enlarging the ensemble size, however note that this would come with an added computational cost since the computational cost of BSP scales linearly with the size of the ensemble. In the stochastic environment ISL performs similarly as it does in the deterministic environment and outperforms BSP for all values of N. Note further that ISL is the only algorithm that is able to solve the task for all values of N for all seeds. Implementations details for *deep sea* and *deep sea stochastic* can be found in appendices 5.G and 5.H, respectively.

5.4.3 Ablation Study

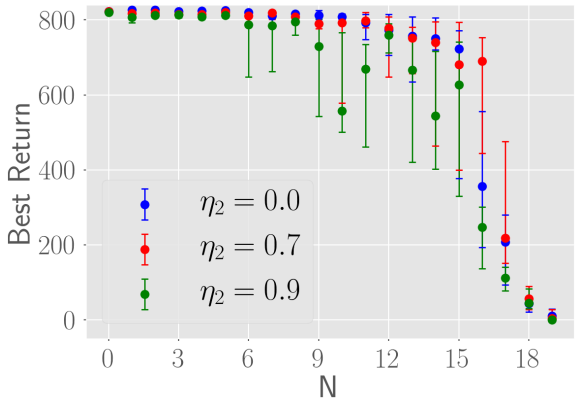
In this section we include an ablation study for the hyperparameters κ, η_1 and η_2 using the cartpole task. We sweep κ through $20 \times 2^{[-4,-3,-2,-1,0,1,2,3,4]}$ and η_1 and η_2 through $[0, 0.1, \cdots, 0.9, 1]$. Results are shown in figure 5.2; to make these plots more visually clear we only show three of the curves in each plot; we note though that the effects of the hyperparameters on the performance of ISL are clear with these three curves. The rest of the

curves can be found in appendix 5.I.

The exploration-exploitation trade-off managed by κ is clear in figure 5.2(a); increasing κ improves results for high values of N (since more exploration is required in these cases) but does so at the expense of exploiting less and therefore the best return diminishes for low values of N (where less exploration is necessary). Figures 5.2(b) and 5.2(c) indicate that the effect of hyperparameters η_1 and η_2 on the performance of ISL are less drastic than that of κ, however in both cases the best performance is obtained for intermediate values of η_1 and η_2 as expected.



(a) κ (b) η_1



(c) η_2

Figure 5.2: In all cases we ran 10 experiments with different seeds, the plots show the median and first and third quartiles.

5.5 Summary

In this chapter we provide a novel and original approach to address the problem of deep exploration. We also make an interesting connection with the literature on maximum entropy RL. In contrast to current RL algorithms and deep exploration strategies, where deriving the learning equations and the deep exploration strategies are treated separately, in our approach, both the learning equations and the deep exploration strategy are derived in tandem as the solution to a unique optimization problem. The main insight of our approach, is that in RL, obtaining point estimates of the quantities of interest is not sufficient, it is also necessary to estimate confidence bounds. Furthermore, we have introduced a practical way of quantifying uncertainty over q estimates that is usable with NNs. We hope this ideas might inspire novel and original research directions.

5.A Proof of Lemma 5.1

In the proof we assume that the actions are ordered following the lemma's assumption. By definition, the KL divergence is given by:

$$\begin{aligned} D_{KL}(u_s^\pi(\widetilde{q})) &= \int_{\widetilde{q}} u_s^\pi(\widetilde{q}) \log\left(\frac{u_s^\pi(\widetilde{q})}{u_s^\bullet(\widetilde{q})}\right) d\widetilde{q} \\ &\overset{(b)}{=} \sum_a \pi(a|s) \int_{\widetilde{q}} d_{(s,a)}(\widetilde{q}) \log\left(\sum_{a'} \pi(a'|s) d_{(s,a')}(\widetilde{q})\right) d\widetilde{q} + \log(\ell_A) \end{aligned} \tag{5.19}$$

where in (b) we used (5.4). Note that since $\delta^{\pi}(s)$ is a piecewise constant distribution, the integral in (5.19) has the following closed form expression:

$$\begin{aligned}
&\int_{\widetilde{q}} d_{(s,a_j)}(\widetilde{q}) \log\left(\sum_{a'} \pi(a'|s) d_{(s,a')}(\widetilde{q})\right) d\widetilde{q} = \int_{-\ell_j}^{\ell_j} (2\ell_j)^{-1} \log\left(\sum_{a'} \pi(a'|s) d_{(s,a')}(\widetilde{q})\right) d\widetilde{q} \\
&= \ell_j^{-1} \int_{\ell_{j-1}}^{\ell_j} \log\left(\sum_{a'} \pi(a'|s) d_{(s,a')}(\widetilde{q})\right) d\widetilde{q} + \ell_j^{-1} \int_{0}^{\ell_{j-1}} \log\left(\sum_{a'} \pi(a'|s) d_{(s,a')}(\widetilde{q})\right) d\widetilde{q} \\
&= \frac{\ell_j - \ell_{j-1}}{\ell_j} \log\left(\sum_{b=0}^{A-j} \pi(A-b|s) \ell_{A-b}^{-1}\right) + \ell_j^{-1} \int_{0}^{\ell_{j-1}} \log\left(\sum_{a'} \pi(a'|s) d_{(s,a')}(b)\right) d\widetilde{q} \\
&= \sum_{n=1}^{j} \frac{\ell_n - \ell_{n-1}}{\ell_j} \log\left(\sum_{b=n}^{A} \frac{\pi(b|s)}{\ell_b}\right) \qquad (5.20)
\end{aligned}$$

Combining (5.19) and (5.20) we get:

$$\sum_{k=1}^{A} \frac{\pi(a_k|s)}{\ell_k} \sum_{n=1}^{k} (\ell_n - \ell_{n-1}) \log\left[\sum_{b=n}^{A} \frac{\pi(b|s)\ell_A}{\ell_b}\right] \qquad (5.21)$$

Rearranging the terms in (5.21) we get:

$$D_{KL}(u_s^{\pi}(\widetilde{q})) = \sum_{n=1}^{A} (\ell_n - \ell_{n-1}) \left(\sum_{k=n}^{A} \frac{\pi(a_k|s)}{\ell_k}\right) \log\left[\sum_{b=n}^{A} \frac{\pi(b|s)\ell_A}{\ell_b}\right] \qquad (5.22)$$

5.B Proof Lemma 5.2

We start proving that for any action a_j that is Pareto dominated by another action a_i it must be the case that $\pi^{\star}(a_j|s) = 0$. We now present an assumption to make notation simpler.

Assumption 5.2. *In this section we assume without loss of generality that actions are ordered such that* $\ell_i > \ell_j \iff i > j$.

We prove the lemma by contradiction. Assume that there is a *uc*-optimal policy π_1 for

which $\pi_1(a_j|s_1) > 0$. We now define policy π_2 as:

$$\pi_2(a|s) = \begin{cases} \pi_1(a|s) - \alpha & \text{if } (s,a) = (s_1, a_j) \\ \pi_1(a|s) + \alpha & \text{if } (s,a) = (s_1, a_i) \\ \pi_1(a|s) & \text{else} \end{cases} \tag{5.23}$$

where $0 < \alpha < \pi_1(a|s)$. We show that $\frac{\partial v^{\pi_2}(s_1)}{\partial \alpha}\big|_{\alpha=0} > 0$ and hence $v^{\pi_2}(s_1) > v^{\pi_1}(s_1)$ for a small enough $\alpha > 0$, which contradicts the claim that π_1 is a *uc*-optimal policy.

$$\begin{aligned} \frac{\partial v^{\pi_2}(s_1)}{\partial \alpha}\bigg|_{\alpha=0} &= \frac{\partial v^{\pi_2}(s_1)}{\partial \pi_1(a_j|s_1)} \frac{\partial \pi_1(a_j|s_1)}{\partial \alpha}\bigg|_{\alpha=0} + \frac{\partial v^{\pi_2}(s_1)}{\partial \pi_1(a_i|s_1)} \frac{\partial \pi_1(a_i|s_1)}{\partial \alpha}\bigg|_{\alpha=0} \\ &= \frac{\partial v^{\pi_2}(s_1)}{\partial \pi_1(a_i|s_1)} - \frac{\partial v^{\pi_2}(s_1)}{\partial \pi_1(a_j|s_1)} \\ &= \underbrace{\widehat{q}^{\pi_2}(s,a_i) - \widehat{q}^{\pi_2}(s,a_j)}_{>0 \text{ (due to Pareto assumption)}} + \kappa \left(\frac{\partial D_{KL}(u_{s_1}^{\pi_2}(\widetilde{q}))}{\partial \pi_1(a_j|s_1)} - \frac{\partial D_{KL}(u_{s_1}^{\pi_2}(\widetilde{q}))}{\partial \pi_1(a_i|s_1)} \right) \end{aligned} \tag{5.24}$$

Using (5.22) we get the following expression for the gradient of the KL term:

$$\begin{aligned} \frac{\partial D_{KL}(u_s^{\pi}(\widetilde{q}))}{\partial \pi(a_j|s)} &= \sum_{b=1}^{j} \frac{(\ell_b - \ell_{b-1})}{\ell_j} \log \left(\sum_{c=b}^{A} \frac{\pi(c|s)}{\ell_c} \right) + \ell_j^{-1} \sum_{b=1}^{j} (\ell_b - \ell_{b-1}) \\ &= \sum_{b=1}^{j} \frac{(\ell_b - \ell_{b-1})}{\ell_j} \log \left(\sum_{c=b}^{A} \frac{\pi(c|s)}{\ell_c} \right) + 1 \end{aligned} \tag{5.25}$$

Combining (5.25) and (5.24) we get:

$$\begin{aligned} \frac{\partial D_{KL}(u_{s_1}^{\pi_2}(\widetilde{q}))}{\partial \pi_1(a_j|s_1)} - \frac{\partial D_{KL}(u_{s_1}^{\pi_2}(\widetilde{q}))}{\partial \pi_1(a_i|s_1)} &= \sum_{b=1}^{j} \frac{(\ell_b - \ell_{b-1})}{\ell_j} \log \left(\sum_{c=b}^{A} \frac{\pi(c|s)}{\ell_c} \right) \\ &\quad - \sum_{b=1}^{i} \frac{(\ell_b - \ell_{b-1})}{\ell_i} \log \left(\sum_{c=b}^{A} \frac{\pi(c|s)}{\ell_c} \right) \\ = \sum_{b=1}^{j} \frac{(\ell_b - \ell_{b-1})(\ell_i - \ell_j)}{\ell_j \ell_i} \log \left(\sum_{c=b}^{A} \frac{\pi(c|s)}{\ell_c} \right) &- \sum_{b=j+1}^{i} \frac{(\ell_b - \ell_{b-1})}{\ell_i} \log \left(\sum_{c=b}^{A} \frac{\pi(c|s)}{\ell_c} \right) \\ \overset{(a)}{>} \sum_{b=1}^{j} \frac{(\ell_b - \ell_{b-1})(\ell_i - \ell_j)}{\ell_j \ell_i} \log \left(\sum_{c=j}^{A} \frac{\pi(c|s)}{\ell_c} \right) &- \sum_{b=j+1}^{i} \frac{(\ell_b - \ell_{b-1})}{\ell_i} \log \left(\sum_{c=j}^{A} \frac{\pi(c|s)}{\ell_c} \right) \end{aligned}$$

$$= \frac{(\ell_i - \ell_j)}{\ell_i} \log\left(\sum_{c=j}^{A} \frac{\pi(c|s)}{\ell_c}\right) - \frac{(\ell_i - \ell_j)}{\ell_i} \log\left(\sum_{c=j}^{A} \frac{\pi(c|s)}{\ell_c}\right)$$

$$= 0 \tag{5.26}$$

where (a) is due to the fact that all terms in $\sum_{c=i}^{A} \frac{\pi(c|s)}{\ell_c}$ are non-negative and *log* is a monotone increasing function. Combining (5.26) with (5.24) we get:

$$\left.\frac{\partial v^{\pi_2}(s_1)}{\partial \alpha}\right|_{\alpha=0} > \widehat{q}^{\pi_2}(s, a_i) - \widehat{q}^{\pi_2}(s, a_j) > 0 \tag{5.27}$$

which completes the proof.

The proof for the Mixed Pareto case follows similarly. We assume that there is a *uc*-optimal policy π_1 that assigns non-zero probability to an action a_k Mixed Pareto dominated by a_i and a_j, $\pi_1(a_k|s_1) > 0$. Since a_k is assumed to be Mixed Pareto dominated, equations (5.9) are satisfied. Similarly, as before, we define a new policy π_2 as:

$$\pi_2(a|s) = \begin{cases} \pi_1(a|s) - \alpha & \text{if } (s,a) = (s_1, a_k) \\ \pi_1(a|s) + \alpha\frac{(\ell_k - \ell_j)\ell_i}{(\ell_i - \ell_j)\ell_k} & \text{if } (s,a) = (s_1, a_i) \\ \pi_1(a|s) + \alpha\frac{(\ell_i - \ell_k)\ell_j}{(\ell_i - \ell_j)\ell_k} & \text{if } (s,a) = (s_1, a_j) \\ \pi_1(a|s) & \text{else} \end{cases} \tag{5.28}$$

The gradient of the value function becomes:

$$\begin{aligned}
\left.\frac{\partial v^{\pi_2}(s_1)}{\partial \alpha}\right|_{\alpha=0} &= \left.\frac{\partial v^{\pi_2}(s_1)}{\partial \pi_1(a_k|s_1)}\frac{\partial \pi_1(a_k|s_1)}{\partial \alpha}\right|_{\alpha=0} + \left.\frac{\partial v^{\pi_2}(s_1)}{\partial \pi_1(a_j|s_1)}\frac{\partial \pi_1(a_j|s_1)}{\partial \alpha}\right|_{\alpha=0} \\
&\quad + \left.\frac{\partial v^{\pi_2}(s_1)}{\partial \pi_1(a_i|s_1)}\frac{\partial \pi_1(a_i|s_1)}{\partial \alpha}\right|_{\alpha=0} \\
&= \frac{\partial v^{\pi_2}(s_1)}{\partial \pi_1(a_i|s_1)}\frac{(\ell_k-\ell_j)\,\ell_i}{(\ell_i-\ell_j)\,\ell_k} + \frac{\partial v^{\pi_2}(s_1)}{\partial \pi_1(a_j|s_1)}\frac{(\ell_i-\ell_k)\,\ell_j}{(\ell_i-\ell_j)\,\ell_k} - \frac{\partial v^{\pi_2}(s_1)}{\partial \pi_1(a_k|s_1)} \\
&= \underbrace{\widehat{q}^{\pi_2}(s,a_i)\frac{(\ell_k-\ell_j)\,\ell_i}{(\ell_i-\ell_j)\,\ell_k} + \widehat{q}^{\pi_2}(s,a_j)\frac{(\ell_i-\ell_k)\,\ell_j}{(\ell_i-\ell_j)\,\ell_k} - \widehat{q}^{\pi_2}(s,a_k)}_{>0 \text{ (due to Pareto assumption)}} \\
&\quad + \kappa\left(\frac{\partial D_{KL}(u^{\pi_2}_{s_1}(\widetilde{q}))}{\partial \pi_1(a_k|s_1)} - \frac{\partial D_{KL}(u^{\pi_2}_{s_1}(\widetilde{q}))}{\partial \pi_1(a_i|s_1)}\frac{(\ell_k-\ell_j)\,\ell_i}{(\ell_i-\ell_j)\,\ell_k} - \frac{\partial D_{KL}(u^{\pi_2}_{s_1}(\widetilde{q}))}{\partial \pi_1(a_j|s_1)}\frac{(\ell_i-\ell_k)\,\ell_j}{(\ell_i-\ell_j)\,\ell_k}\right)
\end{aligned} \tag{5.29}$$

And finally:

$$\begin{aligned}
&\frac{\partial D_{KL}(u^{\pi_2}_{s_1}(\widetilde{q}))}{\partial \pi_1(a_k|s_1)} - \frac{\partial D_{KL}(u^{\pi_2}_{s_1}(\widetilde{q}))}{\partial \pi_1(a_i|s_1)}\frac{(\ell_k-\ell_j)\,\ell_i}{(\ell_i-\ell_j)\,\ell_k} - \frac{\partial D_{KL}(u^{\pi_2}_{s_1}(\widetilde{q}))}{\partial \pi_1(a_j|s_1)}\frac{(\ell_i-\ell_k)\,\ell_j}{(\ell_i-\ell_j)\,\ell_k} \\
&= \sum_{b=1}^{k}\frac{(\ell_b-\ell_{b-1})}{\ell_k}\log\left(\sum_{c=b}^{A}\frac{\pi(c|s)}{\ell_c}\right) - \frac{(\ell_k-\ell_j)}{(\ell_i-\ell_j)\,\ell_k}\sum_{b=1}^{i}(\ell_b-\ell_{b-1})\log\left(\sum_{c=b}^{A}\frac{\pi(c|s)}{\ell_c}\right) \\
&\qquad - \frac{(\ell_i-\ell_k)}{(\ell_i-\ell_j)\,\ell_k}\sum_{b=1}^{j}(\ell_b-\ell_{b-1})\log\left(\sum_{c=b}^{A}\frac{\pi(c|s)}{\ell_c}\right) \\
&= \sum_{b=j+1}^{k}\frac{(\ell_b-\ell_{b-1})}{\ell_k}\log\left(\sum_{c=b}^{A}\frac{\pi(c|s)}{\ell_c}\right) - \frac{(\ell_k-\ell_j)}{(\ell_i-\ell_j)\,\ell_k}\sum_{b=j+1}^{i}(\ell_b-\ell_{b-1})\log\left(\sum_{c=b}^{A}\frac{\pi(c|s)}{\ell_c}\right) \\
&= \sum_{b=j+1}^{k}\frac{(\ell_b-\ell_{b-1})(\ell_i-\ell_k)}{(\ell_i-\ell_j)\ell_k}\log\left(\sum_{c=b}^{A}\frac{\pi(c|s)}{\ell_c}\right) - \frac{(\ell_k-\ell_j)}{(\ell_i-\ell_j)\,\ell_k}\sum_{b=k+1}^{i}(\ell_b-\ell_{b-1})\log\left(\sum_{c=b}^{A}\frac{\pi(c|s)}{\ell_c}\right) \\
&> \sum_{b=j+1}^{k}\frac{(\ell_b-\ell_{b-1})(\ell_i-\ell_k)}{(\ell_i-\ell_j)\ell_k}\log\left(\sum_{c=k}^{A}\frac{\pi(c|s)}{\ell_c}\right) - \frac{(\ell_k-\ell_j)}{(\ell_i-\ell_j)\,\ell_k}\sum_{b=k+1}^{i}(\ell_b-\ell_{b-1})\log\left(\sum_{c=k}^{A}\frac{\pi(c|s)}{\ell_c}\right) \\
&= \frac{(\ell_k-\ell_j)(\ell_i-\ell_k)}{(\ell_i-\ell_j)\ell_k}\log\left(\sum_{c=k}^{A}\frac{\pi(c|s)}{\ell_c}\right) - \frac{(\ell_k-\ell_j)\,(\ell_i-\ell_k)}{(\ell_i-\ell_j)\,\ell_k}\log\left(\sum_{c=k}^{A}\frac{\pi(c|s)}{\ell_c}\right) = 0
\end{aligned} \tag{5.30}$$

Combining (5.29) with (5.30) we get $\left.\frac{\partial v^{\pi_2}(s_1)}{\partial \alpha}\right|_{\alpha=0} > 0$, which completes the proof.

5.C Proof Theorem 5.1

Due to Lemma 5.2 we already know that for Pareto dominated actions $\pi^\star(a|s) = 0$. Therefore, without loss of generality, we assume that all actions are Pareto optimal. Furthermore, to simplify notation in the proof we assume that $\sigma_s(a_j) = j$ and hence we will not use the ordering function $\sigma_s(a)$. We start differentiating (5.7) with respect to $\pi(a|s)$ and equating to zero:

$$\kappa^{-1}\widehat{q}(s,a) - \frac{\partial D_{KL}(u_s^\star(\widetilde{q}))}{\partial \pi^\star(a|s)} = 0 \tag{5.31}$$

Using (5.8) we get the following expression for the gradient of the KL term:

$$\begin{aligned}
\frac{\partial D_{KL}(u_s^\star(\widetilde{q}))}{\partial \pi^\star(a_j|s)} &= \sum_{b=1}^{j} \frac{(\ell_b - \ell_{b-1})}{\ell_j} \log\left(\sum_{c=b}^{A} \frac{\pi^\star(c|s)}{\ell_c}\right) + \ell_j^{-1} \sum_{b=1}^{j} (\ell_b - \ell_{b-1}) \\
&= \sum_{b=1}^{j} \frac{(\ell_b - \ell_{b-1})}{\ell_j} \log\left(\sum_{c=b}^{A} \frac{\pi^\star(c|s)}{\ell_c}\right) + 1 \\
&= \ell_j^{-1}(\ell_j - \ell_{j-1}) \log\left(\sum_{c=j}^{A} \pi^\star(c|s)\ell_c^{-1}\right) + \frac{\ell_{j-1}}{\ell_j}\left(\frac{\partial D_{KL}(u_s^\star(\widetilde{q}))}{\partial \pi(a_{j-1}|s)} - 1\right) + 1 \\
&= \ell_j^{-1}(\ell_j - \ell_{j-1}) \log\left(\sum_{c=j}^{A} \pi^\star(c|s)\ell_c^{-1}\right) + \frac{\ell_{j-1}}{\ell_j}\frac{\partial D_{KL}(u_s^\star(\widetilde{q}))}{\partial \pi^\star(a_{j-1}|s)} + \frac{\ell_j - \ell_{j-1}}{\ell_j}
\end{aligned} \tag{5.32}$$

Now we can solve for each action combining the recursive form given in (5.32) with (5.31). Recall that due to the specific numbering of actions we assumed, a_A is the action who has the greatest uncertainty ℓ_A. Hence, we can start solving for a_A as follows:

$$\begin{aligned}
0 &= \kappa^{-1}\widehat{q}(s,a_A) - \ell_A^{-1}(\ell_A - \ell_{A-1}) \log\left(\pi^\star(a_A|s)\ell_A^{-1}\right) - \frac{\ell_{A-1}}{\ell_A}\frac{\partial D_{KL}(u_s^\star(\widetilde{q})||u_s^\bullet(\widetilde{q}))}{\partial \pi^\star(a_{A-1}|s)} - \frac{\ell_A - \ell_{A-1}}{\ell_A} \\
&= \kappa^{-1}\widehat{q}(s,a_A) - \ell_A^{-1}(\ell_A - \ell_{A-1}) \log\left(\pi^\star(a_A|s)\ell_A^{-1}\right) - \kappa^{-1}\frac{\ell_{A-1}}{\ell_A}\widehat{q}(s,a_{A-1}) - \frac{\ell_A - \ell_{A-1}}{\ell_A} \\
&\rightarrow \pi^\star(a_A|s) \propto \ell_A p_A(s)
\end{aligned} \tag{5.33}$$

where we defined:

$$p_j(s) = \exp\left[\frac{\ell_j(s)\widehat{q}(s,a_j) - \ell_{j-1}(s)\widehat{q}(s,a_{j-1})}{\kappa\left(\ell_j(s) - \ell_{j-1}(s)\right)}\right] \tag{5.34}$$

Following the same procedure as in (5.33) we can solve for $\pi^\star(a_j|s)$.

$$\begin{aligned}
0 &= \kappa^{-1}\widehat{q}(j,s) - \frac{(\ell_j - \ell_{j-1})}{\ell_j}\log\left(\sum_{c=j}^{A}\frac{\pi^\star(c|s)}{\ell_c}\right) - \frac{\ell_{j-1}}{\ell_j}\frac{\partial D_{KL}(u_s^\star(\widetilde{q}))}{\partial\pi^\star(a_{j-1}|s)} - \frac{\ell_j - \ell_{j-1}}{\ell_j} \\
&\overset{(a)}{=} \kappa^{-1}\widehat{q}(s,a_j) - \frac{(\ell_j - \ell_{j-1})}{\ell_j}\log\left(\sum_{c=j}^{A}\frac{\pi^\star(c|s)}{\ell_c}\right) - \kappa^{-1}\frac{\ell_{j-1}}{\ell_j}\widehat{q}(s,a_{j-1}) - \frac{\ell_j - \ell_{j-1}}{\ell_j} \\
&\to \pi^\star(a_j|s) \propto \ell_j p_j(s) e^{-1} - \ell_j\sum_{c=j+1}^{A}\pi^\star(c|s)\ell_c^{-1}
\end{aligned} \tag{5.35}$$

where in (a) we used (5.31). Starting with $j = A-1$, unwinding (5.35) one step of the recursion at a time, and normalizing we get:

$$\pi^\star(a_j|s) = \frac{\ell_j(p_j(s) - p_{j+1}(s))}{\sum_{j=1}^{A}(\ell_j - \ell_{j-1})p_j(s)} \tag{5.36}$$

which completes the proof.

5.D Proof Lemma 5.4

We start by showing that $\mathcal{T}^\ell$ is a contraction mapping. For this we define two mappings $q_1, q_2 : \mathcal{S}\times\mathcal{A} :\to \mathbb{R}$. We start assuming $\mathcal{T}^\ell q_1(s,a) \geq \mathcal{T}^\ell q_2(s,a)$, then:

$$\begin{aligned}
0 \leq \mathcal{T}^\ell q_1(s,a) - \mathcal{T}^\ell q_2(s,a) &= r(s,a) + \gamma\mathbb{E}_{\boldsymbol{s'}}v_1^\star(\boldsymbol{s'}) - r(s,a) - \gamma\mathbb{E}_{\boldsymbol{s'}}v_2^\star(\boldsymbol{s'}) = \gamma\mathbb{E}_{\boldsymbol{s'}}\left(v_1^\star(\boldsymbol{s'}) - v_2^\star(\boldsymbol{s'})\right) \\
&\overset{(a)}{=} \gamma\mathbb{E}_{\boldsymbol{s'}}\left(\max_\pi \mathbb{E}_{\boldsymbol{a}\sim\pi}\left(q_1(\boldsymbol{s'},\boldsymbol{a}) - \kappa D_{KL}(u^\pi_{\boldsymbol{s'}}(\widetilde{q}))\right) - \max_\pi \mathbb{E}_{\boldsymbol{a}\sim\pi}\left(q_2(\boldsymbol{s'},\boldsymbol{a}) - \kappa D_{KL}(u^\pi_{\boldsymbol{s'}}(\widetilde{q}))\right)\right) \\
&\leq \gamma\mathbb{E}_{\boldsymbol{s'}}\max_\pi \mathbb{E}_{\boldsymbol{a}\sim\pi}\left(q_1(\boldsymbol{s'},\boldsymbol{a}) - \kappa D_{KL}(u^\pi_{\boldsymbol{s'}}(\widetilde{q})) - q_2(\boldsymbol{s'},\boldsymbol{a}) + \kappa D_{KL}(u^\pi_{\boldsymbol{s'}}(\widetilde{q}))\right) \\
&= \gamma\mathbb{E}_{\boldsymbol{s'}}\max_\pi \mathbb{E}_{\boldsymbol{a}\sim\pi}\left(q_1(\boldsymbol{s'},\boldsymbol{a}) - q_2(\boldsymbol{s'},\boldsymbol{a})\right) \\
&\leq \gamma\max_{s,a}\left(q_1(s,a) - q_2(s,a)\right)
\end{aligned} \tag{5.37}$$

$$\rightarrow 0 \leq \mathcal{T}^{\ell} q_1(s,a) - \mathcal{T}^{\ell} q_2(s,a) \leq \gamma \max_{s,a} \left(q_1(s,a) - q_2(s,a)\right) \tag{5.38}$$

where in (a) we used $v^{\star}(s) = \max_{\pi} \mathbb{E}_{\boldsymbol{a}\sim\pi} \left(q^{\star}(s,\boldsymbol{a}) - \kappa D_{KL}(u_s^{\pi}(\widetilde{q}))\right)$. Noting that if $\mathcal{T}^{\ell} q_1(s,a) < \mathcal{T}^{\ell} q_2(s,a)$ the same argument applies. Exchanging the roles of q_1 and q_2 we can conclude that for any (s,a) pair it holds:

$$0 \leq |\mathcal{T}^{\ell} q_1(s,a) - \mathcal{T}^{\ell} q_2(s,a)| \leq \gamma \max_{s,a} |q_1(s,a) - q_2(s,a)| \tag{5.39}$$

which concludes the proof that $\mathcal{T}^{\ell}$ is a contraction mapping. Applying Banach's Fixed-Point Theorem concludes the proof (see Theorem 6.2.3 [22]).

5.E Proof Lemma 5.5

The proof follows by noting that due to Lemma 4 after application of ℓ-Policy Evaluation, it will hold $\delta(s,a) = 0$ for any (s,a) pair. Therefore,

$$\ell(s,a) \leftarrow |\delta(s,a)| + \gamma \mathbb{E}_{\boldsymbol{s}'} \max_{a} \ell(\boldsymbol{s}',a) \; = \; \gamma \mathbb{E}_{\boldsymbol{s}'} \max_{a} \ell(\boldsymbol{s}',a) \tag{5.40}$$

and hence $\ell(s,a)$ for any (s,a) pair decays γ-linearly to 0. Combining this result with Remark 5.2 concludes the proof.

5.F Cartpole Swingup Implementation Details

The implementation details are as follows. All implementations used TensorFlow. We used neural networks as function approximators in all cases. All NN's are composed of two hidden layers with fifty units per layer. ReLu's are used in all hidden layers. All output layers are linear, except for the outputs of the networks that approximate the ℓ values whose outputs pass through sigmoid functions with limits $[1e-12, 100]$. We used the ADAM optimizer in all cases. To approximate the ℓ values we used one network with only one output per action instead of one network with A outputs, empirically this provides better performance without

making any difference in terms of computation requirements. All hyperparameters were set by iterating through them and performing individual per-hyperparameter grid-searches; resulting values are shown in table 5.1.

Table 5.1: Hyperparameters for Cartpole Swingup. Where $|\mathcal{R}|$ is the size of the replay buffer, B is the mini-batch size, μ_q is the step-size for q-network (and similarly for μ_ρ, μ_ℓ, μ_v, μ_π and μ_u) and *tup* stands for the target update period.

ISL	BDQN	UBE	SBEED
$\gamma = 0.99$	$\gamma = 0.99$	$\gamma = 0.99$	$\gamma = 0.99$
$\|\mathcal{R}\| = 1e5$	$\|\mathcal{R}\| = 1e5$	$\|\mathcal{R}\| = 1e5$	$\|\mathcal{R}\| = 1e5$
$B = 64$	$B = 128$	$B = 128$	$B = 256$
$\mu_q = 2e-4$	$\mu_q = 5e-4$	$\mu_q = 5e-4$	$\mu_\rho = 1e-3$
$\mu_\rho = 5e-6$	$\epsilon = 0$	$\mu_u = 1e-4$	$\mu_v = 1e-3$
$\mu_\ell = 2e-5$	mask prob $= 0.5$	$\mu = 20$	$\mu_\pi = 1e-3$
tup$= 4$	tup$= 4$	tup$= 4$	$\kappa = 0.5$
$T = 1$	sgd period$= 1$	$T = 1$	$T = 1$
$I = 3$		$I = 1$	$I = 1$
$\eta_1 = 0.8$		β=1	$\eta = 0$
$\eta_2 = 0.7$			
$\kappa = 13$			

5.G Deep Sea Implementation Details

The architecture of the implementation is the same as the one used for the Cartpole Swingup task. All hyperparameters were set by iterating through them and performing individual per-hyperparameter grid-searches; resulting values are shown in table 5.2.

Table 5.2: Hyperparameters for Deep Sea. Where $|\mathcal{R}|$ is the size of the replay buffer, B is the mini-batch size, μ_q is the step-size for q-network (and similarly for μ_ρ, μ_ℓ, μ_v, μ_π and μ_u) and *tup* stands for the target update period.

ISL	BDQN	UBE	SBEED
$\gamma = 0.99$	$\gamma = 0.99$	$\gamma = 0.99$	$\gamma = 0.99$
$\|\mathcal{R}\| = 1e5$	$\|\mathcal{R}\| = 1e5$	$\|\mathcal{R}\| = 1e5$	$\|\mathcal{R}\| = 1e5$
$B = 256$	$B = 128$	$B = 128$	$B = 256$
$\mu_q = 2e-4$	$q\mu_q = 5e-4$	$\mu_q = 5e-4$	$\mu_\rho = 1e-2$
$\mu_\rho = 1e-4$	$\epsilon = 0$	$\mu_u = 1e-4$	$\mu_v = 1e-2$
$\mu_\ell = 5e-5$	mask probability $= 0.5$	$\mu = 10$	$\mu_\pi = 1e-2$
tup$= 2$	tup$= 5$	tup$= 4$	$\kappa = 0.5$
$T = 2$	sgd period$= 1$	$T = 1$	$T = 1$
$I = 1$		$I = 1$	$I = 1$
$\eta_1 = 0.9$		$\beta = 2$	$\eta = 0$
$\eta_2 = 0.1$			
$\kappa = 1$			

5.H Deep Sea Stochastic Implementation Details

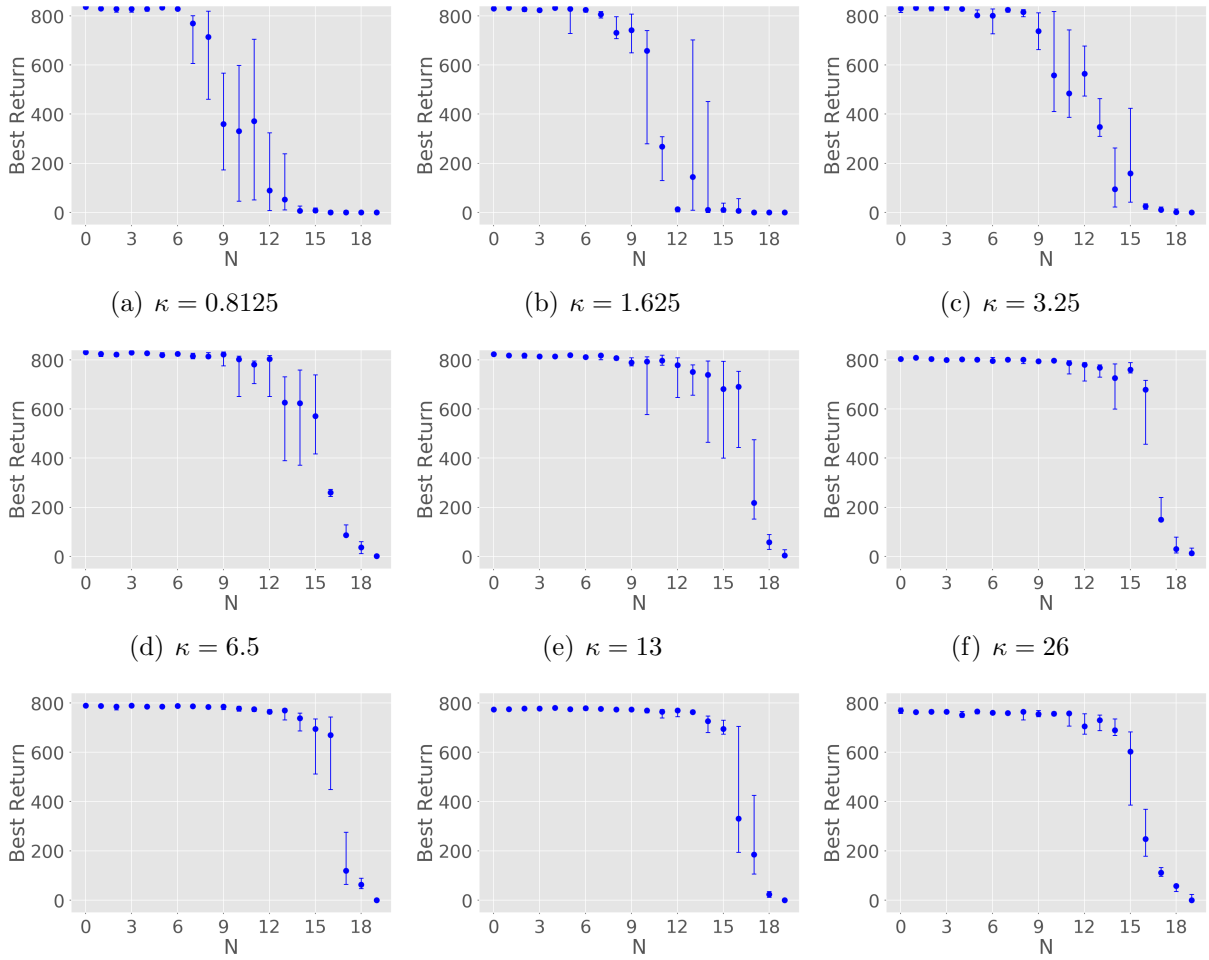
The implementation architecture for the Deep Sea Stochastic game is the same as for Deep Sea, only some hyperparameters change (see table 5.3).

Table 5.3: Hyperparameters for Deep Sea Stochastic. Where $|\mathcal{R}|$ is the size of the replay buffer, B is the mini-batch size, μ_q is the step-size for q-network (and similarly for μ_ρ, μ_ℓ, μ_v, μ_π and μ_u) and *tup* stands for the target update period.

ISL	BDQN	UBE	SBEED
$\gamma = 0.99$	$\gamma = 0.99$	$\gamma = 0.99$	$\gamma = 0.99$
$\|\mathcal{R}\| = 1e5$	$\|\mathcal{R}\| = 1e5$	$\|\mathcal{R}\| = 1e5$	$\|\mathcal{R}\| = 1e5$
$B = 256$	$B = 128$	$B = 128$	$B = 256$
$\mu_q = 1e-4$	$\mu_q = 1e-5$	$\mu_q = 2e-5$	$\mu_\rho = 1e-3$
$\mu_\rho = 1e-4$	$\epsilon = 0$	$\mu_u = 1e-5$	$\mu_v = 1e-3$
$\mu_\ell = 5e-5$	mask probability $= 0.5$	μ=10	$\mu_\pi = 1e-3$
tup= 2	tup= 5	tup= 4	$\kappa = 0.5$
$T = 10$	sgd period= 1	$T = 1$	$T = 1$
$I = 1$		$I = 1$	$I = 1$
$\eta_1 = 1.0$		$\beta = 2$	$\eta = 0.01$
$\eta_2 = 0.5$			
$\kappa = 1$			

5.I Ablation Study Figures

5.I.1 Ablation study for κ



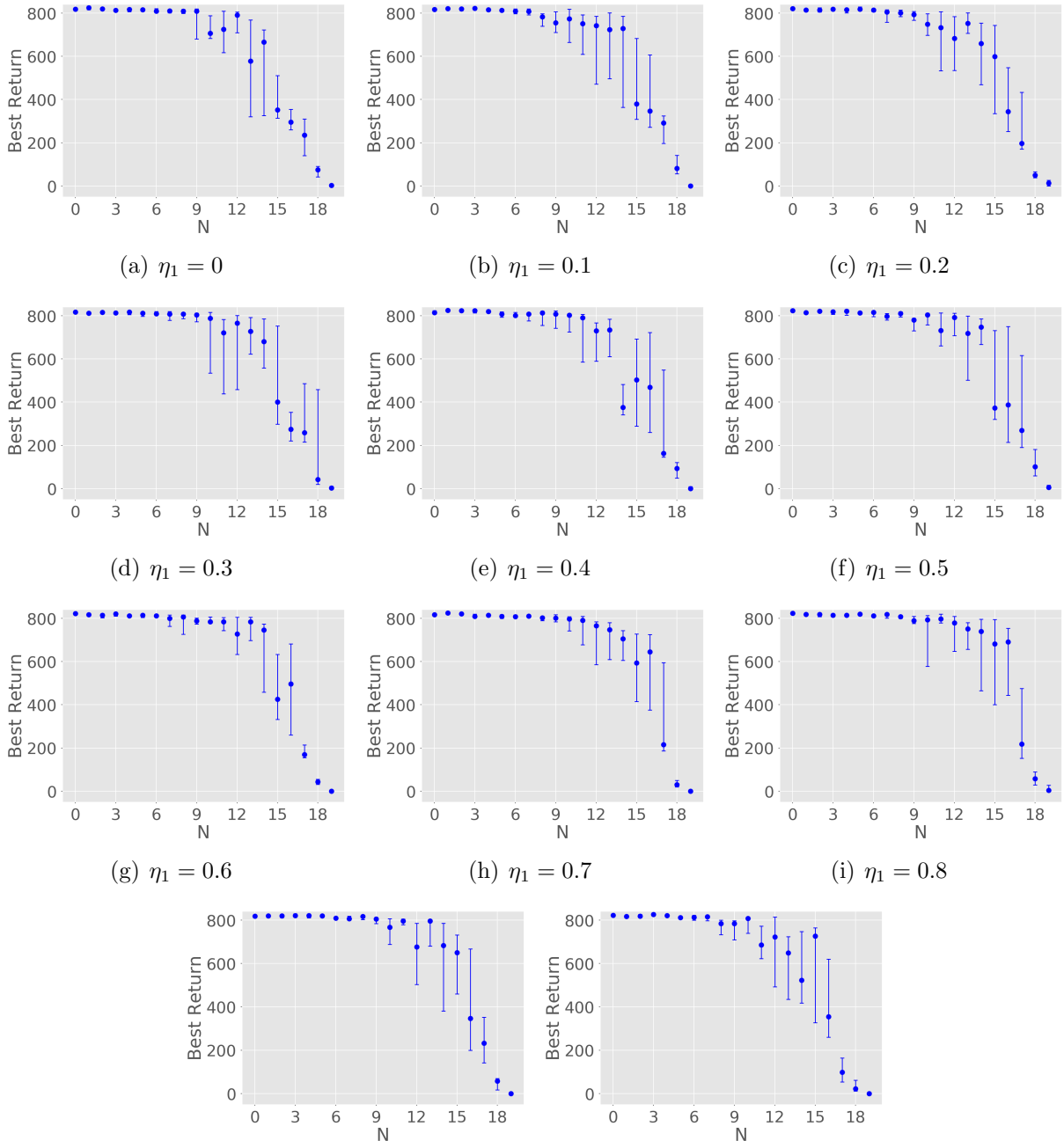
(a) $\kappa = 0.8125$ (b) $\kappa = 1.625$ (c) $\kappa = 3.25$

(d) $\kappa = 6.5$ (e) $\kappa = 13$ (f) $\kappa = 26$

(g) $\kappa = 52$ (h) $\kappa = 104$ (i) $\kappa = 208$

Figure 5.3: Ablation study for κ.

5.I.2 Ablation study for η_1



(a) $\eta_1 = 0$ (b) $\eta_1 = 0.1$ (c) $\eta_1 = 0.2$

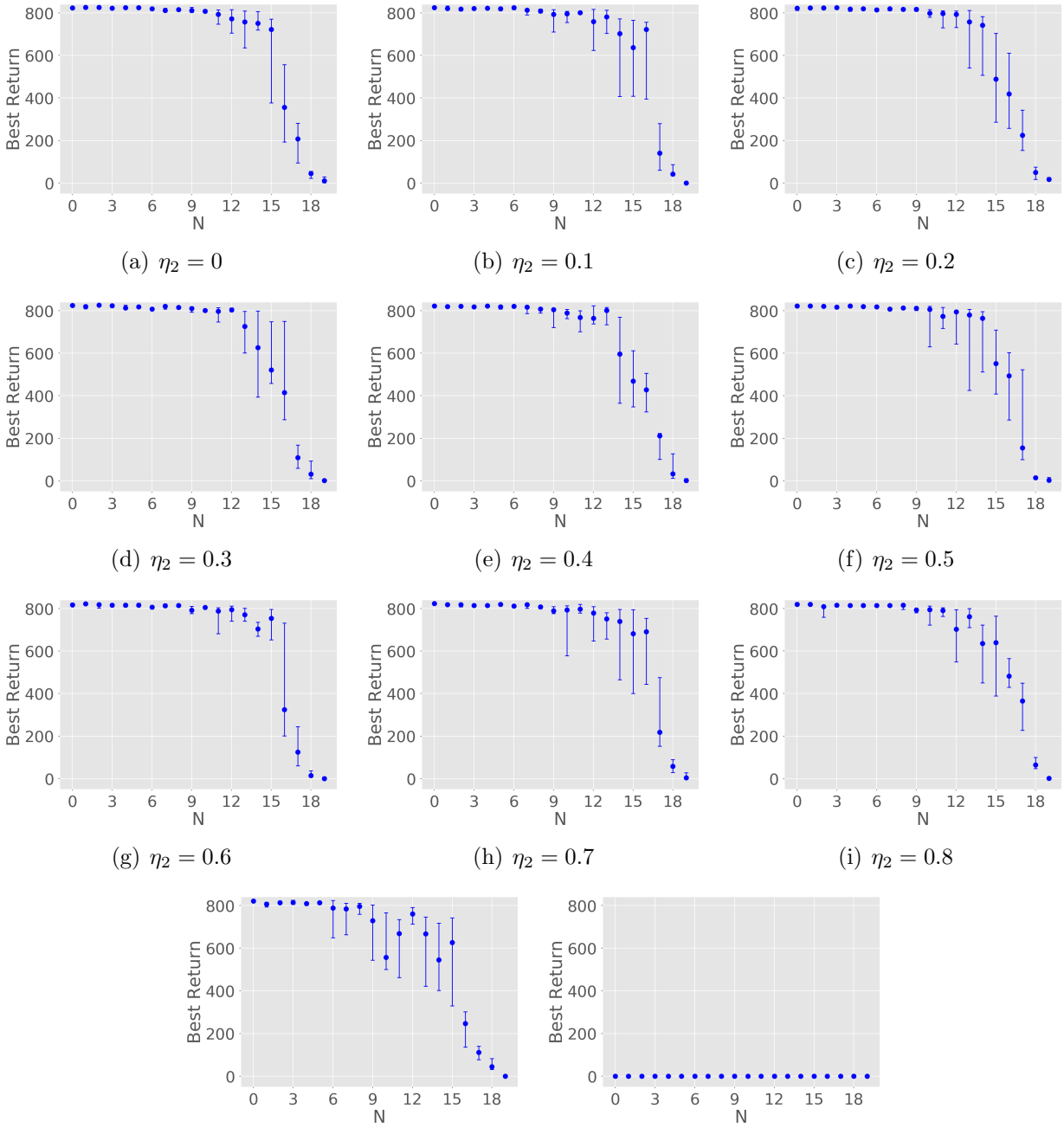
(d) $\eta_1 = 0.3$ (e) $\eta_1 = 0.4$ (f) $\eta_1 = 0.5$

(g) $\eta_1 = 0.6$ (h) $\eta_1 = 0.7$ (i) $\eta_1 = 0.8$

(j) $\eta_1 = 0.9$ (k) $\eta_1 = 1.0$

Figure 5.4: Ablation study for η_1.

5.I.3 Ablation study for η_2



(a) $\eta_2 = 0$ (b) $\eta_2 = 0.1$ (c) $\eta_2 = 0.2$

(d) $\eta_2 = 0.3$ (e) $\eta_2 = 0.4$ (f) $\eta_2 = 0.5$

(g) $\eta_2 = 0.6$ (h) $\eta_2 = 0.7$ (i) $\eta_2 = 0.8$

(j) $\eta_2 = 0.9$ (k) $\eta_2 = 1.0$

Figure 5.5: Ablation study for η_2.

We clarify that figure 5.5(k) shows no progress because the algorithm diverged for the chosen step-sizes in this ablation study.

CHAPTER 6

Concluding Remarks and Future Work

In this dissertation we have addressed four problems and provided solutions to each of them:

- In Chapter 2 we considered the distributed policy evaluation problem and proposed the *Fast Difussion for Policy Evaluation* algorithm. FDPE is designed for the finite sample case and we proved that for this case all agents converge to the correct solution linearly fast. This algorithm can be applied to two distinct scenarios. In the first scenario, multiple agents interact with different instances of the same MDP and collaborate to evaluate the same target policy. Due to the collaboration, all agents still converge to the correct solution even in cases where each agent's behavior policy only explores a portion of the total state-space. The second scenario we considered is the MARL setting with local rewards. FDPE can be use as part of a policy learning loop whose goal is to learn the optimal policy. The main disadvantage of this policy learning approach is that it is not sample efficient, this motivated the work in Chapter 3.

- In Chapter 3 we addressed the problem of learning the optimal team policy in a fully distributed MARL setting with local rewards. To this end, we introduced the *Difussion Team Policy Optimization* algorithm. DTPO is an off-policy algorithm and hence is sample efficient. Experimentation with this algorithm revealed two important issues. The first issue is that of scalability. DTPO does not scale gracefully with the number of agents due to the fact that each agent learns the policy for the team's state-action space. The second issue is that of exploration. DTPO can explore the state-space efficiently only in the tabular setting. In the general case where function approximation is used, it is unclear how to endow the algorithm with a behavior policy that is efficient at deep exploration.

- In Chapter 4 we studied the scalability issue that was identified in chapter 3. This scalability issue in MARL is independent of whether a global reward or local rewards are considered. Hence, in this chapter we considered the MARL setting were the reward is global. We proposed the *Logical Team Q-learning* algorithm. Our proposed solution is off-policy and learns factored functions that only depend on the observation and action of each individual agent. We proved that in the dynamic programming setting our LTQL obtains factored quantities that allow all agents to converge to a team optimal policy.

- Finally, in Chapter 5 we studied the problem of deep exploration. Since deep exploration is still an open question in single-agent RL, in this chapter we considered this domain. We connected the problem of deep exploration with the soft RL framework and derived the *Information Seeking Learner*. We proved that in the dynamic programming setting ISL converges to the optimal q-function. Experimental results showed that ISL achieves state of the art performance in three challenging deep exploration benchmarks.

The ideas and algorithms we introduced in this dissertation give rise to the two following important research problems:

- Derive an algorithm that obtains factored policies in the fully distributed setting where agents observe local instead of global rewards. Potentially this could be done by combining the factored relations we derived in Chapter 4 with the techniques we used in Chapter 3 to obtain DTPO.

- Extend the exploration work we presented in Chapter 5 to the multi-agent setting. In particular, we believe that the ideas we used to derive ISL could be combined with those of Chapters 3 and 4 to re-derive versions of DTPO and LTQL that are more efficient at performing exploration.

- Extend LTQL to the fully distributed case even when agents can only perceive a subset of the team's agents and provide answers to the following questions. Can this be

accomplished without loss of performance leveraging communication among agents? If sharing information is not possible, can expressions be derived, which depend on the TMDPs parameters, to bound the loss of performance?

- Once all the previous research problems are addressed the following natural challenge would be to put it all together to derive an algorithm for the MARL case that combines all the aforementioned features. In other words, the resulting algorithm should be capable of learning off-policy, in a fully distributed manner, perform efficient exploration and have good scalability properties in th setting where agents only have access to their own observations and local rewards.

REFERENCES

[1] A. L. Samuel, "Some studies in machine learning using the game of checkers," *IBM Journal of Research and Development*, vol. 3, no. 3, pp. 210–229, 1959.

[2] D. Michie and R. A. Chambers, "Boxes: An experiment in adaptive control," *Machine Intelligence*, vol. 2, no. 2, pp. 137–152, 1968.

[3] J. H. Holland, "Escaping brittleness: The possibilities of general-purpose learning algorithms applied to parallel rule-based systems," *Machine Learning: An Artifical Intelligence Approach*, vol. 2, pp. 593–623, 1986.

[4] R. S. Sutton, "Learning to predict by the methods of temporal differences," *Machine Learning*, vol. 3, no. 1, pp. 9–44, 1988.

[5] V. Mnih, K. Kavukcuoglu, D. Silver *et al.*, "Human-level control through deep reinforcement learning," *Nature*, vol. 518, no. 7540, p. 529, 2015.

[6] D. Silver, A. Huang, C. J. Maddison *et al.*, "Mastering the game of go with deep neural networks and tree search," *nature*, vol. 529, no. 7587, p. 484, 2016.

[7] D. Silver, J. Schrittwieser, K. Simonyan *et al.*, "Mastering the game of go without human knowledge," *Nature*, vol. 550, no. 7676, pp. 354–359, 2017.

[8] D. Silver, T. Hubert, J. Schrittwieser *et al.*, "A general reinforcement learning algorithm that masters chess, shogi, and go through self-play," *Science*, vol. 362, no. 6419, pp. 1140–1144, 2018.

[9] O. Vinyals, I. Babuschkin, J. Chung *et al.*, "Alphastar: Mastering the real-time strategy game starcraft ii," https://deepmind.com/blog/alphastar-mastering-real-time-strategy-game-starcraft-ii/, 2019.

[10] C. Berner, G. Brockman, B. Chan *et al.*, "Dota 2 with large scale deep reinforcement learning," *arXiv:1912.06680*, 2019.

[11] O. M. Andrychowicz, B. Baker, M. Chociej *et al.*, "Learning dexterous in-hand manipulation," *The International Journal of Robotics Research*, vol. 39, no. 1, pp. 3–20, 2020.

[12] M. Tan, "Multi-agent reinforcement learning: Independent vs. cooperative agents," in *Proceedings of the International Conference on Machine Learning*, 1993, pp. 330–337.

[13] C. Claus and C. Boutilier, "The dynamics of reinforcement learning in cooperative multiagent systems," *AAAI/IAAI*, vol. 1998, no. 746-752, p. 2, 1998.

[14] R. S. Sutton, D. A. McAllester, S. P. Singh, and Y. Mansour, "Policy gradient methods for reinforcement learning with function approximation," in *Advances in Neural Information Processing Systems*, Denver, USA, 2000, pp. 1057–1063.

[15] R. J. Williams, "Simple statistical gradient-following algorithms for connectionist reinforcement learning," *Machine Learning*, vol. 8, no. 3-4, pp. 229–256, 1992.

[16] S. M. Kakade, "A natural policy gradient," in *Advances in Neural Information Processing Systems*, 2002, pp. 1531–1538.

[17] V. R. Konda and J. N. Tsitsiklis, "Actor-critic algorithms," in *Advances in Neural Information Processing Systems*, 2000, pp. 1008–1014.

[18] J. Peters and S. Schaal, "Natural actor-critic," *Neurocomputing*, vol. 71, no. 7-9, pp. 1180–1190, 2008.

[19] V. Mnih, A. P. Badia, M. Mirza *et al.*, "Asynchronous methods for deep reinforcement learning," in *Proceedings of the International Conference on Machine Learning*, New York, USA, June 2016, pp. 1928–1937.

[20] J. Schulman, S. Levine, P. Abbeel *et al.*, "Trust region policy optimization," in *Proceedings of the International Conference on Machine Learning*, New York, USA, 2015, pp. 1889–1897.

[21] J. Schulman, F. Wolski, P. Dhariwal *et al.*, "Proximal policy optimization algorithms," *arXiv:1707.06347*, August 2017.

[22] M. L. Puterman, *Markov Decision Processes.: Discrete Stochastic Dynamic Programming*. Wiley, NY, 2014.

[23] C. Watkins and P. Dayan, "Q-learning," *Machine Learning*, vol. 8, no. 3-4, pp. 279–292, 1992.

[24] H. V. Hasselt, "Double q-learning," in *Advances in Neural Information Processing Systems*, 2010, pp. 2613–2621.

[25] H. Van Hasselt, A. Guez, and D. Silver, "Deep reinforcement learning with double Q-learning," in *Proceedings of the AAAI Conference on Artificial Intelligence*, Arizona, USA, 2016.

[26] Z. Wang, T. Schaul, M. Hessel *et al.*, "Dueling network architectures for deep reinforcement learning," in *Proceedings of the International Conference on Machine Learning*, New York, USA, June 2016, pp. 1995–2003.

[27] T. Lu, D. Schuurmans, and C. Boutilier, "Non-delusional q-learning and value-iteration," in *Advances in Neural Information Processing Systems*, 2018, pp. 9949–9959.

[28] R. Fox, A. Pakman, and N. Tishby, "Taming the noise in reinforcement learning via soft updates," in *Proceedings of the Conference on Uncertainty in Artificial Intelligence*, New York, USA, 2016, pp. 202–211.

[29] T. Haarnoja, H. Tang, P. Abbeel, and S. Levine, "Reinforcement learning with deep energy-based policies," in *Proceedings of the International Conference on Machine Learning*, Sydney, Australia, 2017, pp. 1352–1361.

[30] O. Nachum, M. Norouzi, K. Xu, and D. Schuurmans, "Bridging the gap between value and policy based reinforcement learning," in *Advances in Neural Information Processing Systems*, Long Beach, USA, 2017, pp. 2775–2785.

[31] T. Haarnoja, A. Zhou, P. Abbeel, and S. Levine, "Soft actor-critic: Off-policy maximum entropy deep reinforcement learning with a stochastic actor," in *Proceedings of the International Conference on Machine Learning*, Stockholm, Sweden, 2018, pp. 1856–1865.

[32] O. Nachum, M. Norouzi, K. Xu, and D. Schuurmans, "Trust-PCL: An off-policy trust region method for continuous control," *arXiv:1707.01891*, February 2018.

[33] B. Dai, A. Shaw, L. Li *et al.*, "SBEED: Convergent reinforcement learning with nonlinear function approximation," in *Proceedings of the International Conference on Machine Learning*, Stockholm, Sweden, 2018, pp. 1133–1142.

[34] T. Haarnoja, A. Zhou, S. Ha *et al.*, "Learning to walk via deep reinforcement learning," *arXiv:1812.11103*, 2018.

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

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

[37] L. Cassano, K. Yuan, and A. H. Sayed, "Distributed value-function learning with linear convergence rates," in *Proceedings of the European Control Conference*, Napoli, Italy, June 2019, pp. 505–511.

[38] L. Cassano, K. Yuan, and A. H. Sayed, "Multi-agent fully decentralized value function learning with linear convergence rates," *IEEE Transactions on Automatic Control*, pp. 1–1, 2020.

[39] L. Cassano, S. A. Alghunaim, and A. H. Sayed, "Team policy learning for multi-agent reinforcement learning," in *Proceedings of the IEEE International Conference on Acoustics, Speech and Signal Processing*, Brighton, UK, May 2019, pp. 3062–3066.

[40] L. Cassano and A. H. Sayed, "Logical team Q-learning: An approach towards factored policies in cooperative MARL," *arXiv:2006.03553*, June 2020.

[41] L. Cassano and A. H. Sayed, "ISL: A novel approach for deep exploration," *arXiv:1909.06293*, December 2019.

[42] I. Grondman, L. Busoniu, G. A. Lopes, and R. Babuska, "A survey of actor-critic reinforcement learning: Standard and natural policy gradients," *IEEE Transactions on Systems, Man, and Cybernetics, Part C (Applications and Reviews)*, vol. 42, no. 6, pp. 1291–1307, 2012.

[43] B. Kehoe, A. Matsukawa, S. Candido *et al.*, "Cloud-based robot grasping with the Google object recognition engine," in *IEEE International Conference on Robotics and Automation (ICRA)*, Karlsruhe, Germany, May 2013, pp. 4263–4270.

[44] B. Kehoe, S. Patil, P. Abbeel, and K. Goldberg, "A survey of research on cloud robotics and automation," *IEEE Transactions on Automation Science and Engineering*, vol. 12, no. 2, pp. 398–409, 2015.

[45] S. Gu, E. Holly, T. Lillicrap, and S. Levine, "Deep reinforcement learning for robotic manipulation with asynchronous off-policy updates," in *IEEE International Conference on Robotics and Automation (ICRA)*, Singapore, Singapore, May 2017, pp. 3389–3396.

[46] R. S. Sutton, H. R. Maei, and C. Szepesvári, "A convergent $o(n)$ temporal-difference algorithm for off-policy learning with linear function approximation," in *Advances in Neural Information Processing Systems*, Vancouver, Canada, 2009, pp. 1609–1616.

[47] R. S. Sutton, H. R. Maei, D. Precup *et al.*, "Fast gradient-descent methods for temporal-difference learning with linear function approximation," in *Proceedings of the International Conference on Machine Learning*, Montreal, Canada, 2009, pp. 993–1000.

[48] B. Liu, J. Liu, M. Ghavamzadeh *et al.*, "Finite-sample analysis of proximal gradient td algorithms." in *Proceedings of the Conference on Uncertainty in Artificial Intelligence (UAI)*, Amsterdam, Holland, 2015, pp. 504–513.

[49] H. R. Maei, *Gradient Temporal-Difference Learning Algorithms*. Ph.D. dissertation, University of Alberta, 20011.

[50] H. van Hasselt, A. R. Mahmood, and R. S. Sutton, "Off-policy TD(λ) with a true online equivalence," in *Proceedings of the Conference on Uncertainty in Artificial Intelligence*, Quebec City, Canada, 2014, pp. 330–339.

[51] D. Silver, A. Huang, C. J. Maddison *et al.*, "Mastering the game of Go with deep neural networks and tree search," *Nature*, vol. 529, no. 7587, p. 484, 2016.

[52] R. Johnson and T. Zhang, "Accelerating stochastic gradient descent using predictive variance reduction," in *Advances in Neural Information Processing Systems*, Lake Tahoe, USA, 2013, pp. 315–323.

[53] A. Defazio, F. Bach, and S. Lacoste-Julien, "Saga: A fast incremental gradient method with support for non-strongly convex composite objectives," in *Advances in Neural Information Processing Systems*, Montreal, Canada, 2014, pp. 1646–1654.

[54] S. S. Du, J. Chen, L. Li *et al.*, "Stochastic variance reduction methods for policy evaluation," in *Proceedings of the International Conference on Machine Learning*, Sydney, Australia, 2017, pp. 1049–1058.

[55] B. Ying, K. Yuan, and A. H. Sayed, "Variance-reduced stochastic learning under random reshuffling," *IEEE Transactions on Signal Processing*, vol. 68, pp. 1390–1408, February 2020.

[56] S. V. Macua, J. Chen, S. Zazo, and A. H. Sayed, "Distributed policy evaluation under multiple behavior strategies," *IEEE Transactions on Automatic Control*, vol. 60, no. 5, pp. 1260–1274, 2015.

[57] M. S. Stanković and S. S. Stanković, "Multi-agent temporal-difference learning with linear function approximation: Weak convergence under time-varying network topologies," in *Proceedings of the American Control Conference*, Boston, USA, July 2016, pp. 167–172.

[58] H.-T. Wai, Z. Yang, Z. Wang, and M. Hong, "Multi-agent reinforcement learning via double averaging primal-dual optimization," in *Advances in Neural Information Processing Systems*, 2018, pp. 9649–9660.

[59] B. Dai, A. Shaw, L. Li *et al.*, "Sbeed: Convergent reinforcement learning with nonlinear function approximation," in *Proceedings of the International Conference on Machine Learning*, Stockholm, Sweden, 2018, pp. 1133–1142.

[60] V. Mnih, A. P. Badia, M. Mirza *et al.*, "Asynchronous methods for deep reinforcement learning," in *Proceedings of the International Conference on Machine Learning*, 2016, pp. 1928–1937.

[61] K. Zhang, Z. Yang, H. Liu *et al.*, "Fully decentralized multi-agent reinforcement learning with networked agents," in *Proceedings of the International Conference on Machine Learning*, Stockholm, Sweden, July 2018, pp. 10–15.

[62] A. H. Sayed, "Adaptive networks," *Proceedings of the IEEE*, vol. 102, no. 4, pp. 460–497, 2014.

[63] A. Nedic and A. Ozdaglar, "Distributed subgradient methods for multi-agent optimization," *IEEE Transactions on Automatic Control*, vol. 54, no. 1, p. 48, 2009.

[64] K. Yuan, Q. Ling, and W. Yin, "On the convergence of decentralized gradient descent," *SIAM Journal on Optimization*, vol. 26, no. 3, pp. 1835–1854, 2016.

[65] J. F. Mota, J. M. Xavier, P. M. Aguiar, and M. Püschel, "D-ADMM: A communication-efficient distributed algorithm for separable optimization," *IEEE Transactions on Signal Processing*, vol. 61, no. 10, pp. 2718–2723, 2013.

[66] W. Shi, Q. Ling, G. Wu, and W. Yin, "Extra: An exact first-order algorithm for decentralized consensus optimization," *SIAM Journal on Optimization*, vol. 25, no. 2, pp. 944–966, 2015.

[67] A. Nedic, A. Olshevsky, and W. Shi, "Achieving geometric convergence for distributed optimization over time-varying graphs," *SIAM Journal on Optimization*, vol. 27, no. 4, pp. 2597–2633, 2017.

[68] G. Qu and N. Li, "Harnessing smoothness to accelerate distributed optimization," *IEEE Transactions on Control of Network Systems*, vol. 5, no. 3, pp. 1245–1260, 2017.

[69] C. Xi and U. A. Khan, "Dextra: A fast algorithm for optimization over directed graphs," *IEEE Transactions on Automatic Control*, vol. 62, no. 10, pp. 4980–4993, 2017.

[70] K. Yuan, B. Ying, X. Zhao, and A. H. Sayed, "Exact diffusion for distributed optimization and learning—part I: Algorithm development," *IEEE Transactions on Signal Processing*, vol. 67, no. 3, pp. 708–723, 2019.

[71] K. Yuan, B. Ying, X. Zhao, and A. H. Sayed, "Exact diffusion for distributed optimization and learning—part II: Convergence analysis," *IEEE Transactions on Signal Processing*, vol. 67, no. 3, pp. 724–739, 2019.

[72] A. H. Sayed, "Adaptation, learning, and optimization over networks," *Foundations and Trends in Machine Learning*, vol. 7, pp. 311–801, 2014.

[73] P. Di Lorenzo and G. Scutari, "Next: In-network nonconvex optimization," *IEEE Transactions on Signal and Information Processing over Networks*, vol. 2, no. 2, pp. 120–136, 2016.

[74] S. V. Macua, A. Tukiainen, D. G.-O. Hernández *et al.*, "Diff-DAC: Distributed actor-critic for multitask deep reinforcement learning," *arXiv:1710.10363*, 2017.

[75] S. P. Singh and R. S. Sutton, "Reinforcement learning with replacing eligibility traces," *Machine Learning*, vol. 22, no. 1-3, pp. 123–158, 1996.

[76] C. Dann, G. Neumann, and J. Peters, "Policy evaluation with temporal differences: A survey and comparison," *The Journal of Machine Learning Research*, vol. 15, no. 1, pp. 809–883, 2014.

[77] S.-Q. Shen, T.-Z. Huang, and G.-H. Cheng, "A condition for the nonsymmetric saddle point matrix being diagonalizable and having real and positive eigenvalues," *Journal of Computational and Applied Mathematics*, vol. 220, no. 1-2, pp. 8–12, 2008.

[78] D. S. Bernstein, *Matrix Mathematics*. Princeton University Press, 2005.

[79] M. Lauer and M. Riedmiller, "An algorithm for distributed reinforcement learning in cooperative multi-agent systems," in *Proceedings of the International Conference on Machine Learning*, Palo Alto, USA, 2000, pp. 535–542.

[80] S. Kapetanakis and D. Kudenko, "Reinforcement learning of coordination in cooperative multi-agent systems," in *Proceedings of the AAAI Conference on Artificial Intelligence*, Alberta, Canada, 2002, pp. 326–331.

[81] G. Tesauro, "Extending Q-learning to general adaptive multi-agent systems," in *Advances in Neural Information Processing Systems*, Vancouver, Canada, 2004, pp. 871–878.

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

[83] L. Matignon, G. Laurent, and N. Le Fort-Piat, "Hysteretic Q-learning: an algorithm for decentralized reinforcement learning in cooperative multi-agent teams." in *Proceedings of the IEEE/RSJ International Conference on Intelligent Robots and Systems*, San Diego, USA, 2007, pp. 64–69.

[84] C. Guestrin, D. Koller, and R. Parr, "Multiagent planning with factored MDPs," in *Advances in Neural Information Processing Systems*, 2002, pp. 1523–1530.

[85] C. Guestrin, M. Lagoudakis, and R. Parr, "Coordinated reinforcement learning," in *Proceedings of the International Conference on Machine Learning*, vol. 2, 2002, pp. 227–234.

[86] J. R. Kok and N. Vlassis, "Sparse cooperative Q-learning," in *Proceedings of the International Conference on Machine Learning*, 2004, p. 61.

[87] J. N. Foerster, G. Farquhar, T. Afouras *et al.*, "Counterfactual multi-agent policy gradients," in *Proceedings of the AAAI Conference on Artificial Intelligence*, 2018.

[88] J. K. Gupta, M. Egorov, and M. Kochenderfer, "Cooperative multi-agent control using deep reinforcement learning," in *International Conference on Autonomous Agents and Multiagent Systems*, Sao Paulo, Brazil, 2017, pp. 66–83.

[89] T. Rashid, M. Samvelyan, C. S. De Witt *et al.*, "Monotonic value function factorisation for deep multi-agent reinforcement learning," *arXiv:2003.08839*, 2020.

[90] K. Son, D. Kim, W. J. Kang *et al.*, "Qtran: Learning to factorize with transformation for cooperative multi-agent reinforcement learning," in *Proceedings of the International Conference on Machine Learning*, 2019, pp. 5887–5896.

[91] P. Sunehag, G. Lever, A. Gruslys *et al.*, "Value-decomposition networks for cooperative multi-agent learning based on team reward," in *Proceedings International Conference on Autonomous Agents and Multiagent Systems*, 2018, pp. 2085–2087.

[92] T. Rashid, M. Samvelyan, C. Schroeder *et al.*, "Qmix: Monotonic value function factorisation for deep multi-agent reinforcement learning," in *Proceedings of the International Conference on Machine Learning*, 2018, pp. 4295–4304.

[93] S. Omidshafiei, J. Pazis, C. Amato *et al.*, "Deep decentralized multi-task multi-agent reinforcement learning under partial observability," in *Proceedings of the International Conference on Machine Learning*, 2017, pp. 2681–2690.

[94] A. Tampuu, T. Matiisen, D. Kodelja *et al.*, "Multiagent cooperation and competition with deep reinforcement learning," *PloS one*, vol. 12, no. 4, 2017.

[95] J. Foerster, N. Nardelli, G. Farquhar *et al.*, "Stabilising experience replay for deep multi-agent reinforcement learning," in *Proceedings of the International Conference on Machine Learning*, 2017, pp. 1146–1155.

[96] S. Boyd and L. Vandenberghe, *Convex optimization*. Cambridge University Press, 2004.

[97] R. S. Sutton and A. G. Barto, *Reinforcement Learning: An Introduction*. MIT Press, 1998.

[98] O. Nachum, M. Norouzi, K. Xu, and D. Schuurmans, "Bridging the gap between value and policy based reinforcement learning," in *Advances in Neural Information Processing Systems*, 2017, pp. 2775–2785.

[99] L.-J. Lin, "Self-improving reactive agents based on reinforcement learning, planning and teaching," *Machine Learning*, vol. 8, no. 3-4, pp. 293–321, 1992.

[100] T. de Bruin, J. Kober, K. Tuyls, and R. Babuška, "Experience selection in deep reinforcement learning for control," *Journal of Machine Learning Research*, vol. 19, no. 9, pp. 1–56, 2018.

[101] D. P. Kingma and J. Ba, "Adam: A method for stochastic optimization," *arXiv:1412.6980*, 2014.

[102] F. A. Oliehoek, C. Amato *et al.*, *A Concise Introduction to Decentralized POMDPs*. Springer, 2016, vol. 1.

[103] J. Duchi, E. Hazan, and Y. Singer, "Adaptive subgradient methods for online learning and stochastic optimization," *Journal of Machine Learning Research*, pp. 2121–2159, July 2011.

[104] B. C. Stadie, S. Levine, and P. Abbeel, "Incentivizing exploration in reinforcement learning with deep predictive models," *arXiv:1507.00814*, 2015.

[105] I. Osband, C. Blundell, A. Pritzel, and B. Van Roy, "Deep exploration via bootstrapped DQN," in *Advances in Neural Information Processing Systems*, Barcelona, Spain, 2016, pp. 4026–4034.

[106] B. O'Donoghue, I. Osband, R. Munos, and V. Mnih, "The Uncertainty Bellman Equation and Exploration," in *Proceedings of the International Conference on Machine Learning*, Stockholm, Sweden, July 2018, pp. 3839–3848.

[107] Y. Burda, H. Edwards, D. Pathak *et al.*, "Large-scale study of curiosity-driven learning," *arXiv:1808.04355*, 2018.

[108] I. Osband, J. Aslanides, and A. Cassirer, "Randomized prior functions for deep reinforcement learning," in *Advances in Neural Information Processing Systems*, Montréal, Canada, 2018, pp. 8617–8629.

[109] M. Bellemare, S. Srinivasan, G. Ostrovski *et al.*, "Unifying count-based exploration and intrinsic motivation," in *Advances in Neural Information Processing Systems*, 2016, pp. 1471–1479.

[110] G. Ostrovski, M. G. Bellemare, A. van den Oord, and R. Munos, "Count-based exploration with neural density models," in *Proceedings of the International Conference on Machine Learning*, 2017, pp. 2721–2730.

[111] R. Houthooft, X. Chen, Y. Duan *et al.*, "Vime: Variational information maximizing exploration," in *Advances in Neural Information Processing Systems*, 2016, pp. 1109–1117.

[112] D. Pathak, P. Agrawal, A. A. Efros, and T. Darrell, "Curiosity-driven exploration by self-supervised prediction," in *Proceedings of the International Conference on Machine Learning*, 2017, pp. 2778–2787.

[113] Y. Burda, H. Edwards, A. Storkey, and O. Klimov, "Exploration by random network distillation," *arXiv:1810.12894*, 2018.

[114] I. Osband, Y. Doron, M. Hessel *et al.*, "Behaviour suite for reinforcement learning," *arXiv:1908.03568*, August 2019.

[115] I. Osband, B. Van Roy, and Z. Wen, "Generalization and exploration via randomized value functions," in *Proceedings of the International Conference on Machine Learning*, June 2016, pp. 2377–2386.