MULTI-AGENT REINFORCEMENT LEARNING: ANALYSIS AND APPLICATION

by

Paulo Cesar Heredia

A Dissertation

Submitted to the Faculty of Purdue University

In Partial Fulfillment of the Requirements for the degree of

Doctor of Philosophy

School of Aeronautics and Astronautics

West Lafayette, Indiana

May 2022

THE PURDUE UNIVERSITY GRADUATE SCHOOL
STATEMENT OF COMMITTEE APPROVAL

Dr. Shaoshuai Mou, Chair

School of Aeronautics and Astronautics

Dr. Martin J. Corless

School of Aeronautics and Astronautics

Dr. Dengfeng Sun

School of Aeronautics and Astronautics

Dr. Inseok Hwang

School of Aeronautics and Astronautics

Approved by:

Dr. Gregory A. Blaisdell

ACKNOWLEDGMENTS

I would first like to thank my academic advisor Dr. Shaoshuai Mou for the opportunity and encouragement to pursue a Ph.D., as well as for helping me connect with people that have also helped me in my research and career development. And I appreciate how he has challenged me to question myself and be more critical of my work and others work, which is a valuable skill that will be useful even beyond graduation.

I also would like to thank all my collaborators that have helped me with the research presented in this thesis, including Dr. Zhuoran Yang, Dr. Kaiqing Zhang, Hasan Ghadialy, Dr. Eloy Garcia, and Dr. Jemin George, and the members of my committee, Prof. Martin Corless, Prof. Dengeng Sun, and Prof. Inseok Hwang. I also thank Wanxin Jin, Xuan Wang, Zihao Liang, Jeffrey Hall, Kevin Shi, Jiazhen Zhou, Ayush Rai, and Wenjian Hao who have been good friends, colleagues, and classmates throughout my Ph.D. studies. I would especially like to thank Wanxin Jin and Xuan Wang who have served as important mentors to guide me in the first couple of years of my Ph.D. and were always willing to answer my questions and provide help.

I also want to thank my parents, Efrain Heredia and Ana Aguilar, who have given me love and support throughout my Ph.D. studies and always provided a loving home to go back to at the end of a challenging semester. The hard work and sacrifices they have made to provide for my brother and I, has motivated me to keep moving forward in my studies, with the hope of one day repaying at least a fraction of what they have given me. I also want to thank my younger brother Cesar Heredia for giving me someone to talk with about soccer, movies, and video games, for supporting the same teams I do, and for being a good brother in general.

Finally, I want to thank Krutik Mehta and Kartik Misra for being great roommates and friends, and for helping make my time at Purdue more fun despite the rigors of coursework and research.

TABLE OF CONTENTS

LIST OF FIGURES 7

LIST OF SYMBOLS 9

ABSTRACT 10

1 INTRODUCTION 12

1.1 Research Gaps 13

1.2 Summary of Research Contributions 16

1.3 Summary of Publications 19

2 DISTRIBUTED MULTI-AGENT REINFORCEMENT LEARNING BY ACTOR-CRITIC METHOD 22

2.1 The Update 24

2.1.1 Critic Training 25

2.1.2 Actor Training 28

2.2 Main Result 29

2.3 Simulations 35

2.4 Conclusion 38

3 DISTRIBUTED OFFLINE REINFORCEMENT LEARNING 39

3.1 Introduction 39

3.2 Problem Formulation 41

3.3 Key Idea 43

3.4 Algorithm and Main Result 46

3.5 Proof of Main Result 49

3.6 Simulation Results 56

3.6.1 Linear System 56

3.6.2 Nonlinear System 58

3.7 Conclusion 60

4 DISTRIBUTED REINFORCEMENT LEARNING WITH CROSS MODAL OBSERVATIONS . . . 62
4.1 Introduction . . . 62
4.2 Problem Formulation . . . 64
4.3 Preliminaries . . . 67
4.4 Approach . . . 68
4.5 Main Result . . . 71
4.6 Simulation Results . . . 74
4.7 Conclusion . . . 76

5 FINITE-SAMPLE ANALYSIS OF DISTRIBUTED Q-LEARNING FOR MULTI-AGENT NETWORKS . . . 78
5.1 Preliminaries . . . 79
5.1.1 Multi-Agent MDP . . . 79
5.1.2 Linear Function Approximation . . . 80
5.1.3 A Distributed Q-Learning Algorithm . . . 81
5.2 Finite-Sample Analysis for Q-Learning . . . 82
5.3 Proof of Main Result . . . 84
5.4 Conclusion . . . 88
5.5 Supplementary . . . 89
5.5.1 Proof of Lemma 1 . . . 89
5.5.2 Proof of Lemma 2 . . . 91

6 FINITE-SAMPLE ANALYSIS OF MULTI-AGENT POLICY EVALUATION WITH KERNELIZED GRADIENT TEMPORAL DIFFERENCE . . . 94
6.1 Preliminaries and the Problem . . . 95
6.2 Functional Stochastic Quasi-Gradient Method with Consensus . . . 97
6.3 Main Result . . . 99
6.4 Proof of Main Result . . . 101
6.5 Conclusion . . . 105
6.6 Supplementary . . . 105

6.6.1 Proof of Lemma 6.4.2 . . . 105
6.6.2 Proof of Lemma 6.4.3 . . . 106
6.6.3 Proof of Lemma 6.4.4 . . . 108

7 LEARNING-ASSISTED LOAD CONTROL DESIGN FOR TRANSACTIVE ENERGY SYSTEM . . . 110
7.1 Problem Statement . . . 112
7.2 Learning-assisted Bidding Strategy . . . 113
7.2.1 Tabular Q-learning . . . 115
7.2.2 DQN . . . 115
7.2.3 DDPG . . . 116
7.3 Case Studies . . . 117
7.3.1 Control Performance . . . 120
7.3.2 Training Performance . . . 123
7.3.3 Performance in Non-Competitive Market . . . 125
7.4 Conclusion . . . 129

8 DISTRIBUTED STATE ESTIMATION FOR NONLINEAR SYSTEMS WITH UNKNOWN PARAMETERS . . . 132
8.1 Introduction . . . 132
8.2 Problem Statement . . . 134
8.3 Approach . . . 135
8.4 Stability Analysis of Distributed Observer . . . 138
8.5 Proof of Main Result . . . 141
8.6 Simulations . . . 146
8.7 Conclusion . . . 147

9 SUMMARY . . . 148
9.1 Future Directions . . . 151

REFERENCES . . . 154

LIST OF FIGURES

2.1 Distributed Multi-Agent Reinforcement Learning 24

2.2 Averaged State-Value Parameter for the Critic 36

2.3 Averaged Action-Value Parameters for the Critic 36

2.4 Averaged Policy Parameters for the Actor 37

3.1 Network 56

3.2 Cartpole System 58

3.3 Comparing proposed distributed algorithm, which uses network of agents, with the same algorithm using where agents learn independently without sharing any information. Average of Discounted Rewards is calculated over 100 episodes. Each episode has random initial conditions. 60

4.1 Example of network of multi-agent systems 65

4.2 Network of Multi-Agent Systems interaction with plant 65

4.3 Low level picture of Network of Multi-Agent Systems 66

4.4 Example of Bipartite Graph $\mathbb{G}_B$ for Column Assignment 69

4.5 Squared Error 75

4.6 Squared Consensus Error 76

7.1 Agent Feedback 113

7.2 DQN Algorithm 116

7.3 DDPG Algorithm 117

7.4 Price Sensitivity 118

7.5 Cumulative Reward Over Week. DDPG had a runtime of 2.25 hours and DQN took 2.64 hours. For Q tab we were not able to measure the runtime but it ran for around a day 121

7.6 Cumulative Reward Over Week(Only DQN and DDPG) 122

7.7 Cumulative Rewards for Day 2 123

7.8 Indoor Temperature for Day 2 124

7.9 Price Bids for Day 2 125

7.10 Range of average episode rewards received during training for 10 trials of DQN and DDPG with different random seeds. Rewards are averaged every 10 episodes. Shaded region indicates one standard deviation from mean. 126

7.11 Range of episode rewards received during training for 10 trials of DQN and DDPG with different random seeds. Shaded region indicates one standard deviation from mean. 127

7.12 Averaged Reward for Tabular Q 10x5x19. Rewards averaged every 7 episodes .Resulting agent used in comparison plots 128

7.13 Episode Reward for Tabular Q 10x5x19. Resulting agent used in comparison plots 128

7.14 Cumulative Rewards Over Week, w=0.5 129

7.15 Cumulative Rewards Over Week, w=0.2 130

8.1 Problem Block Diagram 135

8.2 Sum of Norm Squared State Estimation Error 146

8.3 Sum of Norm Squared Parameter Estimation Error 147

LIST OF SYMBOLS

$\text{diag}\{A_1, A_2, ..., A_n\}$	denotes the block diagonal matrix made up of matrices $A_1, A_2, ..., A_n$
$\text{col}\{A_1, A_2, ..., A_n\}$	denotes column-wise stacking of matrices $A_1, A_2, ..., A_n$
I_n	denotes the $n \times n$ identity matrix
$\mathbf{1}_n$	denotes the all ones vector of dimension n
$A^\top$	denotes the transpose of matrix A
$\boldsymbol{\pi}(\cdot\|\cdot)$	denotes a probability distribution, which depending on the chapter might be a probability density function or probability mass function. This will be specified in each chapter.
$y \sim \mathcal{Y}$	denotes "random variable y is sampled from distribution $\mathcal{Y}$"

ABSTRACT

With the increasing availability of data and the rise of networked systems such as autonomous vehicles, drones, and smart girds, the application of data-driven, machine learning methods with multi-agents systems have become an important topic. In particular, reinforcement learning has gained a lot of popularity due to its similarities with optimal control, with the potential of allowing us to develop optimal control systems using only observed data and without the need for a model of a system's state dynamics. In this thesis work, we explore the application of reinforcement learning with multi-agents systems, which is known as multi-agent reinforcement learning (MARL). We have developed algorithms that address some challenges in the cooperative setting of MARL. We have also done work on better understanding the convergence guarantees of some known multi-agent reinforcement learning algorithms, which combine reinforcement learning with distributed consensus methods. And, with the aim of making MARL better suited to real-world problems, we have also developed algorithms to address some practical challenges with MARL and we have applied MARL on a real-world problem.

In the first part of this thesis, we focus on developing algorithms to address some open problems in MARL. One of these challenges is learning with output feedback, which is known as partial observability in the reinforcement learning literature. One of the main assumptions of reinforcement learning in the singles agent case is that the agent can fully observe the state of the plant it is controlling (we note the "plant" is often referred to as the "environment" in the reinforcement learning literature. We will use these terms interchangeably). In the single agent case this assumption can be reasonable since it only requires one agent to fully observe its environment. In the multi-agent setting, however, this assumption would require all agents to fully observe the state and furthermore since each agent could affect the plant (or environment) with its actions, the assumption would also require that agent's know the actions of other agents. We have also developed algorithms to address practical issues that may arise when applying reinforcement learning (RL) or MARL on large-scale real-world systems. One such algorithm is a distributed reinforcement learning algorithm that allows us to learn in cases where the states and actions are both continuous and of large

dimensionality, which is the case for many real-world applications. Without the ability to handle continuous states and actions, many algorithms require discretization, which with high dimensional systems can become impractical. We have also developed a distributed reinforcement learning algorithm that addresses data scalability of RL. By data scalability we mean how to learn from a very large dataset that cannot be efficiently processed by a single agent with limited resources.

In the second part of this thesis, we provide a finite-sample analysis of some distributed reinforcement learning algorithms. By finite-sample analysis, we mean we provide an upper bound on the squared error of the algorithm for a given iteration of the algorithm. Or equivalently, since each iteration uses one data sample, we provide an upper bound of the squared error for a given number of data samples used. This type of analysis had been missing in the MARL literature, where most works on MARL have only provided asymptotic results for their proposed algorithms, which only tells us how the algorithmic error behaves as the number of samples used goes to infinity.

The third part of this thesis focuses on applications with real-world systems. We have explored a real-world problem, namely transactive energy systems (TES), which can be represented as a multi-agent system. We have applied various reinforcement learning algorithms with the aim of learning an optimal control policy for this system. Through simulations, we have compared the performance of these algorithms and have illustrated the effect of partial observability (output feedback) when compared to full state feedback.

In the last part we present some other work, specifically we present a distributed observer that aims to address learning with output feedback by estimating the state. The proposed algorithm is designed so that we do not require a complete model of state dynamics, and instead we use a parameterized model where the parameters are estimated along with the state.

1. INTRODUCTION

The field of reinforcement learning (RL) has seen major success recently in areas such as autonomous driving [1], games [2], [3], and robotics [4], [5]. A major reason for this success can be attributed to the combination of deep neural networks with reinforcement learning algorithms. A subset of these algorithms, called model-free algorithms, are of particular interest since they do not require knowledge of the governing dynamics of the system of interest. Some of the more popular algorithms of this type include: temporal difference learning (TD) [6], Q-learning [7], [8], SARSA [9], and asynchronous advantage actor-critic (A3C) [10].

Reinforcement learning attempts to improve the performance of an agent or system by assigning greater rewards to actions that lead to better outcomes. The generality of this method has made it applicable to various fields of study. Some of the benefits of reinforcement learning are that it does not require knowledge of optimal behavior as in supervised learning, and that because of their sequential nature they can be used with online-learning algorithms [11]. Furthermore, because reinforcement learning can be used for sequential decision making stochastic systems [12], it is highly applicable for developing optimal controllers [13].

The above examples demonstrate the power of reinforcement learning for single agents and how far it has been developed, but multi-agent reinforcement learning (MARL) poses a more challenging problem with relatively few results both empirical and theoretical. Some of the reasons that make MARL more challenging [14] are credit assignment[15]–[21], non-stationary environments due to actions by other agents [22]–[27], and coordination [16], [18], [23], [24], [28]. The exact set of issues that would have to be resolved depends on the variation of MARL problem that is being considered [29].

A simple solution to the multi-agent issues would be to use centralization, where we assign a central node with gathering all information (state, actions, rewards) from every agent in the network. The central node would then process all the information and broadcast the actions assigned to each agent. The problem with this approach is that is not scalable to large netowrks due to the high communication costs that would be incurred in order to send information to the central node, which may be spatially very far from many agents [30].

Furthermore, the central node would need to have a large computational capacity in order to process all the information from each agent, and it would be focal point for malicious attacks[31], [32]. This makes centralized multi-agent reinforcement learning impractical and maybe even infeasible for very large and spatially distributed networks. This motivates our work to further the theory of MARL and as well as its applications in real world systems.

1.1 Research Gaps

There has been a lot of work on MARL providing some theoretical results and empirical success, however there are still many issues and challenges left to be explored. Here we outline some of those challenges which has motivated the work in this thesis:

1) *Most theoretical results assume finite spaces or linear function approximation*

   The early examples of MARL [17], [23], [24], [27], [30], [33] have assumed finite states and actions, which allows them to implement a tabular form of reinforcement learning that does not scale well to infinite states and actions. More recently, some works have assumed linear functions approximations in order to achieve results in MARL[31], [34], [35]. It would of great practical use if these results can be extended to reinforcement learning with more general function approximations, such as deep reinforcement learning or kernelized function approximation. Some works which explore deep neural networks for function approximation are [36]–[38], however these works either have only empirical results or are not in a distributed framework. In [32], [39]–[41] they consider only policy evaluation with some fixed policy, and so these algorithms cannot be immediately used to develop optimal control policies.

2) *Few finite sample results for RL algorithms*

   In most proofs of convergence for reinforcement, the main result is that the error either converges to zero asymptotically or that the error asymptotically converges to finite region around zero (depending on the assumptions). In either case, the result only tells us information about the error after an infinite amount of steps and infinite samples of data have been processed. These results do not tell us about the error for a finite

amount of samples and steps. Such a finite sample analysis of the error is needed so that we can know how much data is needed by a particular algorithm in order to reach a certain level of accuracy. So far these results are relatively few in the literature, and especially so for the multi-agent setting.

3) *Output Feedback(Partial Observability) - Algorithms usually require global state and/or actions of other agents*

An important assumption of most reinforcement algorithms is that the plant(or environment) dynamics are time-invariant (or stationary in the RL literature) from the learning agents perspective. For the plant to be time-invariant to the agent, the plant can not have any time varying components that are not observable to the agent, and so the agent must know all the necessary information needed to predict the evolution of the state of the plant. In the context of Markov Decision Processes, which is the mathematical foundation of reinforcement learning, this means the agent must observe the state of the plant. This is an issue in MARL where either we may only have local information on the plant, or even if we can observe the global state of the plant, if we cannot observe the actions of other agents then they essentially become unobservable components of the plant. In either case the plant becomes time-variant (or non-stationary) to the individual agents and so regular reinforcement learning algorithms break down. Most works in MARL which look to address this issue, pose the problem as a Decentralized Partially Obsevable MDP, but this is still a relatively new and open problem where most approaches make use of some global information or a learned model of global information in order to convert the problem from decentralized learning to centralized learning [29], [42]–[45]. Some papers such as [25], [46], explore problem of learning coordination with independent learners, but do not consider communication among agents. We also see a similar problem in [47], which is motivated by target tracking with sensor networks, but this work assumes that agents are independent of each other's actions. Another work is [44] which presents an algorithm for cooperative and competitive scenarios, but requires each agent to learn a decision making model of other agents.

4) *Algorithms require global rewards*

Just as being able to observe the full state of the plant is important, it is also an important assumption to all reinforcement learning algorithms that an agent can observe or compute the reward resulting from its actions. However, in cooperative MARL settings this becomes an issue because though each agent receives an individual reward, the goal of cooperative MARL problems is to maximize the sum of rewards not just individual rewards. An easy solution would be for each agent to communicate each other's rewards. However, this could lead to privacy issues since the reward could potentially contain sensitive or confidential information that individual agents would not want to share [31]. Though there are several works that have explored the case of private rewards [31], [32], [42], [45], they still require full knowledge of the state. This motivates the need for more distributed RL algorithms where along with partial state information, rewards are private to each agent instead of globally known.

5) *Evaluation and comparison of RL algorithms on real world problems*

There is a need for implementation and comparisons of performance of RL and MARL algorithms for problems of practical interest. In practice, performance of RL algorithms is very problem specific so its not clear what algorithm will work best for a particular problem until it is implemented and tested. There are some works comparing deep RL algorithms, such as policy gradient algorithms and DQN , but they are studied on specific use cases [4], [48]–[53] or on either toy examples or games [46], [54]–[56]. In [57] a very comprehensive comparison of temporal difference algorithms is given but again they are applied to toy examples such as the cartpole problem or the 20 link pole balancing problem, which, though of high dimensionality and difficulty, can not fully capture the nuances and difficulty of a real world problem.

6) *Scalability*

There are some challenges that make implementing reinforcement learning and MARL difficult to implement in large-scale real-world problems. One such problem is learning with large, continuous state and action spaces. As previously discussed, many rein-

forcement learning algorithms have been developed under the assumption that states and actions are discrete [8], [24], [27], which for many real-world systems is not the case. As such, these algorithms would require the discretization of a continuous state space or action space, which could become impractical since the size of the discretization would increase exponentially with the size of state or action vector[58]–[60]. Similarly, another important practical problem deals with data scalability [61]–[63]. Since reinforcement learning is a data driven method, the more complex the system the more data that will be required to learn a control policy for that system [60], [64], [65]. As such, since many real-world systems are very complex, we need reinforcement learning algorithms that scale well with the size of the dataset [66].

1.2 Summary of Research Contributions

Based on the research gaps discussed above, we have made the following contributions with our research and proposed algorithms:

In Chapter 2, we address the scalability challenge that results from systems that have continuous states and actions. As such, we develop a distributed reinforcement learning algorithm that works with both continuous state and action spaces. Due to the high dimensionality of most practical problems, it is important for reinforcement learning algorithms to be able to work with continuous state and action spaces. Otherwise, the states and action spaces need to discretized which can easily lead to an impractical amount of discretized states and actions. In the MARL setting, this is even more important because the number of actions would scale exponentially with the number of agents, therefore even if each agent only decides between a couple of actions, the total number of actions for the network would be large for a large network. Usually, continuous spaces are handles by including some form of function approximation, such as using neural nets or linear function approximation. As such, we have developed an algorithm that uses function approximation in order to handle continuous spaces.

In Chapter 3, we tackle the challenge of data scalability with reinforcement learning. Specifically, we consider the problem of learning from a large, fixed dataset that is too large

to be efficiently processed by a single agent. The issue of data scalability is an important practical problem since many industries and companies have acquired large datasets over time but they have limited computational resources at any single location. In order to learn from all this data in a reasonable amount of time, there a is need for distributed learning algorithms that can split up the computational load over many locations or nodes. To address this problem we have developed a distributed reinforcement learning algorithm where such a large dataset is partitioned and distributed amongst a network of agents. By splitting up that dataset amongst several agents, our approach is not limited by the computational resources of a single agent and so our approach can scale well with the size of the dataset.

In chapter 4 we address the challenge of learning from output feedback (or partial observability in the RL literature). The problem of multi-agent reinforcement learning with partial observability is a very challenging and open problem. The ability to handle partial observability addresses both the issue with not knowing what actions other agents make and also the very realistic case that an agent might not have full information about its plant (or environment). If some form of centralization is allowed, there are many works that have leveraged such a centralized structure to allow each agent to learn optimal policies. If centralization is not allowed, however, the problem is more challenging as each agent can only use local information. As such, we have developed algorithms that can accomplish learning in a distributed manner, only leveraging local information and communication between neighboring agents. In chapter 4, along with partial observability, another important challenge that we address is the issue of private rewards. Many works in MARL have assumed that either all agents receive the same reward or that agents can share the reward with each other. This would not be the case when the reward function contains sensitive, private information. Even in cooperative settings, each agent might not be willing to share their private reward and so it is important to develop algorithms that would not requiring sharing of rewards. Though there are many works that have similarly addressed this issue, they have assumed full observability of the state. Thus, we have addressed both reward privacy and partial observability in our work.

In Chapter 5, we provide a finite sample analysis of a distributed reinforcement learning algorithm. Most of the theoretical results in reinforcement learning have provided proofs of

asymptotic performance, where a bound on the error is given which the true error approaches as the number of algorithm updates approaches infinity. This has, off course, been very important in forming the foundation for reinforcement learning. But until recently, there have been no results providing a bound on the error for a finite amount of algorithm updates. Since each update uses at least one data sample, this is sometimes referred to as a finite-sample analysis. This type of analysis allows us to determine how many data samples are needed to guarantee a certain level of accuracy. Furthermore, such an analysis is even less available in the MARL setting, has motivated our work in providing a finite-sample analysis of reinforcement learning algorithms for the distributed, multi-agent setting.

In Chapter 6, we provide a finite-sample analysis of a reinforcement learning algorithm using a more general function approximation. As will be discussed later on, function approximation is necessary when dealing with continuous states and actions. It is also important, however, that the form of the function approximation is as general as possible so that if theoretical results are achieved, these results can apply to a larger set of functions. So far the literature has mostly made use of linear function approximations, which allows for good theoretical results but does not accept many functions of practical use. Therefore, along with linear function approximation, will have also studied a more general function approximation for our theoretical work.

In Chapter 7, we explore the application of reinforcement learning to a practical system (transactive energy systems). An important part of research is in demonstrating that established or newly developed algorithms can work for practical applications. Though many works exist applying reinforcement learning, especially deep reinforcment learning, many of them are applied to games and toy problems, which though are of high complexity do not offer an immediate practical benefit. There is also work that studies the performance of reinforcement learning on specific and practical uses cases, but since each particular use case presents its own challenges when using reinforcement learning, application to new systems must be studied on a case by case basis. Accordingly, we have studied the performance of reinforcement learning on transactive energy systems. Transactive energy is a new paradigm for demand-side control in power grids, where the interaction between power suppliers and consumers is designed to emulate a market system . A transactive energy system usually con-

sists of a group of distributed energy resources such as smart loads, distributed generation, and even energy storage [67]–[69].

In chapter 8, we address partial observability by developing a distributed observer based on the unscented Kalman filter. The motivation is that such a distributed observer will allow each agent of a multi-agent system to estimate the state, and so the distributed observer could be combined with a distributed reinforcement learning algorithm without needing to observe the full state. Crucially, we developed a distributed observer for both state estimation and parameter estimation. This allows us to relax the assumption of knowledge of state dynamics, which is usually needed with state estimation, and instead we assume we have a parametrized model of the state dynamics where the parameters are unknown. As such, the proposed distributed observer can be combined with a distributed reinforcement learning algorithm to achieve an optimal control policy without fully observing the state and without complete knowledge of state dynamics.

1.3 Summary of Publications

The content of Chapter 2 appears in:

- Paulo C. Heredia, Shaoshuai Mou. "Distributed Multi-Agent Reinforcement Learning by Actor-Critic Method". *IFAC-PapersOnLine*, 52/20, pp. 363-368 (2019). © 2019 International Federation fo Automatic Control. Reproduced with permission from Paulo C. Heredia and Shaoshuai Mou.

The content of Chapter 5 appears in:

- Heredia, Paulo, Hasan Ghadialy, and Shaoshuai Mou. "Finite-Sample Analysis of Distributed Q-learning for Multi-Agent Networks." In *2020 American Control Conference (ACC)*, pp. 3511-3516. IEEE, 2020. © 2020 IEEE. Reprinted, with permission, from Paulo Heredia, Hasan Ghadialy, Shaoshuai Mou.

The content of Chapter 6 appears in:

- Paulo Heredia and Shaoshuai Mou. "Finite-sample analysis of multi-agent policy evaluation with kernelized gradient temporal difference." In *2020 59th IEEE Conference on*

Decision and Control (CDC), pp. 5647-5652. IEEE, 2020. © 2020 IEEE. Reprinted, with permission, from Paulo Heredia and Shaoshuai Mou.

The content of Chapter 8 appears in:

- Paulo Heredia, Eloy Garcia , Shaoshuai Mou. "Distributed State Estimation for Nonlinear Systems with UnknownParameters." Submitted to: *2022 American Control Conference (ACC)*.

PART I

Algorithms for MARL

2. DISTRIBUTED MULTI-AGENT REINFORCEMENT LEARNING BY ACTOR-CRITIC METHOD

Introduction

Multi-agent reinforcement learning (MARL) has recently gained a lot research attention with extensive applications into mobile sensor networks, robotics, UAV swarms, cybersecurity, and so on [70]. Research challenges in MARL mainly come from the fact that each agent has its own local and private reward, and can only coordinate with nearby agents, which usually result in conflicts with other agents in credit assignment and coordinating actions [28]. This has led to a recently booming area of developing distributed algorithms for MARL, in which there is no centralized coordinator and only local coordination among nearby neighbors are allowed. Early results in the direction of distributed MARL usually assume finite states and actions to allow them to implement a tabular form of reinforcement learning [30], [33], which are not applicable to situations requiring infinite states and actions. Further progress has been achieved in [32], [39]–[41] which only consider evaluation of fixed policies and cannot be immediately used to develop optimal control policies. Recently researchers have started to develop distributed MARL based on actor-critic methods in single-agent case in [71], [72]. It has recently been shown that critic training could be reformulated as a primal-dual optimization problem in single-agent case in [38], with further generalization to distributed MARL algorithm in the worst-case by [32], followed by a finite sample analysis in [73]. Perhaps one of the most significant progress in distributed MARL based on actor-critic method are algorithms developed in [31], [74], in which each gent makes its own decision only based on locally observed information and communication among nearby neighbors, and the network connecting agents are time-varying.

Motivated by [31], [74], we in this work have also developed a distributed algorithm for MARL, based on actor-critic methods. With this framework each agent is tasked with training an actor to generate a control input given the state, and a critic to output a scalar value for the performance of the current policy, given a state and input pair. In addition, we consider continuous states/actions as in [74] but with a different variation of the actor-critic algorithm. Different from [31], which considers the expected time-averaged reward and

finite spaces for states/actions, we in this work considered the expected sum of discounted rewards over an infinite time horizon. Under results developed in this work, the policy evaluation algorithm proposed in [32] was used for action-value functions as well as state-value functions, which in turn implies that such policy evaluation algorithm can potentially be used in a distributed actor-critic framework based on [31].

*N*otation Let ∇_a denote the gradient with respect to a parameter a. To indicate the transpose of a matrix A, we use $A^\top$. Furthermore, by $\{a(t)\}$ we mean a sequence of $a(t)$ and by $a \sim d$ we mean "a is sampled from the distribution d". We also use $col\{a_1, a_2, .., a_n\}$ to denote the column-wise stacking of $a_1, .., a_n$.

Problem Formulation

Consider the case in which a network of m autonomous agents operate in an unknown environment (or plant). Let $x(t) \in \mathbb{R}^n$ denote the state of the plant at time t. For each control input $u_{\mathrm{i}}(t)$ from agent i to the plant, a local reward $r_{\mathrm{i}}(x(t), u(t))$ is produced, where $u(t) = col\{u_1(t), u_2(t), ..., u_m(t)\} \in \mathbb{R}^{\bar{n}}$. Here, each $r_{\mathrm{i}}(\cdot)$ is the private reward locally accessible to only agent i, and is not shared with other agents.

Let

$$R(x(t), u(t)) = \sum_{\mathrm{i}=1}^{m} \frac{1}{m} r_{\mathrm{i}}(x(t), u(t)) \tag{2.1}$$

which represents the average reward of all agents in the network. Let $\boldsymbol{\pi}$ denote a stochastic control policy such that $u \sim \boldsymbol{\pi}(x, u)$. Let $Q_{\boldsymbol{\pi}}$ denote the corresponding objective function, which is assumed to be a sum of discounted rewards $R(x(t), u(t))$ when a stochastic control policy $\boldsymbol{\pi}$ is applied to the plant. Namely,

$$Q_{\boldsymbol{\pi}}(x(t), u(t)) = \mathbb{E}[\sum_{k=0}^{\infty} \gamma^k R(x(k), u(k)) \mid x(0) = x(t), u(0) = u(t)]$$

where $\gamma \in (0, 1)$ is a discount factor. The goal of MARL in this paper is to achieve a globally optimal control policy $\boldsymbol{\pi}^*$ to maximize the objective function $Q_{\boldsymbol{\pi}}$.

In a multi-agent network, each agent i usually can only communicate with certain neighboring agents denoted by $\mathcal{N}_{\mathrm{i}}$, which includes agent i. The neighbor relations can be modeled

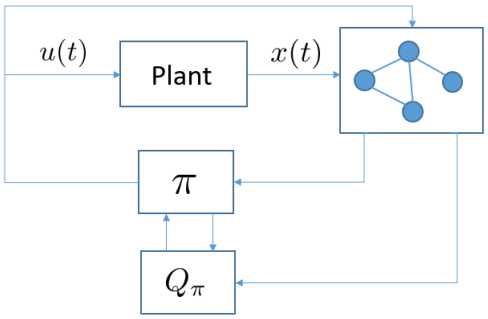

Figure 2.1. Distributed Multi-Agent Reinforcement Learning

by a connected undirected graph $\mathbb{G}$ such that there is an edge between i and j if and only if i and j are neighbors. Suppose each agent i controls $\boldsymbol{\pi}_\mathrm{i}$ (an estimate to the optimal control policy $\boldsymbol{\pi}^*$) and Q_i (the Q_{π_i} corresponding to $\boldsymbol{\pi}_\mathrm{i}$). The **problem** of interest, as indicated in Fig. 2.1, is to develop a distributed algorithm such that each agent achieves an ϵ approximation of the optimal policy $\boldsymbol{\pi}^*$ (denoted by $\boldsymbol{\pi}^* \pm \epsilon$), as well as its corresponding action value function $Q_{\pi^* \pm \epsilon}$, using only coordination with its nearby neighbors, namely,

$$\boldsymbol{\pi}_\mathrm{i} \to \boldsymbol{\pi}^* \pm \epsilon \tag{2.2}$$

$$Q_\mathrm{i} \to Q_{\pi^* \pm \epsilon}. \tag{2.3}$$

Here, $\boldsymbol{\pi}^* \pm \epsilon$ denotes a policy value in the interval $[\boldsymbol{\pi}^* - \epsilon, \boldsymbol{\pi}^* + \epsilon]$.

2.1 The Update

In this section we will develop a distributed algorithm for MARL by introducing an actor and a critic at each agent. That is, each agent is tasked with training an actor to generate a control input given the state (control policy) and a critic to output a scalar value for the performance of the current policy given a state and input pair (action-value function). In the following we will present the updates for both critic training and actor training.

2.1.1 Critic Training

We first assume $\boldsymbol{\pi}$ is fixed, and so the proposed approach is to train each agent's critic to converge to $Q_{\boldsymbol{\pi}}$

As is well known, the Bellman equation for reinforcement learning can be described in terms of $Q_{\boldsymbol{\pi}}$ as follows [75]:

$$Q_{\boldsymbol{\pi}}(x(t),u(t)) = R(x(t),u(t)) + \gamma \mathbb{E}_{x|x(t),u(t)}\left[V_{\boldsymbol{\pi}}(x(t+1))\right],$$

where

$$V_{\boldsymbol{\pi}}(x(t+1)) = \mathbb{E}_{u(t+1)|\boldsymbol{\pi}}\left[Q_{\boldsymbol{\pi}}(x(t+1),u(t+1))\right] \tag{2.4}$$

is the state-value function at $t+1$. The above Bellman equation can be used to directly compute the entries in $Q_{\boldsymbol{\pi}}$, which is however not directly applicable to continuous space of actions and states. To address this, we approximate $Q_{\boldsymbol{\pi}}$ as linear combination of given basis functions [76], that is,

$$Q_w(x,u) = w^\top \phi(x,u), \tag{2.5}$$

where $w \in \mathbb{R}^{q_1}$ is unknown and $\phi(x,u) \in \mathbb{R}^{q_1}$ is a column vector of basis functions.

Similarly, the control policy $\boldsymbol{\pi}$ can also be approximated as a parameterized function $\boldsymbol{\pi}_\theta$, where $\theta \in \mathbb{R}^p$ This can be achieved by defining $\boldsymbol{\pi}_\theta$ as a normal distribution with mean and standard deviation as functions of θ.

Let $\mathbf{R}$, $\mathbf{Q_w}$ and $\mathbf{V}_{\boldsymbol{\pi}}$ denote the vectors from stacking all R in (2.1), Q_w in (2.5), and $V_{\boldsymbol{\pi}}$ in (2.4), respectively, for every (x,u) pair. To ease notation we refer to $\mathbb{E}_{u|\boldsymbol{\pi}}$ as $\mathbb{E}_u$ and $\mathbb{E}_{x|x(t),u(t)}$ as $\mathbb{E}_x$. Then a nice estimate of $Q_{\boldsymbol{\pi}}$ can be achieved by minimizing the following mean squared projected bellman error(MSPBE) with respect to w [32], namely,

$$\min_w \ \text{MSPBE}(w) = \min_w \frac{1}{2}\|\boldsymbol{\Pi}_{\boldsymbol{\Phi}}(\mathbf{Q_w} - \mathbf{R} - \gamma \mathbb{E}_x\left[\mathbf{V}_{\boldsymbol{\pi}}\right])\|^2_{\mathbf{D}} + \rho\|w\|^2, \tag{2.6}$$

where $\mathbf{D} = \text{diag}[\{\mu_{\boldsymbol{\pi}_\theta}(x) \forall x \in \mathbb{R}^n\}]$ is a diagonal matrix with the stationary distribution of $\boldsymbol{\pi}_\theta$ on the diagonal; $\boldsymbol{\Pi}_{\boldsymbol{\Phi}} = \boldsymbol{\Phi}(\boldsymbol{\Phi}^\top \mathbf{D} \boldsymbol{\Phi})^{-1}\boldsymbol{\Phi}^\top \mathbf{D}$ is the projection onto the subspace $\{\boldsymbol{\Phi} w : w \in \mathbb{R}^{q_1}\}$; $\boldsymbol{\Phi}$ is the stacking of $\phi(x,u)$ for every (x,u) pair,and ρ is a free parameter for

regularization of w. From [32], and assuming $\mathbf{A}$ is invertible, we know this can also be rewritten as

$$\begin{aligned}\min_w \ \mathrm{MSPBE}(w) &= \min_w \ \frac{1}{2}\|\mathbf{\Phi}^\top \mathbf{D}(\mathbf{Q_w} - \mathbf{R} - \gamma \mathbb{E}_x\left[\mathbf{V_\pi}\right])\|^2_{(\mathbf{\Phi}^\top \mathbf{D} \mathbf{\Phi})^{-1}} + \rho\|w\|^2 \\ &= \min_w \ \frac{1}{2}\|\mathbf{A}w - \mathbf{b}\|^2_{\mathbf{A}^{-1}} + \rho\|w\|^2,\end{aligned}$$

where

$$\mathbf{A} = \mathbb{E}[A(t)], \mathbf{b} = \mathbb{E}[b(t)]$$

with

$$\begin{aligned}A(t) &= \phi(x(t), u(t))\phi(x(t), u(t))^\top \\ b(t) &= (R(x(t), u(t)) + \gamma V_\pi(x(t+1)))\phi(x(t), u(t))\end{aligned}$$

and $\|v\|_M = \sqrt{v^\top M v}$ for any vector v. Note that $\mathbf{A}$ and $\mathbf{b}$ are usually not available in practice since they are all computed with respect to the stationary distribution of $\boldsymbol{\pi}_\theta$, which denoted by $\mu_{\boldsymbol{\pi}_\theta}$ usually requires the knowledge of state dynamics of the plant. Thus instead of solving (2.7), we will solve its equivalent problem, as shown in [32]:

$$\min_w \frac{1}{m}\sum_{\mathrm{i}=1}^{m} \mathrm{MSPBE}_\mathrm{i}(w) \tag{2.7}$$

where

$$\mathrm{MSPBE}_\mathrm{i}(w) = \frac{1}{2}\|\hat{\mathbf{A}}w - \hat{\mathbf{b}}_\mathrm{i}\|^2_{\hat{\mathbf{A}}^{-1}} + \rho\|w\|^2, \tag{2.8}$$

$\hat{\mathbf{A}} = \frac{1}{T}\sum_{t=1}^{T} A(t)$, $\hat{\mathbf{b}}_\mathrm{i} = \frac{1}{T}\sum_{t=1}^{T} b_\mathrm{i}(t)$, and

$$b_\mathrm{i}(t) = (r_\mathrm{i}(x(t), u(t)) + \gamma V_\pi(x(t+1)))\phi(x(t), u(t)). \tag{2.9}$$

Then the problem of learning a good estimate to Q_π can be achieved by solving the optimization problem in (2.7). Similar to [32], we employ the following update at each agent i:

$$w_{\mathrm{i}}(t+1) = \sum_{\mathrm{j}=1}^{N} W_{\mathrm{i,j}} w_{\mathrm{i}}(t) - \alpha_1 s_{\mathrm{i}}(A(t), t)$$
$$\nu_{\mathrm{i}}(t+1) = \nu_{\mathrm{i}}(t) + \alpha_2 d_{\mathrm{i}}(A(t), b_{\mathrm{i}}(t), t),$$

Here, ν_{i} is the dual variable of agent i with Metropolis weights $W_{\mathrm{i,j}}$ given by

$$W_{\mathrm{i,j}} = \begin{cases} \frac{1}{\max\{\mathrm{e_i}, \mathrm{e_j}\}}, & \text{if } \mathrm{j} \in \mathcal{N}_{\mathrm{i}}, \mathrm{j} \neq \mathrm{i} \\ 1 - \sum_{k \in \mathcal{N}_{\mathrm{i}}, k \neq \mathrm{i}} W_{\mathrm{i,j}}, & \text{if } \mathrm{i} = \mathrm{j} \\ 0, & \text{if } \mathrm{j} \notin \mathcal{N}_{\mathrm{i}} \end{cases},$$

where $\mathrm{e_i}$ is the number of neighbors of agent i, which by our definition of $\mathcal{N}_{\mathrm{i}}$ includes agent i. Here, s_{i} is a surrogate for the gradient of the objective function in (2.7) with respect to w_{i}, and likewise d_{i} is a surrogate for the gradient of the same objective function with respect to ν_{i}. Through these gradient surrogates, each agent attempts to track the actual gradients of the objective function by using only local information and the estimates of its neighbors. As such, the updates of these surrogates use gradients on the locally available function given by (2.8), plus the local average of previous estimates (in the case of s_{i} only). Please refer to [32] for more details on the definition and updates of these gradient surrogates.

Note that computing $b_{\mathrm{i}}(t)$ at each agent i requires $V_\pi(x)$ as shown in (2.9), which is related to $Q_w(x, u)$ by (2.4). Since the expectation $\mathbb{E}_{u(t+1)|\pi}$ in (2.4) cannot be calculated by each agent i without access to $\boldsymbol{\pi}$, we employ a linear function approximation for the state-value function, namely,

$$V_\upsilon(x) = \upsilon^\top \eta(x), \tag{2.10}$$

where $\upsilon \in \mathbb{R}^{q_2}$ and $\eta(x) \in \mathbb{R}^{q_2}$ is a vector of basis functions. Then we need to find proper parameters υ such that $V_\upsilon \to V_\pi$, for which we employ the following updates to improve our estimates of V_υ:

$$\upsilon_{\mathrm{i}}(t+1) = \sum_{\mathrm{j}=1}^{N} W_{\mathrm{i,j}} \upsilon_{\mathrm{i}}(t) - \alpha_3 h_{\mathrm{i}}(C(t), t)$$
$$\kappa_{\mathrm{i}}(t+1) = \kappa_{\mathrm{i}}(t) + \alpha_4 l_{\mathrm{i}}(C(t), D(t), f_{\mathrm{i}}(t), t).$$

Here,

$$C(t) = \eta(x(t))(\eta(t) - \gamma\eta(x(t+1)))^\top$$
$$D(t) = \eta(x(t))\eta(x(t))^\top$$
$$f_{\mathrm{i}}(t) = r_{\mathrm{i}}(x(t), u(t))\eta(x(t)),$$

and κ_{i} is the corresponding dual variable. In addition, we have that h_{i} is the gradient surrogate with respect to υ_{i} and l_{i} is the gradient surrogates with respect to κ_{i}. The definition of gradient surrogates is discussed above.

2.1.2 Actor Training

Now based on the convergence of the critic, we train each agent's actor to converge on the globally optimal control policy. Similar to [31], we will also utilize the policy gradient method for the actor training in this section. A policy best for the whole network will be achieved based on the advantage function

$$A_\pi(t) = Q_\pi(x(t), u(t)) - V_\pi(s),$$

[71], [72], [75]. Though each agent does not know the exact value to this advantage function, we allow each agent to use

$$A_{\mathrm{i}}(t) = Q_{w_{\mathrm{i}}}(x(t), u(t)) - V_{\upsilon_{\mathrm{i}}}(x(t)),$$

which can be looked at as a local estimate to the global advantage function $A_\pi(t)$. Motivated by this we employ the following updates for actor training:

$$\theta_{\mathrm{i}}(t+1) = \Gamma(\theta_{\mathrm{i}}(t) + \beta(t)A_{\mathrm{i}}(t)\psi_{\mathrm{i}}(x(t), u(t))),$$

where Γ is a projection operator and we have:

$$\psi_{\mathrm{i}}(x(t), u(t)) = \frac{\nabla_{\theta_{\mathrm{i}}}(\boldsymbol{\pi}_{\theta_{\mathrm{i}}}(x(t), u(t)))}{\boldsymbol{\pi}_{\theta_{\mathrm{i}}}(x(t), u(t)))},$$

which comes from the gradient of $log(\boldsymbol{\pi}_{\theta_{\mathrm{i}}}(x(t), u(t)))$ with respect to θ_{i}.

To summarize, the proposed distributed update at each agent i is given as follows:

Critic Update:

$$w_{\mathrm{i}}(t+1) = \sum_{\mathrm{j}=1}^{N} W_{\mathrm{i,j}} w_{\mathrm{i}}(t) - \alpha_1 s_{\mathrm{i}}(t) \tag{2.11}$$

$$\nu_{\mathrm{i}}(t+1) = \nu_{\mathrm{i}}(t) + \alpha_2 d_{\mathrm{i}}(t), \tag{2.12}$$

$$\upsilon_{\mathrm{i}}(t+1) = \sum_{\mathrm{j}=1}^{N} W_{\mathrm{i,j}} \upsilon_{\mathrm{i}}(t) - \alpha_3 h_{\mathrm{i}}(t) \tag{2.13}$$

$$\kappa_{\mathrm{i}}(t+1) = \kappa_{\mathrm{i}}(t) + \alpha_4 l_{\mathrm{i}}(t). \tag{2.14}$$

Actor Update:

$$\theta_{\mathrm{i}}(t+1) = \Gamma(\theta_{\mathrm{i}}(t) + \beta(t)A_{\mathrm{i}}(t)\psi_{\mathrm{i}}(x(t), u(t))). \tag{2.15}$$

2.2 Main Result

We will now go over the main result of our paper where we describe what the proposed algorithm can achieve under the following assumptions:

(A1)- The function approximation of the policy , i.e $\boldsymbol{\pi}_\theta$, is greater than 0 for any θ. This is a standard assumption used in [31], [72], [77]

(A2)- $\boldsymbol{\pi}_\theta(x,u)$ is continuously differentiable in θ, as is assumed in [77]

(A3)-The projection operator Γ, which is used in the proposed update, projects any $\theta_{\mathrm{i}}(t)$ onto a compact set. Furthermore, we assume that the compact set Θ is large enough to include a least one local minimum of $V_{\boldsymbol{\pi}_\theta}$.

(A4)-The reward function $r_{\mathrm{i}}(t)$ is uniformly bounded for each agent and for all time. This assumption has been made in works such as [31], [74]

(A5)- The step-sizes $\alpha_1(t)$,$\alpha_3(t)$, $\beta(t)$ satisfy:

$$\sum_{t=1}^{\infty}\alpha_1(t)\to\infty \qquad \sum_{t=1}^{\infty}\alpha_3(t)\to\infty \qquad \sum_{t=1}^{\infty}\beta(t)\to\infty$$
$$\beta(t)=o(\alpha_1(t)) \qquad \alpha_1(t)=o(\alpha_3(t)),\ \text{as } t\to\infty,$$

where $f(t)=o(g(t))$ means for every constant ϵ there exists a constant N such that $|f(t)|\leq \epsilon g(t)$, for all $t\geq N$. In addition, we assume $\sum_t^{\infty}(\alpha_1(t)^2+\alpha_3(t)^2+\beta(t)^2)$ is bounded.

(A6)- Each data sample is selected at least once every T iterations of parameter updates.

(A7)-The matrices $\hat{A}$ and $\hat{C}$ are full rank for large T

(A8)- The sequence of states produced by any policy $\boldsymbol{\pi}$ is a Markov chain that is irreducible and aperiodic.

(A9)- $\boldsymbol{\Phi}$ is full rank , $q_1\leq n$, and $\phi(x(t),u(t))$ is uniformly bounded for all x,u pairs.

Theorem 2.2.1. *We denote $w_{\boldsymbol{\pi}_\theta}=w_\theta$ and $\upsilon_{\boldsymbol{\pi}_\theta}=\upsilon_\theta$, where w_θ and υ_θ are the target parameters such that $|Q_{w_\theta}(x,u)-Q_{\boldsymbol{\pi}_\theta}(x,u)|$ and $|V_{\upsilon_\theta}(x)-V_{\boldsymbol{\pi}_\theta}(x)|$ are minimized for all (x,u) and some parametrized policy $\boldsymbol{\pi}_\theta$. Given assumptions (A1)-(A9) we have the following: For each agent* i *, given a fixed parametrized policy $\boldsymbol{\pi}_\theta$, w_{i} and υ_{i} converge with consensus to parameters w_θ and υ_θ in a linear rate, such that $Q_{w_{\mathrm{i}}}\to Q_{w_\theta}$ and $V_{\upsilon_{\mathrm{i}}}\to V_{\upsilon_\theta}$. Furthermore, given $Q_{w_{\mathrm{i}}}\to Q_{w_\theta}$ and $\epsilon>0$, there exists a $\delta>0$ such that if $\sup_{\theta(t)}\|\mathrm{e}_{\theta(t)}\|<\delta$, then the proposed actor updates on $\theta_{\mathrm{i}}(t)$ converge almost surely to an ϵ neighborhood of a local optimum of $Q_{\boldsymbol{\pi}_\theta}$. Where the local optimum is defined as a θ such that $\nabla_\theta Q_{\boldsymbol{\pi}_\theta}=0$ and furthermore:*

$$\mathrm{e}_{\theta(t)}=\mathbb{E}_x\left[\mathbb{E}_{u|x}\left[((Q_{\boldsymbol{\pi}_\theta}(x,u)-Q_{w_\theta}(x,u))+(V_{\boldsymbol{\pi}_\theta}(x)-V_{\upsilon_\theta}(x)))\psi_{\mathrm{i}}(x,u)\right]\right].$$

Remark: We note that $\mathrm{e}_{\theta(t)}$ expresses the bias due to linear function approximation of Q_{π_θ} and V_{π_θ}. Therefore, as long as this bias is small enough the proposed algorithm can achieve convergence to ϵ neighborhood of the optimal policy, and so we achieve the goal of our paper described by: $\boldsymbol{\pi}_{\theta_\mathrm{i}} \to \boldsymbol{\pi}^* \pm \epsilon$ and $Q_{w_\mathrm{i}} \to Q_{\pi^* \pm \epsilon}$.

In order to prove Theorem 2.2.1, we need the following lemmas, where lemma 1 is used to prove lemma 2.

Lemma 2.2.2. *Given assumption (A5),(A6),(A7), a sufficiently small α_3 with $\alpha_4 = \iota_1 \alpha_3$ where*

$$\iota_1 = 8(\rho + \lambda_{max}(\hat{\mathbf{C}}^\top \hat{\mathbf{D}}^{-1} \hat{\mathbf{C}}))/\lambda_{min}(\hat{\mathbf{D}}),$$

and a policy $\boldsymbol{\pi}_\theta$, then for each agent i *the updates on υ_i from (2.13) converge to network consensus on υ_θ such that $V_{\upsilon_\mathrm{i}} \to V_{\upsilon_\theta}$.*

Proof of Lemma 2.2.2: Using the conditions on $\alpha_3(t)$ relative to $\beta(t)$ from (A5), we apply the two time-scale stochastic approximation method [78] to hold θ fixed in our analysis on the convergence of υ. The result then follows from [32], which proves that the updates from (2.15) converge to network consensus given a fixed policy $\boldsymbol{\pi}_\theta$. ■

From Lemma 1 we have that υ_i converges such that $V_\upsilon \to V_\theta$ and that the network reaches consensus on υ_i, furthermore from (A5) we know that υ_i converges in a faster time-scale than w_i. Therefore, using two-timescale stochastic approximation [78] we have that $V_\upsilon = V_\theta$ for our analysis of updates on w_i. By following the same proof in [32], one then has the following lemma:

Lemma 2.2.3. *If assumptions (A5), (A6), and (A7) hold and the primal step size α_1 is sufficiently small with $\alpha_2 = \iota_2 \alpha_1$ where $\iota_2 = 8(\rho + \lambda_{max}(\hat{\mathbf{A}}))/\lambda_{min}(\hat{\mathbf{A}})$, then for a given policy $\boldsymbol{\pi}_\theta$ the critic algorithm converges to the optimal parameters w_θ, ν_i^*, and $\frac{1}{m}\sum_{\mathrm{i}=1}^m \|w_\mathrm{i}(t) - \bar{w}(t)\|)$ converges to zero, all at a linear rate. More formally we have:*

$$\|\bar{w}(t) - w_\theta\|^2 + \frac{1}{\iota_2 m}\sum_{\mathrm{i}=1}^m \|\nu_\mathrm{i} - \nu_\mathrm{i}^*\|^2 = \mathcal{O}(\sigma^t), \;\; \frac{1}{m}\sum_{\mathrm{i}=1}^m \|w_\mathrm{i}(t) - \bar{w}(t)\| = \mathcal{O}(\sigma^t),$$

where $\bar{w}(t) = \frac{1}{m}\sum_{\mathrm{i}=1}^m w_\mathrm{i}(t)$ and $0 < \sigma < 1$.

While more details of the proof for lemma 2 are provided in [32] and the full details are in the supplementary notes they provide, we will give a short outline of their proof.

Sketch of Proof for Lemma 2.2.3: We now go through an outline of the proof for lemma 2. Overall, the proof focuses on the evolution of the following Lyapunov functions, and on the conditions that guarantee these functions will converge linearly to zero:

$$\|\hat{v}(t)\|^2 = \Theta(\|\bar{w}(t\| - w_\theta)^2 + \frac{1}{\kappa m}\sum_{\mathrm{i}=1}^{m}\|\nu_\mathrm{i} - \nu_\mathrm{i}^*\|^2)$$

$$E_c(t) = \frac{1}{m}\sqrt{\sum_{\mathrm{i}=1}^{m}\|w_\mathrm{i}(t) - \bar{w}(t)\|^2}$$

$$E_g(t) = \frac{1}{m}\sqrt{\sum_{1}^{m}\|s_\mathrm{i}(t) - \frac{1}{mT}\sum_{\mathrm{i}=1}^{m}\sum_{t=1}^{T}\nabla_w J_\mathrm{i}(w,\nu,t)\|^2}$$

Where J_i is the conjugate of the objective function in (2.8) for an agent i, E_c is the consensus error of the primal parameter, and E_g is the consensus error of the primal aggregated gradient.

The proof can then be divided into three parts. In the first part, one iteration of the critic algorithm is analyzed , and in particular the errors due to imperfect tracking of the temporal and spatial gradients are considered. The analysis in the first step then leads to the construction of a Lyapunov vector $\bar{v}(t) = [\|\hat{v}(t)\|, E_c(t), E_g(t)]^\top$, which consists of the Lyapunov functions shown above.

The second part focuses on the evolution of this Lyapunov vector $\bar{v}(t)$ in one iteration of the algorithm, which is refered to as the coupled system. It is then observed that the Lyapunov vector contains delayed terms due the incremental updates employed by the algorithm. The convergence of the delayed and coupled systems are studied, and sufficient conditions for their linear convergence are explored.

The last part of the proof seeks to derive conditions on the step size α_1, so that sufficient conditions for linear convergence are satisfied. ■

Proof of Theorem 2.2.1: For convenience we denote $Q_{\pi_\theta} = Q_\theta$, $w_{\pi_\theta} = w_\theta$, $V_{\pi_\theta} = V_\theta$, $\upsilon_{\pi_\theta} = \upsilon_\theta$, $\psi_\mathrm{i}(x(t), u(t))$as $\psi_\mathrm{i}(t)$, $\mathbb{E}_x[\mathbb{E}_{u|x}[\,\cdot\,]]$ as $\mathbb{E}[\,\cdot\,]$, and for simplicity we denote $\theta(t) = \theta$.

From our problem formulation we have that each agent maintains an estimate of the global optimal policy $\boldsymbol{\pi}_{\mathrm{i}}$, however we also have that each agent only executes a control input u_{i}.This u_{i} is only an element of the global control input estimate $u_{\pi_{\mathrm{i}}}$ sampled from $\boldsymbol{\pi}_{\mathrm{i}}$. We now define an effective global policy $\boldsymbol{\pi}$ such that when u is sampled from $\boldsymbol{\pi}$, we get the actual global control input vector $col\{u_1, u_2, ..., u_m\}$. In addition we define $\boldsymbol{\pi}_\theta$ as the parameterized form of $\boldsymbol{\pi}$, given some parameter vector θ .

Given the above definitions, we begin by writing out the actor update :

$$\theta_{\mathrm{i}}(t+1) = \Gamma(\theta_{\mathrm{i}}(t) + \beta(t)A_{\mathrm{i}}(t)\psi_{\mathrm{i}}(t)).$$

Now let $\mathcal{F}(t) = \sigma(\theta_{\mathrm{i}}(\tau), \tau \leq t)$ be a σ-field (also called σ-algebra). We can then define the following:

$$\xi_1(t+1) = A_{\mathrm{i}}(t)\phi(t) - \mathbb{E}[A_{\mathrm{i}}(t)\psi_{\mathrm{i}}(t) \mid \mathcal{F}(t)]$$
$$\xi_2(t+1) = \mathbb{E}[(A_{\mathrm{i}}(t) - A_\theta(t)\psi_{\mathrm{i}}(t) \mid \mathcal{F}(t)],$$

where $A_\theta(t) = Q_{w_\theta}(x(t), u(t)) - V_{\upsilon_\theta}(x(t))$ is the advantage function after the critic converges to $w_\theta, \upsilon_\theta$ for a given $\boldsymbol{\pi}_\theta$, whereas $A_{\mathrm{i}}(t)$ is a current estimate using the critic of agent i. With this we can rewrite the actor updates as:

$$\theta_{\mathrm{i}}(t+1) = \Gamma(\theta_{\mathrm{i}}(t) + \beta(t)\mathbb{E}[A_\theta(t)\psi_{\mathrm{i}}(t)]\beta(t)\xi_1(t) + \beta(t)\xi_2(t)).$$

From lemma 2 we know that the critic converges, and from our time-step assumptions we know that it converges in a faster time scale than the actor. Therefore, in the actor update time-scale we have that $A_{\mathrm{i}}(t) \rightarrow A_\theta(t)$, and so ξ_2 is in $o(1)$. Furthermore, let $M(t) = \sum_{t=1}^{\infty} \beta(t)\xi_1(t)$, we also note that the sequence $\{M(t)\}$ is a martingale sequence. We also know that from assumption the sequences $\{w_{\mathrm{i}}(t)\}$, $\{\psi_{\mathrm{i}}(t)\}$, and $\{\phi(x(t), u(t))\}$ are bounded,

and so $\{\xi_1(t)\}$ must also be bounded. Using our step-size assumption we then have the following almost surely:

$$\sum_{t=1}^{\infty} \mathbb{E}[\|M(t+1)-M(t)\|^2 \mid \mathcal{F}(t)] = \sum_{t\geq 1}^{\infty} \|\beta(t)\xi_1(t+1)\|^2 < \infty.$$

From the martingale convergence theorem we know that $M(t)$ conveges almost surely, and so we have :

$$\lim_{t\to\infty} \mathbb{P}\left(\sup_{n\geq t} \|\sum_{\tau=t}^{n} \beta(\tau)\xi_1(\tau)\| \geq \epsilon\right) = 0$$

for some $\epsilon > 0$.

We now look at the quantity $\mathbb{E}[A_{\theta_\mathrm{i}}(t)\psi_\mathrm{i}(t)]$, which can be rewritten as the following:

$$\begin{aligned}\mathbb{E}[A_\theta(t)\psi_\mathrm{i}(t)] &= \int_{-\infty}^{\infty} \mu_\theta(x) \int_{-\infty}^{\infty} \boldsymbol{\pi}_\theta(x,u)\psi_\mathrm{i}(t)A_\theta(t)\,du\,dx \\ &= \int_{-\infty}^{\infty} \mu_\theta(x) \int_{-\infty}^{\infty} \boldsymbol{\pi}_\theta(x,u)\psi_\mathrm{i}(x,u)(w_\theta^\top \phi(x,u) - \upsilon_\theta^\top \eta(x))\,du\,dx.\end{aligned}$$

From the above we can show that $\mathbb{E}[A_\theta(t)\psi_\mathrm{i}(t)]$ is continuous in θ_i.It is important to note that θ is the parameter vector of $\boldsymbol{\pi}_\theta$, which when sampled produces the same u_i extracted from each agents $u \sim \boldsymbol{\pi}_{\theta_\mathrm{i}}$. Therefore, as long as the parametrization of $\boldsymbol{\pi}$ can represent any viable(stochastic) policies, then θ can be seen as a continuous function of each agent's θ_i.This observation implies that that if a function is continuous in θ, then it is also continuous in θ_i.

With the above observations of each term in the proposed update equation, the Kushner-Clark lemma [79] tells us that the update converges almost surely to the set of asymptotically stable equilibria of the following ODE:

$$\dot{\theta}_\mathrm{i}(t) = \Gamma(\mathbb{E}[A_\theta \psi_\mathrm{i}(t)]). \tag{2.16}$$

From [71], [72], [75] we know that in order to update the policy towards the optimal policy we must compute $\nabla_\theta V_\theta(x)$, which is the policy gradient, usually expressed as $\nabla J(\theta)$. In [75], [77] we find that:

$$\nabla_\theta V_\theta(x) = \mathbb{E}[(Q_\theta(x,u) - V_\theta(x))\psi_\mathrm{i}(x,u)].$$

We can then rewrite $\mathbb{E}[A_\theta \psi_{\mathrm{i}}(t)]$ similar to [31], [77],in the following way :

$$\mathbb{E}[A_\theta \psi_{\mathrm{i}}(t)] = \nabla_\theta V_\theta(x) + (\mathbb{E}[A_\theta \psi_{\mathrm{i}}(t)] - \mathbb{E}[(Q_\theta(x,u) - V_\theta(x))\psi_{\mathrm{i}}(t)]).$$

By rearranging terms and using the linearity of expectation we then get:

$$\mathbb{E}[A_\theta \psi_{\mathrm{i}}(t)] = \nabla_\theta V_\theta(x) + \mathbb{E}[((Q_{w_\theta}(x,u) - Q_\theta(x,u)) + (V_{v_\theta} - V_\theta(x)))\psi_{\mathrm{i}}(t)],$$

where $\mathbb{E}[(Q_{w_\theta}(x,u) - Q_\theta(x,u)) + (V_{v_\theta} - V_\theta(x))\psi_{\mathrm{i}}(t)]$ expresses the bias due to linear function approximation of V_θ and Q_θ. Therefore, if

$$\sup_{\theta(t)} \|\mathbb{E}[((Q_{w_\theta}(x,u) - Q_\theta(x,u)) + (V_{v_\theta} - V_\theta(x)))\psi_{\mathrm{i}}(t)]\| < \delta$$

for some $\delta > 0$, then (2.16) converges almost surely to an ϵ neighborhood of $\nabla_\theta V_\theta = 0$, which is a local optimum of V_θ and , since $V_\theta(x) = \mathbb{E}_{u\sim\pi_\theta}[Q_\theta(x,u)]$, a local optimum of Q_θ. [77]. ■

2.3 Simulations

As in in [74], we also consider the following nonlinear system:

$$x(t+1) = \varphi|x(t)| + v^\top u + (\sqrt{1-\varphi^2})\varrho(t)$$

where $\varphi = 0.9$, $\varrho(t) \sim \mathcal{N}(0,1)$, and $v \in \mathbb{R}^m$ is selected randomly from $[0,1]^m$.We use $\mathcal{N}(0,1)$ to denote the normal distribution with zero mean and standard deviation of one.

Consider a small network of $m = 4$ agents. Each agent's $\boldsymbol{\pi}_{\mathrm{i}}$ is approximated by a normal distribution $\mathcal{N}(\zeta_{\theta_{\mathrm{i}}}(x), \sigma)$, where $\zeta_{\theta_{\mathrm{i}}}(x) = \theta_{\mathrm{i}}^\top \chi(x)$ and $\sigma = 0.5$. We have that $\chi(x) \in \mathbb{R}^5$ is a vector of Gaussian radial basis functions(RBF) with means randomly selected from $[0,1]$ and a standard deviation of 0.001. Furthermore, each agent observes a reward $r_{\mathrm{i}}(x,u) = k_{0,\mathrm{i}} + k_{1,\mathrm{i}} u_{\mathrm{i}}^2 + k_{2,\mathrm{i}} x^2$, where u_{i} is the scalar control input of agent i. The coefficients $k_{0,\mathrm{i}}, k_{1,\mathrm{i}}, k_{2,\mathrm{i}}$ are selected randomly from the range $[0,1]$ for each agent.

In order to approximate the state-value function $V(x)$, we use a scalar basis function $\eta(x)$ which we implement as a Gaussian radial basis function with mean selected randomly from

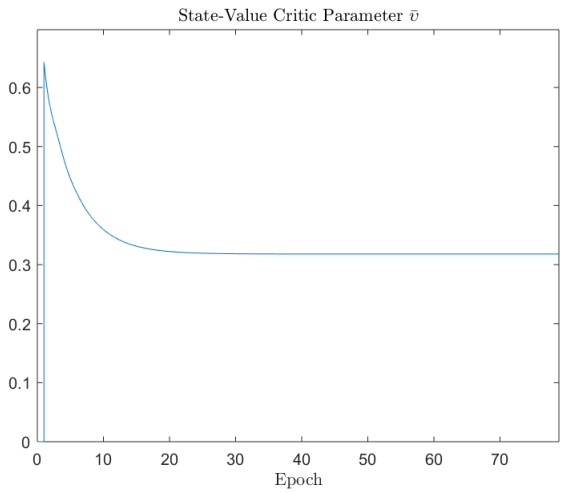

Figure 2.2. Averaged State-Value Parameter for the Critic

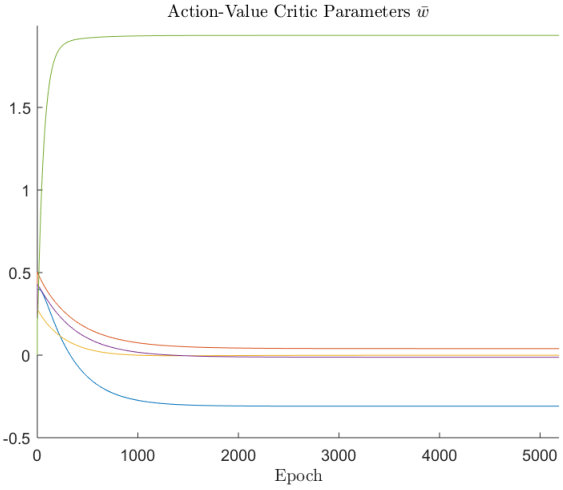

Figure 2.3. Averaged Action-Value Parameters for the Critic

the interval $[-2, 5]$ and standard deviation of 0.1. For the approximation of the action-value function $Q(x, u)$ we use the following structure:

$$Q_{w_{\mathrm{i}}}(x, u) = w_{1,\mathrm{i}} u^{\top} E(x) u + u^{\top} F(x) w_{2:q_1-1,\mathrm{i}} + w_{q_1,\mathrm{i}}$$

where $w_{\mathrm{i}} = col\{w_{1,\mathrm{i}}, w_{2:q_1-1,\mathrm{i}}, w_{q_1,\mathrm{i}}\}$ with $q_1 = 5$. The basis functions $E(x)$ and $F(x)$ are also selected as Gaussian radial basis functions with means randomly selected from $[0, 1]$ and standard deviation of 0.1 for both. The plots of our simulation results, shown in figures

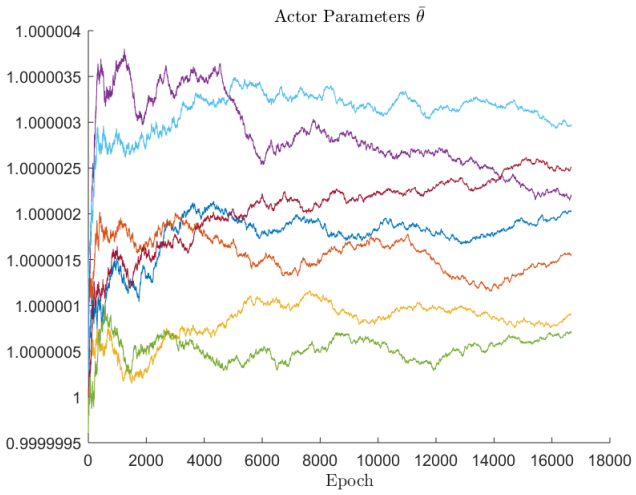

Figure 2.4. Averaged Policy Parameters for the Actor

(2.2, 2.3, 2.4), show the time evolution of the network average of the parameters of interest, namely $\bar{\upsilon}, \bar{w}, \bar{\theta}$. Where $\bar{\upsilon} = \frac{1}{m}\sum_{\mathrm{i}=1}^{m} \upsilon_{\mathrm{i}}$, $\bar{w} = \frac{1}{m}\sum_{\mathrm{i}=1}^{m} w_{\mathrm{i}}$, $\bar{\theta} = \frac{1}{m}\sum_{\mathrm{i}=1}^{m} \theta_{\mathrm{i}}$.We label each x-axis as "epochs". We define an "epoch" as the time step t divided by the number of data samples in memory M, where data samples are the sequences of states and control inputs that have been observed and recorded. For our simulations we used a memory of $M = 1500$ data samples.

From figures (2.2) and (2.3) we can see that the proposed updates on υ_{i} and w_{i} both converge for every agent in the network, and that by design the critic converges much faster for υ_{i} then for w_{i}.

2.4 Conclusion

In this paper we have looked at the problem of distributed multi-agent reinforcement learning where agents only observe their own local rewards. We have presented an actor-critic algorithm that allows agents to use information from their neighbors in order to improve their policies so that the globally averaged reward is maximized. The algorithm has been analyzed based on the two-timescale method used in stochastic approximation problems, and conditions for its convergence have been provided.

3. DISTRIBUTED OFFLINE REINFORCEMENT LEARNING

3.1 Introduction

Reinforcement learning (RL) has recently achieved success in many applications such as autonomous driving, robotics, and so on [1], [3], [4], especially when combined with powerful neural network function approximators. There are, however, still practical challenges when applying reinforcement learning in the real-world. One of these challenges is that reinforcement learning in the classical setting requires continual interaction with the environment in order to learn [75]. In practice, this interaction can be costly and/or dangerous since it might require running experiments with potentially expensive hardware or potentially using a changing control policy, which changes as part of the learning process, and in doing so might explore unsafe actions [80]–[82]. Offline reinforcement learning, where we learn from large, previously collected datasets without further interaction with the environment[80], [82]–[84], offers an alternative to classical reinforcement learning.

Off-policy reinforcement learning, such as Q-learning [7], can in principle enable an agent to learn from large, fixed datasets, but in practice offline reinforcement learning comes with some unique challenges not addressed by off-policy methods [80]. The main challenge is that learning from a fixed, finite dataset can lead to overfitting to the dataset which can result in large extrapolation errors [81], [82].

In the offline reinforcement learning literature, this overfitting issue and the resulting extrapolation errors have been attributed to what is known as distributional shift, which is the difference in data distributions between the data produced by the behaviour policy (the policy used to collected the data) and the data that would have been produced by the learned policy [80], [85]. If distributional shift is not addressed, it has been shown that reinforcement learning algorithms can result in poorly performing policies due to the mentioned issue of overfitting [56], [80], [85], [86]. This motivates us to explore offline reinforcement learning algorithms, so that we could achieve good policies even with fixed, finite datasets.

Recently researchers have also explored RL algorithms for mutli-agent systems due to their applications in formation control [87], [88], coverage [89], social networks [90] and so on. These problems are challenging due to the lack of a centralized coordinator in multi-

agent systems. Despite these challenges, great progress has been achieved for Multi-Agent Reinforcement Learning (MARL) in [31], [74], [91]–[93]. In these algorithms, it is assumed agents operate in the same environment with a shared state representing the environment, and in a lot of these works the state transition depends on the actions of each agent.

In our work we consider a different perspective, where we consider an agent operating in some unknown environment, where the agent has collected a large dataset by interacting with the environment and by choosing actions according to some policy. Given this scenario, we explore how we can utilize a multi-agent network, where agents can only communicate with nearby neighbors, to cooperatively learn a better control policy by splitting up the large dataset amongst the agents. The motivation is that by splitting up the dataset, agents can more efficiently learn from a very large dataset, given that each agent has limited computational power, which is a key idea that has been explored in the distributed machine learning literature [61], [62], [94] and more recently federated learning [95]–[97].

We note that the problem explored in this work is similar to horizontal federated reinforcement learning[98] but with the difference that in our case the data is produced from a single agent and so privacy is not an issue, and unlike in [98], in this work we consider the challenges that come with offline reinforcement learning. We also note the similarity of our work with [81], where they also explore offline reinforcement learning in the multi-agent setting, but in [81] they consider independent agents in the sense that agents are not part of a communication network and so agents cannot share information. As such, in [81] they have not explored how agents can improve their policies by sharing information. In addition, in [81] they propose a model based approach in order for each agent to predict and coordinate with other agents, while our focus is on a model-free approach where agents leverage communication to cooperatively learn. To our knowledge, there are no works that have explored cooperative multi-agent offline reinforcement learning where agents are part of a distributed network and where distributional shift is taken into account.

3.2 Problem Formulation

Consider a scenario where a control system with unknown dynamics operates in an environment. Let $x_t \in \mathcal{X} \subset \mathbb{R}^n$ denote the *state* of the system at time t. Let $a_t \in \mathcal{A}$ denote the *action* of the system at time t, where $\mathcal{A}$ is a finite action space. For each x_t and a_t at time t, let $r(x_t, a_t)$ denote the system's *reward* received from the interaction with the environment, where $r : \mathcal{X} \times \mathcal{A} \to \mathbb{R}$, and correspondingly the state changes to be x_{t+1}. Call $\{x_t, a_t, x_{t+1}, r(x_t, a_t)\}$ a *data point* at time t, which describes the interaction between the system and its operational environment at time t.

Suppose the action a_t follows a control policy $\boldsymbol{\pi} : \mathcal{A} \times \mathcal{X} \to [0, 1]$, i.e. $a_t \sim \boldsymbol{\pi}(\cdot|x_t)$, where $\sim$ means "sampled from a distribution". Let $\boldsymbol{\pi}^*$ denote the *optimal policy* which maximizes long term rewards, i.e.

$$\boldsymbol{\pi}^* = \arg\max_{\boldsymbol{\pi}'} \mathbb{E}\left[\sum_{\tau=0}^{\infty} \gamma^\tau r(x_\tau, a_\tau)\right], \ a_\tau \sim \boldsymbol{\pi}'(\cdot|x_\tau).$$

with $\gamma \in (0, 1)$ is a discount factor. Obviously, learning an optimal policy $\boldsymbol{\pi}^*$ is critical to enable a control system operate in an optimal way in its operational environment. One way to achieve this is by

$$\boldsymbol{\pi}^*(a|x_t) = \begin{cases} 1, \ \text{if } a = \arg\max_u Q_{\boldsymbol{\pi}^*}(x_t, u) \\ 0, \ \text{otherwise.} \end{cases} \tag{3.1}$$

where $Q_{\boldsymbol{\pi}^*}$ is called an *optimal value function*

$$Q_{\boldsymbol{\pi}^*}(x_t, a_t) = \mathbb{E}\left[\sum_{\tau=0}^{\infty} \gamma^\tau r(x_\tau, a_\tau) \middle| x_0 = x_t, a_0 = a_t\right],$$

with $a_\tau \sim \boldsymbol{\pi}^*(\cdot|x_\tau)$. We note that since $\mathcal{A}$ is a finite set, then $\arg\max_u Q_{\boldsymbol{\pi}^*}(x_t, u)$ from (3.1) requires only $|\mathcal{A}|$ calls to $Q_{\boldsymbol{\pi}^*}$. As adopted by the area of reinforcement learning[99], we also assume that $Q_{\boldsymbol{\pi}^*}$ follows a linear combination of known features, i.e.

$$Q_{\boldsymbol{\pi}^*}(x_t, a_t) = \phi(x_t, a_t)^\top \theta^*, \tag{3.2}$$

where $\theta^* \in \mathbb{R}^w$ is a vector of parameters to be determined and $\phi(x_t, a_t) \in \mathbb{R}^w$ is a pre-known vector of linearly independent basis functions. Thus learning an optimal policy can be achieved by estimation of θ^*.

Different from online reinforcement learning, by which we "try-and-correct" estimates of θ^* from the real-time interactions between the system and environment, offline reinforcement learning aims to estimate θ^* based on a fixed set $\mathcal{D}(T)$ consisting of data points for $t = 0, 1, 2, ..., T$. Here, T is a finite positive integer which is usually very large. Other than process the large dataset $\mathcal{D}(T)$ in a single processor, we in this paper are interested in developing a distributed algorithm for offline reinforcement learning in multi-agent systems. We partition $\mathcal{D}(T)$ into a number of m subsets, i.e.

$$\mathcal{D}(T) = \mathcal{D}_1 \cup \mathcal{D}_2 \cup \cdots \cup \mathcal{D}_m$$

where

$$\mathcal{D}_{\mathrm{i}} = \{x_{\mathrm{i},t}, a_{\mathrm{i},t}, x_{\mathrm{i},t+1}, r(x_{\mathrm{i},t}, a_{\mathrm{i},t}) : t \in \mathcal{T}_{\mathrm{i}}\},$$

and $\mathcal{T}_{\mathrm{i}} \subset \{0, 1, 2, ..., T\}$ is the set of time indices associated with $\mathcal{D}_{\mathrm{i}}$. Suppose we are given a network of m agents where each agent i knows $\mathcal{D}_{\mathrm{i}}$, controls a vector $\theta_{\mathrm{i},k}$ as an estimate to θ^* at iteration k and is able to receive information from its nearby neighbors $\mathcal{N}_{\mathrm{i}}$. **The problem of interest** is to develop a rule for each agent i to update $\theta_{\mathrm{i},k}$ only based on information from its nearby neighbors $\mathrm{j} \in \mathcal{N}_{\mathrm{i}}$ such that

$$\lim_{k\to\infty} \sum_{\mathrm{i}=1}^{m} \|\theta_{\mathrm{i},k} - \theta^*\|^2 \leq c(T), \tag{3.3}$$

Remark 3.2.1. *As pointed out in [82], the main challenge in offline reinforcement learning (RL) results from the* distributional shift *, which is the difference between the actual dataset and the data set that would have been observed if the system had followed the optimal policy. More formally, let $d_{\pi'}(x)$ denote a probability density function over $x \in \mathcal{X}$ which describes the probability of a state given an agent takes actions according to a policy $\boldsymbol{\pi}'$. As such, if $\boldsymbol{\pi}'(a|x) \neq \boldsymbol{\pi}^*(a|x)$ for some $x \in \mathcal{X}$ and $a \in \mathcal{A}$, then $d_{\pi'}(x) - d_{\pi^*}(x) \neq 0$, where the difference $|d_{\pi'}(x) - d_{\pi^*}(x)|$ describes the distributional shift induced by a policy $\boldsymbol{\pi}'$. This difference in*

distributions can result in a tendency to overestimate the predicted reward of (state, action) pairs that are not in the dataset. This overestimation is caused by the agent never observing a low or negative reward for these (state, action) pairs since they are not in the dataset. When this overestimation occurs the resulting policy performs poorly when it encounters these (state, action) pairs during test time [80], [82]. Thus in offline reinforcement learning one needs to come up a way to account for distributional shift in order to achieve policies that perform well in (state, action) pairs that are not in the datasets.

3.3 Key Idea

To achieve a nice estimate to θ^* in the literature of reinforcement learning (RL) [75], one usually considers the following temporal difference error

$$\delta_t(\theta) = Q_\theta(x_t, a_t) - r(x_t, a_t) - \gamma \max_{a'} Q_\theta(x_{t+1}, a')$$

since $\mathbb{E}[\frac{1}{2}\delta_t(\theta^*)^2] = 0$. By noting that the term $\max_{a'} Q_\theta(x_{t+1}, a')$ is not differentiable with respect to θ, researchers usually employs $\hat{\delta}_{\mathrm{i},t}(\theta)$ instead as in [36], by replacing $\max_{a'} Q_\theta(x_{t+1}, a'))$ with $\max_{a'} Q_{\theta_{\mathrm{i},k}}(x_{t+1}, a'))$ in $\delta_t(\theta)$, where

$$\hat{\delta}_{\mathrm{i},t}(\theta) = Q_\theta(x_t, a_t) - r(x_t, a_t) - \gamma \max_{a'} Q_{\theta_{\mathrm{i},k}}(x_{t+1}, a'). \tag{3.4}$$

In order to account for distributional shift in offline RL, one introduces the following term as in [82]

$$\beta \mathbb{E}[\log(\sum_{a \in \mathcal{A}} \exp(Q_\theta(x_t, a))) - Q_\theta(x_t, a_t)], \tag{3.5}$$

with $\beta > 0$. This term penalizes overvaluing (state, action) pairs that are not in the training dataset, in the sense that it penalizes an estimate θ that results in

$$\mathbb{E}[\log(\sum_{a \in \mathcal{A}} \exp(Q_\theta(x_t, a)))] > \mathbb{E}[Q_\theta(x_t, a_t)].$$

As such, this term attempts to prevent learning a policy that is biased towards actions that are not in the dataset [82], since we do not know if these actions actually lead to higher rewards.

In the literature of multi-agent reinforcement learning (MARL) [100], for a given estimate $\theta_{\mathrm{i},k}$, one also use the following term

$$\frac{\alpha_k}{2}(\theta-\theta_{\mathrm{i},k})^\top \mathbb{E}[A_t](\theta-\theta_{\mathrm{i},k}), \tag{3.6}$$

with $A_t=\phi(x_t,a_t)\phi(x_t,a_t)^\top \in \mathbb{R}^{w\times w}$ and tunable parameter $\alpha_k>0$ to penalize large changes in the estimate of θ^*. Thus in order for each agent i to achieve a nice estimate to θ^* given $\theta_{\mathrm{i},k}$, one needs to minimize the following $L(\theta)$, which is a sum of terms in (3.4),(3.5) and (3.6):

$$\begin{aligned} L_{\mathrm{i}}(\theta) = \mathbb{E}[\frac{1}{2}\hat{\delta}_{\mathrm{i},t}(\theta)^2] + \frac{\alpha_k}{2}(\theta-\theta_{\mathrm{i},k})^\top \mathbb{E}[A_t](\theta-\theta_{\mathrm{i},k}) \\ + \beta\mathbb{E}[\log(\sum_{a\in\mathcal{A}}\exp(Q_\theta(x_t,a))) - Q_\theta(x_t,a_t)]. \end{aligned} \tag{3.7}$$

This minimization is achieved by setting the gradient equal to zero, which leads to the following system of equations:

$$(\alpha_k+1)\mathbb{E}[A_t]\theta = \mathbb{E}[b_t(\theta_{\mathrm{i},k})] + \beta\mathbb{E}[c_t(\theta)] + \alpha_k\mathbb{E}[A_t]\theta_{\mathrm{i},k}, \tag{3.8}$$

where

$$\begin{aligned} b_t(\theta_{\mathrm{i},k}) &= \phi(x_t,a_t)(r(x_t,a_t) - \gamma\phi(x_{t+1},a)^\top\theta_{\mathrm{i},k}), \\ a &= \arg\max_{a'} \phi(x_{t+1},a')^\top\theta_{\mathrm{i},k}, \\ c_t(\theta) &= \frac{\sum_{a\in\mathcal{A}}\exp(Q_\theta(x_t,a))\phi(x_t,a)}{\sum_{a\in\mathcal{A}}\exp(Q_\theta(x_t,a))} - \phi(x_t,a_t). \end{aligned}$$

In order to achieve further simplification, we use the current estimate $\theta_{\mathrm{i},k}$ to replace θ in the nonlinear term $c_t(\theta)$, i,.e. replace $c_t(\theta)$ by $c_t(\theta_{\mathrm{i},k})$ in (3.8), which leads to

$$(\alpha_k + 1)\mathbb{E}[A_t]\hat{\theta} = \mathbb{E}[\bar{b}_t(\theta_{\mathrm{i},k})] + \alpha_k \mathbb{E}[A_t]\theta_{\mathrm{i},k}, \tag{3.9}$$

where

$$\bar{b}_t(\theta_{\mathrm{i},k}) = b_t(\theta_{\mathrm{i},k}) + \beta c_t(\theta_{\mathrm{i},k}).$$

The above system of equations is linear with respect to $\hat{\theta}$, and so we could solve for $\hat{\theta}$ if we could compute the expectations of A_t and $\bar{b}_t$, which are however not directly available to each agent i. We thus need to introduce approximations of A_t and $\bar{b}_t$ based on only $\mathcal{D}_\mathrm{i}$. For any dataset $\hat{\mathcal{D}} \subset \mathcal{D}_\mathrm{i}$ with $\hat{d} = \left|\hat{\mathcal{D}}\right|$ and its corresponding time index set $\hat{\mathcal{T}} \subset \mathcal{T}_\mathrm{i}$, we let

$$A_\mathrm{i}(\hat{\mathcal{D}}) = \frac{1}{\hat{d}} \sum_{t \in \hat{\mathcal{T}}} A_{\mathrm{i},t}, \bar{b}_\mathrm{i}(\hat{\mathcal{D}}, \theta_{\mathrm{i},k}) = b_\mathrm{i}(\hat{\mathcal{D}}, \theta_{\mathrm{i},k}) + \beta c_\mathrm{i}(\hat{\mathcal{D}}, \theta_{\mathrm{i},k}) \tag{3.10}$$

$$b_\mathrm{i}(\hat{\mathcal{D}}, \theta_{\mathrm{i},k}) = \frac{1}{\hat{d}} \sum_{t \in \hat{\mathcal{T}}} b_{\mathrm{i},t}(\theta_{\mathrm{i},k}),\ c_\mathrm{i}(\hat{\mathcal{D}}, \theta_{\mathrm{i},k}) = \frac{1}{\hat{d}} \sum_{t \in \hat{\mathcal{T}}} c_{\mathrm{i},t}(\theta_{\mathrm{i},k}), \tag{3.11}$$

where

$$A_{\mathrm{i},t} = \phi(x_{\mathrm{i},t}, a_{\mathrm{i},t})\phi(x_{\mathrm{i},t}, a_{\mathrm{i},t})^\top$$

$$\bar{b}_{\mathrm{i},t}(\theta_{\mathrm{i},k}) = b_{\mathrm{i},t}(\theta_{\mathrm{i},k}) + \beta c_{\mathrm{i},t}(\theta_{\mathrm{i},k}) \tag{3.12}$$

$$b_{\mathrm{i},t}(\theta_{\mathrm{i},k}) = \phi(x_{\mathrm{i},t}, a_{\mathrm{i},t})(r(x_{\mathrm{i},t}, a_{\mathrm{i},t}) - \gamma\phi(x_{\mathrm{i},t+1}, u)^\top \theta_{\mathrm{i},k}) \tag{3.13}$$

$$u = \arg\max_{a'} \phi(x_{\mathrm{i},t+1}, a')^\top \theta_{\mathrm{i},k}$$

$$c_{\mathrm{i},t}(\theta) = \frac{\sum_{a \in \mathcal{A}} \exp(Q_\theta(x_{\mathrm{i},t}, a))\phi(x_{\mathrm{i},t}, a)}{\sum_{a \in \mathcal{A}} \exp(Q_\theta(x_{\mathrm{i},t}, a))} - \phi(x_{\mathrm{i},t}, a_{\mathrm{i},t}). \tag{3.14}$$

Then the linear equations in (3.9) with $\mathbb{E}[A_t] \approx A_\mathrm{i}(\hat{\mathcal{D}})$ and $\mathbb{E}[\bar{b}_t(\theta_{\mathrm{i},k})] \approx \bar{b}_\mathrm{i}(\hat{\mathcal{D}}, \theta_{\mathrm{i},k})$ can be approximated by:

$$(\alpha_k + 1)A_\mathrm{i}(\hat{\mathcal{D}})\hat{\theta} = \bar{b}_\mathrm{i}(\hat{\mathcal{D}}, \theta_{\mathrm{i},k}) + \alpha_k A_\mathrm{i}(\hat{\mathcal{D}})\theta_{\mathrm{i},k}. \tag{3.15}$$

For the dataset $\hat{\mathcal{D}} \subset \mathcal{D}_{\mathrm{i}}$, we let $\Phi_{\hat{\mathcal{D}}} \in \mathbb{R}^{w \times \hat{d}}$ be the constant matrix with columns equal to $\phi(x_{\mathrm{i},t}, a_{\mathrm{i},t}), t \in \hat{\mathcal{T}}$. Recall that $\phi \in \mathbb{R}^w$ is a set of linearly independent basis functions. Thus Φ is full rank if $\hat{d} \geq w$. It follows that $A_{\mathrm{i}}(\hat{\mathcal{D}}) = \Phi_{\hat{\mathcal{D}}} \Phi_{\hat{\mathcal{D}}}^{\top}$ is non-singular.

To sum up, for any dataset $\hat{\mathcal{D}} \subset \mathcal{D}_{\mathrm{i}}$ with $\hat{d} = \left|\hat{\mathcal{D}}\right| \geq w$ and its corresponding time index set $\hat{\mathcal{T}} \subset \mathcal{T}_{\mathrm{i}}$, one has

$$\hat{\theta} = \frac{\alpha_k}{\alpha_k + 1} \theta_{\mathrm{i},k} + \frac{1}{\alpha_k + 1} A_{\mathrm{i}}(\hat{\mathcal{D}})^{-1} \bar{b}_{\mathrm{i}}(\hat{\mathcal{D}}, \theta_{\mathrm{i},k}), \tag{3.16}$$

with $A_{\mathrm{i}}(\hat{\mathcal{D}})$ and $\bar{b}_{\mathrm{i}}(\hat{\mathcal{D}}, \theta_{\mathrm{i},k})$ as defined in (3.10).

3.4 Algorithm and Main Result

In this section we present a distributed algorithm to estimate θ^* based on (3.16) and the idea of convex combination.

In order to more efficiently utilize each agent i's data $\mathcal{D}_{\mathrm{i}}$, we create a batch of subsets of $\mathcal{D}_{\mathrm{i}}$ denoted as $\mathcal{D}_{\mathrm{i},1}, \mathcal{D}_{\mathrm{i},2}, ..., \mathcal{D}_{\mathrm{i},N_{\mathrm{i}}(T)}$ such that the union of these subsets is $\mathcal{D}_{\mathrm{i}}$. Correspondingly for each $\mathcal{D}_{\mathrm{i},q}$, one defines

$$A_{\mathrm{i}}(\mathcal{D}_{\mathrm{i},q}), \quad \bar{b}_{\mathrm{i}}(\mathcal{D}_{\mathrm{i},q}, \theta_{\mathrm{i},k}), \quad q = 1, 2, ..., N_{\mathrm{i}}(T)$$

which can be computed as in (3.10) before embarking iterations. At the kth iteration, each agent i first computes

$$\hat{\theta}_{\mathrm{i},k} = \frac{\alpha_k}{\alpha_k + 1} \theta_{\mathrm{i},k} + \frac{1}{\alpha_k + 1} A_{\mathrm{i}}(\mathcal{D}_{\mathrm{i},q(k)})^{-1} \bar{b}_{\mathrm{i}}(\mathcal{D}_{\mathrm{i},q(k)}, \theta_{\mathrm{i},k}), \tag{3.17}$$

with $q(k) = k \mod N_{\mathrm{i}}(T)$ is the remainder of k when divided by $N_{\mathrm{i}}(T)$, then each agent i receives $\hat{\theta}_{\mathrm{j},k}$ from all its neighbors $\mathrm{j} \in \mathcal{N}_{\mathrm{i}}$, and performs the following iterative update

$$\theta_{\mathrm{i},k+1} = \sum_{\mathrm{j} \in \mathcal{N}_{\mathrm{i}}} w_{\mathrm{ij}} \hat{\theta}_{\mathrm{j},k}. \tag{3.18}$$

Here w_{ij} denotes non-negative weights such that $w_{ij} > 0$ if and only if $j \in \mathcal{N}_i$, and

$$\sum_{j=1}^{m} w_{ij} = 1, \quad \sum_{i=1}^{m} w_{ij} = 1.$$

In the following we present our main result that each $\theta_{i,k}$ converges close to the optimal parameter θ^* by deriving an asymptotic upper bound on the sum of norm squared errors for each agent. Before proceeding on, we introduce four assumptions:

Assumption 3.4.1. *We assume*

$$\|\phi(x,a)\| \leq 1$$

for all $x \in \mathcal{X}, a \in \mathcal{A}$.

Since $\mathcal{X}$ is a compact set and $\mathcal{A}$ is a finite set, there is a scalar $\phi_{\max}$ such that $\phi_{\max} > \|\phi(x,a)\|$ and so we can always normalize $\phi(x,a)$ by $\frac{\phi(x,a)}{\phi_{\max}}$.

Assumption 3.4.2. *Let* W *be a matrix such that* w_{ij} *is the* ij *entry of* W *and let* λ_0 *be the smallest eigenvalue of* W. *Let* $\tilde{A}_k = diag\{(A_1(\mathcal{D}_{1,q(k)}), \ldots, (A_m(\mathcal{D}_{m,q(k)})\}$ *and let* λ_1 *denote the smallest eigenvalue of* $\tilde{A}_k$ *for all* k. *We assume that*

$$\lambda_0^2 > \lambda_1^2 \gamma^2.$$

Note that since $\gamma \in (0,1)$ is a design parameter, we can choose a γ such that the needed condition holds.

Assumption 3.4.3. *We assume there exists a non-negative scalar* p *such that*

$$\sup_{a \in \mathcal{A}} |\boldsymbol{\pi}^*(a|x) - \boldsymbol{\pi}(a|x)| \leq p$$

for all $x \in \mathcal{X}$ *and we assume there exists a positive scalar* r_{max} *such that* $|r(x,a)| \leq r_{max}$ *for all* $x \in \mathcal{X}, a \in \mathcal{A}$.

Assumption 3.4.4. *We assume $\boldsymbol{\pi}(a|x) > 0$ for all $x \in \mathcal{X}, a \in \mathcal{A}$ and $d_{\boldsymbol{\pi}}(x) > 0$, for all $x \in \mathcal{X}$, where we recall $d_{\boldsymbol{\pi}}(x)$ is a probability density over all states given a policy $\boldsymbol{\pi}$. This assumption is satisfied by choosing a policy $\boldsymbol{\pi}$ such that an agent following $\boldsymbol{\pi}$ can explore the entire state space $\mathcal{X}$ given enough time.*

Theorem 3.4.5. *Given assumptions (3.4.1-3.4.3), and each agent follows the updates given by (3.17) and (3.18) with $\alpha_k = k$ and $\beta = \frac{2pr_{max}}{1-\gamma}$. We have that for all* i, $\theta_{\mathrm{i},k}$ *converges to a neighborhood of θ^* in the sense that:*

$$\lim_{k\to\infty} \sum_{\mathrm{i}=1}^{m} \|\theta_{\mathrm{i},k} - \theta^*\|^2 \leq c(T). \tag{3.19}$$

where

$$c(T) = c + \frac{\|\bar{h}(T)^{-1}\bar{f}(T) - \tilde{\theta}^*\|^2}{\lambda_0^2 - \lambda_1^2\gamma^2}, \tag{3.20}$$

with $c = \frac{8m\lambda_1^2 p^2 r_{max}^2}{(\lambda_0^2-\lambda_1^2\gamma^2)(1-\gamma)^2}$, $\tilde{\theta}^* = \mathbf{1}_m \otimes \theta^*$,

$$\bar{h}(T) = \frac{1}{\tilde{N}(T)} \sum_{\tau=0}^{\tilde{N}(T)} \tilde{A}_\tau, \;\; \bar{f}(T) = \frac{1}{\tilde{N}(T)} \sum_{\tau=0}^{\tilde{N}(T)} \hat{b}_\tau(\tilde{\theta}^*).$$

Here $\hat{b}_k(\tilde{\theta}^*) = col\{b_1(\mathcal{D}_{1,q(k)}, \theta^*), \ldots, b_m(\mathcal{D}_{m,q(k)}, \theta^*)\}$, $\tilde{N}(T)$ denotes the number of distinct joint batches processed by the proposed algorithm, $\Xi(k) = \{\mathcal{D}_{1,q(k)}, \ldots, \mathcal{D}_{m,q(k)}\}$ denotes the joint batch at iteration k, and $\tilde{A}_k = diag\{A_1(\mathcal{D}_{1,q(k)}, \ldots, A_m(\mathcal{D}_{m,q(k)}\}$. Moreover,

$$\lim_{T\to\infty} c(T) = c. \tag{3.21}$$

Remark 3.4.6. *From (3.21) and (3.20) we can conclude that as the size of the dataset approaches infinity (i.e. $T \to \infty$), the asymptotic error upper bound $c(T)$ decreases from $c + \frac{\|\bar{h}(T)^{-1}\bar{f}(T)-\tilde{\theta}^*\|^2}{\lambda_0^2-\lambda_1^2\gamma^2}$ towards c, where the positive term $\frac{\|\bar{h}(T)^{-1}\bar{f}(T)-\tilde{\theta}^*\|^2}{\lambda_0^2-\lambda_1^2\gamma^2}$ goes to zero.*

3.5 Proof of Main Result

In this section we prove the convergence of our algorithm to a neighborhood of the the optimal solution θ^*. The main idea of our proof is to use the ODE approximation method [100], [101], where we will show that the error dynamics for the proposed algorithm follows an ODE that is globally asymptotically stable. First we present and prove some lemmas that are used in our proof.

Lemma 3.5.1. *Let $\mathcal{Z} = \{\Xi(k) : k = 1 : \tilde{N}\}$ denote the set of all distinct joint batches. Now let g be a continuous function $g : \mathcal{Z} \to \mathbb{R}^{p_1 \times p_2}$, for any $p_1 > 0, p_2 > 0$, and let $\bar{g}(T) = \frac{1}{\tilde{N}(T)} \sum_{\tau=1}^{\tilde{N}(T)} g(\Xi(\tau))$. For any such g, we have the following equation:*

$$\lim_{N\to\infty} \frac{1}{N} \sum_{k=0}^{N} g(\Xi(k)) = \bar{g}(T).$$

Proof: We recall that the number of distinct joint batches from each $\mathcal{D}_\text{i}$ is $\tilde{N}(T)$, and so all these joint batches are processed every $\tilde{N}(T)$ iterations since at each iteration the network of agents jointly processes one joint batch. As such we have that $\Xi(k) = \Xi(k + \tilde{N})$. Let $\lfloor x \rfloor$ denote the "floor" operator which rounds down x to the nearest integer, we have the following:

$$\begin{aligned}
&\lim_{N\to\infty} \frac{1}{N} \sum_{k=0}^{N} g(\Xi(k)) \\
&= \lim_{N\to\infty} \frac{1}{N} \sum_{v=0}^{\lfloor N/\tilde{N}(T) \rfloor} \sum_{\tau=1}^{\tilde{N}(T)} g(\Xi(\tau)) + \frac{1}{N} \sum_{\tau=\lfloor N/\tilde{N}(T) \rfloor + 1}^{N} g(\Xi(\tau)) \\
&= \lim_{N\to\infty} \frac{1}{N} \sum_{v=0}^{\lfloor N/\tilde{N}(T) \rfloor} \bar{g}(T) + 0 = \bar{g}(T),
\end{aligned}$$

where in the last step we make use of the fact that $\frac{1}{N} \sum_{\tau=\lfloor N/\tilde{N}(T) \rfloor + 1}^{N} g(\Xi(\tau))$ is the product of a finite sum and $\frac{1}{N}$, which goes to zero as $N \to \infty$. This concludes our proof.

Lemma 3.5.2. *For some function $V : \mathcal{Z} \to [1, \infty)$ and $s : \mathcal{Z} \to [0, 1]$ and constants $\delta > 0$, $b < \infty$:*

$$\lim_{N\to\infty} \frac{1}{N} \sum_{k=0}^{N} V(\Xi(k+1)) \leq (1-\delta)V(z) + bs(z), \forall\ z \in \mathcal{Z}.$$

Proof: Since each dataset $\mathcal{D}_{\text{i}}$ is finite and therefore $\mathcal{Z}$ is finite, we can compute $V_{max} = \max_z V(z)$ and $V_{min} = \min_z V(z)$. Now let $\delta = \frac{1}{V_{max}}$, $b = V_{max}^2$, and $s(z) = \frac{V_{max} - V_{min} + 1}{V_{max}}$, we then have the following:

$$\begin{aligned}
&(1-\delta)V(z) + bs(z) \\
&\qquad = (1 - \frac{1}{V_{max}})V(z) + V_{max}^2(\frac{V_{max} - V_{min} + 1}{V_{max}}).
\end{aligned}$$

Since by definition $V_{max} \geq V_{min}$, we can lower bound the above with $(1-\frac{1}{V_{max}})V(z)+V_{max}^2(\frac{1}{V_{max}})$, which then gives us:

$$(1-\frac{1}{V_{max}})V(z) + V_{max} \leq (1-\delta)V(z) + bs(z).$$

We note that by definition $V(z) \geq 1$ which implies $V_{max} \geq 1$. It then follows that $(1 - \frac{1}{V_{max}})V(z) \geq 0$, and so we have $V_{max} \leq (1 - \frac{1}{V_{max}})V(z) + V_{max}$. This gives us:

$$V_{max} \leq (1-\delta)V(z) + bs(z). \tag{3.22}$$

Let $\bar{V} = \frac{1}{\tilde{N}(T)} \sum_{\tau=1}^{\tilde{N}(T)} V(\Xi(\tau))$, from lemma 3.5.1 we have that $\lim_{N\to\infty} \frac{1}{N} \sum_{k=0}^{N} V(\Xi(k+1)) = \bar{V}$. We then have that by definition $\bar{V} \leq V_{max}$, and so by using (3.22) we arrive at:

$$\lim_{N\to\infty} \frac{1}{N} \sum_{k=0}^{N} V(\Xi(k+1)) \leq V_{max} \leq (1-\delta)V(z) + bs(z).$$

This concludes our proof.

Proof of Theorem 1

We first look at each agents individual error $\mathrm{e}_{\mathrm{i},k} = \theta_{\mathrm{i},k} - \theta^*$. We subtract θ^* from both sides of (3.18) and unpack using (3.17) in order to get the following equation for the evolution of $\mathrm{e}_{\mathrm{i},k}$:

$$\begin{aligned}\mathrm{e}_{\mathrm{i},k+1} &= \sum_{\mathrm{j}\in\mathcal{N}_\mathrm{i}} w_{\mathrm{ij}}\Big(\frac{\alpha_k}{\alpha_k+1}\theta_{\mathrm{j},k} - \theta^* + \frac{1}{\alpha_k+1}A_\mathrm{j}(\mathcal{D}_{\mathrm{j},q(k)})^{-1}\bar{b}_\mathrm{j}(\mathcal{D}_{\mathrm{j},q(k)},\theta_{\mathrm{j},k})\Big)\\ &= \sum_{\mathrm{j}\in\mathcal{N}_\mathrm{i}} w_{\mathrm{ij}}\Big(\mathrm{e}_{\mathrm{j},k} - \frac{1}{\alpha_k+1}(\theta_{\mathrm{j},k} - A_\mathrm{j}(\mathcal{D}_{\mathrm{j},q(k)})^{-1}\bar{b}_\mathrm{j}(\mathcal{D}_{\mathrm{j},q(k)},\theta_{\mathrm{j},k})\Big). \qquad (3.23)\end{aligned}$$

Now by subtracting the right side of (3.23) by $\frac{1}{\alpha_k+1}(\theta^* - \theta^*)$, we arrive at the following:

$$\mathrm{e}_{\mathrm{i},k+1} = \sum_{\mathrm{j}\in\mathcal{N}_\mathrm{i}} w_{\mathrm{ij}}\Big(\mathrm{e}_{\mathrm{j},k} - \frac{1}{\alpha_k+1}\big(\mathrm{e}_{\mathrm{j},k} - (A_\mathrm{j}(\mathcal{D}_{\mathrm{j},q(k)})^{-1}\bar{b}_\mathrm{j}(\mathcal{D}_{\mathrm{j},q(k)},\theta_{\mathrm{j},k}) - \theta^*)\big)\Big).$$

Now that we have an equation for the error dynamics of each agent, we can represent the error dynamics in compact form by column stacking all of the above equations for $\mathrm{i} = 1,\dots,m$. Let

$$\begin{aligned}\tilde{\mathrm{e}}_k &= \mathrm{col}\{\mathrm{e}_{1,k},\dots,\mathrm{e}_{m,k}\},\ \tilde{\theta}_k = \mathrm{col}\{\theta_{1,k},\dots,\theta_{m,k}\}\\ \tilde{A}_k &= \mathrm{diag}\{(A_1(\mathcal{D}_{1,q(k)}),\dots,(A_m(\mathcal{D}_{m,q(k)})\}\\ \tilde{b}_k(\tilde{\theta}_k) &= \mathrm{col}\{\bar{b}_1(\mathcal{D}_{1,q(k)},\theta_{1,k}),\dots,\bar{b}_m(\mathcal{D}_{m,q(k)},\theta_{m,k})\},\end{aligned}$$

$\tilde{\theta}^* = \mathbf{1}_m \otimes \theta^*$, and let W be a matrix such that w_{ij} is the ij entry of W. Let $\tilde{W} = (W \otimes I_w)$, the following then expresses the error dynamics for the whole network:

$$\tilde{\mathrm{e}}_{k+1} = \tilde{W}\tilde{\mathrm{e}}_k - \frac{1}{\alpha_k+1}\tilde{W}\big(\tilde{\mathrm{e}}_k - (\tilde{A}_k^{-1}\tilde{b}_k(\tilde{\theta}_k) - \tilde{\theta}^*)\big).$$

In order to clean up the analysis, we now denote $\tilde{g}(k) = \tilde{A}_k^{-1}\tilde{b}_k(\tilde{\theta}_k) - \tilde{\theta}^*)$, which allows us to rewrite the above as:

$$\tilde{\mathrm{e}}_{k+1} = \tilde{W}\tilde{\mathrm{e}}_k - \frac{1}{\alpha_k + 1}(\tilde{W}\tilde{\mathrm{e}}_k - \tilde{W}\tilde{g}(k)).$$

We now square both sides of the above equation, which gives us:

$$\begin{aligned}\|\tilde{\mathrm{e}}_{k+1}\|^2 =& \|\tilde{W}\tilde{\mathrm{e}}_k\|^2 - \frac{2}{\alpha_k + 1}(\tilde{W}\tilde{\mathrm{e}}_k)^\top(\tilde{W}\tilde{\mathrm{e}}_k - \tilde{W}\tilde{g}(k)) + \frac{1}{(\alpha_k + 1)^2}\|\tilde{W}\tilde{\mathrm{e}}_k - \tilde{W}\tilde{g}(k)\|^2 \\ \leq& \|\tilde{W}\tilde{\mathrm{e}}_k\|^2 - \frac{1}{\alpha_k + 1}(\|\tilde{W}\tilde{\mathrm{e}}_k\|^2 - \|\tilde{W}\tilde{g}(k)\|^2).\end{aligned}$$

Let λ denote some eigenvalue of W, we recall that W is a doubly stochastic matrix and so we have that $0 < |\lambda| \leq 1$ and that λ must be real. Let λ_0 denote the smallest eigenvalue of W in absolute value, since $\tilde{W} = W \otimes I_w$ and therefore has the same eigenvalues as W, we can upper bound $\|\tilde{\mathrm{e}}_{k+1}\|^2$ with the following:

$$\|\tilde{\mathrm{e}}_{k+1}\|^2 \leq \|\tilde{\mathrm{e}}_k\|^2 - \frac{1}{\alpha_k + 1}(\lambda_0^2\|\tilde{\mathrm{e}}_k\|^2 - \|\tilde{g}(k)\|^2). \tag{3.24}$$

We now look at the term $\|\tilde{g}(k)\|^2$.

$$\begin{aligned}\|\tilde{g}(k)\|^2 &= \|\tilde{A}_k^{-1}\tilde{b}_k(\tilde{\theta}_k) - \tilde{\theta}^*\|^2 \\ &= \|\tilde{A}_k^{-1}\tilde{b}_k(\tilde{\theta}_k) - \tilde{A}_k^{-1}\tilde{b}_k(\tilde{\theta}^*) + \tilde{A}_k^{-1}\tilde{b}_k(\tilde{\theta}^*) - \tilde{\theta}^*\|^2 \\ &\leq \|\tilde{A}_k^{-1}(\tilde{b}_k(\tilde{\theta}_k) - \tilde{b}_k(\tilde{\theta}^*))\|^2 + \|\tilde{A}_k^{-1}\tilde{b}_k(\tilde{\theta}^*) - \tilde{\theta}^*\|^2\end{aligned}$$

Recall that $A_{\mathrm{i}}(\mathcal{D}_{\mathrm{i},q(k)})$ is non singular for all $\mathrm{i} = 1, \ldots, m$, and so there is a $\lambda_1 > 0$ such that $\lambda_1^2 \leq \|\tilde{A}_k\|^2$, and so we can simplify the above as:

$$\|\tilde{g}(k)\|^2 \leq \lambda_1^2\|\tilde{b}_k(\tilde{\theta}_k) - \tilde{b}_k(\tilde{\theta}^*)\|^2 + \|\tilde{A}_k^{-1}\tilde{b}_k(\tilde{\theta}^*) - \tilde{\theta}^*\|^2$$

We now focus on $\tilde{b}_k(\tilde{\theta}_k) - \tilde{b}_k(\tilde{\theta}^*)$ from the above inequality. From assumption (3.4.1) we have $\|\phi(x,a)\| < 1$, and so using equations (3.12)-(3.14), it can be shown that the above inequality is upper bounded by the following:

$$\|\tilde{g}(k)\|^2 \leq \lambda_1^2\gamma^2\|\tilde{\theta}_k - \tilde{\theta}^*\|^2 + \lambda_1^2\beta^2 m + \|\tilde{A}_k^{-1}\tilde{b}_k(\tilde{\theta}^*) - \tilde{\theta}^*\|^2$$

We now look at the $\|\tilde{A}_k^{-1}\tilde{b}_k(\tilde{\theta}^*) - \tilde{\theta}^*\|^2$ term in the above inequality. Using (3.10)-(3.14) and assumption (3.4.1), we can further simplify the above inequality as:

$$\|\tilde{g}(k)\|^2 \leq \lambda_1^2\gamma^2\|\tilde{\theta}_k - \tilde{\theta}^*\|^2 + 2\lambda_1^2\beta^2 m + \|\tilde{A}_k^{-1}\hat{b}_k(\tilde{\theta}^*) - \tilde{\theta}^*\|^2 \tag{3.25}$$

where $\hat{b}_k(\theta^*) = \text{col}\{b_1(\mathcal{D}_{1,q(k)}, \theta^*), \ldots, b_m(\mathcal{D}_{m,q(k)}, \theta^*)\}$.

By combining inequality (3.25) with (3.24) we get the following inequality:

$$\begin{aligned}\|\tilde{\text{e}}_{k+1}\|^2 \leq \|\tilde{\text{e}}_k\|^2 - \frac{1}{\alpha_k + 1}((\lambda_0^2 - \lambda_1^2\gamma^2)\|\tilde{\text{e}}_k\|^2 \\ -2\lambda_1^2\beta^2 m - \|\tilde{A}_k^{-1}\hat{b}_k(\theta^*) - \tilde{\theta}^*\|^2)\end{aligned} \tag{3.26}$$

We recall $\Xi(k) = \{\mathcal{D}_{1,q(k)}, ..., \mathcal{D}_{m,q(k)}\}$, which denotes the data processed by all agents at iteration k. We note that $\tilde{A}_k$ is fully determined by the data in $\Xi(k)$, and so we can define a function h such that $h(\Xi(k)) = \tilde{A}_k$. Similarly, we can define a function f such that $f(\Xi(k)) = \hat{b}_k(\theta^*)$. These pairs of functions (h and f) allows us to then simplify (3.26) as:

$$\begin{aligned}\|\tilde{\text{e}}_{k+1}\|^2 \leq \|\tilde{\text{e}}_k\|^2 - \frac{1}{\alpha_k + 1}((\lambda_0^2 - \lambda_1^2\gamma^2)\|\tilde{\text{e}}_k\|^2 \\ -2\lambda_1^2\beta^2 m - \|h(\Xi(k))^{-1}f(\Xi(k)) - \tilde{\theta}^*\|^2)\end{aligned}$$

We now focus on finding the following limits in order to proceed with using the ODE approximation method: $\lim_{N\to\infty}\frac{1}{N}\sum_{k=0}^{N} f(\Xi(k)), \lim_{N\to\infty}\frac{1}{N}\sum_{k=0}^{N} h(\Xi(k))$.

Let $\bar{f}(T) = \frac{1}{\tilde{N}(T)} \sum_{\tau=1}^{\tilde{N}(T)} f(\Xi(\tau))$, from lemma 3.5.1 it then follows that

$$\lim_{N\to\infty} \frac{1}{N} \sum_{k=1}^{N} f(\Xi(k)) = \bar{f}(T).$$

Let $\bar{h}(T) = \frac{1}{\tilde{N}(T)} \sum_{\tau=1}^{\tilde{N}(T)} h(\Xi(\tau))$, from lemma 3.5.1 we also have that

$$\lim_{N\to\infty} \frac{1}{N} \sum_{k=0}^{N} h(\Xi(k)) = \bar{h}(T).$$

Given $\alpha_k = k$, from the result that the following limit $\lim_{N\to\infty} \frac{1}{N} \sum_{k=0}^{N} f(\Xi(k))$ equals a constant $\bar{f}(T)$ and $\lim_{N\to\infty} \frac{1}{N} \sum_{k=0}^{N} h(\Xi(k))$ equals a constant $\bar{h}(T)$, and using the result of lemma 3.5.2, it follows from Theorem 3.1 in [101] that the upper bound of $\|\tilde{\mathrm{e}}_k\|^2$ converges to the invariant set of the following:

$$\frac{d\mathrm{e}}{dt} = -(\lambda_0^2 - \lambda_1^2\gamma^2)\mathrm{e} + (\lambda_0^2 - \lambda_1^2\gamma^2)c(T),$$

where $c(T) = \frac{1}{(\lambda_0^2 - \lambda_1^2\gamma^2)}(2\lambda_1^2\beta^2 m + \|\bar{h}(T)^{-1}\bar{f}(T) - \tilde{\theta}^*\|^2)$. By noting that $c(T)$ is a constant and that from assumption (3.4.2) we have $\lambda_0^2 > \lambda_1^2\gamma^2$, it can readily be seen that $\frac{d\mathrm{e}}{dt} = -(\lambda_0^2 - \lambda_1^2\gamma^2)\mathrm{e} + c(T)$ is globally asymptotically stable around $c(T)$, and so we can conclude that as $k \to \infty$, $\|\tilde{\mathrm{e}}_k\|^2 \leq c(T)$. We note that $\|\tilde{\mathrm{e}}_k\|^2 = \sum_{\mathrm{i}=1}^{m} \|\mathrm{e}_{\mathrm{i},k}\|^2 = \sum_{\mathrm{i}=1}^{m} \|\theta_{\mathrm{i},k} - \theta^*\|^2$, and so it follows that as $k \to \infty$:

$$\sum_{\mathrm{i}=1}^{m} \|\theta_{\mathrm{i},k} - \theta^*\|^2 \leq c(T).$$

From the above inequality we can conclude that each $\theta_{\mathrm{i},k}$ converges to a neighborhood of θ^*.

Furthermore, we note that as $T \to \infty$ it follows that $\tilde{N}(T) \to \infty$, and so given assumption 3.4.4 and recalling that $\mathcal{X}$ is a compact set, it then follows that in the limit we have the following:

$$\lim_{T\to\infty} \bar{h}(T) = \lim_{T\to\infty} \frac{1}{\tilde{N}(T)} \sum_{\tau=1}^{\tilde{N}(T)} \tilde{A}_\tau = \mathbb{E}[\tilde{A}_k]$$
$$\lim_{T\to\infty} \bar{f}(T) = \lim_{T\to\infty} \frac{1}{\tilde{N}(T)} \sum_{\tau=1}^{\tilde{N}(T)} \hat{b}_\tau(\theta^*) = \mathbb{E}[\hat{b}_k(\theta^*)]$$

We recall $\tilde{A}_k = \text{diag}\{A_1(\mathcal{D}_{1,q(k)}), \ldots, A_m(\mathcal{D}_{m,q(k)})\}$, and therefore:

$$\begin{aligned}\mathbb{E}[\tilde{A}_k] &= \text{diag}\{\mathbb{E}[A_1(\mathcal{D}_{1,q(k)})], \ldots, \mathbb{E}[A_m(\mathcal{D}_{m,q(k)})]\} \\ &= I_m \otimes \mathbb{E}[A_t], \end{aligned} \tag{3.27}$$

where the last step is a consequence of the agents sampling data from the same distribution, and so the expectation for each agent is identical. Similarly, we have

$$\mathbb{E}[\hat{b}_k(\theta^*)] = \mathbf{1}_m \otimes (\mathbb{E}[b_t(\theta^*)]). \tag{3.28}$$

Let $\mathbf{A}_t = I_m \otimes A_t$ and let $\mathbf{b}_t(\theta^*) = \mathbf{1}_m \otimes b_t(\theta^*)$. Using equations (3.27) and (3.28) , we have the following limit for $c(T)$:

$$\lim_{T\to\infty} c(T) = \frac{2\lambda_1^2\beta^2 m + \|\mathbb{E}[\mathbf{A}_t]^{-1}\mathbb{E}[\mathbf{b}_t(\theta^*)] - \tilde{\theta}^*\|^2}{\lambda_0^2 - \lambda_1^2\gamma^2}.$$

We recall $[\frac{1}{2}\delta_t(\theta^*)^2] = 0$, and so it then follows that $\mathbb{E}[\phi(x_t, a_t)\delta_t(\theta^*)] = 0$ which can be rewritten as $\mathbb{E}[A_t\theta^* - b_t(\theta^*)] = 0$ and so it follows that $\theta^* = \mathbb{E}[A_t]^{-1}\mathbb{E}[b_t(\theta^*)]$. Let $c = \frac{8m\lambda_1^2 p^2 r_{\max}^2}{(\lambda_0^2 - \lambda_1^2\gamma^2)(1-\gamma)^2}$, using $\theta^* = \mathbb{E}[A_t]^{-1}\mathbb{E}[b_t(\theta^*)]$ and recalling that $\beta = \frac{2pr_{\max}}{1-\gamma}$ we then have the following:

$$\lim_{T\to\infty} c(T) = c$$

This concludes our proof.

3.6 Simulation Results

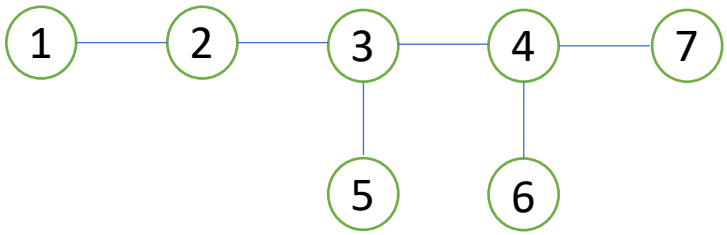

Figure 3.1. Network

In this section we present some simulation results of applying the proposed algorithm. First, we apply the algorithm on a linear time-invariant (LTI) system and show that the algorithm can converge close to the well known LQR solution. Then we apply the algorithm on the cartpole problem, which has been widely used as a benchmark for reinforcement learning algorithms and which has nonlinear state dynamics and a nonlinear reward function. For both cases, we simulate a network of agents that is described by the undirected graph in Fig. 3.1. We used lazy Metropolis weights for w_{ij} such that:

$$w_{\mathrm{ij}} = \begin{cases} \frac{1}{\max\{d_{\mathrm{i}}, d_{\mathrm{j}}\}}, & \text{if } \mathrm{j} \in \mathcal{N}_{\mathrm{i}}, \mathrm{j} \neq \mathrm{i} \\ 1 - \sum_{k \in \mathcal{N}_{\mathrm{i}}, k \neq \mathrm{i}} w_{\mathrm{i}k}, & \text{if } \mathrm{i} = \mathrm{j} \\ 0, & \text{if } \mathrm{j} \notin \mathcal{N}_{\mathrm{i}}, \end{cases}$$

where $d_{\mathrm{i}} = |\mathcal{N}_{\mathrm{i}}|$ is the degree of agent i.

3.6.1 Linear System

We first test the proposed algorithm on an LTI system. Though the state dynamics are simple, by applying the algorithm on a linear system we can compare the final policy to the LQR solution, which is known to be optimal but requires knowledge of the state dynamics to compute.

We simulated an agent controlling an unstable, controllable LTI system. We used the following equation to simulate the dynamics of the system:

$$x_{t+1} = Ax_t + Ba_t,$$

where:

$$A = \begin{bmatrix} 1 & 1 & 3 & 5 \\ 0 & 0.1 & 0 & 1 \\ 0 & 0 & 2 & 5 \\ 1 & 0 & 0 & 1 \end{bmatrix}, \qquad B = \begin{bmatrix} 1 \\ 1 \\ 1 \\ 0 \end{bmatrix},$$

where a_t is the control action at time t. The rewards for this scenario are computed using the following equation: $R(x_t, a_t) = -x_t^\top Q x_t - R a_t^2$, where $Q = I_4, R = 1$. By using $a_t = -Kx_t + n_t,\ n_t \in U(0, 0.7)$ as the control policy for the agent, we created a dataset $\mathcal{D}$, where K was designed to be an unstable state feedback control gain using pole placement. The dataset $\mathcal{D}$ was then partitioned as $\mathcal{D} = \{\mathcal{D}_1, \ldots, \mathcal{D}_7\}$ and distributed amongst a network of 7 agents. The network of agents can be represented by the undirected graph illustrated in Fig. 3.1.

In addition, because of the linearity of the dynamics we can derive a set of basis functions to represent the state-action value function. This set is the set of all quadratic terms using state and action variables, or more precisely:

$$\phi(x, a) = \text{col}\{a^2, ax_1, ax_2, ..., x_3^2, x_3x_4, x_4^2\},$$

where col{} denotes column-wise stacking.

As such, we apply the proposed algorithm to learn the optimal policy for this LTI using a network of 7 agents.

The results of applying the proposed algorithm is a final estimate of θ^* for each agent. Using these estimates of θ^*, we can use the following equation to compute a set of gains for a state feedback controller:

$$K_\theta = \frac{1}{2\theta_1}[\theta_2, \theta_3, \theta_4, \theta_5].$$

The resulting state feedback controller is an approximation of the optimal policy for the LTI. Using the average of each nodes final $\theta_{\mathrm{i},k}$, which we denote $\bar{\theta}$, we arrive at the following gains:

$$K_{\bar{\theta}} = [1.1606, 0.4416, 2.0789, 5.0336].$$

In comparison, by using knowledge of the A and B matrices we can compute the optimal gains using the LQR solution:

$$K_{\mathrm{LQR}} = [1.2336, 0.4744, 2.1480, 5.1455].$$

As we can see, the gains returned by the proposed algorithm are close to the optimal gains, which shows us that the proposed algorithm is capable of finding the optimal policy using suboptimal, offline data.

3.6.2 Nonlinear System

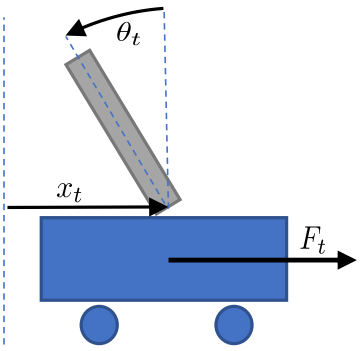

Figure 3.2. Cartpole System

We now apply the proposed algorithm in the scenario where an agent controls a nonlinear system, namely the cartpole system. The cartpole problem consist of a pole on top of cart where the objective is to balance the pole by applying a force on the cart. The state consists of the the position of the cart x_t, the speed of the cart $\dot{x}_t$, the angle of the pole from vertical θ_t, and the rate of change of the angle $\dot{\theta}_t$. There are two possible actions, which are to apply a force $F_t = \{10 \text{ N}, -10 \text{ N}\}$. Each attempt at balancing the pole is referred to as an episode, where an episode terminates if the pole is more than 12 degrees from vertical or if the cart moves more than 2.4 m from the center. For each time step the episode does not terminate the controlling agent is given a reward of $+1$, and when the episode terminates it is given a reward of -5.

In order to collect data, we trained a policy using the well known DQN algorithm [3] and stopped training before the policy could converge on an optimal solution. Using the policy from DQN, we then simulated various episodes using the same suboptimal policy but with different initial conditions and recorded the necessary data.

For this system, we approximate the state-action value function using radial basis functions where a radial basis function has the following form: $\mathrm{e}^{\frac{\|x-c\|^2}{r}}$. The parameters c and r denote the center and radius of the radial basis function, respectively. We used two sets of 250 centers (one for each discrete action). The value for these centers were computed using K-Means clustering on simulated data. This data was generated by using an exploration policy, which with probability ϵ returned the output of a suboptimal policy(trained using DQN algorithm) and with probability $1 - \epsilon$ returned a random action. In addition, for each center we used two different sets of radii, and so in total 2 sets (one for each action) of 500 radial basis functions were used to implement ϕ.

In Fig. 3.3 we see a comparison of the proposed distributed algorithm, which uses a network of agents to cooperatively learn better policies, and compare it to the scenario where a similar algorithm is used but agents do not share any information and so are regarded as independent. In terms of implementation, the differences between the two compared algorithms is that for the independent agents case $w_{\mathrm{ij}} = 1$ if and only if i = j, otherwise $w_{\mathrm{ij}} = 0$, thereby enforcing the absence of communication amongst agents.

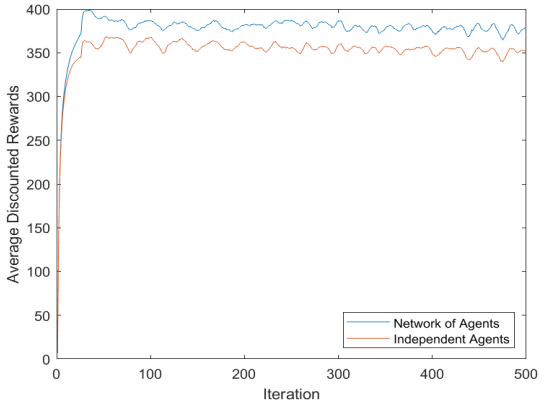

Figure 3.3. Comparing proposed distributed algorithm, which uses network of agents, with the same algorithm using where agents learn independently without sharing any information. Average of Discounted Rewards is calculated over 100 episodes. Each episode has random initial conditions.

More specifically, in Fig. 3.3, we show the average of discounted rewards $\sum_{t=1}^{N} \gamma^t r(x_t, a_t)$ with $\gamma = 0.99999$ achieved by each agent's learned policy over 100 episodes where N denotes the duration of an episode, and where each episode has random initial conditions. In order to account for random initial conditions, we fix the random seed when evaluating all learned policies. As was expected, the plot shows us that for the same number of iterations, using the network of agents allow us to reach a higher average reward when compared to using independent agents that do not share any information. This demonstrates that by sharing estimates the network of agents can learn better policies than agents that learn independently. This difference could potentially be larger for cases where the network is larger and/or the system has a larger state dimension, since this would amplify the advantage of the network over a single independent agent. Moreover, this simulation result shows the benefit of extending offline reinforcement learning to a distributed scheme, as we have done in this work.

3.7 Conclusion

In this paper we have developed a distributed offline reinforcement learning algorithm for a multi-agent system, where agents can communicate over a distributed network. The dis-

tributed offline algorithm combines Q-learning and offline regularization with weighted local averaging so that agents make use of their neighbor's estimates to improve their own. In this way the network of agents can collaborate to improve their estimates while each agent only needs to process a fraction of the total dataset. We have proven that the proposed algorithm converges in the sense that the norm squared error summed over all agents is upper bounded by a constant $c(T)$. From our analysis, we have also shown that $c(T)$ decreases towards a constant c as T goes to infinity. We have supported this theoretical result with simulations that demonstrate that the proposed algorithm can be used to learn control policies for a linear system and a nonlinear system, namely the cartpole system. Some potential future directions would be to consider the case where the dataset $\mathcal{D}(T)$ contains data from different agents, where these agents observed rewards from different reward functions. Similarly, we could explore the case where the dataset was collected by different agents using different policies.

4. DISTRIBUTED REINFORCEMENT LEARNING WITH CROSS MODAL OBSERVATIONS

4.1 Introduction

Recently, multi-agent reinforcement learning (MARL) has gained a lot of popularity with applications into mobile sensor networks, robotics, UAV swarms, and so on .The main challenges in MARL arise from the constraint that agents can sometimes only coordinate with nearby agents and that sometimes agents have different goals and so might observe different rewards. This gives rise to problems such as credit assignment [102], [103] and coordinating actions [28], and partial observability of the state [104], [105]. This has led to the development of distributed algorithms for MARL, in which there is no central node or coordinator and only communication between nearby neighbors is available. Early results in the direction of distributed MARL have usually assumed discrete states and actions to allow them to implement a tabular form of reinforcement learning [30], [33], which are not applicable to situations requiring continuous states and actions. Further progress has been achieved in [32], [39]–[41] for reinforcment learning with continuous states. We note that these works are specifically concerned with the policy evaluation problem of reinforcement learning, where the goal is to learn a metric (known as the value function) that will allows us to compute the long-term expected reward of using a given control policy [75]. We similarly focus on the so called policy evaluation in this work. Policy evaluation is an important step in reinforcement learning since it can be combined with actor-critic methods to develop optimal policies [106], [107].

In our work we consider a network of multi-agent systems that share the same environment but have different and potentially conflicting goals. For example, consider a network of robotic teams where each team has a separate task, and where the performance of each task is affected by the actions of other teams as well. We, in addition, consider the case where each team member can observe a value or metric indicating the team performance but cannot observe its own individual contribution to the team performance. This constraint has been explored in the literature as the credit assignment problem [102], [108]–[110] and is still an open problem. Though the actions of each team could conflict, it is also the case

that each team could achieve a better team performance by cooperating with other teams and so the teams are incentivized to cooperate instead of competing [14], [46]. However, though the teams are cooperative they do not fully trust each other, and so would not want to share information about their performance metric with other teams since this information could be used against them if a team decides to dissent. This motivates the notion of privacy where sensitive information such as performance values cannot be shared with other teams. This notion of privacy has been explored in [32], [111]–[113].

In this work we are also interested in exploring scenarios with cross-modal observations. Cross-modality has recently gained attention in the machine learning literature in the context of transfer learning, where the goal is to use different types of data to learn the same model [114]. In this setting, the literature mainly focuses on how to transform different data types to a common latent space [114]–[116], usually this involves learning functions (different one for each data type) that maps data to a latent vector. In this work we focus on a different aspect of cross-modality where the the transformation to a latent space is known but each agent cannot compute a full latent vector. Moreover, we consider the problem of how to use observations from different sensors where each individual sensor by itself can not fully capture the state of the environment, but the combination of the observations from all sensors can fully observe the state.

This scenerio where the full state of the environment is not observed by any agent has been explored in the reinforcement learning field as a partially observable markov decision process(POMDP) problem [117]. As was previously mentioned, this notion of partial observability has also been studied in the context of MARL, where the state of the environment is said to be partially observed because agents do not know the actions of other agents[118], [119]. In these works, each agent is assumed to only know an observation of the environment instead of the full state [104], [120], and usually based on these observations an estimate of the state is used to carry out the learning process [121]–[123]. Though we also assume each agent recieves an observation and not the full state vector, we differ from traditional POMDP settings in that we further consider that each agent's observation cannot by itself be used to calculate a full estimate of the state. As such we explore a different approach that does not rely on state estimation at the agent level.

The problem that we explore in this work can be decomposed into two levels or two layers. At the high level is the network of multi-agents systems. In this level the problem is driven by the private rewards observed by each multi-agent system and the goal of cooperation to learn the value function for the whole network. At the lower level is a particular multi-agent system. In this level the problem is driven by cross-modal observations where each agent has different, incomplete observation of the environment's state and so must cooperate in order to form a complete observation of the state. We not only need to consider the challenges at each level but the challenges arising from the interaction between the two layers. As such, the approach proposed in this work addresses the challenges in both levels and in how the two interact.

4.2 Problem Formulation

Consider a network of q multi-agent systems controlling a plant with unknown dynamics. We will refer to each multi-agent system as a team. Each team j will consists of an aggregator j and m_{j} agents, where $\sum_{\mathrm{j}=1}^{m} m_{\mathrm{j}}$ is the total number of agents. The aggregator is a node that can communicate with every agent in a team as well as certain other aggregators of other teams. We have that aggregators can communicate if and only if they are aggregator neighbors, where aggregator j's aggregator neighbor is denoted $\mathcal{N}_{\mathrm{j}}$. The aggregator neighbor relations can be modeled by an undirected graph $\mathbb{G}$ such that there is an edge between team j and $\bar{\mathrm{j}}$ if and only if j and $\bar{\mathrm{j}}$ are aggregator neighbors. The aggregator is specialized for communication and not computation, therefore we assume it cannot do any computation. We will label each agent with two indices where the second index indicates team membership, such that ij is the i'th agent of the j'th team. Within each team, agents in the network can communicate if and only if they are agent neighbors, where we denote the set of agent ij's agent neighbors as $\mathcal{N}_{\mathrm{ij}}$. The agent neighbor relations for each team j can be modeled by an undirected graph $\mathbb{G}_{\mathrm{j}}$ such that there is an edge between ij and $\overline{\mathrm{ij}}$ if and only if ij and $\overline{\mathrm{ij}}$ are neighbors.

Let $x_t \in \mathcal{X}$ denote the state of the environment at time t, where $\mathcal{X} \subset \mathbb{R}^n$. Let $a_{\mathrm{ij}}(t) \in \mathcal{A}_{\mathrm{ij}}$ denote the control action of agent ij at time t, where $\mathcal{A}_{\mathrm{ij}} \subset \mathbb{R}^{p_{\mathrm{ij}}}$. Now let $a_{\mathrm{j}}(t)$ denote the

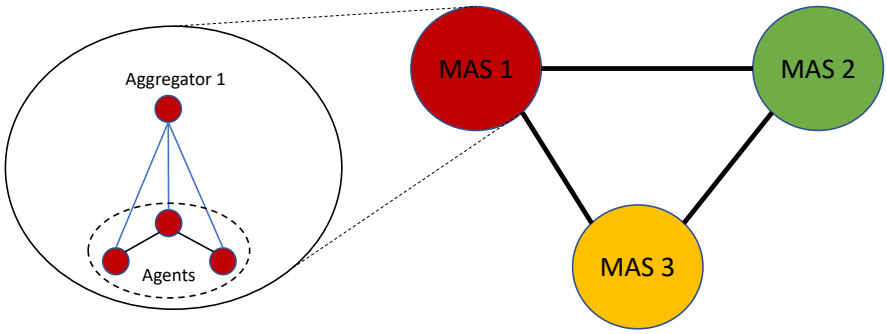

Figure 4.1. Example of network of multi-agent systems

joint control action of team j such that $a_{\mathrm{j}}(t) = \mathrm{col}\{a_{1\mathrm{j}}(t), ..., a_{m_{\mathrm{j}}\mathrm{j}}(t)\}$, and let $a(t)$ denote the joint control action of the entire network such that $a_t = \mathrm{col}\{a_1(t), .., a_q(t)\}$. At every time step t, the network interacts with the environment through the joint control action a_t. As a result, each agent ij receives a team reward $r_{\mathrm{j}}(x_t, a_t)$ which is distinct from the reward of other teams. An example of this interaction is shown in Fig. 4.2, where the details of the Network of Multi-Agent Systems can be seen in Fig. 4.3.

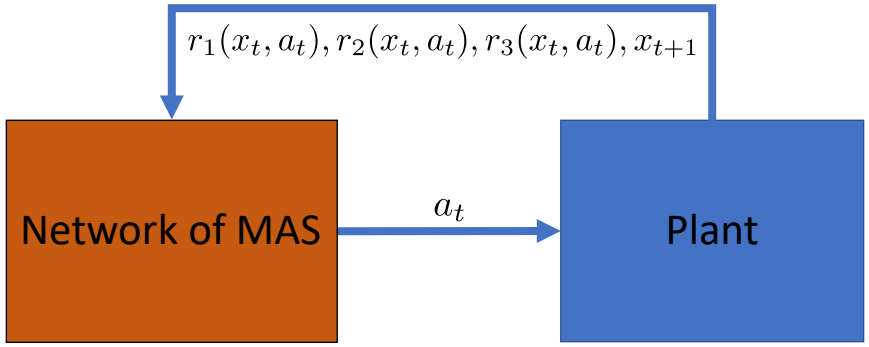

Figure 4.2. Network of Multi-Agent Systems interaction with plant

In this work we consider the case where each agent ij observes a vector $y_{\mathrm{ij}}(t) \in \mathcal{Y}_{\mathrm{ij}}$ where $\mathcal{Y}_{\mathrm{ij}} \subseteq \mathbb{R}^{n_{\mathrm{ij}}}$, and $y_{\mathrm{ij}}(t) = h_{\mathrm{ij}}(x_t)$ for some nonlinear function of the state $h_{\mathrm{ij}} : \mathcal{X} \to \mathcal{Y}_{\mathrm{ij}}$. We note that the observations for each agent may be different which takes into account cross-modal observations, where each agent could use different sensors to observe the state of the environment.

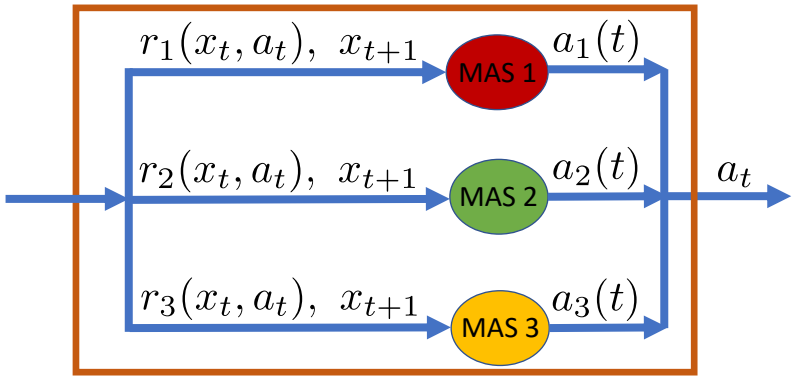

Figure 4.3. Low level picture of Network of Multi-Agent Systems

In addition, we have that the team reward can not be shared with team members in other teams. This constraint is motivated by team privacy concerns, where knowledge of team j's reward could potentially be used by other teams to gain an advantage over team j. We now describe how each agent ij computes its action $a_{ij}(t)$ at time t. We have that each agent ij follows a control policy $\boldsymbol{\pi}_{ij}(\cdot|\cdot)$, where $\boldsymbol{\pi}_{ij}$ is a probability density over $\mathcal{A}_{ij} \times \mathcal{Y}_{ij}$ such that $a_{ij}(t) \sim \boldsymbol{\pi}_j(\cdot|y_{jt})$. Recall $y_{ij}(t) = h_{ij}(x_t)$, let $\boldsymbol{\pi}(\cdot|\cdot)$ be global policy such that $\boldsymbol{\pi}(a_t|x_t) = \prod_{j=1}^{q} \prod_{i=1}^{m_j} \boldsymbol{\pi}_{ij}(a_{ij}(t)|h_{ij}(x_t))$, which describes how the entire network of teams computes a joint action, or more formally we have that $a_t \sim \boldsymbol{\pi}(a|x_t)$. We note no team or agent knows $\boldsymbol{\pi}$ since it describes global behavior resulting from each team following their own policy.

Let $R(x_t, a_t)$ be the average of team rewards , which we define as:

$$R(x_t, a_t) = \frac{1}{q} \sum_{j=1}^{q} r_j(x_t, a_t).$$

Now let $V_{\boldsymbol{\pi}}$ denote a metric for evaluating the policy $\boldsymbol{\pi}$, which is referred to in the literature [75] as the state-value function. $V_{\boldsymbol{\pi}}$, which is defined as:

$$V_{\boldsymbol{\pi}}(x_t) = \mathbb{E}\left[\sum_{\tau=0}^{\infty} \gamma^{\tau} R(x_\tau,\ a_\tau)|x_0 = x_t\right],\ \gamma \in (0, 1)$$

where γ is a discount factor. In this work we are interested in the policy evaluation problem of reinforcement learning, which entails learning the value function $V_{\boldsymbol{\pi}}$. This is an

important problem in reinforcement learning because $V_{\boldsymbol{\pi}}$ can be used to improve the control policy $\boldsymbol{\pi}$ using actor-critic methods.

We consider the case where $V_{\boldsymbol{\pi}}$ is equal to a linear combination of basis functions, or more formally suppose there exists a function $\phi : \mathcal{X} \to \mathbb{R}^w$ such that we have the following:

$$V_{\boldsymbol{\pi}}(x_t) = \phi(x_t)^\top \theta.$$

We have that θ is unknown to all agents, and each agent ij knows ϕ but without knowledge of x_t can only compute certain entries of $\phi(x_t)$. Let $\mathcal{W}_{\mathrm{ij}} \subset \{1, \ldots, w\}$, more formally we have that each agent ij can compute the v'th entry of $\phi(x_t)$ if and only if $v \in \mathcal{W}_{\mathrm{ij}}$. Furthermore, suppose each team j can compute $\phi(x_t)$ such that $|\mathcal{W}_{\mathrm{ij}}| \geq 1$, $\left|\bigcup_{\mathrm{i}=1}^{m_{\mathrm{j}}} \mathcal{W}_{\mathrm{ij}}\right| = w$, and $\left|\bigcup_{\mathrm{j}=1}^{q} \bigcap_{\mathrm{i}=1}^{m_{\mathrm{j}}} \mathcal{W}_{\mathrm{ij}}\right| = w$. As such, each team needs to learn θ in order to compute $V_{\boldsymbol{\pi}}(x_t) \forall x_t \in \mathcal{X}$.

Let each agent ij control $\theta_{\mathrm{ij}}(k) \in \mathbb{R}^w$, and let $\theta_{\mathrm{j}}(k) = \frac{1}{m_{\mathrm{j}}} \sum_{\mathrm{i}=1}^{m_{\mathrm{j}}} \theta_{\mathrm{ij}}(k)$. The goal of the network is for each $\theta_{\mathrm{ij}}(k)$ to converge to a constant vector θ_{ij}^* as $k \to \infty$ such that $\| \sum_{\mathrm{j}=1}^{q} \theta_{\mathrm{j}}(k) - \theta \|^2 \to \epsilon$ which would allow the network to jointly approximate $V_{\boldsymbol{\pi}}$.

4.3 Preliminaries

In [124] it is shown that the solution to the fully observable problem of policy evaluation (θ) is the solution to the following linear equation:

$$\mathbb{E}[A_t]\theta = \mathbb{E}[\bar{b}_t],$$

where $A_t = \phi(x_t)(\phi(x_t) - \gamma\phi(x_{t+1}))^\top \in \mathbb{R}^{w,w}$ and $\bar{b}_t = \frac{1}{m} \sum_{\mathrm{i}=1}^{m} b_{\mathrm{i}t} = \phi(x_t) R(x_t, a_t)$. With enough samples, i.e. T is very large, the expectations can be approximated using the empirical mean as in [32], giving us:

$$\mathbb{E}[A_t] \approx \frac{1}{T} \sum_{t=1}^{T} A_t, \ \mathbb{E}[\bar{b}_t] \approx \frac{1}{T} \sum_{t=1}^{T} \bar{b}_t,$$

and so the linear equations can be approximated by:

$$\mathbf{A}\hat{\theta} = \mathbf{b}, \tag{4.1}$$

where $\mathbf{A} = \frac{1}{T}\sum_{t=1}^{T} A_t$ and $\mathbf{b} = \frac{1}{T}\sum_{t=1}^{T} \bar{b}_t$. Since each agent ij cannot observe x_t but instead observes $y_{\mathrm{ij}}(t)$ and only has access to it's team reward r_{j}, then each agent can only construct part of $\mathbf{A}$ and $\mathbf{b}$. These limitations motivate a distributed approach to solving the system of linear equations in (4.1).

4.4 Approach

Let $\mathbf{A}_{rc}$ denote the entry at row r and column c of $\mathbf{A}$, let ϕ_{rt} denote the r'th entry of $\phi(x_t)$, and let $\mathbf{b}_{r\mathrm{j}} = \frac{1}{Tq}\sum_{t=1}^{T} \phi_{rt} r_{\mathrm{j}}(x_t, a_t)$. We note that $\mathbf{b}_r = \sum_{\mathrm{j}=1}^{q} \mathbf{b}_{r\mathrm{j}}$ where $\mathbf{b}_r$ denotes the entry at row r of $\mathbf{b}$. We also note that the values $\phi_{r,t}$,$\phi_{c,t}$, and r_{j} are needed in order to compute A_{rc} and $b_{r\mathrm{j}}$. As such, in order for the network to cooperatively solve (4.1) we require each team j to be assigned a set of column indices $\mathcal{C}_{\mathrm{j}}$, such that for every ij in team j and every $c \in \mathcal{C}_{\mathrm{j}}$ we have $c \in \mathcal{W}_{\mathrm{ij}}$, and such that every column index is assigned to one team. Furthermore, we require each agent ij is assigned a set of row indices $\mathcal{R}_{\mathrm{ij}}$ such that $\mathcal{R}_{\mathrm{ij}} \subseteq \mathcal{W}_{\mathrm{ij}}$, $\left|\bigcup_{\mathrm{i}=1}^{m_{\mathrm{j}}} \mathcal{R}_{\mathrm{ij}}\right| = w$, and $\left|\bigcap_{\mathrm{i}=1}^{m_{\mathrm{j}}} \mathcal{R}_{\mathrm{ij}}\right| = 0$. We will now describe how all columns and rows will be assigned.

We first describe how each team is assigned columns of $\mathbf{A}$ to meet this condition. Let $\mathcal{H}_{\mathrm{j}} = \bigcap_{\mathrm{i}=1}^{m_{\mathrm{j}}} \mathcal{W}_{\mathrm{ij}}$ denote the set of indices indicating which entries of $\phi(x_t)$ are computable by all agents in team j. Based on each team's $\mathcal{H}_{\mathrm{j}}$, all teams must come to an arrangement where each team j is assigned at least one column c such that $c \in \mathcal{H}_{\mathrm{j}}$ and such that each column is assigned to one team. For the case $q = w$, this assignment process can be formulated as a maximum matching problem for some bipartite graph $\mathbb{G}_B$. Let $\mathbb{G}_B = (U, V, E)$ where each team j is a node in U, each column c is a node in V, and where edge $(\mathrm{j}, c) \in E$ if and only if $c \in \mathcal{H}_{\mathrm{j}}$. An example of such a bipartite graph is illustrated by Fig. 4.4. As such, one possible assignment is given by a maximum matching of $\mathbb{G}_B$, which can be obtained efficiently using the Hopcroft-Karp algorithm [125]. For the case $w > q$, we would need an iterative procedure. We first input $\mathbb{G}_B$ to the Hopcroft-Karp algorithm, which gives us a

maximum matching M_1. M_1 is then used to assign columns to teams. Every c that has been assigned a team in M_1 is then removed from $\mathbb{G}_B$. This new graph is used as input to the Hopcroft-Karp algorithm which gives us a new matching M_2. This process continues until all columns have been assigned. We denote $\mathcal{C}_{\text{j}}$ the set of column indices assigned to team j, which results from using the process outlined above. We note that for column assignment, we assume the assignment process is done by a selected team that knows the $\mathcal{H}_{\text{j}}$ of each team j beforehand. After that chosen team has computed the column assignment, it then sends the assignment to its neighbors through aggregator communication, and its neighbors sends the assignment to their neighbors, thereby propagating the assignment throughout the whole network.

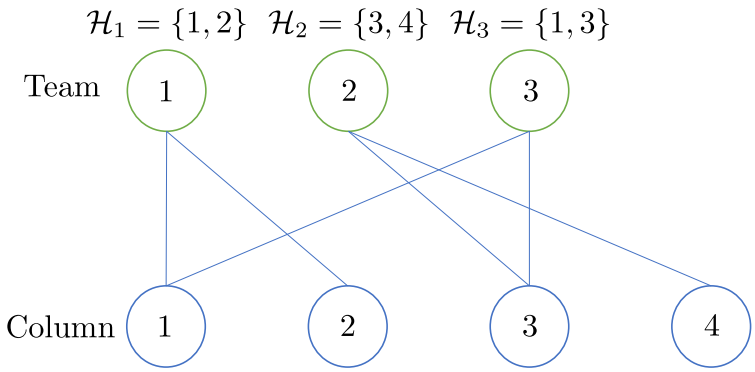

Figure 4.4. Example of Bipartite Graph $\mathbb{G}_B$ for Column Assignment

Now we describe how each agent ij is assigned a row index r. For every team j, agents must come to an agreement where each ij is assigned at least one row index r such that $r \in \mathcal{W}_{\text{ij}}$ and such that each row is assigned to one agent. Similar to the process of assigning columns, for the case $m_{\text{j}} = w$ we can formulate the assignment of rows as a maximum matching problem for a bipartite graph $\mathbb{G}_{B_{\text{j}}}$. For each team j, let $\mathbb{G}_{B_{\text{j}}} = (U, V, E)$ where each agent ij is a node in U, each row r is a node in V, and where edge $(\text{ij}, r) \in E$ if and only if $r \in \mathcal{W}_{\text{ij}}$. Again we can use the Hopcroft-Karp algorithm to compute a maximum matching. For the case $w > m_{\text{j}}$, we again use an iterative application of the Hopcroft-Karp algorithm (see previous paragraph for more details) where after each iteration the assigned rows are removed from the graph $\mathbb{G}_{B_{\text{j}}}$. We denote the resulting set of row indices assigned to agent ij as $\mathcal{R}_{\text{ij}}$. The row assignment process for team j can be computed by any agent ij. Once the

assignment has been computed, agent ij then communicates the assignment to all agents in team j through the aggreagator j.

Distributed Update

We now present a set of update equations that each agent will follow in order to improve their estimate of part of θ. The distributed update is based on work on solving a system of linear equations with scalar states [126]. The update equations will require that we use networks $\mathbb{G}$ and $\mathbb{G}_{\mathrm{j}}$ and it introduces additional states $x_{rc}(k) \in \mathbb{R}$ and $z_{rc}(k) \in \mathbb{R}$ for all $r \in \{1, \ldots, w\}$ and $c \in \{1, \ldots, w\}$, where each agent will control one or more x_{rc} and z_{rc}. Let Z_{j} denote the set of all z_{rc} that are updated by the agents of team j, or more formally we have $Z_{\mathrm{j}} = \{z_{rc} : r \in \{1, \ldots, w\}, c \in \mathcal{C}_{\mathrm{j}}$. We suppose each aggregator j can receive $Z_{\bar{\mathrm{j}}}(k)$ for all $\bar{\mathrm{j}} \in \mathcal{N}_{\mathrm{j}}$ using aggregator communication, and then aggregator j can distribute $\{z_{rv} : r \in \mathcal{R}_{\mathrm{ij}}, v \in \mathcal{C}_{\bar{\mathrm{j}}}\} \subset Z_{\bar{\mathrm{j}}}$ to each agent ij. Furthermore, within each team j, agent ij can recieve $\{x_{vc} : v \in \mathcal{R}_{\bar{\mathrm{ij}}}, c \in \mathcal{C}_{\mathrm{j}}\}$ for all $\bar{\mathrm{ij}} \in \mathcal{N}_{\mathrm{ij}}$.

As such, each agent ij computes the following updates for all $r \in \mathcal{R}_{\mathrm{ij}}, c \in \mathcal{C}_{\mathrm{j}}$:

$$\dot{x}_{rc} = -\mathbf{A}_{rc}(\mathbf{A}_{rc}x_{rc} - \frac{1}{|\mathcal{C}_{\mathrm{j}}|}\mathbf{b}_{r\mathrm{j}} - \sum_{k \in \mathcal{N}_{\mathrm{j}}} \sum_{v \in \mathcal{C}_k} (z_{rc} - z_{rv})) - \sum_{k\mathrm{j} \in \mathcal{N}_{\mathrm{ij}}} \sum_{v \in \mathcal{R}_{k\mathrm{j}}} (x_{rc} - x_{vc}) \tag{4.2}$$

$$\dot{z}_{rc} = \mathbf{A}_{rc}x_{rc} - \frac{1}{|\mathcal{C}_{\mathrm{j}}|}\mathbf{b}_{r\mathrm{j}} - \sum_{k \in \mathcal{N}_{\mathrm{j}}} \sum_{v \in \mathcal{C}_k} (z_{rc} - z_{rv})). \tag{4.3}$$

Let $\theta_{\mathrm{ij},c}$ denote the c'th entry of θ_{ij}. Given initial estimate $x_{rc}(0)$ for all $c = 1, ..., w$ and $r = 1, ..., w$, each agent ij has an initial estimate $\theta_{\mathrm{ij}}(0)$ with the following form:

$$\theta_{\mathrm{ij},c}(0) = \begin{cases} \frac{1}{|\mathcal{R}_{\mathrm{ij}}|} \sum_{r \in \mathcal{R}_{\mathrm{ij}}} x_{rc}(0), \text{ if } c \in \mathcal{C}_{\mathrm{j}} \\ 0, \text{ otherwise.} \end{cases} \tag{4.4}$$

As such, each agent ij updates their θ_{ij} according to:

$$\dot{\theta}_{\mathrm{ij},c} = \begin{cases} \frac{1}{|\mathcal{R}_{\mathrm{ij}}|} \sum_{r \in \mathcal{R}_{\mathrm{ij}}} \dot{x}_{rc}, & \text{if } c \in \mathcal{C}_{\mathrm{j}} \\ 0, & \text{otherwise.} \end{cases} \tag{4.5}$$

4.5 Main Result

Assumption 4.5.1. *Let $\mathcal{A} = \prod_{\mathrm{j}=1}^{q} \prod_{\mathrm{i}=1}^{m_{\mathrm{j}}} \mathcal{A}_{\mathrm{ij}}$. We assume $\boldsymbol{\pi}(a|x) > 0$ for all $x \in \mathcal{X}, a \in \mathcal{A}$ and $d_{\boldsymbol{\pi}}(x) > 0$, for all $x \in \mathcal{X}$, where we recall $d_{\boldsymbol{\pi}}(x)$ is a probability density over all states given a policy $\boldsymbol{\pi}$. This assumption is satisfied if $\boldsymbol{\pi}$ is such that an agent following $\boldsymbol{\pi}$ can explore the entire state space $\mathcal{X}$ given enough time, which is a necessary and common assumption in the RL literature [7], [75].*

Theorem 4.5.2. *Suppose that assumption (4.5.1) is met, $\mathbb{G}$ is connected and $\mathbb{G}_{\mathrm{j}}$ is connected for every team* j. *Then using the distributed updates in (4.2)-(4.5), for each agent* ij, θ_{ij} *converges exponentially to θ^*_{ij}, such that as $k \to \infty$:*

$$\| \sum_{\mathrm{j}=1}^{q} \theta_{\mathrm{j}}(k) - \theta \|^2 \to \epsilon(T),$$

where $\epsilon(T) = \|\mathbf{A}(T)^{-1}\mathbf{b}(T) - \theta\|^2$ and

$$\lim_{T \to \infty} \epsilon(T) = 0.$$

Proof

To prove our main result we will show that (4.2) and (4.3) can be reduced to the updates in [126].

Let $\mathrm{j}(c) = \{\mathrm{j} : c \in \mathcal{C}_{\mathrm{j}}\}$. Since each column c is assigned one team j, then $|\mathrm{j}(c)| = 1$ for all $c = 1, ..., w$. As such, we can define $\hat{r}_c = \frac{1}{|\mathcal{C}_{\mathrm{j}(c)}|} r_{\mathrm{j}(c)}$ for all $c = 1, ..., w$. Recall $\mathbf{b}_{r\mathrm{j}} = \frac{1}{Tq} \sum_{t=1}^{T} \phi_{rt} r_{\mathrm{j}}(x_t, a_t)$, using $\hat{r}_c$, we can then define $\hat{b}_{rc} = \frac{1}{Tq} \sum_{t=1}^{T} \phi_{rt} \hat{r}_c(x_t, a_t)$ for all $c = 1, ..., w$. We now note that for each agent ij, for all $c \in \mathcal{C}_{\mathrm{j}}$ and all $r \in \mathcal{R}_{\mathrm{ij}}$ we have

$\hat{b}_{rc} = \frac{1}{|\mathcal{C}_{\mathrm{j}(c)}|}\mathbf{b}_{r\mathrm{j}(c)} = \frac{1}{|\mathcal{C}_{\mathrm{j}}|}\mathbf{b}_{r\mathrm{j}}$, since by definition $\mathrm{j}(c) = \mathrm{j} \forall c \in \mathcal{C}_{\mathrm{j}}$, and so we can rewrite (4.2) and (4.3) as:

$$\begin{aligned}
\dot{x}_{rc} =& -\mathbf{A}_{rc}(\mathbf{A}_{rc}x_{rc} - \hat{b}_{rc} - \sum_{k\in\mathcal{N}_{\mathrm{j}}}\sum_{v\in\mathcal{C}_k}(z_{rc} - z_{rv})) - \sum_{k\mathrm{j}\in\mathcal{N}_{\mathrm{ij}}}\sum_{v\in\mathcal{R}_{k\mathrm{j}}}(x_{rc} - x_{vc}) \\
\dot{z}_{rc} =& \mathbf{A}_{rc}x_{rc} - \hat{b}_{rc} - \sum_{k\in\mathcal{N}_{\mathrm{j}}}\sum_{v\in\mathcal{C}_k}(z_{rc} - z_{rv})).
\end{aligned}$$

Now let $\mathcal{N}_{rc}^R = \{rv : v \in \mathcal{C}_k \forall k \in \mathcal{N}_{\mathrm{j}}\}$ and let $\mathcal{N}_{rc}^C = \{vc : v \in \mathcal{R}_{k\mathrm{j}} \forall k\mathrm{j} \in \mathcal{N}_{\mathrm{ij}}\}$. We can then rewrite the above two equations as:

$$\begin{aligned}
\dot{x}_{rc} =& -\mathbf{A}_{rc}(\mathbf{A}_{rc}x_{rc} - \hat{b}_{rc} - \sum_{rv\in\mathcal{N}_{rc}^R}(z_{rc} - z_{rv})) - \sum_{vc\in\mathcal{N}_{rc}^C}(x_{rc} - x_{vc}) \\
\dot{z}_{rc} =& \mathbf{A}_{rc}x_{rc} - \hat{b}_{rc} - \sum_{rv\in\mathcal{N}_{rc}^R}(z_{rc} - z_{rv})).
\end{aligned}$$

We now show that $\sum_{c=1}^{w}\hat{b}_{rc} = \mathbf{b}_r$ through the following:

$$\begin{aligned}
\sum_{c=1}^{w}\hat{b}_{rc} =& \sum_{c=1}^{w}\frac{1}{|\mathcal{C}_{\mathrm{j}}|}\mathbf{b}_{r\mathrm{j}} \\
=& \sum_{\mathrm{j}=1}^{q}\sum_{c\in\mathcal{C}_{\mathrm{j}}}\frac{1}{|\mathcal{C}_{\mathrm{j}}|}\mathbf{b}_{r\mathrm{j}} \\
=& \sum_{\mathrm{j}=1}^{q}\mathbf{b}_{r\mathrm{j}} \\
=& \mathbf{b}_r
\end{aligned}$$

Since $\sum_{c=1}^{w}\hat{b}_{rc} = \mathbf{b}_r$ and by recalling that each $\mathbf{A}_{rc}$ is a scalar entry of matrix $\mathbf{A}$, it then follows that (4.2) and (4.3) are equivalent to equations (5) and (6) in [126].

Given the conditions stated in our theorem are met, it then follows from the proof of Theorem 1 in [126] that each x_{rc} converges exponentially to a constant vector x_{rc}^* such that

for each $r = 1, ..., w$, $\sum_{c=1}^{w}(\mathbf{A}_{rc}x^*_{rc} - \hat{b}_{rc}) = 0$ and $x^*_{1,c} = x^*_{2,c} = ... = x^*_{w,c}$. Since each θ_{ij} is updated according to (4.5), it then follows that each θ_{ij} converges to a θ^*_{ij} such that:

$$\dot{\theta}^*_{\text{ij},c} = \begin{cases} \frac{1}{|\mathcal{R}_{\text{ij}}|}\sum_{r\in\mathcal{R}_{\text{ij}}} x^*_{rc}, \text{ if } c \in \mathcal{C}_{\text{j}} \\ 0, \text{ otherwise.} \end{cases}$$

$$= \begin{cases} x^*_{rc}, \text{ if } c \in \mathcal{C}_{\text{j}} \forall r = 1, ..., w \\ 0, \text{ otherwise.} \end{cases}$$

As such, we then have that $\mathbf{A}\sum_{\text{j}=1}^{q}\frac{1}{m_{\text{j}}}\sum_{\text{i}=1}^{m_{\text{j}}}\theta^*_{\text{ij}} = \mathbf{b}$. which implies

$$\sum_{\text{j}=1}^{q}\frac{1}{m_{\text{j}}}\sum_{\text{i}=1}^{m_{\text{j}}}\theta^*_{\text{ij}} = \hat{\theta}.$$

Recall $\theta_{\text{j}}(k) = \frac{1}{m_{\text{j}}}\sum_{\text{i}=1}^{m_{\text{j}}}\theta_{\text{ij}}(k)$, since each $\theta_{\text{ij}} \to \theta^*_{\text{ij}}$, then from the above we have:

$$\sum_{\text{j}=1}^{q}\theta_{\text{j}}(k) \to \hat{\theta} \tag{4.6}$$

We now examine the squared error $\|\theta_{\text{j}}(k) - \theta\|^2$:

$$\begin{aligned} \|(\sum_{\text{j}=1}^{q}\theta_{\text{j}}(k)) - \theta\|^2 &= \|((\sum_{\text{j}=1}^{q}\theta_{\text{j}}(k)) - \hat{\theta}) + (\hat{\theta} - \theta)\|^2 \\ &= \|(\sum_{\text{j}=1}^{q}\theta_{\text{j}}(k)) - \hat{\theta}\|^2 + 2\|(\sum_{\text{j}=1}^{q}\theta_{\text{j}}(k)) - \hat{\theta}\|\|(\hat{\theta} - \theta)\|\cos(\beta) + \|(\hat{\theta} - \theta)\|^2 \end{aligned} \tag{4.7}$$

We recall that $\hat{\theta} = \mathbf{A}^{-1}\mathbf{b}$ and recall that $\mathbf{A}$ and $\mathbf{b}$ depend on the number of data samples T, therefore we can express $\hat{\theta}$ as a function of T: $\hat{\theta}(T)$. Now let $\epsilon(T) = \hat{\theta}(T) - \theta$, by combining (4.6) and equation (4.7) it then follows that:

$$\|(\sum_{\text{j}=1}^{q}\theta_{\text{j}}(k)) - \theta\|^2 = \epsilon(T).$$

Furthermore, we note that given assumption (4.5.1) and recalling that $\mathcal{X}$ is a compact set, it then follows that as $T \to \infty$ we have the following: $\lim_{T\to\infty} \frac{1}{T}\sum_{t=1}^{T} A_t = \mathbb{E}[A_t]$, $\lim_{T\to\infty} \frac{1}{T}\sum_{t=1}^{T} b_t = \mathbb{E}[b_t]$. As such, recall $\mathbf{A} = \frac{1}{T}\sum_{t=1}^{T} A_t$ and $\mathbf{b} = \frac{1}{T}\sum_{t=1}^{T} b_t$ and so it follows that since $\theta = \mathbb{E}[A_t]^{-1}\mathbb{E}[b_t]$ and $\hat{\theta} = \mathbf{A}^{-1}\mathbf{b}$ we have the following limit:

$$\lim_{T\to\infty} \epsilon(T) = 0$$

This concludes our proof.

4.6 Simulation Results

In this section we present simulation results for our proposed approach. We have simulated a network consisting of 4 multi-agent teams with 5 agents in each team. The network controls a system with the following dynamics:

$$x_{t+1} = \tilde{A}x_t + \tilde{B}a_t$$
$$y_{\text{ij}} = h_{\text{ij}}(x_t)$$

where $x_t \in \mathbb{R}^3$ is the state of the system, y_{ij} is a nonlinear observation, and $a_t \in \mathbb{R}$ is the joint control input or action of the network, which follows a stabilizing control policy $\boldsymbol{\pi}$. The goal of the network is to evaluate how this policy maximizes the sum of team rewards, which are given by:

$$r_1(x_t, a_t) = -x_1^2 - \frac{1}{4}a_t^2,\ r_2(x_t, a_t) = -x_2^2 - \frac{1}{4}a_t^2$$
$$r_3(x_t, a_t) = -x_3^2 - \frac{1}{4}a_t^2,\ r_4(x_t, a_t) = -x_1x_2 - \frac{1}{4}a_t^2.$$

In order to accomplish this goal, each agent follows the updates given by (4.5) and (4.3), using the following basis functions:

$$\phi(x) = \text{col}\{x_1^2, x_2^2, x_3^2, x_1x_2, x_2x_3, x_1x_3\}, \tag{4.8}$$

We recall $\mathcal{W}_{\text{ij}}$ is a set of indices such that the v'th entry of $\phi(x_t)$ is computable by agent ij if and only if $v \in \mathcal{W}_{\text{ij}}$. The following equations describes which entries of $\phi(x_t)$ are computable by each agent:

$$\mathcal{W}_{11} = \{1\}, \mathcal{W}_{21} = \{1,2\}, \mathcal{W}_{31} = \{1,3\}, \mathcal{W}_{41} = \{1,4\},$$
$$\mathcal{W}_{51} = \{1,5,6\}, \mathcal{W}_{12} = \{1,2\}, \mathcal{W}_{22} = \{2\}, \mathcal{W}_{32} = \{2,3\},$$
$$\mathcal{W}_{42} = \{2,4\}, \mathcal{W}_{52} = \{2,5,6\}, \mathcal{W}_{13} = \{1,3,4\},$$
$$\mathcal{W}_{23} = \{2,3,4\}, \mathcal{W}_{33} = \{3,4\}, \mathcal{W}_{43} = \{3,4\},$$
$$\mathcal{W}_{53} = \{3,4,5,6\}, \mathcal{W}_{14} = \{1,5,6\}, \mathcal{W}_{24} = \{2,5,6\},$$
$$\mathcal{W}_{34} = \{3,5,6\}, \mathcal{W}_{44} = \{4,5,6\}, \mathcal{W}_{54} = \{5,6\}$$

With the above constraints, we use the method outlined in the Approach section to assign row indices and column indices to each agents, and then we used updates $(4.2-4.5)$ for each agent. We have presented the results of using the proposed algorithm in Fig.4.6 and Fig. 4.5. In Fig. 4.6 we show the average squared consensus error of the parameter estimates. This shows us that the consensus error goes to zero as the time step increases, indicating the network achieves consensus in their parameter estimates. In Fig. 4.5 we see that the average squared error goes to zeros as the time step increases, this and the fact the network achieves consensus indicates that each agent converges to the solution θ.

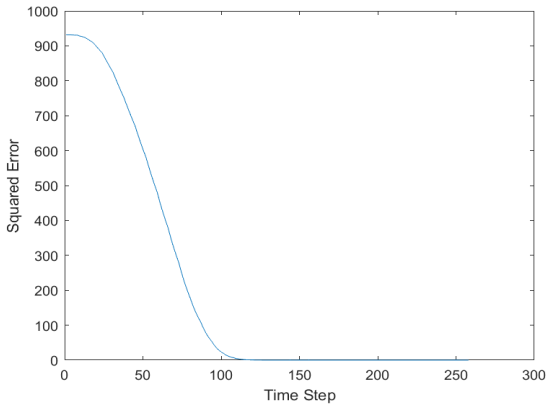

Figure 4.5. Squared Error

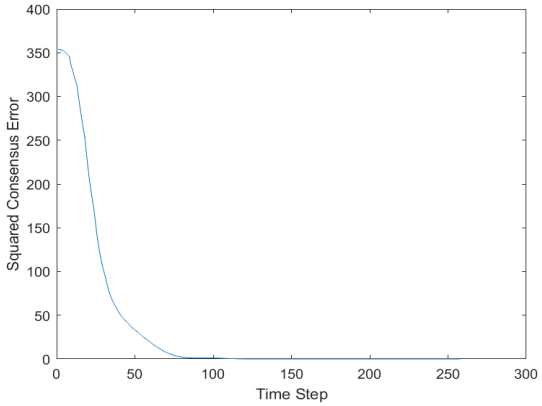

Figure 4.6. Squared Consensus Error

4.7 Conclusion

In this work we have developed a distributed reinforcement learning algorithm for policy evaluation with cross-modal observations. Here, by cross-modal observations we mean each agent observes a different nonlinear observation of the environment. As such, each agent can only compute part of the vector of basis functions ϕ given their observation. Based on which entries of ϕ each agent can compute, we have proposed an approach by which each team can be efficiently assigned columns of $\mathbf{A}$ and each agent is assigned a row. The main idea of this approach is to reduce the assignment problem to a maximum matching problem, which has known and provably efficient algorithms such as the Hopcroft-Karp algorithm. Given an efficient assignment process, an update for each agent's estimate was developed so that each team can converge to part of the θ vector, and such that the network jointly computes the entire θ vector. The main result in this work is that by following the proposed distributed algorithm, each agent converges exponentially to a solution such that the network jointly converges to θ. We have also provided simulations results that demonstrate how the algorithm can be used for distributed policy evaluation of a control system. Our simulations show that the average squared error converges exponentially to zero, which supports our theoretical result. A potential future direction would be to extend the results to include policy improvement by incorporating actor-critic methods, where the critic training would be based on the algorithm proposed in this work.

PART II

Finite-Sample Analysis

5. FINITE-SAMPLE ANALYSIS OF DISTRIBUTED Q-LEARNING FOR MULTI-AGENT NETWORKS

Introduction

Reinforcement learning (RL) algorithms has recently be used for various applications such as autonomous driving, robotics, and so on [1], [4]. As a model-free RL algorithm, Q-learning is able to provide an estimate to the so-called *Q function*, which is a state-action value function to assign a scalar value to each state-action pair according to a given reward function and discount factor [8]. Such Q-functions can be used to achieve optimal control policies and allows for online implementation [127]. Recently researchers have turned their attention to develop RL algorithms for mutli-agent systems due to their extensive applications into formation control [87], [128], coverage [89], social networks [90] and so on. Such development is challenging due to the lack of a centralized coordinator in multi-agent systems. Challenges aside, great progress has been achieved for Multi-Agent Reinforcement Learning (MARL) in [31], [74]. In these algorithms, all agents operate in the same environment with a shared state representing the environment. The state transition depends on the actions of each agent and the unknown state dynamics of the environment. Each agent is given a local reward by the environment depending on the joint action of the agents and the current state. Due to privacy, agents cannot share their rewards with other agents and can only communicate with neighbors (nearby agents). The goal of MARL is then to enable each agent to cooperate with neighbors in order to ensure the entire network of agents develops a policy that maximizes the global average of local rewards. Along this MARL setting, we in this paper will investigate distributed algorithms for Q-learning with the aim of providing a finite-sample analysis, namely, a convergence analysis of the algorithm for any finite number of iterations.

In the literature of MARL, a distributed gossiping TD(0) algorithm is developed in [41], followed by formulation of policy evaluation with linear function approximation as a primal-dual optimization problem[32], as well as for actor-critic algorithms in [129], [130]. Along the direction of distributed Q-learning in multi-agent systems, there have been various empirical results [23], [24], [131]–[133] and theoretical results for convergence [27], [134], [135] with

finite state and action spaces. Distributed Q-learning with continuous states and actions have recently been investigated in [25], [46], [136], [137] by incorporating function approximation with neural networks.

After reviewing the literature we have seen that there are few convergence results for *online* Q-learning for continuous state spaces with function approximation, except [138] which considers fitted Q-iteration-based algorithms. Finite-sample analysis has also been achieved for distributed TD algorithms in [34] under the assumption of i.i.d samples, and in [35] with non i.i.d samples. In the context of Q-learning, there are results of finite-sample analysis for tabular Q-learning in [139], Q-learning with discretized states in [140], Q-learning for high-dimensional stopping problems in [141], and very recently for Q-learning with linear function approximation and non i.i.d samples in [142], [143]. This work provides a finite-sample analysis of a distributed Q-learning algorithm with non i.i.d samples, which to the best of our knowledge has not been addressed in the literature. The finite-sample analysis is needed in order to determine the convergence rate in terms of sample complexity [142]. In other words, a finite-sample analysis will allow us to determine how many samples are needed in order to achieve a given level of accuracy [141].

5.1 Preliminaries

In this section we introduce the multi-agent Markov Decision Process (MDP) of interest and the method of linear function approximation.

5.1.1 Multi-Agent MDP

Consider the case in which a network of m autonomous agents operate in an unknown environment (or plant). Let $x(t) \in \mathcal{X}$ denote the state of the plant at time t which is observed by each agent, where $\mathcal{X}$ is a continuous state space. The state dynamics of the plant can be described by an irreducible Markov chain $\{x(t)\}$, where irreduciblity describes the property that any state is accessible from any other state. For each joint control action $a(t)$ from the network to the plant, a local reward $r_{\mathrm{i}}(x(t), a(t))$ is produced, where $a(t) \in \mathcal{A}$

is observed by each agent and $\mathcal{A}$ is a finite action set. Here, each $r_{\mathrm{i}}(\cdot,\cdot)$ is the private reward locally accessible to only agent i, and is not shared with other agents. Let

$$R(x(t),a(t)) = \sum_{\mathrm{i}=1}^{m} \frac{1}{m} r_{\mathrm{i}}(x(t),a(t)), \tag{5.1}$$

which represents the average reward of all agents in the network. Let $\boldsymbol{\pi}$ denote a fixed stochastic control policy that maps a state $x \in \mathcal{X}$ to a probability distribution $\boldsymbol{\pi}(\cdot \mid x)$ over that action space $\mathcal{A}$, i.e. $a \sim \boldsymbol{\pi}(\cdot \mid x)$. Let $Q_{\boldsymbol{\pi}}$ denote the corresponding state-action value function, which in this work is defined as a sum of discounted rewards $R(x(t),a(t))$ when a stochastic control policy $\boldsymbol{\pi}$ is applied to the plant. Namely,

$$Q_{\boldsymbol{\pi}}(x(t),a(t)) = \mathbb{E}\left[\sum_{k=0}^{\infty} \gamma^k R(x(k),a(k)) \mid x(0)=x(t), a(0)=a(t)\right]$$

where $\gamma \in (0,1)$ is a discount factor. Furthermore, in a multi-agent network, each agent i usually can only communicate with certain neighboring agents denoted by $\mathcal{N}_{\mathrm{i}}(t)$, which includes agent i. The neighbor relations can be modeled by a series of time-varying undirected graphs $\mathbb{G}(t)$ such that there is an edge between i and j if and only if i and j are neighbors at time t.

The goal of multi-agent Q-learning is to enable each agent to achieve $Q_{\boldsymbol{\pi}^*}$ asymptotically, where $\boldsymbol{\pi}^*$ is the global optimal policy, using only local information.

5.1.2 Linear Function Approximation

Since in practice it is not feasible to compute all values of $Q_{\boldsymbol{\pi}^*}$ we will only consider the case where $Q_{\boldsymbol{\pi}^*}(x,a) = \phi(x,a)^\top \theta^*$, where $\phi(x,a) \in \mathbb{R}^p$ is a feature vector and $\theta^* \in \mathbb{R}^p$ is the optimal parameter vector. We note that in general $Q_{\boldsymbol{\pi}^*}$ cannot be exactly expressed using linear combinations of features, and so the best we can do is to converge to a neighborhood of $Q_{\boldsymbol{\pi}^*}$. In addition, we have that θ^* lies in a finite Euclidean-norm ball with radius R, or more

formally $\|\theta^*\| \leq R$. Suppose each agent i controls a θ_{i} (an estimate to the optimal parameter θ^*), and therefore a corresponding estimate of Q_{π^*} which we will denote $Q_{\theta_{\mathrm{i}}}$, where

$$Q_{\theta_{\mathrm{i}}}(x,a) = \phi(x,a)^\top \theta_{\mathrm{i}}.$$

The goal of multi-agent Q-learning then becomes enabling θ_{i} to converge to θ^* asymptotically, where each agent can only share θ_{i} with its neighbours.

5.1.3 A Distributed Q-Learning Algorithm

Based on linear approximation, the goal of Q-learning can be achieved by a distributed update to enable all θ_{i} to converge to θ^*. Towards this end, we will investigate a consensus based distributed Q-learning algorithm, which is based on the distributed TD(0) algorithm given in [34] and similar to other consensus based distributed optimization algorithms [144], [145]. Under this algorithm the update for each agent i is given by:

$$\theta_{\mathrm{i}}(t+1) = \Pi\left(\sum_{\mathrm{j}\in\mathcal{N}_{\mathrm{i}}(t)} W_{\mathrm{ij}}(t)\theta_{\mathrm{j}}(t) + \alpha(t) g_{\mathrm{i}}(\theta_{\mathrm{i}}, t)\right). \tag{5.2}$$

Here, $\mathcal{N}_{\mathrm{i}}(t)$ denotes the set of neighbors of agent i at time t, $W_{\mathrm{ij}}(t)$ denotes the (i,j) element of some doubly stochastic weight matrix $W(t)$, $\alpha(t)$ is the step-size; and $\Pi(\cdot)$ is a projection such that

$$\Pi(\theta) = \arg\min_{\theta':\|\theta'\|\leq R} \|\theta - \theta'\|,$$

Furthermore,

$$g_{\mathrm{i}}(\theta_{\mathrm{i}}, t) = (r_{\mathrm{i}}(x(t), a(t)) + \gamma \max_{b\in\mathcal{A}} Q_{\theta_{\mathrm{i}}}(x(t+1), b) - Q_{\theta_{\mathrm{i}}}(x(t), a(t)))\nabla_\theta Q_{\theta_{\mathrm{i}}}(x(t), a(t))$$

is the semi-gradient [75] of the objective function $J(t, \theta_{\mathrm{i}}(t))$, where

$$J(t, \theta_{\mathrm{i}}(t)) = \frac{1}{2}(r_{\mathrm{i}}(x(t), a(t)) + \gamma \max_{b\in\mathcal{A}} Q_{\theta_{\mathrm{i}}}(x(t+1), b) - Q_{\theta_{\mathrm{i}}}(x(t), a(t)))^2.$$

5.2 Finite-Sample Analysis for Q-Learning

Asymptotic convergence of online distributed Q-learning algorithms such as (5.2) have not been well established and even less known is known of the error bounds of these algorithms for a finite number of iterations. In this section we will provide such a finite-sample analysis to the distributed Q-learning algorithm (5.2), which will be our main result.

To begin, we rewrite the semi-gradient in (5.2) as

$$g_{\mathrm{i}}(\theta_{\mathrm{i}}, t) = b_{\mathrm{i}}(t) + c(\theta_{\mathrm{i}}, t) - A(t)\theta_{\mathrm{i}}(t), \tag{5.3}$$

where:

$$\begin{aligned} A(t) &= \phi(x(t), a(t))\phi(x(t), a(t))^{\top} \\ b_{\mathrm{i}}(t) &= \phi(x(t), a(t))(r_{\mathrm{i}}(x(t), a(t))) \\ c(\theta_{\mathrm{i}}, t) &= \phi(x(t), a(t))\gamma \max_{b\in\mathcal{A}} Q_{\theta_{\mathrm{i}}}(x(t+1), b). \end{aligned}$$

For any fixed $\theta \in \mathbb{R}^p$ and $x \in \mathcal{X}$, we define

$$a^*(x, \theta) = \arg\max_{a\in\mathcal{A}} \phi(x, a)^{\top}\theta.$$

We then define

$$\mathbb{E}[A^*(\theta)] = \mathbb{E}[\phi(x, a^*(x, \theta))\phi(x, a^*(x, \theta))^{\top}].$$

Based on the projection imposed in the update of $\theta(t)$ we can bound the gradient according to

$$\|g_{\mathrm{i}}(\theta_{\mathrm{i}}, t)\| \leq G, \ \forall \theta_{\mathrm{i}} : \|\theta_{\mathrm{i}}\| \leq R,$$

where $G = 2R + r_{\max}$[142].

As in [34], [142], we will make the following assumptions:

1. The feature vectors $\{\phi(x(t), a(t))\}$ are linearly independent for $t \in \{1, 2, .., p\}$ where $\phi(x, a) \in \mathbb{R}^p$. In addition we assume $\|\phi(x, a)\| \leq 1, \quad \forall x, a.$

2. There exists an integer $\mathcal{B}$ such that the following graph is connected for all positive integer l:

$$(\mathcal{V}, \mathcal{E}(l\mathcal{B}) \cup \mathcal{E}(l\mathcal{B}+1) \cup ... \cup \mathcal{E}((l+1)\mathcal{B}-1).$$

Moreover, there exists a constant β such that $W_{\mathrm{ij}}(t) \in [\beta, 1)$ if $\mathrm{j} \in \mathcal{N}_{\mathrm{i}}(t)$, and $W_{\mathrm{ij}}(t) = 0$ if $\mathrm{j} \notin \mathcal{N}_{\mathrm{i}}(t)$.

3. Rewards are uniformly bounded such that $\|r_{\mathrm{i}}(x,a)\| \leq r_{\max}$ for all (x,a) pairs.

4. There are constants $l > 0$ and $\rho \in (0,1)$ such that

$$\sup_{x \in \mathcal{X}} d_{TV}(\mathbb{P}(X_t \in \cdot \mid X_0 = x), P_{\boldsymbol{\pi}}) \leq l\rho^t, \ \forall t \geq 0, \tag{5.4}$$

where $d_{TV}(P,Q) = \sup_A |P(A) - Q(A)|$ denotes the total-variation distance between probability measures P and Q, and $P_{\boldsymbol{\pi}}$ is the stationary distribution of $\boldsymbol{\pi}$.

5. $\mathbb{E}[A]$ is positive definite with $\lambda_1 < \mathbb{E}[A] < \lambda_2$. In addition, we assume $\mathbb{E}[A^*(\theta)] < \lambda_3$ and $\lambda_1 - \gamma^2\lambda_3 \geq w > 0$ for all $\theta : \|\theta\| \leq R$.

Then we have the following main result:

Theorem 5.2.1. *Given assumptions (1-5) are satisfied and each agent follows the update described by (5.2) with a diminishing step-size* $\alpha(t) = \frac{1}{w(t+1)}$, *we then have the following upper bound for the average squared error:*

$$\begin{aligned}\frac{1}{m}\sum_{\mathrm{i}=0}^{m} \mathbb{E}[\|\theta_{\mathrm{i}}(t+1) - \theta^*\|^2] \leq \frac{2\beta_1}{w^2}\frac{1}{t+1} + \frac{2\beta_2}{w^2}\frac{\ln(t+1)}{t+1} + \frac{4\|\Theta(0)\|^2}{m\eta^2}\delta^{2t+2} \\ + \frac{32mG^2}{w^2\eta^2(1-\delta)^2}\left(\delta^{2\lceil t/2\rceil+2} + \frac{1}{(\lceil t/2\rceil+1)^2}\right).\end{aligned} \tag{5.5}$$

where

$$\eta = min\{1 - 1/(2m^3), \sup_{t\geq 0} \sigma_2(W(t))\},$$

$\sigma_2(W(t))$ *denotes the second largest singular value of* $W(t)$, $\delta = \eta^{\frac{1}{B}} < 1$, *and* $\Theta = col(\theta_1, ..., \theta_m)$ *is the augmented parameter vector. In addition,* $\lceil x \rceil$ *denotes the "ceiling" operator , which rounds up any real number* x *to the nearest integer , and similarly* $\lfloor x \rfloor$

denotes the "floor" operator, which rounds down to the nearest integer. Furthermore we have:

$$\beta_1 = \left(12\tau_0 G^2 \ln(\tau_0) + (9+24\tau_0)G^2\right), \ \beta_2 = \left((8+12\tau_0)G^2\right)$$

where $\tau_0 = min\{t \geq 0 : l\rho^t \leq \alpha(t)\}$ *describes a minimum mixing time relative to our step-size.*

Remark 5.2.2. *Below we provide an analysis of the upper bound (5.5). The first term in (5.5) clearly goes to zero as* $t \to \infty$ *and the second term also go to zero as* $t \to \infty$ *due to the fact that* $\frac{1}{t}$ *decays faster than* $\ln(t)$ *grows (this can be verified using L'Hospital's rule). The third and fourth terms decay exponentially to zero since* $\delta < 1$ *and the last term decays to zero as* $\frac{1}{t^2}$*. Therefore, since we have shown that each term of (5.5) goes to zero then the sum must go to zero and so :*

$$\frac{1}{m}\sum_{\mathrm{i}=0}^{m} \mathbb{E}[\|\theta_{\mathrm{i}}(t+1) - \theta^*\|^2] \to 0$$

as $t \to \infty$*.*

5.3 Proof of Main Result

In this section we will provide a proof for Theorem 1 by looking at the squared error of the average parameter vector :$\bar{\theta}(t) = \frac{1}{m}\sum_{\mathrm{i}=1}^{m} \theta_{\mathrm{i}}(t)$. We first introduce some lemmas, whose proofs will be given in the Supplementary section with the exception of Lemma 3 which is proved in [34].

Lemma 5.3.1. *Given assumptions* $(1-5)$ *hold and each agent executes the update described by (5.2), then we have the following upper bound for the squared error of the average parameter vector:*

$$\mathbb{E}[\|\bar{\theta}(t+1) - \theta^*\|^2] \leq \mathbb{E}[(1-\alpha(t)w)\|\bar{\theta}(t) - \theta^*\|_2^2 + \alpha(t)^2 G^2 + 2\alpha(t)\Lambda(\bar{\theta}, \Theta, t)]. \tag{5.6}$$

where $\Lambda(\bar{\theta}, \Theta, t) = \langle \bar{\theta}(t) - \theta^*, \bar{g}(\Theta, t) - \mathbb{E}[\bar{g}(\Theta, t)]\rangle$ and $\bar{g}(\Theta, t) = \frac{1}{m}\sum_{\mathrm{i}=1}^{m} g_{\mathrm{i}}(\theta_{\mathrm{i}}, t)$.

Lemma 5.3.2. *For any $\tau_0 = min\{t \geq 0 : l\rho^t \leq \alpha(t)\}$ we have the following for $t \leq \tau_0$:*

$$\mathbb{E}[\Lambda(\bar{\theta}, \Theta, t)] \leq 6G^2 \sum_{\mathrm{i}=0}^{t-1} \alpha(\mathrm{i}), \tag{5.7}$$

and for $t > \tau_0$:

$$\mathbb{E}[\Lambda(\bar{\theta}, \Theta, t)] \leq (4 + 6\tau_0)G^2\alpha(t - \tau_0). \tag{5.8}$$

Lemma 5.3.3. *From Remark 1 in [34]: Given $\alpha(t) = \frac{1}{w(t+1)}$ we have the following:*

$$\sum_{\mathrm{i}=1}^{m} \|\bar{\theta}(k) - \theta_{\mathrm{i}}(k)\|^2 \leq \left(\frac{\delta^{t+1}}{\eta}\|\Theta(0)\| + \frac{2mG}{w\eta(1-\delta)}\left(\delta^{\lceil t/2 \rceil + 1} + \frac{1}{(\lceil t/2 \rceil + 1)}\right)\right)^2. \tag{5.9}$$

Proof of Theorem 1

Using Lemma 5.3.1 we can express the squared error of the global averaged parameter vector as the following:

$$\mathbb{E}[\|\bar{\theta}(t+1) - \theta^*\|^2] \leq \mathbb{E}[(1 - \alpha(t)w)\|\bar{\theta}(t) - \theta^*\|_2^2 + \alpha(t)^2G^2 + 2\alpha(t)\Lambda(\bar{\theta}, \Theta, t)].$$

Let's now consider a diminishing step size described by $\alpha(t) = \frac{1}{w(t+1)}$. We note that this relies on w being positive which is a result of assumption (5). Substituting for $\alpha(t)$ into (5.6) we get:

$$\mathbb{E}[\|\bar{\theta}(t+1) - \theta^*\|^2] \leq \mathbb{E}[\frac{t}{t+1}\|\bar{\theta}(t) - \theta^*\|_2^2 + \frac{1}{w^2(t+1)^2}G^2 + \frac{2}{w(t+1)}\Lambda(\bar{\theta}, \Theta, t)]. \tag{5.10}$$

We can apply the above inequality recursively which gives us:

$$\mathbb{E}[\|\bar{\theta}(t+1) - \theta^*\|^2] \leq \mathbb{E}[\frac{G^2}{w^2(t+1)}\sum_{k=0}^{t}\frac{1}{k+1} + \frac{2}{w(t+1)}\sum_{k=0}^{t}\Lambda(\bar{\theta}, \Theta, t)], \tag{5.11}$$

since $\sum_{k=0}^{t} \frac{1}{k+1}$ is a harmonic series we then have

$$\sum_{k=0}^{t} \frac{1}{k+1} \leq 1 + \ln(t+1). \tag{5.12}$$

Now let's look at the Λ term. Using the bounds described by Lemma 5.3.2 we can rewrite the sum as:

$$\begin{aligned}
\sum_{k=0}^{t} \mathbb{E}[\Lambda(\bar{\theta}, \Theta, k)] &= \sum_{k=0}^{\tau_0} \mathbb{E}[\Lambda(\bar{\theta}, \Theta, k)] + \sum_{k=\tau_0+1}^{t} \mathbb{E}[\Lambda(\bar{\theta}, \Theta, k)] \\
&\leq \sum_{k=0}^{\tau_0} \left(6G^2 \sum_{k=0}^{k-1} \frac{1}{w(k+1)} \right) + \sum_{k=\tau_0+1}^{t} \left((4+6\tau_0)G^2 \frac{1}{w(k-\tau_0+1)} \right) \\
&\leq \frac{6G^2\tau_0}{w}(\ln(\tau_0)+1) + \frac{(4+6\tau_0)G^2}{w}(\ln(t)+1).
\end{aligned} \tag{5.13}$$

Combining (5.11),(5.12), and (5.13), we get the following:

$$\begin{aligned}
\mathbb{E}[\|\bar{\theta}(t+1) - \theta^*\|^2] \leq \frac{G^2(\ln(t+1)+1)}{w^2(t+1)} + \frac{2}{w(t+1)}\Big(&\frac{6G^2\tau_0}{w}(\ln(\tau_0)+1) \\
&+ \frac{(4+6\tau_0)G^2}{w}(\ln(t)+1)\Big).
\end{aligned}$$

Now let's expand terms and then combine like terms to get:

$$\begin{aligned}
\mathbb{E}[\|\bar{\theta}(t+1) - \theta^*\|^2] = \left(12\tau_0 G^2 \ln(\tau_0) + (9+24\tau_0)G^2\right) &\frac{1}{w^2(t+1)} \\
+ \left((8+12\tau_0)G^2\right) &\frac{\ln(t)}{w^2(t+1)} + G^2 \frac{\ln(t+1)}{w^2(t+1)}.
\end{aligned}$$

In order to simplify our expression we upper bound the above using $\ln(t) < ln(t+1)$, which gives us:

$$\mathbb{E}[\|\bar{\theta}(t+1) - \theta^*\|^2] = \left(12\tau_0 G^2 \ln(\tau_0) + (9+24\tau_0)G^2\right) \frac{1}{w^2(t+1)} + \left((9+12\tau_0)G^2\right) \frac{\ln(t+1)}{w^2(t+1)}.$$

Now let us define the following constants:

$$\beta_1 = \left(12\tau_0 G^2 \ln(\tau_0) + (9+24\tau_0)G^2\right), \ \beta_2 = \left((9+12\tau_0)G^2\right).$$

We can then simplify our expression to :

$$\mathbb{E}[\|\bar{\theta}(t+1) - \theta^*\|^2] \leq \frac{\beta_1}{w^2(t+1)} + \frac{\beta_2 \ln(t+1)}{w^2(t+1)} \tag{5.14}$$

We now have a converging upper bound for the squared error of the average parameter vector, but we still want to provide a converging upper bound for $\frac{1}{m}\sum_{\mathrm{i}=0}^{m}\mathbb{E}[\|\theta_{\mathrm{i}}(t+1) - \theta^*\|^2]$, i.e the average squared error. Fortunately, we can use the Cauchy-Schwarz inequality to relate the squared error of the average parameter vector and the average squared error with the following inequality:

$$\mathbb{E}[\|\theta_{\mathrm{i}}(t+1) - \theta^*\|^2] \leq 2\mathbb{E}[\|\bar{\theta}(t+1) - \theta^*\|^2] + 2\mathbb{E}[\|\theta_{\mathrm{i}}(t+1) - \bar{\theta}(t+1)\|^2].$$

We can sum both sides over all agents and divide by m to get:

$$\frac{1}{m}\sum_{\mathrm{i}=0}^{m}\mathbb{E}[\|\theta_{\mathrm{i}}(t+1) - \theta^*\|^2] \leq 2\mathbb{E}[\|\bar{\theta}(t+1) - \theta^*\|^2] + \frac{2}{m}\sum_{\mathrm{i}=0}^{m}\mathbb{E}[\|\theta_{\mathrm{i}}(t+1) - \bar{\theta}(t+1)\|^2].$$

Using Lemma 1 from [34] the above can be rewritten as:

$$\begin{aligned}\frac{1}{m}\sum_{\mathrm{i}=0}^{m}&\mathbb{E}[\|\theta_{\mathrm{i}}(t+1) - \theta^*\|^2 \\ &\leq 2\mathbb{E}[\|\bar{\theta}(t+1) - \theta^*\|^2] + \frac{2}{m}\left(\frac{\delta^{t+1}}{\eta}\|\Theta(0)\| + \frac{2mG}{\eta}\sum_{k=0}^{t}\delta^{t-k}\frac{1}{w(k+1)}\right)^2.\end{aligned} \tag{5.15}$$

Using (5.9) from Lemma 5.3.3 we can simplify the second term as:

$$\frac{2}{m}\left(\frac{\delta^{t+1}}{\eta}\|\Theta(0)\|+\frac{2mG}{\eta}\sum_{k=0}^{t}\delta^{t-k}\frac{1}{w(k+1)}\right)^2$$
$$\leq \frac{2}{m}\left(\frac{\delta^{t+1}}{\eta}\|\Theta(0)\|+\frac{2mG}{w\eta(1-\delta)}\left(\delta^{\lceil t/2\rceil+1}+\frac{1}{(\lceil t/2\rceil+1)}\right)\right)^2.$$

Then by using the Cauchy-Schwarz inequality we can make the following simplification:

$$\left(\frac{\delta^{t+1}}{\eta}\|\Theta(0)\|+\frac{2mG}{\eta}\sum_{k=0}^{t}\delta^{t-k}\frac{1}{w(k+1)}\right)^2$$
$$\leq \frac{2\delta^{2t+2}}{\eta^2}\|\Theta(0)\|^2+\frac{8m^2G^2}{w^2\eta^2(1-\delta)^2}\left(2\delta^{2\lceil t/2\rceil+2}+\frac{2}{(\lceil t/2\rceil+1)^2}\right). \tag{5.16}$$

And now by plugging in (5.16) and (5.14) into (5.15) we arrive at:

$$\frac{1}{m}\sum_{\mathrm{i}=0}^{m}\mathbb{E}[\|\theta_{\mathrm{i}}(t+1)-\theta^*\|^2]$$
$$\leq\frac{2\beta_1}{w^2}\frac{1}{t+1}+2\beta_2\frac{\ln(t+1)}{w^2(t+1)}+\frac{4\|\Theta(0)\|^2}{m\eta^2}\delta^{2t+2}+\frac{32mG^2}{w^2\eta^2(1-\delta)^2}\left(\delta^{2\lceil t/2\rceil+2}+\frac{1}{(\lceil t/2\rceil+1)^2}\right). \tag{5.17}$$

This concludes our proof.

5.4 Conclusion

This paper has studied a distributed version of the Q-learning algorithm, which incorporates consensus into its update with the goal that all agents in the multi-agent system converge to the same optimal policy and action-value function, such that the global average of rewards is maximized. The main result of this paper is a finite time bound on the average of all parameter errors for a given diminishing step-size $\alpha(t)$. The result supports earlier works which have empirically demonstrated convergence of distributed Q-learning algorithms, and furthermore characterizes the convergence rate of such algorithms without assuming i.i.d data samples. An interesting direction for this work may be to incorporate

policy improvement such as actor-critic algorithms, or to conduct a similar analysis but without the use of a projection step in the algorithm [1]

5.5 Supplementary

5.5.1 Proof of Lemma 1

Using the update of the proposed algorithm described in (5.2), we write the squared error of the average parameter vector as the following:

$$\begin{aligned}
&\mathbb{E}[\|\bar{\theta}(t+1)-\theta^*\|^2] \\
&= \mathbb{E}[\|\Pi(\bar{\theta}(t)+\alpha(t)\bar{g}(\Theta,t))-\Pi(\theta^*))\|^2] \\
&\leq \mathbb{E}[\|\bar{\theta}(t)+\alpha(t)\bar{g}(\Theta,t)-\theta^*)\|^2] \\
&= \mathbb{E}[\|\bar{\theta}(t)-\theta^*\|^2+\alpha(t)^2\|\bar{g}(\Theta,t)\|^2+2\alpha(t)\langle\bar{\theta}(t)-\theta^*,\bar{g}(\Theta,t)\rangle] \\
&\leq \mathbb{E}[\|\bar{\theta}(t)-\theta^*\|_2^2+\alpha(t)^2G^2+2\alpha(t)\langle\bar{\theta}(t)-\theta^*,\mathbb{E}[\bar{g}(\Theta,t)]\rangle \\
&\qquad +2\alpha(t)\langle\bar{\theta}(t)-\theta^*,\bar{g}(\Theta,t)-\mathbb{E}[\bar{g}(\Theta,t)]\rangle]
\end{aligned} \tag{5.18}$$

Let us take a closer look at $\bar{g}(\Theta,t)$. Using (5.3) we have :

$$\bar{g}(\Theta,t)=\bar{b}(t)+\bar{c}(\Theta,t)-A(t)\bar{\theta}(t). \tag{5.19}$$

Now let

$$h(\theta,t)=\phi(x(t),a(t))\left(\gamma\max_{b\in\mathcal{A}}Q_\theta(x(t+1),b)-\phi(x(t),a(t))^\top\theta(t)+\sum_{\mathrm{i}=1}^{m}\frac{1}{m}r_{\mathrm{i}}(x(t),a(t))\right)$$

be the semi-gradient computed for a centralized version of our algorithm, where the rewards of each agent are known to a central agent. Similarly to $g(t)$ we can write

$$h(\theta,t)=\bar{b}(t)+c(\theta,t)-A(t)\theta(t).$$

[1]↑The authors give their sincere thanks to Zhuoran Yang and Kaiqing Zhang for their valuable advice

We note that since the centralized algorithm has access to all rewards, its fixed point will correspond to our target parameter θ^*. This implies that the expected gradient of a centralized algorithm will be zero at $\theta(t) = \theta^*$. Therefore, by taking the expectation of $h(t)$ at $\theta(t) = \theta^*$ we can write:

$$\mathbb{E}[\bar{b}(t)] = \mathbb{E}[A(t)]\theta^* - \mathbb{E}[c(\theta^*, t)], \tag{5.20}$$

where we make use of $\mathbb{E}[A(t)\theta_{\mathrm{i}}(t)] = \mathbb{E}[A(t)]\theta_{\mathrm{i}}(t)$. This can be shown by noting that the expectation is with respect to the behaviour policy, which in Q-learning is independent of the learning policy and so is independent of θ.

Then by taking the expectation of (5.19) and then plugging in (5.20) we have:

$$\begin{aligned}\mathbb{E}[\bar{g}(\Theta, t)] &= \mathbb{E}[A(t)]\theta^* - \mathbb{E}[c(\theta^*, t)] + \mathbb{E}[\bar{c}(t)] - \mathbb{E}[A(t)]\bar{\theta}(t) \\ &= -\mathbb{E}[A(t)](\bar{\theta}(t) - \theta^*) + \mathbb{E}[\bar{c}(\Theta, t)] - \mathbb{E}[c(\theta^*, t)].\end{aligned}$$

Now let's go back to (5.18) and take a closer look at the third term. We can do the following decomposition:

$$2\langle \bar{\theta}(t) - \theta^*, \mathbb{E}[\bar{g}(\Theta, t)]\rangle = -2\langle \bar{\theta}(t) - \theta^*, \mathbb{E}[A(t)](\bar{\theta}(t) - \theta^*)\rangle + 2\langle \bar{\theta}(t) - \theta^*, \mathbb{E}[\bar{c}(t)] - \mathbb{E}[c(\theta^*, t)]\rangle.$$

Now using assumption (5) we can lower bound the first term which then gives us:

$$2\langle \bar{\theta}(t) - \theta^*, \mathbb{E}[\bar{g}(\Theta, t)]\rangle \leq -2\lambda_1 \|\bar{\theta}(t) - \theta^*\|^2 + 2\langle \bar{\theta}(t) - \theta^*, \mathbb{E}[\bar{c}(t)] - \mathbb{E}[c(\theta^*, t)]\rangle. \tag{5.21}$$

Let's focus on the second term above. For convenience we denote $A(t) = A, x = x(t), x' = x(t+1), a = a(t)$, and $\tilde{\theta} = \bar{\theta}(t) - \theta^*$:

$$\begin{aligned}2\langle \tilde{\theta}, \mathbb{E}[\bar{c}(t)] - \mathbb{E}[c(\theta^*, t)]\rangle &= 2\langle \tilde{\theta}, \mathbb{E}[\phi(x, a)(\max_{b \in \mathcal{A}}(\phi(x', b)^\top \bar{\theta}) - \max_{b \in \mathcal{A}}(\phi(x', b)^\top \theta^*))]\rangle \\ &\leq 2\gamma \sqrt{\tilde{\theta}^\top \mathbb{E}[A]\tilde{\theta}} \sqrt{\max\{\tilde{\theta}^\top \mathbb{E}[A^*(\theta)]\tilde{\theta}, \tilde{\theta}^\top \mathbb{E}[A^*(\theta^*)]\tilde{\theta}\}} \\ &\leq 2\gamma \sqrt{\lambda_1 \lambda_3} \|\bar{\theta}(t) - \theta^*\|^2.\end{aligned} \tag{5.22}$$

Now combining (5.21) and (5.22) we arrive at the following:

$$2\langle \bar{\theta}(t) - \theta^*, \mathbb{E}[\bar{g}(\Theta, t)]\rangle = -2(\lambda_1 - \gamma\sqrt{\lambda_1\lambda_3})\|\bar{\theta}(t) - \theta^*\|^2.$$

The above can then be upper bounded by:

$$2\langle \bar{\theta}(t) - \theta^*, \mathbb{E}[\bar{g}(\Theta, t)]\rangle \leq -2\sqrt{\lambda_1}\frac{\lambda_1 - \gamma^2\lambda_3}{\sqrt{\lambda_1} + \gamma\sqrt{\lambda_3}}\|\bar{\theta}(t) - \theta^*\|^2.$$

Now using assumption (5) we can simplify and then upper bound the above in the following way:

$$2\langle \bar{\theta}(t) - \theta^*, \mathbb{E}[\bar{g}(\Theta, t)]\rangle \leq -w\|\bar{\theta}(t) - \theta^*\|^2. \tag{5.23}$$

Let $\Lambda(\bar{\theta}, \Theta, t) = \langle \bar{\theta}(t) - \theta^*, \bar{g}(\Theta, t) - \mathbb{E}[\bar{g}(\Theta, t)]\rangle$. We can now plug in (5.23) to get another upper bound on (5.18):

$$\begin{aligned}
&\mathbb{E}[\|\bar{\theta}(t+1) - \theta^*\|^2] \\
&\leq \mathbb{E}[(1 - \alpha(t)w)\|\bar{\theta}(t) - \theta^*\|_2^2 + \alpha(t)^2G^2 + 2\alpha(t)\Lambda(\bar{\theta}, \Theta, t)],
\end{aligned}$$

where the last step just combines some terms and makes use of our bound on the gradient.

5.5.2 Proof of Lemma 2

First we need to show that $\Lambda(\bar{\theta}, \Theta, t)$ is Lipschitz continuous. We start by considering the following difference :

$$\begin{aligned}
&\Big|\Lambda(\bar{\theta}_1, \Theta_1, t) - \Lambda(\bar{\theta}_2, \Theta_2, t))\Big| \\
=&\Big|\langle \bar{\theta}_1 - \theta^*, \bar{g}(\Theta_1, t) - \mathbb{E}[\bar{g}(\Theta_1, t)]\rangle - \langle \bar{\theta}_2 - \theta^*, \bar{g}(\Theta_2, t) - \mathbb{E}[\bar{g}(\Theta_2, t)]\rangle\Big| \\
\leq& 2R\|\bar{g}(\Theta_1, t) - \mathbb{E}[\bar{g}(\Theta_1, t)] - (\bar{g}(\Theta_2, t) - \mathbb{E}[\bar{g}(\Theta_2, t)])\| + 2G\|\bar{\theta}_1 - \bar{\theta}_2\|
\end{aligned} \tag{5.24}$$

The first term in 5.24 can be upperbounded with the following:

$$\begin{aligned}\|\bar{g}(\Theta_1,t)-\mathbb{E}[\bar{g}(\Theta_1,t)]-(\bar{g}(\Theta_2,t)-\mathbb{E}[\bar{g}(\Theta_2,t)])\| \\ \leq \|\bar{g}(\Theta_1,t)-\bar{g}(\Theta_2,t)\|+\|\mathbb{E}[\bar{g}(\Theta_1,t)]-\mathbb{E}[\bar{g}(\Theta_2,t)]\|.\end{aligned}$$

Now let's look at the first term on the right hand side where we denote $x = x(t)$ and $a = a(t)$. We do the following :

$$\begin{aligned}\|\bar{g}(\Theta_1,t)-\bar{g}(\Theta_2,t)\| &\leq \left|\frac{\gamma}{m}\sum_{\mathrm{i}=0}^{m}(\max_{b\in\mathcal{A}}\phi(x',b)^\top\theta_{\mathrm{i},1}-\max_{b\in\mathcal{A}}\phi(x',b)^\top\theta_{\mathrm{i},2})\right|+\left|\phi(x,a)^\top(\bar{\theta}_1-\bar{\theta}_2)\right| \\ &\leq (1+\gamma)\|\bar{\theta}_1-\bar{\theta}_2\|.\end{aligned}$$

The last step is done by using the following arguement. Consider the following:

$$\max_{b\in\mathcal{A}}\phi(x',b)^\top\theta_{\mathrm{i}1}-\max_{b\in\mathcal{A}}\phi(x',b)^\top\theta_{\mathrm{i}2}=\phi(x',a_1)^\top\theta_{\mathrm{i}1}-\phi(x',a_2)^\top\theta_{\mathrm{i}2},$$

which is lower bounded by $\phi(x',a_2)^\top(\theta_{\mathrm{i},1}-\theta_{\mathrm{i},2})$ and upper bounded by $\phi(x',a_1)^\top(\theta_{\mathrm{i},1}-\theta_{\mathrm{i},2})$. We then have:

$$\begin{aligned}&\left|\frac{1}{m}\sum_{\mathrm{i}=0}^{m}(\max_{b\in\mathcal{A}}\phi(x',b)^\top\theta_{\mathrm{i},1}-\max_{b\in\mathcal{A}}\phi(x',b)^\top\theta_{\mathrm{i},2})\right| \\ &\leq \max\{\left|\phi(x',a_2)^\top(\bar{\theta}_1-\bar{\theta}_2)\right|,\left|\phi(x',a_1)^\top(\bar{\theta}_1-\bar{\theta}_2)\right|\} \\ &\leq \|\bar{\theta}_1-\bar{\theta}_2\|.\end{aligned}$$

It then similarly follows that

$$\|\mathbb{E}[\bar{g}(\Theta_1,t)]-\mathbb{E}[\bar{g}(\Theta_2,t)]\|\leq(1+\gamma)\|\bar{\theta}_1-\bar{\theta}_2\|.$$

We therefore arrive at the following:

$$\left|\Lambda(\bar{\theta}_1,\Theta_1,t)-\Lambda(\bar{\theta}_2,\Theta_2,t))\right|\leq 6G\|\bar{\theta}_1-\bar{\theta}_2\|,$$

which demonstrates Lipschitz continuity.

We now look at upper bounding $\|\bar{\theta}_1 - \bar{\theta}_2\|$ by analyzing the update of $\bar{\theta}$ according to our algorithm. We have the following:

$$\begin{aligned}
&\|\bar{\theta}(t+1) - \bar{\theta}(t)\| \\
&\quad = \|\frac{1}{m}\sum_{\mathrm{i}=1}^{m}\Pi(\sum_{\mathrm{j}\in\mathcal{N}_\mathrm{i}(t)} W_{\mathrm{ij}}\theta_\mathrm{j}(t) + \alpha(t)g_\mathrm{i}(\theta_\mathrm{i}, t)) - \Pi(\bar{\theta}(t))\| \\
&\quad \leq \|\frac{1}{m}\sum_{\mathrm{i}=1}^{m}(\sum_{\mathrm{j}\in\mathcal{N}_\mathrm{i}(t)} W_{\mathrm{ij}}\theta_\mathrm{j}(t) + \alpha(t)g_\mathrm{i}(\theta_\mathrm{i}, t)) - \bar{\theta}(t)\| \\
&\quad = \|\frac{1}{m}\sum_{\mathrm{i}=1}^{m}(\theta_\mathrm{i}(t) + \alpha(t)g_\mathrm{i}(\theta_\mathrm{i}, t)) - \bar{\theta}(t)\| \\
&\quad = \|\alpha(t)\bar{g}(\Theta, t)\| \leq G\alpha(t),
\end{aligned} \tag{5.25}$$

where we use the fact that W is doubly stochastic for the second equality. Given assumption 4, our proof then follows the proof of Lemma 15 from [142].

6. FINITE-SAMPLE ANALYSIS OF MULTI-AGENT POLICY EVALUATION WITH KERNELIZED GRADIENT TEMPORAL DIFFERENCE

Introduction

Due to recent success of applying reinforcement learning into feedback control systems, autonomous driving and robotics [1], [4], [127], [146], and also extensive application of multi-agent systems [128], [147]–[149], significant research attention has been given to Multi-Agent Reinforcement Learning (MARL), which aims to enable agents to cooperate with nearby neighbors in order to achieve a value function that accurately evaluates the global policy in terms of the average of local rewards. Although such development is challenging due to the lack of a centralized coordinator in multi-agent systems, there has been significant progress in algorithm development for MARL [31], [74], [91]–[93]. In these distributed algorithms, all agents are usually assumed to operate in the same environment with a shared state representing the environment, and each agent is given a local reward by the environment depending on the joint action of the agents and the current state. Various algorithms in the direction of MARL have been developed, such as the distributed TD(0) algorithm based on gossiping [41], followed by formulation of policy evaluation with linear function approximation as a primal-dual optimization problem[32]; distributed algorithms for actor-critic learning in multi-agent systems [150], communication efficient distributed RL [151], Q-learning algorithms [152], [153], and so on.

With more and more MARL algorithms developed recently, most of them rely on parametric models, where the value function is approximated as a linear combination of features [124], [154]. This motivates this work consideration of non-parametric models by using functions in a Repeated Kernel Hilbert Space(RKHS), which gives the learning a much more general class of nonlinear functions compared to parametric methods [155]. We also notice that policy evaluation is at the core of reinforcement learning algorithms, which assigns a scalar value to each state given a fixed policy and based on a reward function [75]. Once the policy evaluation is established, one can use the resulted value function to improve the

current policy using actor-critic algorithms [72], [77]. However, many reinforcement learning algorithms based on RKHS have mainly focused on policy searching [156], [157], where the policy is assumed to lie in an RKHS. That is why this work has investigated the case where the value function lies in an RKHS. Although in [158], [159], both the value function and policy are assumed to be in an RKHS, they are not directly applicable to the case of multi-agent systems. We also notice that existing results in MARL usually establish the asymptotic or exponential convergence while a finite-sample analysis is not as common. Here, by a *finite-sample analysis* is meant a convergence analysis of the algorithm for any finite number of iterations. Such an analysis is essential to determine the convergence rate in terms of sample complexity [142], in other words, determine how many samples are needed in order to achieve a given level of accuracy [141]. Finite-sample analysis to MARL has been limited to distributed TD algorithms in [34], [35] and Q-learning [139], [140]. In recognition of all these, this work has provided a finite-sample analysis for distributed algorithms for MARL where the value function is in an RKHS.

6.1 Preliminaries and the Problem

Consider a network of m autonomous agents that operate in an unknown environment (or plant). Let $x_t \in \mathcal{X}$ denote the state of the plant at time t which is observed by each agent, where $\mathcal{X}$ is continuous state space and where $n = |\mathcal{X}|$. Suppose dynamics of the plant state can be described by an irreducible Markov chain $\{x_t\}$, where irreducibility describes the property that any state is accessible from any other state. For each joint control action a_t from the multi-agent network to the plant, a local reward $r_{\mathrm{i}}(x_t, a_t)$ is produced, where $a_t \in \mathcal{A}$ is observed by all agents and $\mathcal{A}$ is a continuous action space. Here, each $r_{\mathrm{i}}(\cdot)$ is the private reward locally accessible to only agent i, which is not shared with other agents. Let

$$R(x_t, a_t) = \sum_{\mathrm{i}=1}^{m} \frac{1}{m} r_{\mathrm{i}}(x_t, a_t), \tag{6.1}$$

which represents the average reward of all agents in the network. Let $\boldsymbol{\pi}$ denote a fixed stationary control policy such that $a \sim \boldsymbol{\pi}(a \mid x)$. Let $V_{\boldsymbol{\pi}}$ denote the corresponding state-value function defined as:

$$V_{\boldsymbol{\pi}}(x_t) = \mathbb{E}\left[\sum_{k=0}^{\infty} \gamma^k R(x_k, a_k) \mid x_0 = x_t\right]$$

where $\gamma \in (0, 1)$ is a discount factor.

MARL algorithms are developed for all V_{i} to converge to $V_{\boldsymbol{\pi}}$, where V_{i} denotes agent i's estimation to $V_{\boldsymbol{\pi}}$. Towards this end, one usually introduces the following objective function:

$$J(V) = \frac{1}{m}\sum_{\mathrm{i}=1}^{m} \mathbb{E}[l_{\mathrm{i}}(V_{\mathrm{i}}) + \frac{\lambda}{2}\|V_{\mathrm{i}}\|_{\mathcal{H}}^2], \tag{6.2}$$

where $l_{\mathrm{i}}(V_{\mathrm{i}}, t) = \frac{1}{2}(r_{\mathrm{i}}(x_t, a_t) + \gamma V_{\mathrm{i}}(x_{t+1}) - V_{\mathrm{i}}(x_t))^2$, $V = \mathrm{col}\{V_1(\cdot), V_2(\cdot), ..., V_m(\cdot)\}$ and $\mathcal{H}$ is a repeated kernel Hilbert space (RKHS) as in [160], and $\mathcal{E}$ is the set of all agent pairs. It has been shown in [32], [160], [161] that the MARL problem can be posed as the following regularized optimization problem:

$$[V_{\boldsymbol{\pi}}]_m = \arg\min_{V \in \mathcal{H}^m} J(V) \quad \text{s.t. } V_{\mathrm{i}} = V_{\mathrm{j}},\ (\mathrm{i}, \mathrm{j}) \in \mathcal{E}, \tag{6.3}$$

with $[V_{\boldsymbol{\pi}}]_m$ the column-wise stacking of m copies of $V_{\boldsymbol{\pi}}$.

Note that a repeated kernel Hilbert space is defined by a kernel function κ that has the following properties for all functions $f \in \mathcal{H} : \mathcal{X} \to \mathbb{R}$:

$$\langle f, \kappa(x, \cdot)\rangle_{\mathcal{H}} = f(x),\ \mathcal{H} = \overline{\mathrm{span}\{\kappa(x, \cdot)\}}$$

According to the Representer Theorem [161], [162] one can describe $V_{\boldsymbol{\pi}}$ as a sum of kernel evaluations only over training data, namely

$$V_{\boldsymbol{\pi}}(x_t) = \sum_{n=1}^{N} \theta_n^* \kappa(x_n', x_t), \tag{6.4}$$

where $\kappa(x'(\mathrm{i}), x_t) \in \mathbb{R}$ is a kernel function and $\theta^* \in \mathbb{R}^N$ is the optimal parameter vector, with θ^*_n denoting the n^{th} element of θ^* and N denoting the number of data samples observed. Suppose the proposed objective function satisfies the Representer Theorem. The goal of MARL then becomes for each agent to find the θ^*.

We now discuss some aspects of the multi-agent network. In such a network, each agent i usually can only communicate with certain neighboring agents denoted by $\mathcal{N}_\mathrm{i}(t)$, which includes agent i. The neighbor relations can be modeled by a series of time-varying undirected graphs $\mathbb{G}(t)$ such that there is an edge between i and j if and only if i and j are neighbors at time t. Suppose each agent i controls a θ_i (an estimate to the optimal parameter θ^*), and V_i which as agent i's estimate to V_{π^*} is given by

$$V_\mathrm{i}(x_t) = \sum_{n=1}^{N} \theta_{\mathrm{i},n} \kappa(x'_n, x_t). \tag{6.5}$$

In the above we write $\theta_{\mathrm{i},n}$ to denote the n^{th} element of the vector θ_i. Then the goal of MARL under RKHS is for each θ_i to converge to θ^*, or equivalently for $V_\mathrm{i} \to V_\pi$, where each agent can only share θ_i with its neighbours.

6.2 Functional Stochastic Quasi-Gradient Method with Consensus

We will use a functional stochastic quasi-gradient method with kernel approximation and consensus. Later on the KOMP algorithm [160] will be introduced for a finite-sum representation of V_{θ_i}. Since the policy evaluation is of our particular interest, we also adopt the assumption of a fixed global behaviour policy $\boldsymbol{\pi}$, namely, local updates to each agent's parameter does not affect the global behaviour policy.

Let $V_{\mathrm{i},t}$ denote agent i's estimate of V_π at step t and let $x_{\mathrm{i},t}$ correspond to a dictionary of stored data samples.The update of each agents value function estimate is the following:

$$V_{\mathrm{i},t+1}(\cdot) = P_{X_{\mathrm{i},t+1}}(\sum_{\mathrm{j}\in\mathcal{N}_\mathrm{i}(t)} W_{\mathrm{ij}}(t) V_{\mathrm{j},t}(\cdot) - \alpha(t) g(V_{\mathrm{i},t}, t)), \tag{6.6}$$

where $W_{\mathrm{ij}}(t)$ denotes the (i,j) element of some doubly stochastic matrix $W(t)$ at time t with lazy Metropolis weights [163], $P_{X_{\mathrm{i}}(t+1)}$ is a projection defined by,

$$P_{X_{\mathrm{i}}(t+1)}(f') = \mathrm{argmin}_{f \in \mathcal{H}_{X_{\mathrm{i}}(t+1)}} \|f - f'\|^2_{\mathcal{H}},$$

and $\mathcal{H}_{X_{\mathrm{i},t+1}}$ is a subspace of $\mathcal{H}$ defined by $\mathcal{H}_{X_{\mathrm{i},t+1}} = \mathrm{span}\{\kappa(\chi_1,\cdot),...,\kappa(\chi_{M(t+1)}),\cdot)\}$, where $M(t+1)$ is the number of entries in $X_{\mathrm{i},t}$ at time $t+1$ and χ_{j} is the jth entry of $X_{\mathrm{i},t}$. Furthermore, we have:

$$\begin{aligned} g(V_{\mathrm{i},t},t) &= z_{\mathrm{i},t+1}(\kappa(x_t,\cdot) - \gamma\kappa(x_{t+1},\cdot)) + \lambda V_{\mathrm{i},t}(\cdot) \\ z_{\mathrm{i},t+1} &= (1-\beta(t))z_{\mathrm{i},t} + \beta(t)(\delta_{\mathrm{i}}(t)), \end{aligned} \tag{6.7}$$

where z_{i} is a running average of the temporal difference observed by agent i, which we denote δ_{i} and define as $\delta_{\mathrm{i}}(t) = r_{\mathrm{i}}(x_t,a_t) + \gamma V_{\mathrm{i},t}(x_{t+1}) - V_{\mathrm{i},t}(x_t)$. For notational convenience we do not write g as an explicit function of $z_{\mathrm{i},t+1}$, but we will make this explicit for clarity in parts of our analysis. Then the above updates corresponds to the following updates of our dictionary and parameters.

$$X_{\mathrm{i},t+1} = [x_{\mathrm{i},t}, x_{\mathrm{i},t}, x_{\mathrm{i},t+1}], \tag{6.8}$$

$$\theta_{\mathrm{i}}(t+1) = [(1-\alpha(t)\lambda)\theta_{\mathrm{i}}(t), -\alpha(t)z_{\mathrm{i},t+1},\ \alpha(t)\gamma z_{\mathrm{i},t+1}] \tag{6.9}$$

where $\alpha(t)$ is the step-size.

In order to fit our data exactly we could keep storing every new data sample to our dictionary and parameter vector according to (6.8) and (6.9), but this would be impractical since it would require infinite memory as $t \to \infty$. Therefore, every time a new data sample is recorded we use a model order regulation algorithm called Kernel Orthogonal Matching Pursuit(KOMP) [160], [164]. At each step, this algorithm will try to remove entries from the dictionary such that the Hilbert norm error between the original function approximation and the reduced order function approximation is minimized. The algorithm continues to remove entries and recalculates weights until the approximation error becomes greater than

a specified error tolerance $\epsilon(t)$. For the sake of brevity we will not go into the full details of this algorithm but they can be found in [160], [164].

Furthermore, because the KOMP algorithm acts as a projection on our function estimates then according to [160] we also have that:

$$\|V_{\mathrm{i},t}\|_{\mathcal{H}} \leq R, \, \forall t, \mathrm{i} \tag{6.10}$$

for some constant R

6.3 Main Result

In this section we will present the main result, which provides an upper bound for the average squared error that is defined for each iteration of the presented algorithm.

Before presenting the main theorem, we give some assumption which have been widely adopted in existing literature [32], [124], [154], [160], [161].

Assumptions

1. Data samples are i.i.d and drawn from the stationary distribution $\mu_{\boldsymbol{\pi}}$.

2. The state space $\mathcal{X}$ and action space $\mathcal{A}$ are compact.

3. The difference of reproducing kernels has finite conditional variance. More formally, we have the following: $\mathbb{E}[\|\kappa(x,\cdot) - \gamma\kappa(x,\cdot)\|^2_{\mathcal{H}} \mid \mathcal{F}] \leq G^2$, where $\mathcal{F}_t$ is the data history at up until time t.

4. The average temporal difference $\bar{\delta} = \frac{1}{m}\sum_{\mathrm{i}=1}^{m} \delta_{\mathrm{i}}$ has finite variance. The average estimate of the expected temporal difference $\bar{z} = \frac{1}{m}\sum_{\mathrm{i}=1}^{m} z_{\mathrm{i}}$ has finite conditional second moments from zero. In other words, we have:$\mathbb{E}[(\bar{\delta} - \mathbb{E}[\bar{\delta}])^2 \mid \mathcal{F}] \leq \sigma^2_{\delta}, \;\; \mathbb{E}[\bar{z}^2 \mid x, \boldsymbol{\pi}(x)] \leq \sigma^2_{z}$.

5. The expectation of the temporal difference $\mathbb{E}[\delta_{\mathrm{i}}]$, for each agent, is Lipschitz continuous with respect to the value function V. Formally, for any two distinct δ_{i} and δ'_{i} we have: $|\mathbb{E}[\delta_{\mathrm{i}}] - \mathbb{E}[\delta'_{\mathrm{i}}]| \leq L\|V - V'\|_{\mathcal{H}}$.

6. Let $\mathcal{V}$ be the node set and let $\mathcal{E}(t)$ be the edge set of graph $\mathbb{G}(t)$ for all t. There exists an integer $\mathcal{B}$ such that the following graph is connected for all positive integer l: $(\mathcal{V}, \mathcal{E}(l\mathcal{B}) \cup \mathcal{E}(l\mathcal{B}+1) \cup ... \cup \mathcal{E}((l+1)\mathcal{B}-1)$. Moreover, there exists a constant c such that $W_{\mathrm{ij}}(t) \in [c, 1)$ if $\mathrm{j} \in \mathcal{N}_{\mathrm{i}}(t)$, and $W_{\mathrm{ij}}(t) = 0$ if $\mathrm{j} \notin \mathcal{N}_{\mathrm{i}}(t)$.

With the above assumptions we then have:

Theorem 6.3.1. *Suppose that assumptions (1-6) are satisfied and each agent follows the update described by (6.6) along with the KOMP algorithm [160], with fixed step-sizes $\alpha(t) = \alpha, \beta(t) = \beta$ and tolerance parameter $\epsilon(t) = C\alpha(t)^2, C > 0$. Furthermore, suppose that $0 < \beta < 1$, $\alpha > 0$, and the regularization parameter is $\lambda = \frac{\alpha}{\beta}G^2 + \frac{\lambda_0}{\alpha}$ with $0 < \lambda_0 < 1$. Then we have the following upper bound for the average squared error:*

$$\frac{1}{m}\sum_{\mathrm{i}=1}^{m}\mathbb{E}[\|V_{\mathrm{i},t+1} - V^*\|^2] \leq 2(1-\lambda_0)^t\mathbb{E}[\|V_0 - V^*\|_{\mathcal{H}}^2] + \frac{2\zeta^{2t}}{m\eta^2}\|V_0\|_{\mathcal{H}}^2 + D(\alpha, \beta, \lambda_0),$$

where $\eta = \min\{1 - \frac{1}{2m^3}, \sup_{k\geq 0}\sigma_2(W(k)))\}$, $\sigma_2(W(k)))$ denotes the second largest singular value of $W(k)$, $\zeta = \eta^{\frac{1}{\mathcal{B}}}$, and $D(\alpha, \beta, \lambda_0)$ is a function of design parameters but constant with time.

Remark 6.3.2. *From the above we can conclude that as $t \to \infty$ the Hilbert norm error converges to $D(\alpha, \beta, \lambda_0)$, where*

$$D(\alpha, \beta, \lambda_0) = \mathcal{O}\left(\frac{\alpha^2}{\lambda_0} + \frac{\beta^2}{\lambda_0} + \frac{\alpha^2}{\beta\lambda_0} + \frac{\alpha^4}{\beta^3\lambda_0} + \frac{\alpha^2\lambda_0}{\beta} + \frac{\alpha^4}{\beta\lambda_0} + \alpha^2\right),$$

and $\sigma = \sigma_z G + \lambda R + mC\alpha$. From the above we see that if $\alpha \approx \beta$ then the above simplifies to $\mathcal{O}(\alpha + \frac{\alpha}{\lambda_0} + \frac{\alpha^2}{\lambda_0} + \frac{\alpha^3}{\lambda_0} + \alpha^2)$, which for a fixed λ_0 will go to zero as α goes to zero.

Remark 6.3.3. *From the above analysis we see that there are two regimes of interest that behave differently with respect to the free parameters λ_0 and α. In the first regime where $\alpha < 1$, the result indicates that $\frac{\alpha}{\lambda_0}$ dominates the error upper bound and so by making the term very small we will converge to a very small neighborhood around the optimal value. However, in this regime λ is dominated by λ_0/α, and so we cannot make $\frac{\alpha}{\lambda_0}$ arbitrarily small*

since that would result in a very large λ which would bias our results towards the trivial solution of $V = 0$. In the regime where $\alpha > 1$, the error upper bound is dominated by $\frac{\alpha^3}{\lambda_0}$ and λ is now dominated by the $\frac{\alpha}{\beta}G^2$ term. Therefore, an increase in λ_0, and thereby a potential increase in convergence speed, would not significantly increase λ and bias the solution, but it would increase the error upper bound.

6.4 Proof of Main Result

In this section we will provide a proof for Theorem 1. First we begin by introducing some lemmas. The proofs of the lemmas 6.4.2 - 6.4.4 are given in the Supplementary section. For brevity we have left out the proof of lemma 6.4.1 and we refer to [160] for its proof.

Lemma 6.4.1. *Suppose each agent follows the KOMP algorithm[160] and the update described by (6.6), from [160] we have the following result:*

$$\|g'(V_t, z, t) - g(V_t, z, t)\|_{\mathcal{H}} \leq \frac{m\epsilon(t)}{\alpha(t)}.$$

Lemma 6.4.2. *Suppose each agent follows the update described by (6.6) along with the KOMP algorithm [160] and , we then have the following results:*

$$\mathbb{E}[\|g(V_t, t)\|_{\mathcal{H}}] \leq \sigma_z G + \lambda R, \quad \mathbb{E}[\|g'(V_t, t)\|_{\mathcal{H}}] \leq \sigma, \tag{6.11}$$

where $\sigma = \sigma_z G + \lambda R + \frac{m\epsilon_{max}}{\alpha_{min}}$, ϵ_{max} is the maximum value of $\epsilon(t)$, and α_{min} is the minimum value of $\alpha(t)$ for all t. Furthermore, we have:

$$\mathbb{E}[\bar{V}_{t+1} - \bar{V}_t\|_{\mathcal{H}}^2 \leq 4\alpha(t)^2(\sigma_z^2 G^2 + \lambda^2 R^2) + 2\epsilon(t)^2. \tag{6.12}$$

Lemma 6.4.3. *Given assumptions (1-3) are satisfied and each agent follows the update described by 6.6 along with the KOMP algorithm [160], and suppose $\alpha(t), \beta(t) > 0$, then we have the following result for the expected error of the average value function estimate $\bar{V}_t$:*

$$\begin{aligned}\mathbb{E}[\|\bar{V}_{t+1}-V^*\|_{\mathcal{H}}^2 \mid \mathcal{F}_t] \\ &\leq (1+\frac{\alpha(t)^2}{\beta(t)}G^2)\|\bar{V}_t - V^*\|_{\mathcal{H}}^2 - \alpha(t)\lambda\|\bar{V}_t - V^*\|_{\mathcal{H}}^2 \\ &\quad + 2\epsilon(t)\|\bar{V}_t - V^*\|_{\mathcal{H}} + \ \alpha(t)^2\sigma^2 + \beta(t)\mathbb{E}[((\bar{z}_{t+1}-\mathbb{E}[\bar{\delta}_t])^2 \mid \mathcal{F}_t]. \end{aligned} \tag{6.13}$$

Lemma 6.4.4. *Given assumptions 4 and 5 are satisfied and $0 < \beta(t) < 1$, we then have the following upper bound for the expected squared estimation error of the average Bellman error:*

$$\mathbb{E}[(\bar{z}_{t+1} - \mathbb{E}[\bar{\delta}_t])^2 \mid \mathcal{F}_t] \leq (1-\beta(t))(\bar{z}_t - \mathbb{E}[\bar{\delta}_{t-1}])^2 + 2\beta(t)^2\sigma_\delta^2 + \frac{1}{\beta(t)}L^2\|\bar{V}_t - V_{t-1}\|_{\mathcal{H}}^2. \tag{6.14}$$

Proof of Theorem 1

In order to prove the main result we will first derive an upper bound for the expected squared error of the average value function, then we will derive an upper bound for the expected squared consensus error, and finally we will combine these upper bounds using Cauchy-Schwartz to arrive at the main result.

We start by analyzing the difference between the expected temporal difference $\mathbb{E}[\delta]$ and its estimate z, namely $\mathbb{E}[(\bar{z}(t+1) - \mathbb{E}[\bar{\delta}_t])^2]$. With $\beta(t) = \beta$, we now take the total expectation of both sides of (6.14) from lemma (6.4.4) , and using lemma 6.4.2 we plug in (6.12) into the last term of (6.14) with constant step size and compression budget such that $\alpha(t) = \alpha$ and $\epsilon(t) = \epsilon$. We can then apply the resulting relation recursively from $t = 0$. Then by applying the initial conditions $z_0 = 0$ and $\delta_{-1} = 0$, and noting some terms have constant upper bounds we arrive at:

$$\mathbb{E}[(\bar{z}_{t+1} - \mathbb{E}[\bar{\delta}_t])^2] \leq (2\beta\sigma_\delta^2 + \frac{2L^2}{\beta^2}(\alpha^2(\sigma_z^2 G^2 + \lambda^2 R^2) + \epsilon^2)), \tag{6.15}$$

We now proceed with analyzing the evolution of the Hilbert norm error of the value function $\mathbb{E}[\|\bar{V}_{t+1} - V^*\|_{\mathcal{H}}^2 \mid \mathcal{F}_t]$. Accordingly, we take the total expectation of both sides of (6.13) from lemma 6.4.3,with $\alpha(t) = \alpha$, and plug in (6.15). Now by combining terms, using $\|V\|_{\mathcal{H}} < R$ to bound the third term of (6.13), and setting $\epsilon = C\alpha^2$ with $C > 0$ we then arrive at:

$$\mathbb{E}[\|\bar{V}_{t+1} - V^*\|_{\mathcal{H}}^2] \leq (1 + \frac{\alpha^2}{\beta} G^2 - \alpha\lambda)\mathbb{E}[\|\bar{V}_t - V^*\|_{\mathcal{H}}^2] + A,$$

where $A = 4C\alpha^2 R + \alpha^2\sigma^2 + 2\beta^2\sigma_\delta^2 + \frac{2L^2}{\beta}(\alpha^2(\sigma_z^2 G^2 + \lambda^2 R^2) + C^2\alpha^4)$. With the above inequality it is clear that the Hilbert norm error will converge if we set $\lambda = \frac{\alpha}{\beta} G^2 + \frac{\lambda_0}{\alpha}$, with $0 < \lambda_0 < 1$. Setting λ accordingly, we then have:

$$\mathbb{E}[\|\bar{V}_{t+1} - V^*\|_{\mathcal{H}}^2] \leq (1 - \lambda_0)\mathbb{E}[\|\bar{V}_t - V^*\|_{\mathcal{H}}^2] + A.$$

Then applying the above inequality recursively we arrive at:

$$\mathbb{E}[\|\bar{V}_{t+1} - V^*\|_{\mathcal{H}}^2] \leq (1 - \lambda_0)^t \mathbb{E}[\|V_0 - V^*\|_{\mathcal{H}}^2] + \frac{A}{\lambda_0}. \tag{6.16}$$

where in the last step we use the fact that $\sum_{k=1}^{t}(1 - \lambda_0)^k$ is a geometric sequence and that $\lambda_0 < 1$.

Now that we have an upper bound for the expected squared error of the average value function, we will now look to derive an upper bound for the expected consensus error.

We define $\tilde{W}(t) = W(t) \otimes I_n$, $\tilde{V}$ to be the column wise stacking of m copies of $\bar{V}$, i.e. $\tilde{V} = \mathbf{1}_m \otimes \bar{V}$, and V to be the column-wise stacking of each agent's V_{i}, i.e $V = \text{col}\{V_1, ..., V_m\}$. Since we construct $W(t)$ to be doubly stochastic, we have the following: $\tilde{W}(t)(V_t - \tilde{V}_t) = \tilde{W}(t)(V_t - \tilde{V}_t)$.

Let $Q = (I_m - \frac{1}{m}\mathbf{1}_m\mathbf{1}_m^\top) \otimes I_n$. We now analyze the evolution of the consensus error $(V_{t+1} - \tilde{V}_{t+1})$:

$$QV_{t+1} = (V_{t+1} - \tilde{V}_{t+1}) = \tilde{W}(t)QV_t - \alpha(t)Qg'(V_t, t).$$

We now apply the above equation recursively from $t = 0$, and then by taking the Hilbert norm of both sides and using the triangle inequality we arrive at:

$$\|QV_{t+1}\|_{\mathcal{H}} \leq \|\prod_{k=0}^{t} \tilde{W}(k)QV_0\|_{\mathcal{H}} + \sum_{k=0}^{t} \alpha(k)\| \prod_{l=k+1}^{t} [\tilde{W}(l)]Qg'(V_k, k)\|_{\mathcal{H}}. \tag{6.17}$$

Given assumption 6 is satisfied, from [124], [163] we have that if $W(t)$ is a doubly stochastic matrix with lazy Metropolis weights then:

$$\|W(k)W(k+1)...W(k+\mathcal{B}-1)QX\| \leq \eta\|QX\|, \forall X \tag{6.18}$$

for some matrix or vector X where $\eta = \min\{1 - \frac{1}{2m^3},\ \sup_{k\geq 0} \sigma_2(W(k)))\}$, and $\sigma_2(W(k)))$ denotes the second largest singular value of $W(k)$. Using (6.18) we can then upper bound the products in (6.17) which then allows us to write:

$$\|QV_{t+1}\|_{\mathcal{H}} \leq \eta^{\lfloor (t+1)/\mathcal{B} \rfloor}\|V_0\|_{\mathcal{H}} + \sum_{k=0}^{t} \alpha(k)\eta^{\lfloor (t-k)/\mathcal{B} \rfloor}\|g'(V_k, k)\|_{\mathcal{H}},$$

where in the last step we use $\|Q\| \leq 1$. To facilitate further analysis we let $\zeta = \eta^{\frac{1}{\mathcal{B}}}$. Then by using $\eta^{\lfloor (t+1)/\mathcal{B} \rfloor} \leq \eta^{\lfloor (t+1)/\mathcal{B} \rfloor - 1} \leq \zeta^{t+1}/\eta$ and $\|g'(V_k, k)\|_{\mathcal{H}} \leq \sigma$ from lemma 6.4.2 we can simplify the above inequality. We now set $\alpha(k) = \alpha$, square both sides, and use the Cauchy-Schwartz inequality to arrive at:

$$\|QV_{t+1}\|_{\mathcal{H}}^2 \leq 2\frac{\zeta^{2t+2}}{\eta^2}\|V_0\|_{\mathcal{H}}^2 + \frac{2\sigma^2\alpha^2}{\eta^2(1-\zeta)^2}. \tag{6.19}$$

We will now proceed with combining (6.16) with (6.19). First, using the Cauchy-Shwartz inequality we produce an inequality expressing an upper bound on $\mathbb{E}[\|V_{\mathrm{i},t+1} - V^*\|_{\mathcal{H}}^2]$. We sum both sides of this inequality from $\mathrm{i} = 1$ to m and divide by m:

$$\frac{1}{m}\sum_{\mathrm{i}=1}^{m} \mathbb{E}[\|V_{\mathrm{i},t+1} - V^*\|_{\mathcal{H}}^2] \leq 2\mathbb{E}[\|\bar{V}_{t+1} - V^*\|_{\mathcal{H}}^2] + \frac{2}{m}\mathbb{E}[\|QV_t\|_{\mathcal{H}}^2].$$

Let $D(\alpha, \beta, \lambda_o) = \frac{2A}{\lambda_0} + \frac{2\sigma^2\alpha^2}{m\eta^2(1-\zeta)^2}$, we now upper bound the second term in the above with (6.19), and we upper bound the first term using our result from (6.16).This completes the proof.

6.5 Conclusion

This paper has studied a distributed gradient temporal difference algorithm for policy evaluation using a kerenelized approximation of the value function. The distributed algorithm uses a local weighted average of estimates in order to drive the multi-agent system towards consensus while also driving each agents estimate towards the actual value function V_π. The main result of this paper is that, under certain conditions, if each agent follows the described algorithm then the expected squared error for each agent is bounded and can approach zero as the step-size approaches zero. Furthermore, these bounds are provided for every time step, which completes our goal of conducting a finite-sample analysis. An interesting future direction of this work would be to incorporate policy improvement such as in actor-critic methods, which would follow the work in [155].

6.6 Supplementary

6.6.1 Proof of Lemma 6.4.2

First we analyze the unprojected gradient $g(V_t, t)$. From the definition of $g(V_t, t)$ we have the following:

$$\begin{aligned}\|g(V_t,t)\|_{\mathcal{H}} &= \|z_{t+1}(\kappa(x_t,\cdot) - \gamma\kappa(x_{t+1},\cdot)) + \lambda V_t(\cdot)\|_{\mathcal{H}} \\ &\leq \|z_{t+1}\|\|(\kappa(x_t,\cdot) - \gamma\kappa(x_{t+1},\cdot))\|_{\mathcal{H}} + \lambda\|V_t(\cdot)\|_{\mathcal{H}}.\end{aligned}$$

Using inequality (6.10) and taking the expectation of both sides we simplify the last term to λR. Then using the Law of Total Expectation we have:

$$\mathbb{E}[\|g(V_t,t)\|_{\mathcal{H}}] \leq \mathbb{E}[\|z_{t+1}\| \mid x_t, \boldsymbol{\pi}(x_t)]\mathbb{E}[\|(\kappa(x_t,\cdot) - \gamma\kappa(x_{t+1},\cdot))\|_{\mathcal{H}} \mid \mathcal{F}_t] + \lambda R.$$

Using the property that $\mathbb{E}[Y^2] \geq \mathbb{E}[Y]^2$ and assumptions 3 and 4, we can upper bound the above which gives us our first result: $E[\|g(V_t, t)\|_{\mathcal{H}}] \leq \sigma_z G + \lambda R$.

Now using the result from lemma 6.4.1 and with $\epsilon_{\max} \geq \epsilon(t)$ and $\alpha_{\max} \geq \alpha(t)$ for all t, we then arrive at our second result: $\mathbb{E}[\|g'(V_t, t)\|_{\mathcal{H}}] \leq \sigma_z G + \lambda R + \frac{m\epsilon_{\max}}{\alpha_{\max}} = \sigma$.

With the above results we now focus on the Hilbert norm difference between value functions at one time step and the next. We start by writing out this quantity using (6.6), which gives us:

$$\begin{aligned}\|\bar{V}_{t+1} - \bar{V}_t\|_{\mathcal{H}}^2 &= \|\bar{V}_t + \alpha(t)g'(\bar{V}_t, t) - \bar{V}_t\|_{\mathcal{H}}^2 \\ &\leq 2\alpha(t)^2\|g(\bar{V}_t, t)\|_{\mathcal{H}}^2 + 2\alpha(t)^2\|g'(\bar{V}_t, t) - g(\bar{V}_t, t)\|_{\mathcal{H}}^2,\end{aligned}$$

where the last step makes use of the triangle inequality on the expression $\|(g'(\bar{V}_t, t) - g(\bar{V}_t, t)) + g(\bar{V}_t, t)\|_{\mathcal{H}}^2$. We can then use lemma 6.4.1 to upper bound the second term of the above and then by taking the expectation we arrive at:

$$\mathbb{E}[\|\bar{V}_{t+1} - \bar{V}_t\|_{\mathcal{H}}^2] \leq 2\alpha(t)^2\mathbb{E}[\|g(\bar{V}_t, t)\|_{\mathcal{H}}^2 + 2\epsilon(t)^2.$$

Now we use (6.11) to get an upper bound on $\mathbb{E}[\|g(\bar{V}_t, t)\|_{\mathcal{H}}^2$, and then using the Cauchy-Schwarz inequality we arrive at the final result. This completes the proof of lemma 6.4.2.

6.6.2 Proof of Lemma 6.4.3

We start by looking at the squared average error:

$$\|\bar{V}_{t+1} - V^*\|_{\mathcal{H}}^2 = \|\bar{V}_t - \alpha(t)g'(\bar{V}_t, t) - V^*\|_{\mathcal{H}}^2.$$

We expand out the above product and then add and subtract the functional stochastic gradient $g(\bar{V}_t, t)$ giving us:

$$\begin{aligned}\|\bar{V}_{t+1} - V^*\|_{\mathcal{H}}^2 =& \|\bar{V}_t - V^*\|_{\mathcal{H}}^2 - 2\alpha(t)\langle\bar{V}_t - V^*, g(\bar{V}_t, t)\rangle_{\mathcal{H}} \\ &- 2\alpha(t)\langle\bar{V}_t - V^*, g'(\bar{V}_t, t) - g(\bar{V}_t, t)\rangle_{\mathcal{H}} + \alpha(t)^2\|g'(\bar{V}_t, t)\|_{\mathcal{H}}^2.\end{aligned}$$

Then using lemma 6.4.1 and the Cauchy-Schwarz inequality we can upper bound the third term above by $2\epsilon(t)\|\bar{V}_t - V^*\|_{\mathcal{H}}$. For convenience let $\mathbb{E}[\bar{\delta}_t] = \mathbb{E}[\bar{\delta}_t \mid x_t, \boldsymbol{\pi}(x_t]$. We now remind ourselves that that stochastic quasi-gradient g is also a function of z by writing $g(\bar{V}_t, \bar{z}_{t+1}, t)$, and then we add and subtract $g(\bar{V}_t, \mathbb{E}[\bar{\delta}_t], t)$, to the above equation which gives us:

$$\begin{aligned}\|\bar{V}_{t+1} - V^*\|^2_{\mathcal{H}} \leq & \|\bar{V}_t - V^*\|^2_{\mathcal{H}} - 2\alpha(t)\langle \bar{V}_t - V^*, g(\bar{V}_t, \mathbb{E}[\bar{\delta}_t], t)\rangle_{\mathcal{H}} \\ & + 2\epsilon(t)\|\bar{V}_t - V^*\|_{\mathcal{H}} + \alpha(t)^2\|g'(\bar{V}_t, \bar{z}_{t+1}, t)\|^2_{\mathcal{H}} \\ & + 2\alpha\langle g(\bar{V}_t, \bar{z}_{t+1}, t) - g(\bar{V}_t, \mathbb{E}[\bar{\delta}_t, t], \bar{V}_t - V^*\rangle. \end{aligned} \tag{6.20}$$

Let $v(t) = 2\alpha(t)\langle(\bar{z}(t+1) - \mathbb{E}[\bar{\delta}_t])(\kappa(x_t, \cdot) - \gamma\kappa(x_{t+1}, \cdot)), \bar{V}_t - V^*\rangle$, and now using the definition of g we can then simplify the last term of (6.20) to $v(t)$ by canceling out like terms. We then compute the expectation on both sides of the above inequality, conditional $\mathcal{F}_t$ which gives us:

$$\begin{aligned}\mathbb{E}[\|\bar{V}_{t+1} - V^*\|^2_{\mathcal{H}} \mid \mathcal{F}_t] \leq & \|\bar{V}_t - V^*\|^2_{\mathcal{H}} - 2\alpha(t)\langle \bar{V}_t - V^*, \mathbb{E}[g(\bar{V}_t, \mathbb{E}[\bar{\delta}_t], t) \mid \mathcal{F}_t]\rangle_{\mathcal{H}} \\ & + 2\epsilon(t)\|\bar{V}_t - V^*\|_{\mathcal{H}} + \alpha(t)^2\sigma^2 + \mathbb{E}[v(t) \mid \mathcal{F}_t]. \end{aligned} \tag{6.21}$$

where because of assumption (1) we have that $\bar{V}_t$ is independent on the gradient given $\mathcal{F}_t$. We have also made use of lemma 6.4.2 to upper bound the second to last term. We now note that the regularization term $(\lambda/2)\|V\|^2_{\mathcal{H}}$ in our objective function implies that $J(V)$ is λ-strongly convex in $V \in \mathcal{H}$ since the Hessian of $J(V)$ is lower bounded by λ. We can then substitute the second term in (6.21) with $2\alpha(t)\lambda\|\bar{V}_t - V^*\|^2_{\mathcal{H}}$, which give us:

$$\begin{aligned}\mathbb{E}[\|\bar{V}_{t+1} - V^*\|^2_{\mathcal{H}} \mid \mathcal{F}_t] \leq & \|\bar{V}_t - V^*\|^2_{\mathcal{H}} - 2\alpha(t)\lambda\|\bar{V}_t - V^*\|^2_{\mathcal{H}} \\ & + 2\epsilon(t)\|\bar{V}_t - V^*\|_{\mathcal{H}} + \alpha(t)^2\sigma^2 + \mathbb{E}[v(t) \mid \mathcal{F}_t]. \end{aligned} \tag{6.22}$$

Let us now momentarily focus on $v(t)$. By using the Cauchy-Schwartz inequality and its corollary $2ab \leq \rho a^2 + b^2/\rho$ for $\rho, a, b > 0$, we can simplify $v(t)$. Then by taking the expectation of $v(t)$ and using assumption 3, we arrive at:

$$\mathbb{E}[v(t) \mid \mathcal{F}_t] \leq \beta(t)\mathbb{E}[(\bar{z}_{t+1} - \mathbb{E}[\bar{\delta}_t])^2 \mid \mathcal{F}_t] + \frac{\alpha(t)^2}{\beta(t)} G^2 \|\bar{V}_t - V^*\|_{\mathcal{H}}^2.$$

Finally, we use this upper bound for $\mathbb{E}[v(t) \mid \mathcal{F}_t]$ in (6.22). This completes the proof.

6.6.3 Proof of Lemma 6.4.4

For brevity we will only give an outline of the proof. We start by defining a scalar quantity $\mathrm{e}(t)$ as $\mathrm{e}(t) = (1 - \beta(t))(\mathbb{E}[\bar{\delta}_t] - \mathbb{E}[\bar{\delta}_{t-1}])$. We then analyze the evolution of $\mathbb{E}[(\bar{z}_{t+1} - \mathbb{E}[\bar{\delta}_t] + \mathrm{e}(t))^2 \mid \mathcal{F}_t]$. We use assumption 4 to upper bound terms containing $\mathbb{E}[(\bar{\delta}_t - d\mathbb{E}[\bar{\delta}_t])^2]$, and note that some terms go to zero in expectation. This gives us the following:

$$\mathbb{E}[(\bar{z}_{t+1} - \mathbb{E}[\bar{\delta}_t] + \mathrm{e}(t))^2 \mid \mathcal{F}_t] \leq (1 - \beta(t))^2(\bar{z}_t - \mathbb{E}[\bar{\delta}_{t-1}])^2 + \beta(t)^2\sigma_\delta^2. \tag{6.23}$$

Then we use Lipschitz continuity of $\mathbb{E}[\delta_{\mathrm{i}}(t)]$ with respect to $V(t)$ from assumption 5 to derive the following upper bound on $\mathrm{e}(t)$:

$$|\mathrm{e}(t)| = (1 - \beta(t))\|\mathbb{E}[\bar{\delta}_t] - \mathbb{E}[\bar{\delta}_{t-1}]\| \leq (1 - \beta(t))L\|\bar{V}_t - V_{t-1})\|_{\mathcal{H}}. \tag{6.24}$$

Finally, we use the following result $\|a + b\|^2 \leq (1 + \rho)\|a\|^2 + (1 + \frac{1}{\rho})\|b\|^2$, which holds for any $\rho > 0$, to derive an upper bound for $\mathbb{E}[(\bar{z}_{t+1} - \mathbb{E}[\bar{\delta}_t])^2 \mid \mathcal{F}_t]$ in terms of $\mathbb{E}[(\bar{z}_{t+1} - \mathbb{E}[\bar{\delta}_t] + \mathrm{e}(t))^2 \mid \mathcal{F}_t]$ and $\mathrm{e}(t)$.

$$\mathbb{E}[(\bar{z}_{t+1} - \mathbb{E}[\bar{\delta}_t])^2 \mid \mathcal{F}_t] \leq (1 - \beta(t))(\bar{z}_t - \mathbb{E}[\bar{\delta}_{t-1}])^2 + 2\beta(t)^2\sigma_\delta^2 + \frac{\beta(t) + 1}{\beta(t)}\mathrm{e}(t)^2.$$

The final result then follows by replacing $\mathbb{E}[(\bar{z}_{t+1} - \mathbb{E}[\bar{\delta}_t] + \mathrm{e}(t))^2 \mid \mathcal{F}_t]$ and $\mathrm{e}(t)$ with their upper bounds expressed in (6.23) and (6.24), respectively.

PART III

Applications

7. LEARNING-ASSISTED LOAD CONTROL DESIGN FOR TRANSACTIVE ENERGY SYSTEM

Introduction

Transactive energy has emerged as a paradigm for demand-side control in power grids [67]–[69]. A transactive energy system is usually concerned with the coordination and control of a group of distributed energy resources such as smart loads, distributed generations, and even energy storage. It can be modeled by a multi-agent system with three different types of agents: coordinator, supplier, and customer, where the coordinator represents the market operator, the supplier is the electricity seller, and the customer is the electricity buyer [165]. The underlying coordination and control have a hierarchical structure which consists of two decision-making levels including resource level and supervisory level[166].

There are some works that have applied RL aglorithms to transactive energy problems, such as in [53] where a modified version of DDPG is proposed for strategic bidding in electricity markets, and in [167] where a DQN is proposed for HVAC control. Similiarly, in [168] a multi-zone HVAC control problem is tackled using DQN with a heuristic mechanism for dealing with large action space, while in [169] an actor-attention-critic algorithm is used for a multi-agent formulation of the multi-zone HVAC control problem. Though these results are promising and validate the applicability of RL for HVAC control in TES, they do not compare multiple algorithms against each other. Such a comparison would allow us to see the benefits and drawbacks of using one algorithm over the other for this particular task.

To our knowledge, there are no works that compare different reinforcement learning approaches when applied to transactive energy systems. There are works comparing DQN and DDPG, and other policy gradient algorithms , but they are studied on specific use cases, such as UAV control, cybersecurity, networking, data mining, robotics, and power grid voltage control [4], [48]–[52], which are not in our field of interest, or on either toy examples or games [46], [54]–[56]. In [57] a comprehensive comparison of temporal difference algorithms is given but they are applied to examples such as the cartpole problem or the 20 link pole balancing problem, which, though of high dimensionality and difficulty, do not capture the multi-objective and market-based features that characterize the transactive energy en-

vironment. There are some works that have applied RL aglorithms to transactive energy problems, such as in [53] where a modified version of DDPG is proposed for strategic bidding in electricity markets, and in [167] where a DQN is proposed for HVAC control. Similiarly, in [168] a multi-zone HVAC control problem is tackled using DQN with a heuristic mechanism for dealing with large action space, while in [169] an actor-attention-critic algorithm is used for a multi-agent formulation of the multi-zone HVAC control problem.

In order for transactive control to be effectively realized there is a need for the coordinator to aggregate the load demand of all the customers. This requires that all consumers send a representation of their individual demand curves to the coordinator. Besides the issue of potentially large communication costs from sending demand curves to the coordinator, there is also the issue that each consumer would need to accurately compute a mathematical representation of their demand. Consumers would need to accurately model the load dynamics of their loads, as well as their specific user preferences. This presents a challenge for practical deployment of transactive load control because individual consumer may not have an accurate model for its load or its user preference in mathematical form.

Because the issues of unknown user preferences and difficulty in modeling dynamics of specific loads, there is a need to develop methods that do not require demand curves of each individual or mathematical models of dynamics. This motivates our exploration of model-free methods such as reinforcement learning algorithms. Reinforcement learning can achieve optimal control of systems by learning from observed data that is created from interacting with the environment, where the environment consists of all components that are not controllable. An important part of the data is feedback in the form of a rewards, which determines what type of control inputs or actions are favorable for a given state of the environment. In our case, the reward function will be influenced by user preferences, and the influence of user-preference can be abstracted to the value of a single parameter. This is better than model-based methods where there might be many parameters that the user will have to determine, making it difficult to design for a wide range of users.

7.1 Problem Statement

In this work we consider a transactive energy system that is made up of a network of residential air conditioners (ACs) at the resource level, and a central coordinator and supplier at the supervisory level. Since we are interested in large network of small power loads, the market is close to perfectly competitive, where the market clearing price, as determined by the coordinator, is independent of any **individual** resource level decisions. As such, the resource level individuals are price takers and do not consider decisions of other individuals in their own decision making. Moreover, we focus on the control decisions of one individual, which we call the agent, where the control goal of the agent is to minimize the thermal comfort and energy costs of the agent.

To this end, the agent controls the AC by giving On/Off commands, denoted $u(t)$, at the beginning of every market period t. The command $u(t)$ is determined as follows: The agent decides on a price bid λ_{bid}, then once the agent receives the market clearing price λ_{clear} it turns on the AC for the whole market period if its price bid is greater than or equal to the clearing price, otherwise the agent turns the AC off. Or more formally:

$$u(t) = \begin{cases} \text{On}, \text{ if } \lambda_{\text{bid}}(t) \geq \lambda_{\text{clear}}(t) \\ \text{Off}, \text{ otherwise} \end{cases}$$

The dynamics of the AC can be described by the following equivalent thermal parameter (ETP) model [170] with both discrete and continuous states,

$$\dot{T}_a = \frac{H_m}{C_a}(T_m - T_a) + \frac{U_a}{C_a}(T_o - T_a) + \frac{Q}{C_a} \tag{7.1a}$$

$$\dot{T}_m = \frac{H_m}{C_m}(T_a - T_m) + \frac{Q}{C_m} \tag{7.1b}$$

with

$$Q(t) = \begin{cases} Q_{\text{i}} + Q_s + Q_h & \text{if } u(t) = \text{On} \\ Q_{\text{i}} + Q_s & \text{if } u(t) = \text{Off}, \end{cases}$$

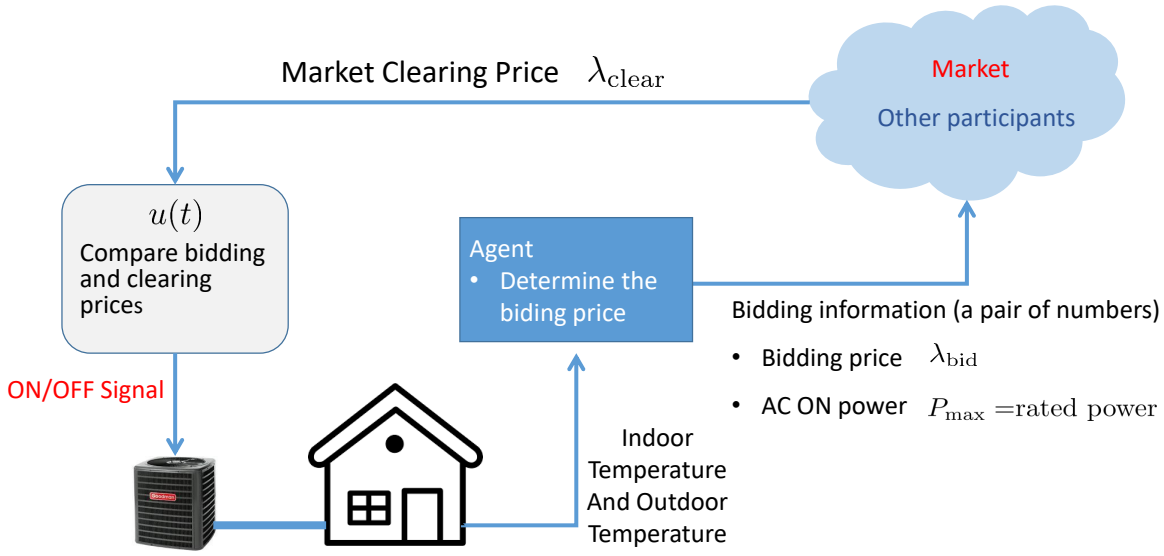

Figure 7.1. Agent Feedback

where T_a, is the indoor air temperature, T_m is the inner mass temperature (due to the building materials and furnishings), U_a is the conductance of the building envelope, T_o is the outdoor air temperature, H_m is the conductance between the inner air and inner solid mass, C_a is the thermal mass of the air, C_m is the thermal mass of the building materials and furnishings, Q is the total heat flux consisting of the heat gain from the internal load Q_{i}, the solar heat gain Q_s and the heat gain from the heating/cooling system Q_h.

With the above structure, the control problem of the agent is to compute λ_{bid} at every market period t.

7.2 Learning-assisted Bidding Strategy

In previous works such as in [165] and the AEP gridSMART project, a bidding strategy is designed based on models of the building temperature dynamics, such as (7.1), and models of user preferences. This approach, however, would be difficult to apply to the realistic case of large networks with heterogeneous consumers, since this would require the impractical task of developing good models for every consumer, which itself potentially operates various appliances with complex power dynamics. With this in mind we are motivated to develop a model-free approach, and so due to its model-free nature and its success in literature we

explore reinforcement learning for the purpose of training intelligent consumer agents to produce optimal bidding strategies.

As such, we consider a learning agent that can only use data about the state $s(t)$ of its environment at the current time step, i.e. market period, t and a reward signal $r(t)$, as well as a finite amount of historical data. We define the state $s(t)$ of the environment as $s(t) = [T_{\text{in}}(kt), T_{\text{out}}(t)]$, where $T_{\text{a}}(t)$ is the indoor temperature and $T_{\text{out}}(t)$ is the outdoor temperature at t. In accordance with the goals of minimizing thermal discomfort and energy cost, the reward $r(t)$ is defined as:

$$r(t) = -wU\left(\Delta T(t)\right) - (1-w)C(P_{\text{avg}}, t) \tag{7.2}$$

where $C(P_{\text{avg}}, t)$ is the cost of energy consumption, $U(\Delta T(t))$ represents the disutility from thermal discomfort, $\Delta T(t) = \bar{T}_{\text{a}}(t) - T_{\text{desired}}$, $\bar{T}_{\text{in}}(t)$ is the average indoor temperature over the market period t, $P_{\text{avg}}(t)$ is the average power consumed over the market period t, and w is a weighting factor.

The goal of the learning agent is to develop an optimal price bidding strategy $\boldsymbol{\pi}(s) : s \rightarrow \lambda_{\text{bid}}$, which outputs a price bid for every state s and maximizes the rewards received, using only data gathered from its environment. The overall feedback loop of the agent with its environment (market and building) is shown in 7.1.

We note that some previous works on developing learning-assisted strategies have used a tabular RL algorithm [171], which required the discretization of the action and state space. This discretization can become impractical as the action and state space becomes large, which would then require an impractical amount of memory in order to save values for each action. In the RL literature this issue is often referred to as the "curse of dimensionality".

In light of this issue, along with tabular forms we will also explore deep reinforcement learning. Deep reinforcement learning makes use of deep neural networks to approximate key functions such as the policy and/or value functions. The main benefit of this is that it allows us to use RL methods on continuous state and action spaces without the need for discretization, thereby addressing the issues with large state and action spaces.

We now introduce the reinforcement algorithms of interest, which we will apply to the problem of learning an optimal price bidding policy.

7.2.1 Tabular Q-learning

The Q-learning algorithm introduced in [7], was developed for environments with discrete action and state spaces. As such, the Q function in traditional Q-learning is represented as a data table of values, and so it is sometimes referred to as tabular Q-learning. The Q-learning algorithm provides a way to update the the entries of this table so that it can better represent the values for the optimal policy. Once the values converge to the optimal values, the table can be used to look up what is the best action for a given state. The algorithm is given by the following equation:

$$Q(s_t, a_t) = Q(s_t, a_t) + \alpha(r_t + \gamma \max_u Q(s_{t+1}, u) - Q(s_t, a_t))$$

where $\alpha \in (0, 1)$ is the learning rate and $\gamma \in (0, 1)$ is a discount factor. Both of these parameters are user-defined. In order to guarantee convergence of the algorithm, it is necessary that all actions and states are visited infinitely many times. This is accomplished by using an exploratory policy to collect data, instead of always taking the greedy action. As with most reinforcement learning algorithms, the need for exploration will also be noted in the next two algorithms.

7.2.2 DQN

We will look at a deep Q-learning approach, which is called Deep Q-Network (DQN). With this approach the Q-function, which in traditional Q-learning is represented as a table of values, will now be represented by a neural network. Once this neural network has been appropriately trained using simulated data and the DQN algorithm, it can then be used to construct a policy similar to [171] where the policy outputs a price bid given an indoor temperature. The main advantage of using the DQN approach is that it does not require the discretization of the feasible space of indoor temperature values, which allows it to scale

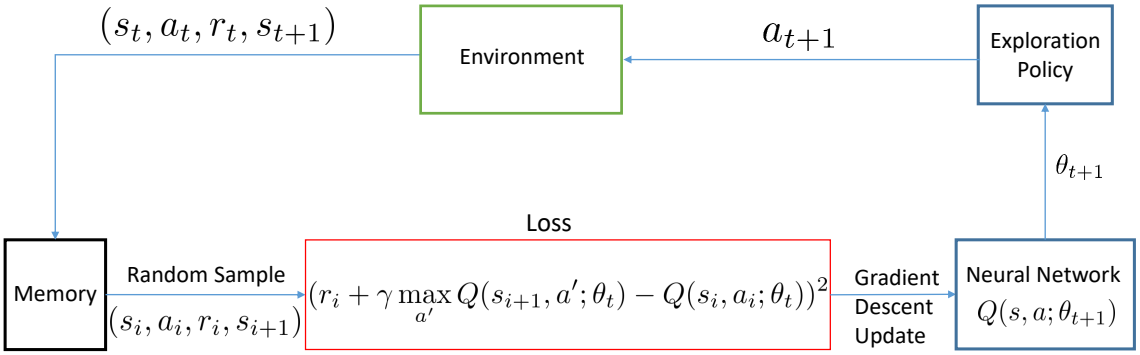

Figure 7.2. DQN Algorithm

better. This neural network can be trained to map the state of the system to a vector of values where each element of the vector corresponds to a discrete action. In this way the best action can be computed by finding the index of the largest value of the output vector.

The overall algorithm is illustrated in Fig. 7.2. First the agent observes a data tuple (s_t, a_t, r_t, s_{t+1}) which is then saved in the agent's memory, where s_t,a_t, and r_t refer to the state, action taken, and reward received at time step t. Then the agent randomly selects a data tuple and this data tuple is used to calculate the gradient of the DQN loss function. The loss function is shown in Fig. 7.2. This gradient is used to update the weights θ_t of the neural network $Q(s, a; \theta_t)$ through a method known as "backpropagation", which is essentially a gradient descent update that uses the chain rule to propagate backwards from the output layer of the neural network. Once the weights have been updated they are used to update the exploration policy, and then finally the updated exploration policy is used to determine the next action. This action is applied to the environment which responds with a new data tuple, thereby continuing the learning cycle.

7.2.3 DDPG

DDPG is a deep learning reinforcement learning algorithm that belongs to the family of policy gradient algorithms and uses an actor-critic structure. The algorithm is similar to DQN, in the DDPG algorithm we represent the Q value function as a neural network, which is referred to as the critic, but DDPG incorporates a second neural network to represent the

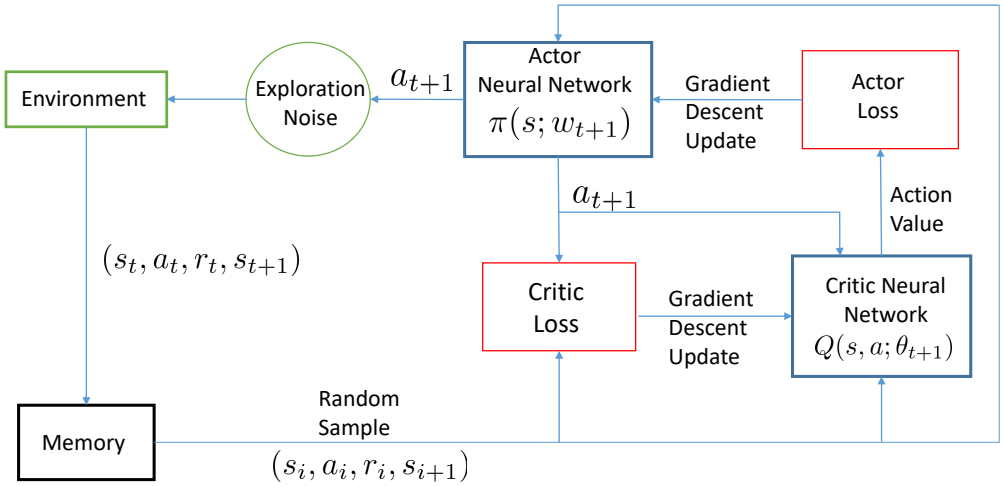

Figure 7.3. DDPG Algorithm

learned policy, referred to as the actor. This addition is needed in order for the algorithm to be applicable for environments that have continuous action spaces, which is not the case for DQN. As such, DDPG is a direct extension of DQN from environments with discrete action spaces to environments with continuous action spaces. We can see the over all structure of the DDPG algorithm in Fig. 7.3. From 7.3, the critic loss for one sample L_{i} is $L_{\mathrm{i}} = (r_{\mathrm{i}} + \gamma Q(s_{\mathrm{i}+1}, a_{t+1}; \theta_t) - Q(s_{\mathrm{i}}, a_{\mathrm{i}}; \theta_t))^2$ and the actor loss for one sample J_{i} is $J_{\mathrm{i}} = Q(s_{\mathrm{i}}, \boldsymbol{\pi}(s_{\mathrm{i}}; w); \theta_t)$. We note that the loss from one sample does not necessarily determine the update for the actor and critic. This is because usually a small batch of samples (called minibatch) is randomly selected from memory to compute multiple losses and multipe gradients of those losses. And so the resulting update will be in the direction of the average of the these gradients. We note the exploration noise block in 7.3, which represents the addition of noise to the calculated greedy action a_{t+1}. This is needed so that the learning agent explores the state space and action space, otherwise the agent could get stuck on a local optimum that is far from the global optimum.

7.3 Case Studies

In this section we will explore case studies that evaluate the algorithms of interest in different contexts. We evaluate them in terms of their control performance and training per-

formance, using metrics such as sample complexity, random seed sensitivity, and robustness to increasingly non-competitive markets given that agents are trained with the assumption of competitive markets. We first describe details regarding our simulations and learning implementation.

In order to emulate a nearly competitive environment, we created a simulated TES environment of 1000 houses where each house has its own user preferences. We confirmed that 1000 houses was enough to approach a competitive market by conducting a price sensitivity analysis. This involved running simulations where we picked a house and fixed its bidding price throughout the day, recorded the clearing prices, and repeated with a different bidding price for the selected house. The results are shown in Fig. 7.4. They indicate that throughout most of the day the clearing prices are not affected by the differences in the biding price of individual house, and that prices are only slightly affected during hours of peak consumption.

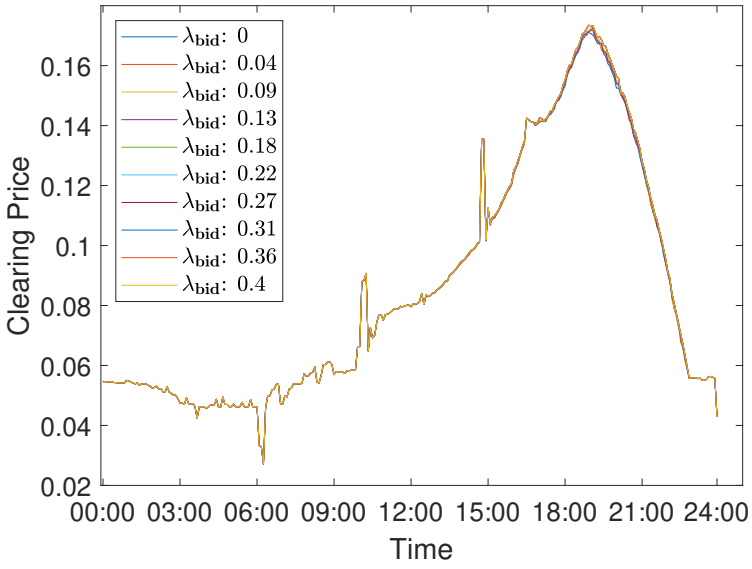

Figure 7.4. Price Sensitivity

For the first two case studies, we trained three agents (one for each algorithm of interest) using a modified version of the simulated TES environment. As part of setting up this training environment, we first collected the clearing prices created by simulating 1000

houses interacting with a central supplier, where each house uses a simple linear model for determining a bidding priced based on the inside temperature. Since we concluded that 1000 houses approaches a competitive environment, we used these precomputed clearing prices instead of simulating the market during agent training. This significantly reduced the time complexity of our simulations and so increased the speed of training. It is important to note that these clearing prices were not given as inputs to the agent, since that would not be realistically accessible to an agent. We trained these agents over 250 episodes using our training environment, where for our problem an episode is defined as one day containing 288 market periods. In addition, each market period contains ten 0.5 minute intervals in which we simulate the evolution of indoor temperatures given the control input set at the beginning of the market period.

In order to create an environment that is compatible with the reinforcement learning algorithms, we also implement a suitable reward function based on equation (7.2). For all the case studies, the form of the reward function is defined by $U(\Delta T(t)) = a\left(\Delta T(t)\right)^2$, and $C(P_{\text{avg}}(t), t) = \lambda_{\text{clear}}(t)P_{\text{avg}}(t)/12$. We adopt the disutility function used in a similar work [172], and a is chosen as $1/240$ with unit $\$/\left(^\circ F\right)^2$ to ensure that the disutility term and cost term are of similar magnitude and the same unit; the cost term has factor $1/12$ since 5 min is $1/12$ hour. Thus, our reward function becomes:

$$r(t) = -wa\left(\Delta T(t)\right)^2 - (1-w)\lambda_{\text{clear}}(t)P_{\text{avg}}(t)/12.$$

To compare the different learning algorithms we use $w = 0.5$ since it balanced disutility minimization and cost minimization, but for our study of uncompetitive markets we in addition use $w = 0.2$ so that we can have a better idea of how the costs affect the performance of the agent.

We now describe some of the details specific to the implementation of the learning algorithms. The neural network architecture and algorithm hyperparameters for the DDPG and DQN agents are manually tuned so that they could achieve good training performance. In order to make the algorithms more comparable, we use a similar neural network architecture for the critic of DDPG and DQN, and also use a neural network of similar magnitude for

the actor of DDPG. In order to describe neural network architecture we use the following notation: $A \times B \times C$. This describes a neural network with one hidden layer, where A is the number of inputs, B is the number of nodes in the hidden layer, and C is the number of outputs. With this notation in mind, we use a $2 \times 150 \times 200 \times 1$ neural network for DQN agents and a $2 \times 150 \times 200 \times 1$ neural network for the critic of DDPG agents and $2 \times 100 \times 200 \times 1$ neural network for the actor of DDPG agents. In terms of hyper parameters, for DDPG and DQN agents we use a mini-bath size of 50, target update frequency of 100, discount factor of 0.99, and learning rate of 0.01 for DQN and DDPG critic but use $1e-4$ for DDPG actor. We note that we use a slightly modified version of DQN called Double DQN, which improves that stability of DQN.

In order to use the tabular Q-learning algorithm, we need to decide on a proper range and discretization for the state space and action space. Since the discretization determines the size of the Q value table, this is essentially analogous to determining a proper neural net architecture. As such, tuning the discretization and range is one of the most important factors to the performance of the algorithm and its sample complexity. After some trial and error, we fixed the discretization such that the indoor temperatures lied in the range (60,78) with 10 discrete values , outdoor temperatures lied in (63,92) with 5 discrete values, and bidding price lied in (0,0.18) with 19 discrete values.

We now present our empirical results from implementing the discussed algorithms. We first demonstrate the control performance of agents trained by each of the three reinforcement learning algorithms. Then we compare the performance of the learning algorithms for our particular environment, using metrics that are widely used in the reinforcement learning literature to compare algorithms.

7.3.1 Control Performance

In this section we demonstrate how well each agent, which was trained by a different learning algorithm, performed on the control task of interest. In particular we look at the rewards received throughout a week of interacting with the simulated TES environment. Since temperature control is a major part of maximizing rewards, we also demonstrate how

well each agent maintains the indoor temperature close to the user defined desired temperature. We also show the price bids of each agent over a day and compare it to the evolution of the outside temperature. This allows us to see how price-bidding behaviour is affected by outside temperature, and demonstrates the differences in bidding behaviour of each agent. Lastly, we test the DDPG agents on their robustness to non-competitive environments.

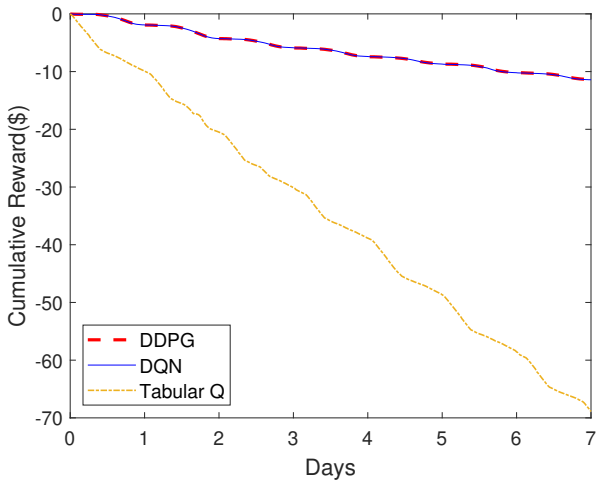

Figure 7.5. Cumulative Reward Over Week. DDPG had a runtime of 2.25 hours and DQN took 2.64 hours. For Q tab we were not able to measure the runtime but it ran for around a day

In Fig. 7.5 we can see that the DQN and DDPG agents achieve higher rewards over the entire week than tabular Q-learning. In fact, DDPG and DQN outperform tabular Q-learning for every day of the week. This is to be expected because the environment's state space is continuous, which makes it more suitable for DDPG and DQN.

In addition, by taking a closer look at the DDPG and DQN agents, we can see from Fig. 7.6 that both agents perform similarly over the entire week. Though by a small margin, we can also see that the DDPG agent achieves a higher cumulative reward by the end of the week. We next present results for Day 2 in Fig. 7.7 and Fig. 7.8. We choose Day 2 since it has the highest temperatures of the week. In Fig. 7.7 it can be seen that DDPG and DQN agents perform similarly and they both outperform tabular Q-learning. In Fig. 7.8 we see that DDPG and DQN agents maintain the indoor temperature close to the desired temperature throughout the day. We also note that for all agents the indoor temperature

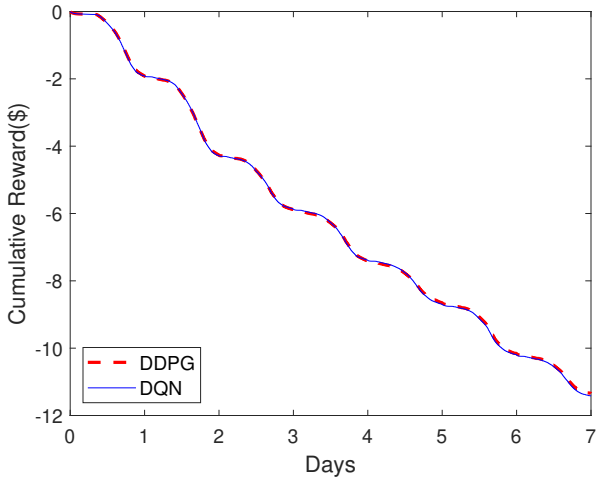

Figure 7.6. Cumulative Reward Over Week(Only DQN and DDPG)

trajectories oscillate with high frequency since we use on/off control mechanism for our system. With a different control mechanism that could for example reduce AC power as needed, we could expect to see smoother trajectories for the indoor temperatures.

In Fig. 7.9 we compare the trajectories of price bids of each agent. One observation we can make is that the trajectories of all agents oscillate with high frequencies throughout the day, and that the amplitude of these oscillations grows larger as the outside temperature increases. As with indoor temperature trajectories, high oscillations can be explained by the on/off control mechanism of the system. The tendency of the amplitudes to increases with outside temperatures, on the other hand, is likely due to the correlation between clearing prices and outside temperatures. We also observe that for all agents, as they increase their high price bids, they also increase their low price bids. Intuitively, we would expect the low price bid to always be zero in order to make sure we are below the clearing price. But since this was not incentivized by the given reward function, none of the agents learned this behavior. It is also important to note that the action space for DQN is constrained to be in the range $(0, 0.35)$ in order to stabilize the learning process. The DDPG algorithm showed stable learning with the full action space $(0, 1)$, and we constrain it to $(0, 0.35)$ so that it would be a fair comparison with DQN. We further constrain the action space in the range $(0, 0.18)$ for tabular Q-learning. These ranges were chosen by using our knowledge of the

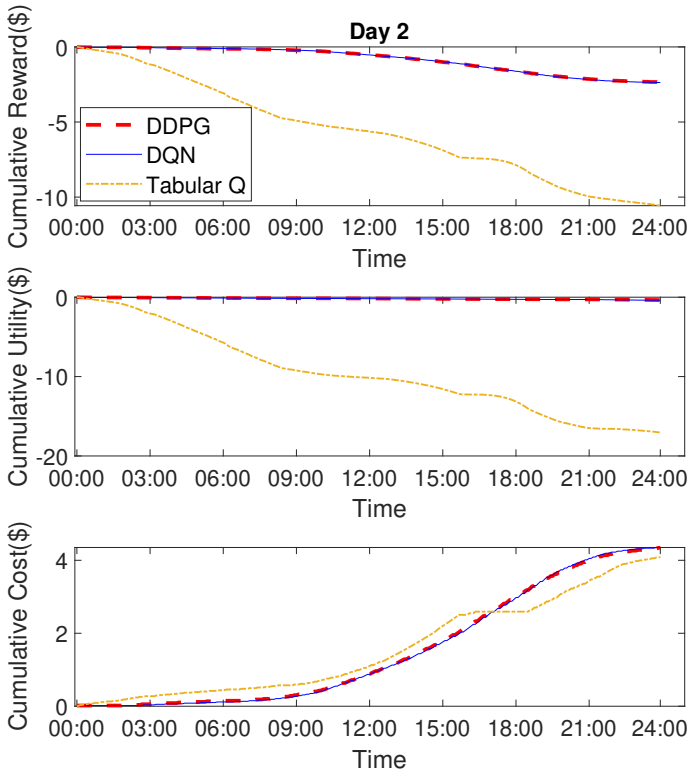

Figure 7.7. Cumulative Rewards for Day 2

clearing prices, which was found to be closely bounded by $(0, 0.18)$. In practical settings, we could make use of historical prices to reasonably constrain the action space if clearing prices are not known beforehand.

7.3.2 Training Performance

In this section we illustrate the performance of each algorithm in terms of their average reward/episode reward. These metrics are commonly used in empirical studies of reinforcement learning algorithms [54], [173], [174]). In [54] and [175] it is emphasized that in order to better evaluate learning algorithms it is important to run many training trials with different random seeds. This would allows us to test the variability of a particular algorithm with respect to the random seeds.

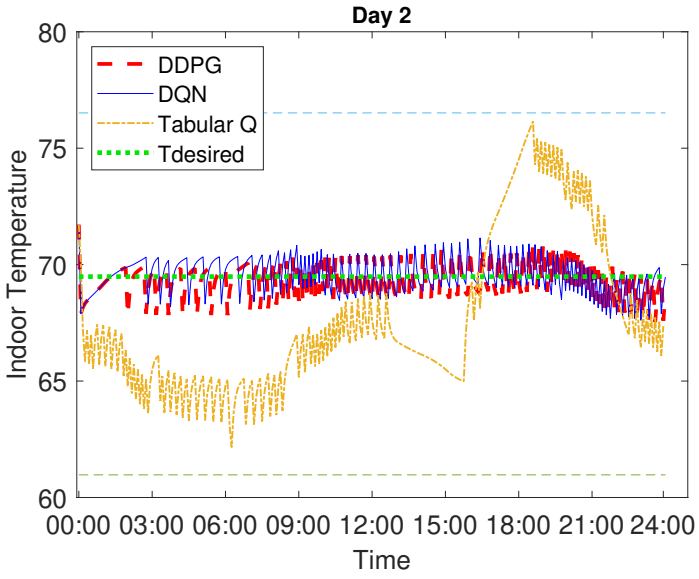

Figure 7.8. Indoor Temperature for Day 2

In Fig. 7.10 and Fig. 7.11 we see that the DDPG algorithm has higher variability in the initial phases of training compared to the DQN algorithm. On the other hand, we also see that the DQN algorithm experienced occasional drops in performance that persisted throughout the training process. These drops in performance are not seen as often nor with the same magnitude in the DDPG training process (except for the first 50 episodes). This indicates that the DDPG algorithm has better long-term stability than DQN for our use case. This might be partly due to the fact that exploration can decay to zero with the DDPG algorithm without triggering divergence during training, while the DQN algorithm requires a minimum amount of exploration throughout training otherwise the algorithm can become unstable. This last point is important because, given a fixed set of hyperparameters, this would also allow DDPG to get closer to the globally optimal policy, whereas DQN will be occasionally pushed away from the globally optimal policy because of the need for a minimum amount of exploration.

In Fig. 7.12 and Fig. 7.13 we can see the training performance of the tabular Q-learning algorithm. We see immediately that the algorithm suffers from high sample complexity, where over 5000 episodes are needed in order for the algorithm to start converging. This is the reason why we have decided to plot the Q-learning training progress separate from

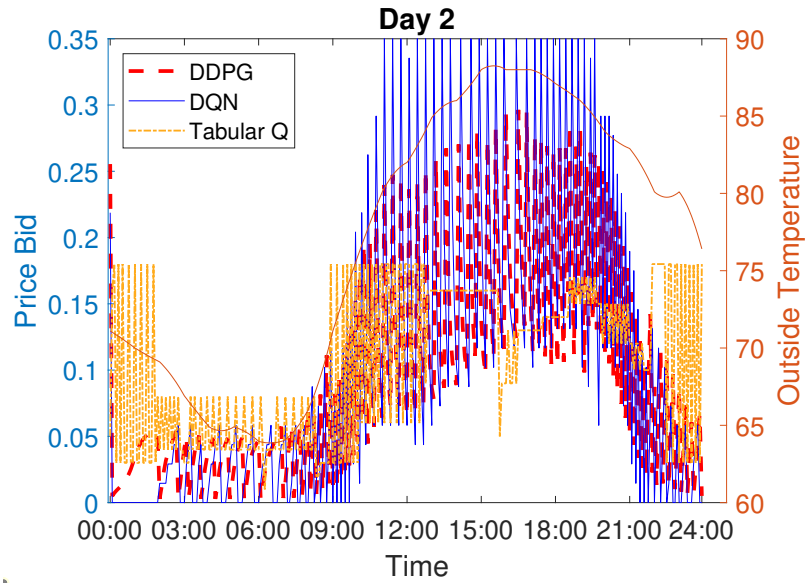

Figure 7.9. Price Bids for Day 2

DQN and DDPG. In addition, we see that Q-learning converges at episode reward of around -15, whereas the DDPG and DQN converge closer to -10. Overall, it seems that DQN and DDPG both perform a lot better than Q-learning for the problem of interest and with a much better training efficiency, in terms of number of episodes needed to converge.

7.3.3 Performance in Non-Competitive Market

Here we show how DDPG agents perform when the market is non-competitive, i.e. the clearing price is affected by individual price bids. We only look at DDPG agents because we can no longer assume a lower upper bound on the actual clearing prices, which greatly increases the discrete action space of DQN agents making them unsuitable for non-competitive markets. As such, we train two DDPG agents on a competitive environment with full bidding price range $(0, 1)$ instead of the smaller range of $(0, 0.35)$ used for comparisons with DQN. One agent is trained with weighting factor $w = 0.5$ and the other is trained with $w = 0.2$. We did this in order to test how the value of w can affect performance, particularly since lower values of w will emphasize the cost of bidding and so this should highlight the affects of a non-competitive market. After training, the agents are then evaluated on a

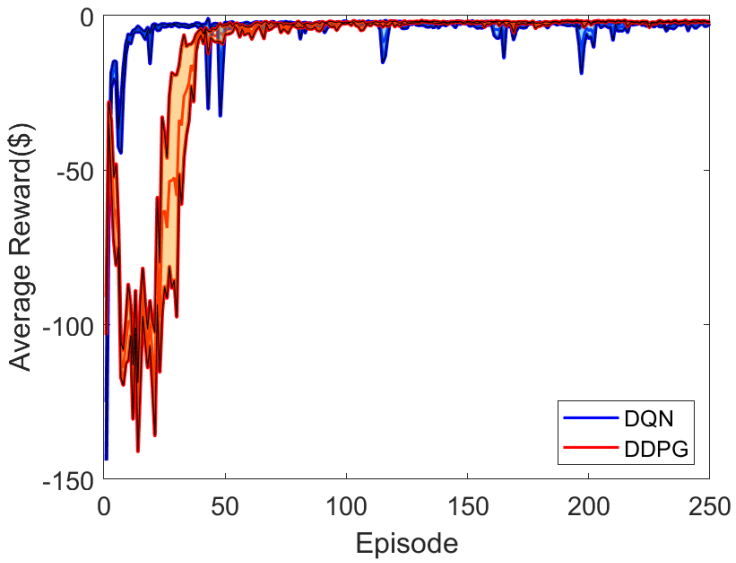

Figure 7.10. Range of average episode rewards received during training for 10 trials of DQN and DDPG with different random seeds. Rewards are averaged every 10 episodes. Shaded region indicates one standard deviation from mean.

different environment that incorporates a non-competitive market. This simply means that an extra function was called to compute the clearing price at the beginning of every market period. As such, this function collects approximations of the agent's demand curve, which is constructed from the price bid and power ratings, as well as the demand curves of other simulated houses. Once these demand curves are all gathered, a suitable clearing price is calculated based on the feeder capacity limits and base price of the supplier. Using this pricing mechanism, we test the agent for environments containing 1000 houses, 500 houses, 100 houses, and 50 houses. These set of tests will allow us to determine how sensitive the agents are to increasingly less competitive markets. Reinforcement learning theory tells us this decrease in market competition should lead to bad performance since it breaks the underlying assumption of a stationary and fully observable environment [29].

The results illustrated in Fig. 7.14 indicate that despite the partial observability of the environment, i.e. lack of knowledge of other houses' actions, the DDPG agents still perform well when compared to our baselines. We also note that there is a drop in performance as the number of houses decreases. The baselines are agents that use a linear piece-wise function as

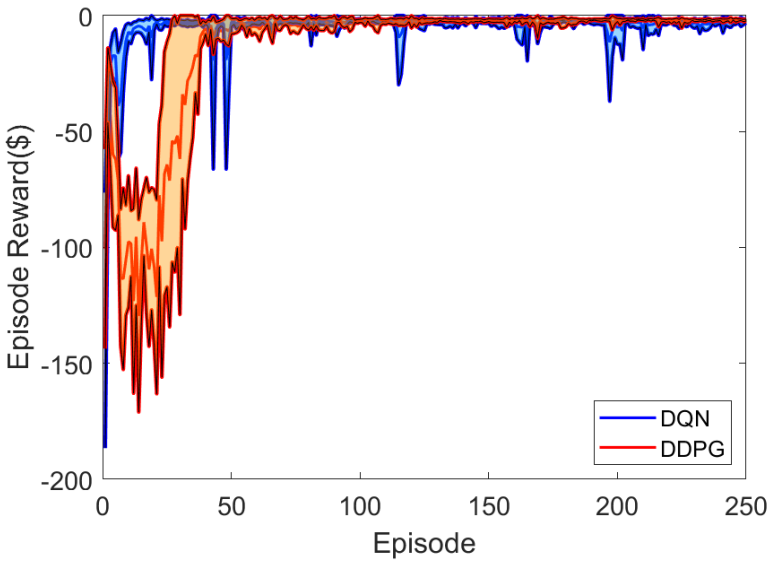

Figure 7.11. Range of episode rewards received during training for 10 trials of DQN and DDPG with different random seeds. Shaded region indicates one standard deviation from mean.

a bidding strategy, where the slope of the function is referred to as the ratio and it represents how sensitive the user is to indoor temperature. A higher ratio means that users are more sensitive to indoor temperature, and so would bid higher for every degree increase in indoor temperature. More formally we have that baseline price bid is given by:

$$\lambda_{\text{bid}} = \begin{cases} 0, \text{if } T_{\text{min}} > T_{\text{a}} \\ \lambda_{\text{avg}} + k\frac{(T_{\text{a}} - T_{\text{desired}})\lambda_{\text{dev}}}{T_{\text{max}} - T_{\text{min}}}, \text{if } T_{\text{min}} \leq T_{\text{a}} \leq T_{\text{max}} \\ 1, \text{if } T_{\text{max}} < T_{\text{a}} \end{cases}$$

where λ_{avg} is the average price over the last 24 hours, λ_{dev} is the standard deviation of prices over the last 24 hours, T_{max} is the user-defined maximum temperature limit, T_{min} is the user-defined minimum temperature limit, and k is the baseline ratio. The maximum and minimum temperature limits where held fixed for all baselines.

It can be seen from Fig. 7.15 that the drop in performance is more drastic for the agent trained with $w = 0.2$. This is expected because as the number of houses drops each individual house gains market power and thus can have a greater influence on the clearing price. This in

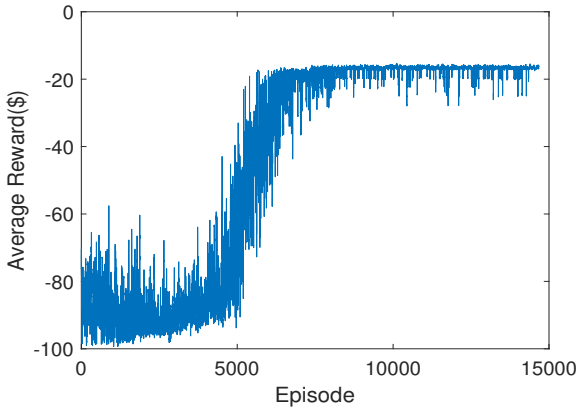

Figure 7.12. Averaged Reward for Tabular Q 10x5x19. Rewards averaged every 7 episodes .Resulting agent used in comparison plots

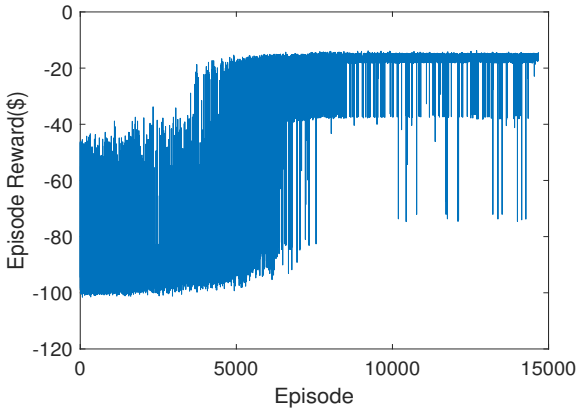

Figure 7.13. Episode Reward for Tabular Q 10x5x19. Resulting agent used in comparison plots

turn leads to less observability from the learning agent's perspective, since it cannot observe the price bids of the other houses, and therefore result in increasingly suboptimal price bids. Overall, these test are promising in that it demonstrates that the learned policies have a certain level of robustness to the changes in the market environment when utility and cost are weighed equally. However, when cost is weighted more and the number of houses is low, we see that the agents' inability to deal with a non-competitive market makes them worse than most of the base cases.

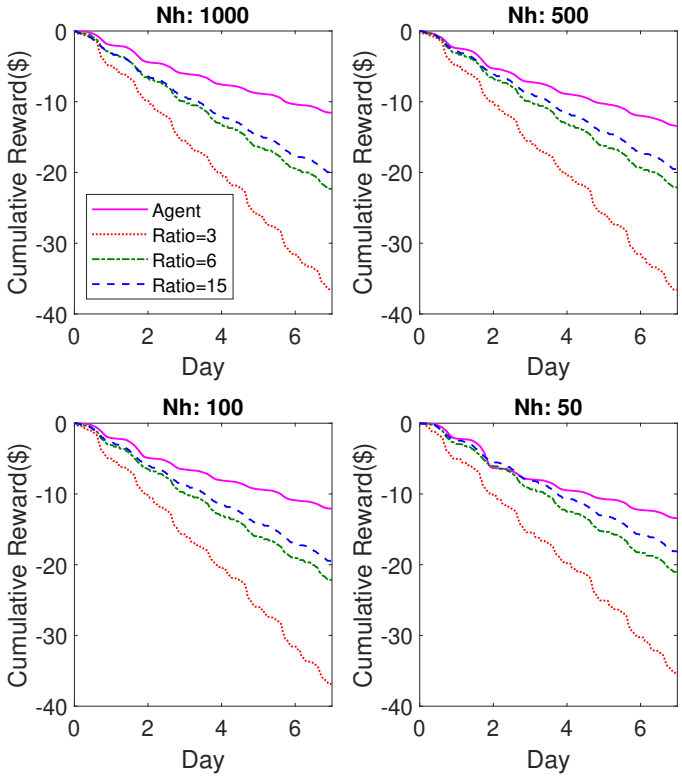

Figure 7.14. Cumulative Rewards Over Week, w=0.5

7.4 Conclusion

We explored three popular reinforcement algorithms for the purpose of finding an optimal bidding strategy for an individual house in transactive energy system. We evaluated these algorithms on the control performance of their resulting bidding strategies and their training performance. From our empirical studies we have found that DDPG and DQN far outperform tabular Q-learning. This was expected since our environment consists of continuous states and tabular Q-learning is better suited for problems with discrete state spaces. We have also found that DDPG and DQN perform similar in control performance. They do have some differences in training performance, where DQN appears to have lower sensitivity to randomness in picking initial weights and randomness in the particular exploration trajectory used during training. However, we have also seen that this sensitivity only exists during initial training of DDPG, and after many episodes of training DDPG shows better stability

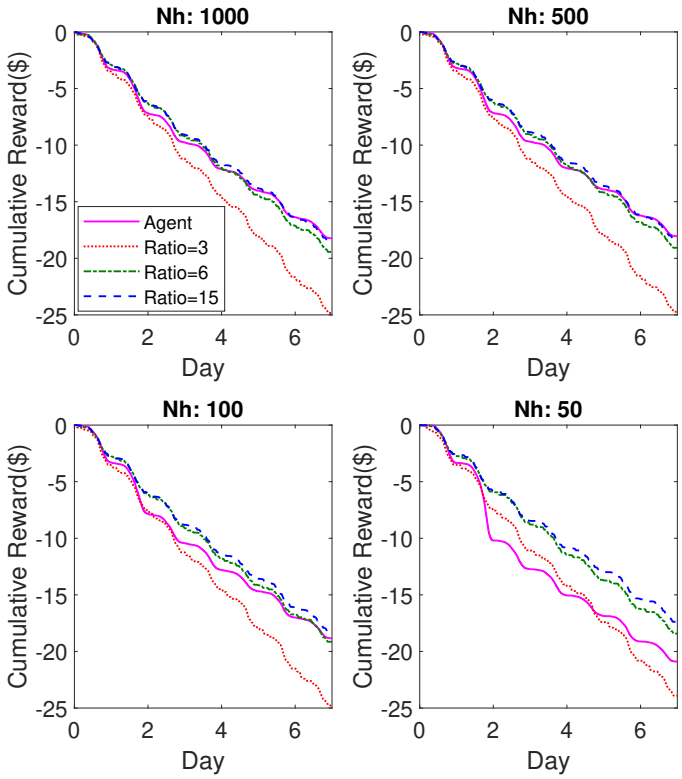

Figure 7.15. Cumulative Rewards Over Week, w=0.2

than DQN. In addition, we note that DQN was the limitation that it requires a discrete action space, while DDPG can work with continous actions. For competitive markets this can be handled by reducing the action space based on historical data of clearing prices, but for non-competitive scenarios, we can no longer do this and so DQN would no longer be a viable option. With that in mind, we have trained an agent using DDPG using a competitive environment, and have tested it on increasingly non-competive markets. This showed us that DDPG agents had a degree of robustness to decreases in competition, but it also showed us that once costs were weighed more than utility, the agent performance suffers a lot in non-competitive markets. This last case study showed us that a more sophisticated approach is needed to handle non-competitive markets, which motivates future work and which could include reformulating the problem as a multi-agent reinforcement learning problem.

PART IV

Other Work

8. DISTRIBUTED STATE ESTIMATION FOR NONLINEAR SYSTEMS WITH UNKNOWN PARAMETERS

8.1 Introduction

Due to the increasing use of large networked systems spurred on by improvements in wireless capabilities, there has been a surge of interest in the problem of estimating the state of these large systems using sensor networks. Early works in this domain have assumed that the state of the global system is observable to each sensor [176]–[178]. The assumption that the state is observable by one sensor has since been relaxed to the assumption that the state is jointly observable to the network of sensors. This problem has been studied under sensor-fusion and multi-sensor tracking research [179]–[182], and others have studied the problem as a resilience problem in the face of communication issues [183], [184].

Decentralized observers have been proposed in [185] in order to overcome the computational bottlenecks of central processors in sensor fusion algorithms, however this decentralized approach carries great communication costs [186] due to the need for all nodes in the network to communicate. Distributed observers, on the other hand, can accomplish decentralized state estimation with only communication between neighbor nodes. This ability makes distributed observers a problem of great interest and research. As such, there is a lot of literature that explores the design of distributed observers, a few of them are [126], [184], [187], [188]. The above papers all successfully derive methods for designing distributed observers for LTI systems, where [184] considers discrete-time systems while [187] and [126] consider continuous-time systems. Though these results are promising, they all only consider linear systems and so cannot be directly applied to many real world systems that have nonlinear dynamics.

Therefore, we draw our attention to nonlinear state estimators. In [189] a distributed observer was developed for a network of nonlinear systems, where the system is decomposed into linear and nonlinear parts. There has also been work on the use of stochastic methods such as particle filters in [190], which work well with nonlinear systems, however suffer from large computational costs. This brings us to the very popular unscented Kalman filters[191], which like particle filters work well with nonlinear systems but have the advantage

of being more computationally efficient[190]. There is also work on extended Kalman filters for nonlinear state estimation [192]–[194], however we will focus on unscented Kalman filter algorithms because they do not require linearization of the nonlinear system, which can result in sub-optimal performance for the extended Kalman filter [191]. There are stability results in [195] for the unscented Kalman filter, which uses a similar strategy as in [196] and [197] for deriving a bound for the filter's estimation error. These papers prove that for stochastic systems the expectation of the unscented Kalman filter estimation squared error is exponentially bounded, or equivalently they proved the error is bounded in mean square.

Kalman filters have also been extended to distributed state estimation. We see this in [198] which improves upon [199] by considering missing measurements. Both of the these papers use a consensus based distributed unscented Kalman filter and assumes complete knowledge of dynamics. A stability result is achieved in [199] for the distributed unscented Kalman filter.

The aforementioned nonlinear estimators have the limitation that full knowledge of state dynamics is required, which in real-world applications might not always be available. Relaxing this requirement will motivate our research. Fortunately, though these state estimators require knowledge of dynamics, they can also be used for parameter estimation of system dynamics. In particular there is literature on dual unscented Kalman filters [200], [201] which uses two unscented Kalman filters to simultaneously estimate model parameters and the system state. These algorithms would enable state estimation with only partial knowledge of dynamics in the form of a parametrized function with unknown parameters.

Despite the convergence results on distributed unscented Kalman filters, to our knowledge there are no analytical results for the convergence of simultaneous parameter-state estimation using dual estimation methods. We note the difference between dual estimation and joint estimation, which is discussed in [202]. With joint estimation the parameters are regarded as another set of state variables and so by representing the dynamics of the states and parameters in compact form, the problem becomes equivalent to regular state estimation. Convergence is then guaranteed from earlier results. In contrast, dual estimation treats the parameter and state estimation as two problems and so a separate Kalman filter is used for each, and accordingly there are then two sets of update equations. As such, convergence

of each Kalman filter in the dual approach cannot be guaranteed separately because they are coupled through the state dynamics. In this work we focus on dual estimation due to empirical results in the literature such as [203] and commentary of [202] that indicate that dual estimation methods are more robust than joint estimation methods. In particular, according to the simulated experiments conducted in [203], the dual method performs better than the joint method when there is uncertainty in the noise covariances.

8.2 Problem Statement

Consider a multi-agent network where each agent i calculates the estimates $x_{\mathrm{i}}(t)$ and $\theta_{\mathrm{i}}(t)$ of the central plant's state $x(t)$ at time step t and parameter vector θ, respectively. Furthermore, each agent in the network can communicate information with a set of neighboring agents $\mathcal{N}_{\mathrm{i}}$ in order to reach consensus on the plant's actual state and parameters. This network will be modeled by a connected undirected graph $\mathbb{G}$, where $\mathrm{j} \to \mathrm{i}$ and $\mathrm{i} \to \mathrm{j}$ if $\mathrm{j} \in \mathcal{N}_{\mathrm{i}}$, and $\mathrm{i} \in \mathcal{N}_{\mathrm{i}}$. In addition, $\mathcal{N}$ refers to the set of all nodes on the graph. The plant is a discrete nonlinear system with the following description:

$$x(t) = f(x(t-1), u(t-1), \theta). \tag{8.1}$$

Furthermore, each agent measures an output according to:

$$y_{\mathrm{i}}(t) = C_{\mathrm{i}} x(t) + w_{\mathrm{i}}(t), \tag{8.2}$$

where $x(t) \in \mathbb{R}^n$ is the plant's state at time step t, $u(t) \in \mathbb{R}^{\hat{n}}$ is an exogenous input at time step t, $y_{\mathrm{i}}(t) \in \mathbb{R}^{n_{\mathrm{i}}}$ is the state measurement output vector, $C_{\mathrm{i}} \in \mathbb{R}^{n_{\mathrm{i}},n}$, and $\theta \in \mathbb{R}^{\bar{n}}$ is the parameters vector. In addition, $f(\cdot)$ is a nonlinear function assumed to be continuously differentiable. $w_{\mathrm{i}}(t)$ is measurement noise, which is assumed to be zero-mean white Gaussian noise. Furthermore, each agent will know the covariance matrix $W_{\mathrm{i}}(t)$ of the noise term.

Let $C = \mathrm{col}\{C_1, C_2, \ldots, C_m\}$ where m is the number of agents, we also have that:

$$\mathrm{rank}(C_{\mathrm{i}}) < n, \ \mathrm{rank}(C) = n. \tag{8.3}$$

This last condition means each agent can not directly solve for the state vector x by itself, but the network of agents can solve for state vector x by sharing information.

As such, the network of agents will implement a distributed observer in order to reach consensus on the estimate of the plant's parameters and state $x(t)$ simultaneously. The following describes the goals of our network:

$$x_{\mathrm{i}}(t) \to x(t) \qquad\qquad \theta_{\mathrm{i}}(t) \to \theta.$$

The following block diagram demonstrates the flow of information in our problem, as well as the connection between the central plant and our multi-agent network,

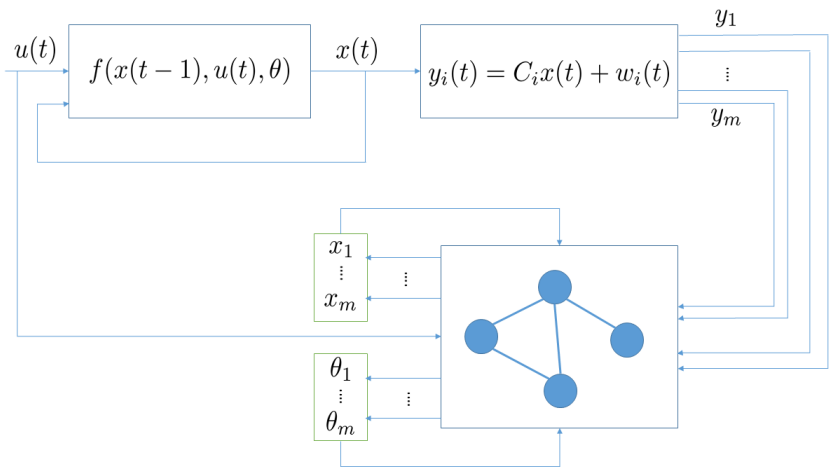

Figure 8.1. Problem Block Diagram

8.3 Approach

This paper will look to solve the above problem by combining dual parameter-state estimation, with a distributed observer for nonlinear systems inspired by the unscented Kalman filter algorithm. This combination will produce our distributed observer for dual parameter-state estimation. First we introduce the distributed observer, and then we extend this to a dual parameter and state estimation scheme by representing the parameters as another state-space to which the distributed observer can be applied.

Distributed Observer

We use a distributed observer with unscented transformations based on the algorithm in [199]. Each agent will share the information pair $(x_{\mathrm{i}}(t), \hat{P}_{\mathrm{i}}(t))$ where $\hat{P}_{\mathrm{i}}(t)$ is agent i's estimate of the state estimation error covariance matrix. The updates for both these estimates are given by the following:

$$\begin{aligned} x_{\mathrm{i}}(t) =& \bar{x}_{\mathrm{i}}(t) + K_{\mathrm{i}}(t)\left(y_{\mathrm{i}}(t) - \bar{y}_{\mathrm{i}}(t)\right) \\ \hat{P}_{\mathrm{i}}(t) =& P_{\mathrm{i},1} - K_{\mathrm{i}}(t) P_{\mathrm{i},2} (K_{\mathrm{i}}(t))^{\top}, \end{aligned}$$

where

$$\begin{aligned} P_{\mathrm{i},1} =& \sum_{s=0}^{2n} q_s (\bar{\chi}_{\mathrm{i},s}(t) - \bar{x}_{\mathrm{i}}(t))(\bar{\chi}_{\mathrm{i},s}(t) - \bar{x}_{\mathrm{i}}(t))^{\top} + U(t-1), \;\; K_{\mathrm{i}}(t) = P_{\mathrm{i},3}(P_{\mathrm{i},2})^{-1}, \\ P_{\mathrm{i},2} =& \sum_{s=0}^{2n} q_s \left(\gamma_{\mathrm{i},s}(t) - y_{\mathrm{i}}(t)\right)\left(\gamma_{\mathrm{i},s}(t) - \bar{y}_{\mathrm{i}}(t)\right)^{\top} + W_{\mathrm{i}}(t), \\ P_{\mathrm{i},3} =& \sum_{s=0}^{2n} q_s \;\; \left(\bar{\chi}_{\mathrm{i},s}(t) - \bar{x}_{\mathrm{i}}(t)\right)\left(\gamma_{\mathrm{i},s}(t) - \bar{y}_{\mathrm{i}}(t)\right)^{\top}, \\ \bar{x}_{\mathrm{i}}(t) =& \sum_{s=0}^{2n} q_s \bar{\chi}_{\mathrm{i},s}(t), \\ \bar{y}_{\mathrm{i}}(t) =& \sum_{s=0}^{2n} q_s \gamma_{\mathrm{i},s}(t) \end{aligned}$$

and where $\bar{x}_{\mathrm{i}}(t)$, $P_{\mathrm{i},1}$, and $K_{\mathrm{i}}(t)$ are the predicted state, state prediction error covariance matrix, and the Kalman gain matrix, respectively.

Furthermore, $\bar{y}_{\mathrm{i}}(t)$, $P_{\mathrm{i},2}$, $P_{\mathrm{i},3}$ are the predicted measurement, predicted measurement covariance , and state-measurement cross-covariance matrix, respectively. $U(t-1)$ is a positive definite matrix that is to be designed to ensure that $\hat{P}_{\mathrm{i}}(t)$ is positive definite, however the bigger the magnitude of $U(t-1)$ the greater the estimation error. We also have $\bar{\chi}_{\mathrm{i},s}(t) = f\left(\chi_{\mathrm{i},s}(t-1), u(t-1), \theta_{\mathrm{i}}(t-1)\right)$ and $\gamma_{\mathrm{i},s}(t) = C_{\mathrm{i}}\bar{\chi}_{\mathrm{i},s}(t)$ for $s = 0, \ldots, 2n$, where $q_s = \frac{\kappa}{n+\kappa}$ for $s = 0$ and $q_s = \frac{1}{2(n+\kappa)}$ for $s = 1, \ldots, 2n$, and where κ is a tunable scaling factor.

Let $\Sigma = \sqrt{(n+\kappa)\sum_{\mathrm{j}\in N_{\mathrm{i}}} d_{\mathrm{i,j}} \hat{P}_{\mathrm{j}}(t-1)}$. The sigma points $\chi_{\mathrm{i},s}$ are calculated through the following: $\chi_{\mathrm{i},0}(t-1) = \sum_{\mathrm{j}\in\mathcal{N}_{\mathrm{i}}} d_{\mathrm{i,j}} x_{\mathrm{j}}(t)$, $\chi_{\mathrm{i},s}(t-1) = \sum_{\mathrm{j}\in\mathcal{N}_{\mathrm{i}}} d_{\mathrm{i,j}} x_{\mathrm{j}}(t) + \Sigma_s$ for $s = 1, \ldots, n$, and

$\chi_{\mathrm{i},s}(t-1) = \sum_{\mathrm{j}\in\mathcal{N}_\mathrm{i}} d_{\mathrm{i,j}} x_\mathrm{j}(t) - \Sigma_{s-n}$ for $s = n+1,\ldots,2n$, where Σ_s denotes the s^{th} row of Σ and $d_{\mathrm{i,j}}$ are Metropolis weights according to the following:

$$d_{\mathrm{i,j}} = \begin{cases} \frac{1}{\max\{d_\mathrm{i}, d_\mathrm{j}\}}, \text{ if } \mathrm{j}\in\mathcal{N}_\mathrm{i}, \mathrm{j}\neq\mathrm{i} \\ 1 - \sum_{k\in\mathcal{N}_\mathrm{i}, k\neq\mathrm{i}} d_{\mathrm{i},k}, \text{ if } \mathrm{i}=\mathrm{j} \\ 0, \text{ if } \mathrm{j}\notin\mathcal{N}_\mathrm{i}, \end{cases}$$

where d_i is the degree of agent i.

Parameter Estimation

Dual parameter-state estimators, such as dual Kalman filters, can be used to simultaneously make better state and parameter estimates without complete knowledge of system dynamics. This corresponds to describing the system with two state-space representations, one for the states and one for the parameters. From equations (8.1) and (8.2) we get our state-space for x, and for our parameters θ we just need 8.2 since θ is time-invariant.

Given this state-space model we can now apply the same algorithm but for parameter estimation. The main changes in the algorithm are that the state dynamics, previously $f(\cdot)$, is now just the identity. The rest of the algorithm only changes in that instead of $x_\mathrm{i}(t-1)$ we now use $\theta_\mathrm{i}(t-1)$, and instead of $\bar{y}_\mathrm{i}(t)$ we use $z_\mathrm{i}(t)$. As such, each agent will also share the information pair $(\theta_\mathrm{i}(t), \bar{P}_\mathrm{i}(t))$ along with $(x_\mathrm{i}(t), \hat{P}_\mathrm{i}(t))$, where $\bar{P}_\mathrm{i}(t)$ is agent i's estimate of the parameter estimation error covariance matrix. The corresponding updates are given by the following:

$$\begin{aligned} \theta_\mathrm{i}(t) =& \bar{\theta}_\mathrm{i}(t) + \bar{K}_\mathrm{i}(t)\left(y_\mathrm{i}(t) - z_\mathrm{i}(t)\right) \\ \bar{P}_\mathrm{i}(t) =& \bar{P}_{\mathrm{i},1} - \bar{K}_\mathrm{i}(t)\bar{P}_{\mathrm{i},2}\left(\bar{K}_\mathrm{i}(t)\right)^\top, \end{aligned}$$

where

$$\begin{aligned}
\bar{P}_{\mathrm{i},1} &= \sum_{s=0}^{2n} q_s(\Theta_{\mathrm{i},s}(t-1) - \bar{\theta}_{\mathrm{i}}(t))(\Theta_{\mathrm{i},s}(t-1) - \bar{\theta}_{\mathrm{i}}(t))^\top + R(t-1), \\
\bar{K}_{\mathrm{i}}(t) &= \bar{P}_{\mathrm{i},3}(\bar{P}_{\mathrm{i},2})^{-1}, \\
\bar{P}_{\mathrm{i},2} &= \sum_{s=0}^{2n} q_s \left(\gamma_{\mathrm{i},s}(t) - z_{\mathrm{i}}(t)\right)\left(\gamma_{\mathrm{i},s}(t) - z_{\mathrm{i}}(t)\right)^\top + W_{\mathrm{i}}(t), \\
\bar{P}_{\mathrm{i},3} &= \sum_{s=0}^{2n} q_s \left(\Theta_{\mathrm{i},s}(t-1) - \bar{\theta}_{\mathrm{i}}(t)\right)\left(\gamma_{\mathrm{i},s}(t) - z_{\mathrm{i}}(t)\right)^\top, \\
z_{\mathrm{i}}(t) &= \sum_{s=0}^{2n} q_s \gamma_{\mathrm{i},s}(t), \\
\bar{\theta}_{\mathrm{i}}(t) &= \sum_{s=0}^{2n} q_s \Theta_{\mathrm{i},s}(t-1).
\end{aligned}$$

Similar to state estimation, we add a positive definite matrix $R(t-1)$ to be designed so that in this case $\bar{P}_{\mathrm{i}}(t)$ is positive definite. Furthermore we have $\gamma_{\mathrm{i},s}(t) = C_{\mathrm{i}} f(x_{\mathrm{i}}(t-1), u(t-1), \Theta_{\mathrm{i},s}(t-1))$ for $s = 0, \ldots, 2\bar{n}$, where $q_s = \frac{\bar{\kappa}}{\bar{n}+\bar{\kappa}}$ for $s = 0$, and $q_s = \frac{1}{2(\bar{n}+\bar{\kappa})}$ for $s = 1, \ldots, 2\bar{n}$, and where $\bar{\kappa}$ is again a tunable scaling factor.

The sigma points $\Theta_{\mathrm{i},s}(t-1)$ are calculated through the following:

$$\begin{aligned}
\Theta_{\mathrm{i},0}(t-1) &= \sum_{\mathrm{j}\in\mathcal{N}_{\mathrm{i}}} d_{\mathrm{i,j}}\theta_{\mathrm{j}}(t), \\
\Theta_{\mathrm{i},s}(t-1) &= \sum_{\mathrm{j}\in\mathcal{N}_{\mathrm{i}}} d_{\mathrm{i,j}}\theta_{\mathrm{j}}(t) + \bar{\Sigma}_s \text{ for } s = 1, \ldots, \bar{n}, \\
\Theta_{\mathrm{i},s}(t-1) &= \sum_{\mathrm{j}\in\mathcal{N}_{\mathrm{i}}} d_{\mathrm{i,j}}\theta_{\mathrm{j}}(t) - \bar{\Sigma}_{s-\bar{n}} \text{ for } s = \bar{n}+1, \ldots, 2\bar{n},
\end{aligned}$$

where $\bar{\Sigma} = \sqrt{(n+\kappa)\sum_{\mathrm{j}\in N_{\mathrm{i}}} d_{\mathrm{i,j}}\hat{P}_{\mathrm{j}}(t-1)}$ and $\bar{\Sigma}_s$ denotes the s^{th} row of $\bar{\Sigma}$.

8.4 Stability Analysis of Distributed Observer

In this section we will analyze the stability of our distributed observer algorithm. This analysis is similar to [198] and [199]. First we will follow the method in [196] where $x(t)$ is expressed as Taylor series expansion, and then consider only the linear term as an approximation. Unlike in [196], we will perform the Taylor expansion around $(x, u, \theta) = (0, u(t-1), 0)$.

Assuming $f(0, u(t-1), 0) = 0$, we can therefore express an approximation of $x(t)$ as: $x(t) \approx F_x(t-1)x(t-1) + F_\theta(t-1)\theta(t-1)$, where $F_x(t-1) = \nabla_x f(x', u', \theta')|_{(x',u',\theta')=(0,u(t-1),0)}$ and $F_\theta(t-1) = \nabla_x f(x', u', \theta')|_{(x',u',\theta')=(0,u(t-1),0)}$. We can take into account the residual of our approximation of $x(t)$ by multiplying by an unknown diagonal matrices $\Omega_x(t)$ and $\Omega_\theta(t)$ to get the following equality: $x(t) = \Omega_x(t-1)F_x(t-1)x(t-1) + \Omega_\theta(t-1)F_\theta(t-1)$. This strategy is used in [195][197][198], and [196] and is crucial to their analysis and ours as well. This gives us a linear representation of the nonlinear time-evolution of our state, and since our parameters and measurement outputs have a linear time-evolution we can now represent our entire system in a linear fashion.

Similarly, we can define a linearization of our system in terms of our parameters θ by Taylor expanding $f(\cdot)$ around $(x', u', \theta) = (x(t-1), u(t-1), 0)$. Assuming that $f(x(t-1), u(t-1), 0) = 0$, this results in the following approximation: $x(t) \approx \bar{F}(t-1)\theta(t-1)$. Again we can take into account the residual of our approximation by introducing an unknown diagonal matrix $\beta(t)$, which gives us the following equality: $x(t) = \beta(t-1)\bar{F}(t-1)\theta(t-1)$ where $\bar{F}(t-1) = \nabla_\theta f(x', u', \theta')|_{(x',u',\theta')=(x(t-1),u(t-1),0)}$. Now in order to analyze the stability of the algorithm we must take into account the state estimates of each agent, as well as the parameter estimates of each agent since the two are coupled. We take this coupling of state and parameter estimates into account by column-wise stacking the parameter and state estimates into one state vector $\hat{x}_{\mathrm{i}}(t) = \mathrm{col}\{x_{\mathrm{i}}(t), \theta_{\mathrm{i}}(t)\}$. We similarly stack the predicted measurement vectors such that $\hat{y}_{\mathrm{i}}(t) = \mathrm{col}\{\bar{y}_{\mathrm{i}}(t), z_{\mathrm{i}}(t)\}$. Let $\Omega(t) = [\Omega_x(t), \Omega_\theta(t)]$ and $F(t) = [F_x(t), F_\theta(t)]$, we now modify all other relevant matrices and vectors in similar way.

$$
\begin{aligned}
&\tilde{x}(t) = \mathrm{col}\{x(t), \theta\}, && \hat{x}_{\mathrm{i}}(t) = \mathrm{col}\{x_{\mathrm{i}}(t), \theta_{\mathrm{i}}(t)\},\\
&\tilde{y}_{\mathrm{i}}(t) = \mathrm{col}\{y_{\mathrm{i}}(t), y_{\mathrm{i}}(t)\}, && \hat{y}_{\mathrm{i}}(t) = \mathrm{col}\{\bar{y}_{\mathrm{i}}(t), z_{\mathrm{i}}(t)\},\\
&\tilde{w}_{\mathrm{i}}(t) = \mathrm{col}\{w_{\mathrm{i}}(t), w_{\mathrm{i}}(t)\}, && \tilde{\Omega}(t) = \mathrm{col}\{\Omega(t), [\mathbf{0}, I_{\bar{n}}]\},\\
&\tilde{F}(t) = \mathrm{col}\{F(t), [\mathbf{0}, I_{\bar{n}}]\}, && \tilde{R}(t) = \mathrm{diag}\{U(t), R(t)\},\\
&\tilde{W}_{\mathrm{i}}(t) = \mathrm{diag}\{W_{\mathrm{i}}(t), W_{\mathrm{i}}(t)\}, && \tilde{P}_{\mathrm{i},1} = \mathrm{diag}\{P_{\mathrm{i},1}, \bar{P}_{\mathrm{i},1}\},\\
&\tilde{P}_{\mathrm{i}}(t) = \mathrm{diag}\{\hat{P}_{\mathrm{i}}(t), \bar{P}_{\mathrm{i}}(t)\}, && \tilde{K}_{\mathrm{i}}(t) = \mathrm{diag}\{K_{\mathrm{i}}(t), \bar{K}_{\mathrm{i}}(t)\},\\
&\tilde{C}_{\mathrm{i}}(t) = \mathrm{diag}\{C_{\mathrm{i}}, C_{\mathrm{i}}\beta_{\mathrm{i}}(t-1)\bar{F}_{\mathrm{i}}(t-1)\}.
\end{aligned}
$$

Using this we can express our entire system with the following equations:

$$\tilde{x}(t) = \tilde{\Omega}(t-1)\tilde{F}(t-1)\tilde{x}(t-1) \tag{8.4}$$

$$\tilde{y}_{\mathrm{i}}(t) = \tilde{C}_{\mathrm{i}}(t)\tilde{x}(t) + \tilde{w}_{\mathrm{i}}(t)$$

$$\hat{x}_{\mathrm{i}}(t) = \tilde{\Omega}(t-1)\tilde{F}(t-1)\sum_{\mathrm{j}\in\mathcal{N}} d_{\mathrm{i,j}}\hat{x}_{\mathrm{j}}(t-1) + \tilde{K}_{\mathrm{i}}(t)(\tilde{y}_{\mathrm{i}}(t) - \hat{y}_{\mathrm{i}}(t)) \tag{8.5}$$

$$\hat{y}_{\mathrm{i}}(t) = \tilde{C}_{\mathrm{i}}(t)\tilde{\Omega}(t-1)\tilde{F}(t-1)\sum_{\mathrm{j}\in\mathcal{N}} d_{\mathrm{i,j}}\hat{x}_{\mathrm{j}}(t-1). \tag{8.6}$$

Furthermore, we can express our update equations as the following:

$$\tilde{P}_{\mathrm{i},1} = \Gamma(t-1)\sum_{\mathrm{j}\in\mathcal{N}} d_{\mathrm{i,j}}\tilde{P}_{\mathrm{j}}(t-1)\Gamma(t-1)^{\top} + \tilde{R}(t-1) \tag{8.7}$$

$$\tilde{K}_{\mathrm{i}}(t) = (\tilde{P}_{\mathrm{i},1} - \tilde{R}(t-1))(\tilde{C}_{\mathrm{i}}(t))^{\top}$$
$$[\tilde{C}_{\mathrm{i}}(t)(\tilde{P}_{\mathrm{i},1} - \tilde{R}(t-1))(\tilde{C}_{\mathrm{i}}(t))^{\top} + \tilde{W}_{\mathrm{i}}(t)]^{-1} \tag{8.8}$$

$$\tilde{P}_{\mathrm{i}}(t) = \tilde{P}_{\mathrm{i},1} - \tilde{K}(t)\tilde{C}_{\mathrm{i}}(t)(\tilde{P}_{\mathrm{i},1} - \tilde{R}(t-1)), \tag{8.9}$$

where $\Gamma(t) = \tilde{\Omega}(t)\tilde{F}(t)$. We now discuss some assumptions that are used in proving our theorem.

Assumption 8.4.1. *There exists real, non-zero numbers* $\underline{f}, \underline{\Omega}, \underline{c}, \bar{f}, \bar{\Omega}, \bar{c}$, *and,* λ_c *such that the following bounds are fulfilled for every* $k \geq 0$ *and all* $\mathrm{i} \in \mathcal{N}$*:*

$$\underline{f}^2 I \leq \tilde{F}(t)\tilde{F}(t)^{\top} \leq \bar{f}^2 I,\ \underline{\Omega}^2 I \leq \tilde{\Omega}_{\mathrm{i}}(t)\tilde{\Omega}_{\mathrm{i}}(t)^{\top} \leq \bar{\Omega}^2 I$$
$$\underline{c}^2 I \leq \tilde{C}_{\mathrm{i}}(t)\tilde{C}_{\mathrm{i}}(t)^{\top} \leq \bar{c}^2 I,\ \lambda_c^2 I \leq \tilde{C}(t)^{\top}\tilde{C}(t) \leq m\bar{c}^2 I,$$

where $\tilde{C}(t) = col\{\tilde{C}_1(t), ..., \tilde{C}_m(t)\}$. *We note that though* $\tilde{C}_{\mathrm{i}}(t)$ *is not full rank, it is reasonable to assume* λ_c *is non-zero because of (8.3).*

Assumption 8.4.2. *There exists positive real numbers* $\bar{v}, \underline{v}, \underline{w}, \bar{w}, \underline{p}, \bar{p} > 0$ *such that the following bounds are fulfilled for every* $k \geq 0$*:*

$$\underline{r} I \leq \tilde{R}(t) \leq \bar{r} I,\ \underline{w} I \leq \tilde{W}_{\mathrm{i}}(t) \leq \bar{w} I,\ \underline{p} I \leq \tilde{P}_{\mathrm{i}}(t) \leq \bar{p} I.$$

With the above assumptions we then have our main result:

Theorem 8.4.3. *Consider a nonlinear stochastic system given in (8.1) and the proposed distributed observer algorithm. We define the estimation error of each agent as* $\mathrm{e_i}(t) = \hat{x}_\mathrm{i}(t) - \tilde{x}(t)$*, and the augmented estimation error of the systems as* $\mathrm{e}(t) = col\{\mathrm{e_i}(t), \mathrm{i} \in \mathcal{N}\}$*. If assumptions 8.4.1-8.4.2 hold, then we have the following upper bound for the expectation of the squared augmented estimation error:*

$$\mathbb{E}[\|\mathrm{e}(t)\|^2] \leq \frac{\underline{p}^{-1}}{\bar{p}^{-1}}\mathbb{E}[\|\mathrm{e}(0)\|^2](1-\lambda)^t + \frac{\mu}{\bar{p}^{-1}}\sum_{\mathrm{i}=1}^{t-1}(1-\lambda)^\mathrm{i}, \tag{8.10}$$

where $\mu = \bar{k}^2\underline{p}^{-1}m\bar{w}^2$*,* $\bar{k} = \frac{\bar{\Omega}^2\bar{f}^2\bar{p}\bar{c}}{\underline{c}^2\underline{\Omega}^2\underline{f}^2\underline{p}+\underline{w}}$*,* $\lambda = \frac{\underline{\Omega}^2\underline{f}^2\underline{p}\lambda_c^2}{m(\bar{c}^2\bar{\Omega}^2\bar{f}^2\bar{p}+\bar{w})}$*, and* $0 < \lambda < 1$*.*

Remark 8.4.4. *We note that as* $t \to \infty$*, the first term on right side of (8.10) goes to zero and so the upper bound of the expectation of the squared error converges to* $\frac{\mu}{\bar{p}^{-1}\lambda}$*. By unpacking* μ *and* λ *we find that this value is a function of* $\bar{w}^2$ *and* $\bar{w}^3$*.*

8.5 Proof of Main Result

In this section we will provide a proof for Theorem 1 by constructing and then analyzing a candidate Lyapunov function $V(\mathrm{e}(t))$. We first introduce lemmas which will be needed for the proof:

Lemma 8.5.1. *Assume that* $\xi(t)$ *and* $V(\xi(t))$ *are stochastic processes, and that there are real numbers* $v_{min} > 0, v_{max} > 0$ *,* $\mu > 0$*, and* $0 < \lambda \leq 1$ *such that* $\forall t$ *we have*

$$v_{\min}\|\xi(t)\|^2 \leq V(\xi(t)) \leq v_{\max}\|\xi(t)\|^2$$
$$\mathbb{E}[V(\xi(t))|\xi(t-1)] - V(\xi(t-1)) \leq \mu - \lambda V(\xi(t-1)).$$

Then $\xi(t)$ *is bounded in mean square:*

$$\mathbb{E}[\|\xi(t)\|^2] \leq \frac{v_{\max}}{v_{\min}}\mathbb{E}[\|\xi_0\|^2](1-\lambda)^t + \frac{\mu}{v_{\min}}\sum_{\mathrm{i}=1}^{t-1}(1-\lambda)^\mathrm{i}.$$

Proof: This result is theorem 2 in [204] and is also referenced in [205] as lemma 2.1. For brevity we refer to their proofs.

Lemma 8.5.2. *Given an integer $N \geq 2$, N positive definite matrices M_{i}, and N vectors v_{i} the following inequality holds*

$$\left(\sum_{\mathrm{i}=1}^{N} M_{\mathrm{i}} v_{\mathrm{i}}\right)^{\top} \left(\sum_{\mathrm{i}=1}^{N} M_{\mathrm{i}}\right)^{-1} \left(\sum_{\mathrm{i}=1}^{N} M_{\mathrm{i}} v_{\mathrm{i}}\right) \leq \sum_{\mathrm{i}=1}^{N} v_{\mathrm{i}}^{\top} M_{\mathrm{i}} v_{\mathrm{i}}.$$

Proof: This result is lemma 2 in [206]. For brevity we refer to their proof.

Proof of Theorem 1

Using (8.4) - (8.6),we can express the estimation error as:

$$\begin{aligned} \mathrm{e}_{\mathrm{i}}(t) &= \hat{x}_{\mathrm{i}}(t) - \tilde{x}(t) \\ &= \Xi_{\mathrm{i}}(t)\Gamma(t-1) \sum_{\mathrm{j}\in\mathcal{N}} d_{\mathrm{i,j}} \mathrm{e}_{\mathrm{j}}(t-1) + \tilde{K}_{\mathrm{i}}(t)\tilde{w}_{\mathrm{i}}(t), \end{aligned} \tag{8.11}$$

where $\Xi_{\mathrm{i}}(t) = (I - \tilde{K}_{\mathrm{i}}(t)\tilde{C}_{\mathrm{i}}(t))$ and $\Gamma(t) = \tilde{\Omega}(t)\tilde{F}(t)$. We choose the following as a candidate Lyapunov function:

$$V(\mathrm{e}(t)) = \sum_{\mathrm{i}\in\mathcal{N}} (\mathrm{e}_{\mathrm{i}}(t))^{\top} (\tilde{P}_{\mathrm{i}}(t))^{-1} \mathrm{e}_{\mathrm{i}}(t), \tag{8.12}$$

where from assumption 2 we have:

$$\bar{p}^{-1} \|\mathrm{e}(t)\|^2 \leq V(\mathrm{e}(t)) \leq \underline{p}^{-1} \|\mathrm{e}(t)\|^2. \tag{8.13}$$

Now by plugging in (8.11) into (8.12) and taking the expectation conditioned on the previous time-step, we get:

$$\mathbb{E}[V(\mathrm{e}(t))|\mathrm{e}(t-1)] = \mathbb{E}\left[\sum_{\mathrm{i}\in\mathcal{N}} (\mathrm{e}_{\mathrm{i}}(t))^{\top} (\tilde{P}_{\mathrm{i}}(t))^{-1} \mathrm{e}_{\mathrm{i}}(t) \ |\mathrm{e}(t-1)\right]. \tag{8.14}$$

Now we expand (8.14) and by noting that in expectation all the cross terms with our augmented noise vector and augmented state vector $\tilde{x}(t)$ go to zero, we get:

$$\mathbb{E}[V(\mathrm{e}(t) \mid \mathrm{e}(t-1))] = \Lambda_1 + \Lambda_2, \tag{8.15}$$

where:

$$\Lambda_1 = \mathbb{E}\left[\sum_{\mathrm{i}\in\mathcal{N}}(\sum_{\mathrm{j}\in\mathcal{N}} d_{\mathrm{i,j}}\mathrm{e_j}(t-1))^\top \Gamma(t-1)^\top \Xi_\mathrm{i}(t)^\top (\tilde{P}_\mathrm{i}(t))^{-1}\Xi_\mathrm{i}(t)\Gamma(t-1)\sum_{\mathrm{j}\in\mathcal{N}} d_{\mathrm{i,j}}\mathrm{e_j}(t-1)\right] \tag{8.16}$$

$$\Lambda_2 = \mathbb{E}\left[\sum_{\mathrm{i}\in\mathcal{N}} \tilde{w}_\mathrm{i}(t)^\top \tilde{K}(t)^\top (\tilde{P}_\mathrm{i}(t))^{-1}\tilde{K}(t)\tilde{w}_\mathrm{i}(t)\right]. \tag{8.17}$$

We now have to find an estimate for each term in (8.15). Starting with Λ_1, we plug in (8.9) into (8.16) which gives us:

$$\begin{aligned}\Lambda_1 = \mathbb{E}\Bigg[&\sum_{\mathrm{i}\in\mathcal{N}}(\sum_{\mathrm{j}\in\mathcal{N}} d_{\mathrm{i,j}}\mathrm{e_j}(t-1))^\top \Gamma(t-1)^\top \Xi_\mathrm{i}(t)^\top \\ &[\Xi_\mathrm{i}(t)(\tilde{P}_{\mathrm{i},1} - \tilde{R}(t)) + \tilde{R}(t)]^{-1}\Xi_\mathrm{i}(t)\Gamma(t-1)\sum_{\mathrm{j}\in\mathcal{N}} d_{\mathrm{i,j}}\mathrm{e_j}(t-1)\Bigg].\end{aligned}$$

Under assumption 8.4.2, we know $\tilde{R}(t)$ is positive definite so it can be shown the following is true: $(\Xi_\mathrm{i}(t)(\tilde{P}_{\mathrm{i},1} - \tilde{R}(t)))^{-1} > [\Xi_\mathrm{i}(t)(\tilde{P}_{\mathrm{i},1} - \tilde{R}(t)) + \tilde{R}(t)]^{-1}$, and so we have the following upper bound on Λ_1:

$$\begin{aligned}\Lambda_1 \leq \mathbb{E}\Bigg[&\sum_{\mathrm{i}\in\mathcal{N}}(\sum_{\mathrm{j}\in\mathcal{N}} d_{\mathrm{i,j}}\mathrm{e_j}(t-1))^\top \Gamma(t-1)^\top \Xi_\mathrm{i}(t)^\top \\ &(\Xi_\mathrm{i}(t)(\tilde{P}_{\mathrm{i},1} - \tilde{R}(t)))^{-1}\Xi_\mathrm{i}(t)\Gamma(t-1)\sum_{\mathrm{j}\in\mathcal{N}} d_{\mathrm{i,j}}\mathrm{e_j}(t-1)\Bigg].\end{aligned}$$

Let $\Psi_\mathrm{i}(t) = \sum_{\mathrm{j}\in\mathcal{N}} \Xi_\mathrm{i}(t)\Gamma(t-1)d_{\mathrm{i,j}}\mathrm{e_j}(t-1)$. We plug in (8.7) which gives us:

$$\Lambda_1 \leq \mathbb{E}[\sum_{\mathrm{i}\in\mathcal{N}} \Psi_\mathrm{i}(t)^\top [(\sum_{\mathrm{j}\in\mathcal{N}} \Xi_\mathrm{i}(t)\Gamma(t-1)d_{\mathrm{i,j}}\tilde{P}_\mathrm{j}(t-1)\Gamma(t-1)^\top)]^{-1}\Psi_\mathrm{i}(t)].$$

Now we expand the term $\Psi_{\mathrm{i}}(t)$ and factor out the following:

$$\tilde{P}_{\mathrm{j}}(t-1)\Gamma(t-1)^{\top}(\tilde{P}_{\mathrm{j}}(t-1)\Gamma(t-1)^{\top})^{-1}.$$

By letting $M_{\mathrm{ij}}(t) = [\Xi_{\mathrm{i}}(t)\Gamma(t-1)d_{\mathrm{i,j}}\tilde{P}_{\mathrm{j}}(t-1)\Gamma(t-1)^{\top}]$ we can then rewrite the above upperbound as:

$$\begin{aligned}\Lambda_1 \leq & \mathbb{E}\left[\sum_{\mathrm{i}\in\mathcal{N}}\left[\sum_{\mathrm{j}\in\mathcal{N}} M_{\mathrm{ij}}(t)(\tilde{P}_{\mathrm{j}}(t-1)\Gamma(t-1)^{\top})^{-1}\mathrm{e}_{\mathrm{j}}(t-1)\right]^{\top}\right. \\ & \left.\left[\sum_{\mathrm{j}\in\mathcal{N}} M_{\mathrm{ij}}(t)\right]^{-1}\left[\sum_{\mathrm{j}\in\mathcal{N}} M_{\mathrm{ij}}(t)(\tilde{P}_{\mathrm{j}}(t-1)\Gamma(t-1)^{\top})^{-1}\mathrm{e}_{\mathrm{j}}(t-1)\right]\right].\end{aligned}$$

We now use lemma 8.5.2 to upper bound the above, then by unpacking M_{ij}, moving the outer sum and d_{ij} inside, and simplifying we arrive at:

$$\Lambda_1 \leq \mathbb{E}[\sum_{\mathrm{j}\in\mathcal{N}}(\mathrm{e}_{\mathrm{j}}(t-1))^{\top}(\Gamma(t-1)\tilde{P}_{\mathrm{j}}(t-1))^{-1} \quad (I-\sum_{\mathrm{i}\in\mathcal{N}} d_{\mathrm{ij}}\tilde{K}_{\mathrm{i}}(t)\tilde{C}_{\mathrm{i}}(t))\Gamma(t-1)\mathrm{e}_{\mathrm{j}}(t-1))]. \tag{8.18}$$

Let us focus on $(I-\sum_{\mathrm{i}\in\mathcal{N}} d_{\mathrm{ij}}\tilde{K}_{\mathrm{i}}(t)\tilde{C}_{\mathrm{i}}(t))$. We use the expansion

$$\tilde{K}_{\mathrm{i}}(t)\tilde{C}_{\mathrm{i}}(t) = \frac{1}{m}\sum_{\mathrm{i}\in\mathcal{N}}\tilde{K}_{\mathrm{i}}(t)\tilde{C}_{\mathrm{i}}(t) + (\tilde{K}_{\mathrm{i}}(t)\tilde{C}_{\mathrm{i}}(t) - \frac{1}{m}\sum_{\mathrm{i}\in\mathcal{N}}\tilde{K}_{\mathrm{i}}(t)\tilde{C}_{\mathrm{i}}(t)),$$

to get: $(I-\sum_{\mathrm{i}\in\mathcal{N}} d_{\mathrm{ij}}\tilde{K}_{\mathrm{i}}(t)\tilde{C}_{\mathrm{i}}(t)) = (I-\frac{1}{m}\sum_{\mathrm{i}\in\mathcal{N}}\tilde{K}_{\mathrm{i}}(t)\tilde{C}_{\mathrm{i}}(t) - (\sum_{\mathrm{i}\in\mathcal{N}}(d_{\mathrm{ij}}-\frac{1}{m})\tilde{K}_{\mathrm{i}}(t)\tilde{C}_{\mathrm{i}}(t)))$. Under assumptions 1 and 2 it can be shown that $\tilde{K}_{\mathrm{i}}(t)\tilde{C}_{\mathrm{i}}(t)$ is positive semi-definite, and then using the fact that by construction $0 \leq (d_{\mathrm{ij}}-\frac{1}{m}) \leq 1$, we have: $(I-\sum_{\mathrm{i}\in\mathcal{N}} d_{\mathrm{ij}}\tilde{K}_{\mathrm{i}}(t)\tilde{C}_{\mathrm{i}}(t)) \leq (I-\frac{1}{m}\sum_{\mathrm{i}\in\mathcal{N}}\tilde{K}_{\mathrm{i}}(t)\tilde{C}_{\mathrm{i}}(t))$, then again using the bounds in assumptions 1 and 2, we arrive at the following:

$$(I-\sum_{\mathrm{i}\in\mathcal{N}} d_{\mathrm{ij}}\tilde{K}_{\mathrm{i}}(t)\tilde{C}_{\mathrm{i}}(t)) \leq (1-\lambda), \tag{8.19}$$

where $\lambda = \frac{\underline{\Omega}^2 \underline{f}^2 p \lambda_c^2}{m(\bar{c}^2 \bar{\Omega}^2 \bar{f}^2 \bar{p} + \bar{w})}$ and $0 < \lambda < 1$. Now by using (8.19) in (8.18) and then simplifying by noting that from assumption 1 $\Gamma(t-1)$ is invertible, we arrive at the following upper bound for (8.18): $\Lambda_1 \leq (1-\lambda)\mathbb{E}[\sum_{\mathrm{j}\in\mathcal{N}}(\mathrm{e_j}(t-1))^\top(\tilde{P}_\mathrm{j}(t-1))^{-1}\mathrm{e_j}(t-1)]$. Now using (8.12) we then get:

$$\Lambda_1 \leq (1-\lambda)\mathbb{E}\left[V(\mathrm{e}(t-1))\right]. \tag{8.20}$$

Proceeding now with Λ_2 we can make use of assumptions 1 and 2 and equations (8.8) and (8.7) to get:

$$\tilde{K}(t)^\top \tilde{K}(t) \leq \bar{k}^2 I, \tag{8.21}$$

where $\bar{k}^2 = \frac{(\bar{\Omega}^4 \bar{f}^4 \bar{p}^2)\bar{c}^2}{(\underline{c}^2 \underline{\Omega}^2 \underline{f}^2 \underline{p} + \underline{w})^2}$. Under assumptions 1 and 2, using equation (8.17) and inequality (8.21) we can derive the following upper bound for Λ_2:

$$\Lambda_2 \leq \bar{k}^2 \underline{p}^{-1} m \bar{w}^2. \tag{8.22}$$

Let $\mu = \bar{k}^2 \underline{p}^{-1} m \bar{w}^2$. Combining (8.14), (8.20), and (8.22) we arrive at:

$$\mathbb{E}[V(\mathrm{e}(t)) \mid \mathrm{e}(t-1)] - V(\mathrm{e}(t-1)) \leq \mu - \lambda V(\mathrm{e}(t-1)). \tag{8.23}$$

Considering that $0 < \lambda < 1$, we can combine (8.23) with (8.13) and apply Lemma 1 to arrive at the conclusion that expectation of the squared norm of $\mathrm{e}(t)$ is bounded by (8.10). ■

8.6 Simulations

We simulate 10 agents that are in a connected, undirected network, which implement the proposed distributed observer to track the state of a nonlinear system. The following describes the nonlinear system:

$$x(t) = 5 \begin{bmatrix} \theta_1 \cos(x_1(t-1)) + \sin(u(t-1)) \\ \theta_2 \cos(x_2(t-1))^2 + \cos(u(t-1)) \\ \theta_3 \sin(x_3(t-1))^2 + u(t-1) \end{bmatrix}.$$

With agent measurements $y_{\mathrm{i}}(t) = C_{\mathrm{i}} x(t) + w_{\mathrm{i}}(t)$, and input $u(t) = sin(t-1)$, where C_{i} are randomly generated matrices with values in the interval $(0, 1)$ and $n_{\mathrm{i}} < 3$, $w_{\mathrm{i}}(t)$ is zero-mean white Gaussian noise, $\theta = [0.3, 0.5, .1]^\top$, and $x(0) = [1, 1, 1]^\top$. The covariance matrices used by the algorithm are the following: $U(t) = 10^{-7} I_n$, $R(t) = 10^{-7} I_{\bar{n}}$, $W_{\mathrm{i}}(t) = 10^{-2} I_{n_{\mathrm{i}}}$, where $W_{\mathrm{i}}(t)$ corresponds to the measurement noise and $U(t)$ and $R(t)$ are designed according to our algorithm and our particular example. We define two functions as metrics for state and parameter estimation performance(one for each): $V_x(t) = \frac{1}{2} \sum_{\mathrm{i}=1}^{m} \|x_{\mathrm{i}}(t) - x(t)\|^2$, $V_\theta(t) = \frac{1}{2} \sum_{\mathrm{i}=1}^{m} \|\theta_{\mathrm{i}}(t) - \theta\|^2$.

The plots in Fig. 8.2-8.3 show the functions $V_x(t)$and $V_\theta(t)$ which we have just defined. From these plots we can see how the estimation errors(for both parameter and state estimates) for all agents decay over time.

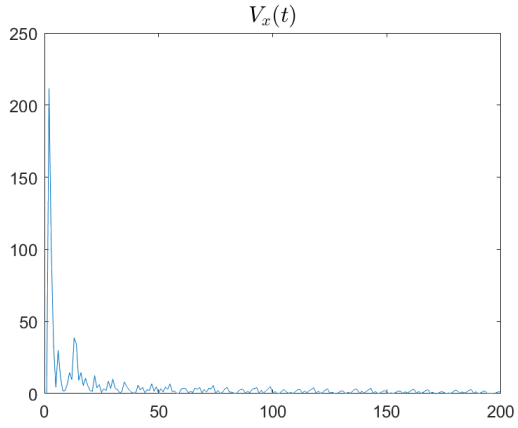

Figure 8.2. Sum of Norm Squared State Estimation Error

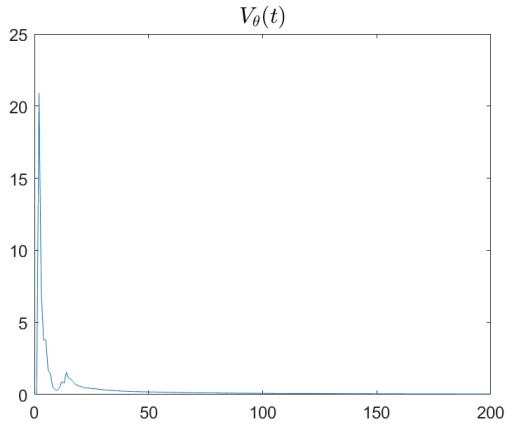

Figure 8.3. Sum of Norm Squared Parameter Estimation Error

8.7 Conclusion

We have developed a distributed observer algorithm based on unscented Kalman filters for both parameter and state estimation. Our observer uses the dependence of state dynamics on the parameters in order to compute errors in parameter estimation from only state measurements. We have proven that this technique results in bounded estimation errors that converge to a finite value as long as certain conditions are satisfied, and have characterized the evolution of this bound. Furthermore, simulations show that our algorithm can successfully estimate the state and parameters of non-linear systems. Though the observer can reliably track the states, we cannot guarantee that the estimation error will converge to zero, which is as expected with nonlinear, stochastic systems. Despite this limitation, our analysis also tells us that if the measurement noise is zero, then as $t \to \infty$, zero estimation error can be guaranteed, which is also as expected.

9. SUMMARY

This thesis has explored multi-agent reinforcement learning (MARL). We have developed distributed algorithms to address some challenges with MARL and have also provided theoretical results to better understand some existing algorithms. The contributions made by this thesis are summarized below:

Contribution 1: Improving scalability of MARL.

We have developed two distributed reinforcement learning algorithms that address some challenges with scalability in MARL. One of these algorithms is a distributed actor-critic algorithm that allows agents to learn optimal policies in cases with continuous state and action spaces. In this proposed algorithm, each agent communicates with neighbors in order to improve their policies so that the globally averaged reward is maximized, given that agents only observe their own individual reward. We provided a theoretical result based on the two-timescale methods, which is used in the field of stochastic approximation. The result provides the conditions under which each agent can converge close to the optimal control policy by following the proposed algorithm.

Another algorithm that we have developed to improve scalability is a distributed offline reinforcement learning algorithm. The proposed distributed offline algorithm combines Q-learning and a method known as offline regularization to improve offline reinforcement learning, where agents can only learn from a fixed dataset. Moreover, the proposed approach leverages a multi-agent system to split up the dataset and therfore the computational load amonsgt the network of agents. As such, the distributed algorithm uses a weighted local averaging step so that agents can make use of their neighbor's estimates to improve their own. In this work we have also provided a theoretical result that shows that the norm squared error (summed over all agents) is upper bounded by a constant and where this constant converges asymptotically to a smaller constant as the amount of data goes to infinity. Along with this result of convergence, we have also provided simulation results that have demonstrated that the proposed algorithm can achieve good performance when applied to both a linear and nonlinear system. In particular, with the linear systems our results indicate that the algorithm can converge to the optimal policy.

Contribution 2: Learning with cross-modal observations.

We have developed a distributed reinforcement learning algorithm for policy evaluation with cross-modal observations, where cross-modal observations means each agent observes a different nonlinear observation of the environment. The concept of cross-modal observations is motivated by scenario where agents have different sensors. Moreover, by considering cross-modal observations we address the challenge of output feedback (partial observability) since each agent cannot observe the full state. With this constraint, each agent can only compute part of the vector ϕ, which is needed for the policy evaluation algorithm. In order for the network to cooperate in the policy evaluation problem, each team needs to be assigned a column of a matrix $\mathbf{A}$ and each agent needs to be assigned a row. For a large number of rows, columns, and number of agents, this assignment problem can be very difficult to do efficiently. As such, part of our contribution is that we have reduced this assignment problem to a maximum matching problem. Through this reduction we can then achieve an assignment efficiently by using known and provably efficient algorithms for maximum matching. Along with an efficient assignment approach, our other contribution is that we have provided a distributed update for each agent, such that the network jointly computes an approximation to θ given finite amount of data. We have also proved that if each agent follows the proposed distributed algorithm, the network converges exponentially to the an approximation $\hat{\theta}$ and that this approximation converges to θ as the amount of data goes to infinity. Through simulations we have supported the theoretical results by illustrating that the squared error converges close to zero exponentially.

Contribution 3: Finite-sample analysis of MARL algorithms. We have studied two MARL algorithms and have provided a finite-sample analysis for both. The first algorithm we studied is a distributed version of the Q-learning algorithm, which combines consensus with local updates so that all agents in the multi-agent system can learn the same optimal policy. Where the optimal policy is optimal in the sense that if each agent follows it, then global average of individual rewards will be maximized. For a given diminishing step-size $\alpha(t)$, we have derived a finite-sample bound on the average of each agent's error. This result supports earlier works which have provided asymptotic convergence results for distributed Q-learning algorithms. Moreover, our finite-sample result characterizes the

convergence rate of distributed Q-learning without assuming independent and indentically distributed data samples, which is an important relaxation since in practice this assumption is not always true. Such a finite-sample analysis of distributed Q-learning had previously been missing in the MARL literature.

The second algorithm we studied is a distributed gradient temporal difference algorithm for policy evaluation, where a kernelized approximation of the value function is used. As in the distributed Q-learning algorithm, this algorithm uses a local weighted average of estimates to push towards consensus while also adding the local innovation of each agents estimate of the actual value function V_π. Under certain described conditions, we have derived an upper bound for the expected squared error for each agent, and have found that the error can approach zero as the step-size approaches zero. These bounds are defined for every time step and so provides a finite-sample bound. Furthermore, by considering a kernelized approximation of the value function, we have provided a theoretical result for a more general function approximation than the linear function approximations used in previous results.

Contribution 4: Application to a real-world system.

We have applied three popular reinforcement algorithms for the purpose of finding an optimal bidding strategy for an individual house in a transactive energy system. We evaluated these algorithms on the control performance of their resulting bidding strategies and their training performance. The three algorithms we have evaluated are Q-learning, DQN, and DDPG. From our simulations we have seen that DDPG and DQN far outperform tabular Q-learning, which is as expected since our environment consists of continuous states and tabular Q-learning is better suited for problems with discrete state spaces.This has shown the importance of algorithms that are designed for continuous states when applying reinforcement learning on real-world problems. From our simulations we have also found that DDPG and DQN have similar control performance, but with the difference that DDPG shows better training stability at least for a large number of iterations. Another part of this study was to explore the effect of partial observability in real-world problem. As such we trained a DDPG agent with a non-competitive scenario, where the environment is affected by others agents. Since the DDPG agent does not observe the actions of other agents, this means the environment is partially observable. This experiment showed us that the DDPG agent has

a degree of robustness to partial observability, but it also showed us that for low number of agents, where each agent affects the environment more, then the DDPG agent's performance suffers more due to the partial observability.

Contribution 5: Distributed observer for state estimation with incomplete model of dynamics.

We have developed a distributed observer algorithm based on the unscented Kalman filter in order to address the challenge of output feedback (partial observability) in MARL. The proposed distributed observer is designed for both parameter and state estimation so that a complete model of state dynamics is not needed. The developed observer leverages the dependence of predicted state dynamics with unknown parameters in order use state estimation to improve parameter estimation. We have derived an upper bound on estimation errors of the proposed algorithm, and have shown that the upper bound converges to a finite value given that certain conditions are satisfied. We have also provided simulation results that show that the proposed algorithm can successfully estimate the state and parameters of a non-linear system with measurement noise. The developed observer is shown to reliably track the states, but cannot be guaranteed to perfectly track the state since the system is stochastic due to the measurement noise. As such, our theoretical result also tells us that if the measurement noise is zero, then as $t \to \infty$, zero estimation error can be guaranteed. The development of this distributed observer addresses the challenge of partial observability in MARL by providing a way for agents to estimate the full state vector given noisy measurements and and an incomplete, parameterized model of state dynamics.

9.1 Future Directions

In this section we outline some possible future research directions in the field of MARL, as well as some possible extensions for the work presented in this thesis.

Topic 1: Resilience for MARL

An interesting topic to pursue in future research is how to improve resilience of MARL. By resilience we mean the ability of MARL algorithms to maintain some level of performance even when subjected to some form of attack. The attack of type will determine how to

approach the problem of resilience. One such attack type is the corruption of data, where all agents are still controllable but the data has been maliciously altered. In this case we could consider how to recover some part of the data or we could consider the possibility that the data is fully corrupted. In the latter case a more conservative approach would probably be needed, whereas the former could potentially still guarantee optimality. Another attack type to explore is the case where some agents are uncontrollable and act maliciously. In this case the goal would be to isolate or minimize the effect these malicious agents have on the rest of the network. These are just a few scenarios and questions that could be explored in this topic.

Topic 2: Analysis of Deep Learning for MARL

In this work we have provided theoretical results of MARL using linear function approximation and kernelized approximations, but these results do not necessarily hold when we consider function approximation with deep neural networks. Deep neural networks provide a very powerful tool for approximating functions. In fact, the impressive empirical results from the combination of deep neural networks with reinforcement learning has popularized reinforcement learning, and machine learning in general. However, due to the high complexity of deep neural networks, it has proven difficult to analyze their behaviour when applied with learning algorithms, and so there are very few theoretical results for reinforcement learning with deep neural networks. As such, any theoretical results that could be established for the combination of reinforcement learning and deep neural networks would be of great value to the field, and even more so if these results could be extended to the multi-agent setting.

Topic 3: Offline MARL with Partial Observability

In this thesis we have developed distributed algorithms for the offline reinforcement setting and for the partial observability setting. A natural follow up would be to consider the offline setting where each agent has the limitation of partial observability, where partial observability could be such that each agent observes some non-linear observation of the state. This would further improve the applicability of MARL to real-world problems since many times we do not have the full state of the environment but instead only have sensor measurements. A possible approach for such a setting would be to incorporate distributed state estimation methods such as the one proposed in chapter 8, or we could even incorporate

deep neural networks so that we could accomplish state estimation without the need for a parameterized model of state dynamics. Alternatively, if we assume linear function approximation we could use an approach similar to 4 where we make use of this linear structure in order to develop a distributed update for each agent, where each agent only has access to its local observations and can only communicate with neighbors.

REFERENCES

[1] C. Chen, A. Seff, A. Kornhauser, and J. Xiao, "Deepdriving: Learning affordance for direct perception in autonomous driving," in *Proceedings of the IEEE International Conference on Computer Vision*, 2015, pp. 2722–2730.

[2] D. Silver, A. Huang, C. J. Maddison, A. Guez, L. Sifre, G. V. D. Driessche, J. Schrittwieser, I. Antonoglou, V. Panneershelvam, M. Lanctot, *et al.*, "Mastering the game of go with deep neural networks and tree search," *nature*, vol. 529, no. 7587, p. 484, 2016.

[3] V. Mnih, K. Kavukcuoglu, D. Silver, A. A. Rusu, J. Veness, M. G. Bellemare, A. Graves, M. Riedmiller, A. K. Fidjeland, G. Ostrovski, *et al.*, "Human-level control through deep reinforcement learning," *Nature*, vol. 518, no. 7540, p. 529, 2015.

[4] S. Gu, E. Holly, T. Lillicrap, and S. Levine, "Deep reinforcement learning for robotic manipulation with asynchronous off-policy updates," in *2017 IEEE international conference on robotics and automation (ICRA)*, IEEE, 2017, pp. 3389–3396.

[5] J. Peters, S. Vijayakumar, and S. Schaal, "Reinforcement learning for humanoid robotics," in *Proceedings of the third IEEE-RAS international conference on humanoid robots*, 2003, pp. 1–20.

[6] S. R. Sutton, "Learning to predict by the methods of temporal differences," *Machine learning*, vol. 3, no. 1, pp. 9–44, 1988.

[7] C. Watkins and P. Dayan, "Q-learning," *Machine learning*, vol. 8, no. 3-4, pp. 279–292, 1992.

[8] C. Watkins, "Learning from delayed rewards," Ph.D. dissertation, King's College, Cambridge, 1989.

[9] G. A. Rummery and M. Niranjan, *On-line Q-learning using connectionist systems*. University of Cambridge, Department of Engineering Cambridge, England, 1994, vol. 37.

[10] V. Mnih, A. P. Badia, M. Mirza, A. Graves, T. Lillicrap, T. Harley, D. Silver, and K. Kavukcuoglu, "Asynchronous methods for deep reinforcement learning," in *International conference on machine learning*, 2016, pp. 1928–1937.

[11] Y. Li, "Deep reinforcement learning: An overview," *arXiv preprint arXiv:1701.07274*, 2017.

[12] A. Y. Ng and M. I. Jordan, "Shaping and policy search in reinforcement learning," Ph.D. dissertation, University of California, Berkeley Berkeley, 2003.

[13] F. L. Lewis, D. Vrabie, and K. G. Vamvoudakis, "Reinforcement learning and feedback control: Using natural decision methods to design optimal adaptive controllers," *IEEE Control Systems Magazine*, vol. 32, no. 6, pp. 76–105, 2012.

[14] L. Panait and S. Luke, "Cooperative multi-agent learning: The state of the art," *Autonomous agents and multi-agent systems*, vol. 11, no. 3, pp. 387–434, 2005.

[15] T. Balch, "Reward and diversity in multirobot foraging," 1999.

[16] T. Balch *et al.*, "Learning roles: Behavioral diversity in robot teams," *College of Computing Technical Report GIT-CC-97-12, Georgia Institute of Technology, Atlanta, Georgia*, vol. 73, 1997.

[17] M. J. Mataric, "Learning to behave socially," in *Third international conference on simulation of adaptive behavior*, vol. 617, 1994, pp. 453–462.

[18] Y.-H. Chang, T. Ho, and L. P. Kaelbling, "All learning is local: Multi-agent learning in global reward games," in *Advances in neural information processing systems*, 2004, pp. 807–814.

[19] S. Ontañón and E. Plaza, "A bartering approach to improve multiagent learning," in *Proceedings of the first international joint conference on Autonomous agents and multiagent systems: part 1*, ACM, 2002, pp. 386–393.

[20] D. H. Wolpert and K. Tumer, "Optimal payoff functions for members of collectives," in *Modeling complexity in economic and social systems*, World Scientific, 2002, pp. 355–369.

[21] P. Tangamchit, J. M. Dolan, and P. K. Khosla, "The necessity of average rewards in cooperative multirobot learning," in *Proceedings 2002 IEEE International Conference on Robotics and Automation (Cat. No. 02CH37292)*, IEEE, vol. 2, 2002, pp. 1296–1301.

[22] J. M. Vidal and E. H. Durfee, "The moving target function problem in multi-agent learning," in *Proceedings International Conference on Multi Agent Systems (Cat. No. 98EX160)*, IEEE, 1998, pp. 317–324.

[23] C. Claus and C. Boutilier, "The dynamics of reinforcement learning in cooperative multiagent systems," *AAAI/IAAI*, vol. 1998, no. 746-752, p. 2, 1998.

[24] M. Tan, "Multi-agent reinforcement learning: Independent vs. cooperative agents," in *Proceedings of the tenth international conference on machine learning*, 1993, pp. 330–337.

[25] S. Omidshafiei, J. Pazis, C. Amato, J. How, and J. Vian, "Deep decentralized multi-task multi-agent reinforcement learning under partial observability," in *Proceedings of the 34th International Conference on Machine Learning-Volume 70*, JMLR. org, 2017, pp. 2681–2690.

[26] M. Bowling, "Convergence problems of general-sum multiagent reinforcement learning," in *ICML*, 2000, pp. 89–94.

[27] J. Hu and M. Wellman, "Multiagent reinforcement learning: Theoretical framework and an algorithm.," in *ICML*, Citeseer, vol. 98, 1998, pp. 242–250.

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

[29] L. Busoniu, R. Babuska, and B. De Schutter, "A comprehensive survey of multiagent reinforcement learning," *IEEE Transactions on Systems, Man, and Cybernetics, Part C (Applications and Reviews)*, vol. 38, no. 2, pp. 156–172, 2008.

[30] C.-K. Tham and J.-C. Renaud, "Multi-agent systems on sensor networks: A distributed reinforcement learning approach," in *2005 International Conference on Intelligent Sensors, Sensor Networks and Information Processing*, IEEE, 2005, pp. 423–429.

[31] K. Zhang, Z. Yang, H. Liu, T. Zhang, and T. Basar, "Fully decentralized multi-agent reinforcement learning with networked agents," in *International Conference on Machine Learning*, 2018, pp. 5867–5876.

[32] H. 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.

[33] J. Schneider, W.-K. Wong, A. Moore, and M. Riedmiller, "Distributed value functions," in *ICML*, 1999, pp. 371–378.

[34] T. T. Doan, S. T. Maguluri, and J. Romberg, "Finite-time analysis of distributed td (0) with linear function approximation on multi-agent reinforcement learning," in *International Conference on Machine Learning*, 2019, pp. 1626–1635.

[35] T. T. Doan, S. T. Maguluri, and J. Romberg, "Finite-time performance of distributed temporal difference learning with linear function approximation," *arXiv preprint*, Jun. 2019.

[36] V. Mnih, K. Kavukcuoglu, D. Silver, A. Graves, I. Antonoglou, D. Wierstra, and M. Riedmiller, "Playing atari with deep reinforcement learning," *arXiv preprint arXiv:1312.5602*, 2013.

[37] A. Nair, P. Srinivasan, S. Blackwell, C. Alcicek, R. Fearon, A. De Maria, V. Panneershelvam, M. Suleyman, C. Beattie, S. Petersen, *et al.*, "Massively parallel methods for deep reinforcement learning," *arXiv preprint arXiv:1507.04296*, 2015.

[38] B. Dai, A. Shaw, L. Li, L. Xiao, N. He, Z. Liu, J. Chen, and L. Song, "Sbeed: Convergent reinforcement learning with nonlinear function approximation," in *ICML 2018*, 2018.

[39] D. Lee, H. Yoon, and N. Hovakimyan, "Primal-dual algorithm for distributed reinforcement learning: Distributed gtd," in *2018 IEEE Conference on Decision and Control (CDC)*, IEEE, 2018, pp. 1967–1972.

[40] S. Kar, J. M. F. Moura, and H. V. Poor, "Qd-learning: A collaborative distributed strategy for multi-agent reinforcement learning through consensus+innovations," *IEEE Transactions on Signal Processing*, vol. 61, no. 7, pp. 1848–1862, 2013.

[41] A. Mathkar and V. S. Borkar, "Distributed reinforcement learning via gossip," *IEEE Transactions on Automatic Control*, vol. 62, no. 3, pp. 1465–1470, 2017.

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

[43] J. Foerster, F. Song, E. Hughes, N. Burch, I. Dunning, S. Whiteson, M. Botvinick, and M. Bowling, "Bayesian action decoder for deep multi-agent reinforcement learning," in *International Conference on Machine Learning*, 2019, pp. 1942–1951.

[44] R. Lowe, Y. I. Wu, A. Tamar, J. Harb, O. P. Abbeel, and I. Mordatch, "Multi-agent actor-critic for mixed cooperative-competitive environments," in *Advances in neural information processing systems*, 2017, pp. 6379–6390.

[45] J. R. Kok and N. Vlassis, "Collaborative multiagent reinforcement learning by payoff propagation," *Journal of Machine Learning Research*, vol. 7, no. Sep, pp. 1789–1828, 2006.

[46] J. Gupta, M. Egorov, and M. Kochenderfer, "Cooperative multi-agent control using deep reinforcement learning," in *International Conference on Autonomous Agents and Multiagent Systems*, Springer, 2017, pp. 66–83.

[47] C. Zhang and V. R. Lesser, "Coordinated multi-agent reinforcement learning in networked distributed pomdps.," in *AAAI*, 2011.

[48] K. Wan, X. Gao, Z. Hu, and G. Wu, "Robust motion control for uav in dynamic uncertain environments using deep reinforcement learning," *Remote sensing*, vol. 12, no. 4, p. 640, 2020.

[49] M. Lopez-Martin, B. Carro, and A. Sanchez-Esguevillas, "Application of deep reinforcement learning to intrusion detection for supervised problems," *Expert Systems with Applications*, vol. 141, p. 112 963, 2020.

[50] N. C. Luong, D. T. Hoang, S. Gong, D. Niyato, P. Wang, Y.-C. Liang, and D. I. Kim, "Applications of deep reinforcement learning in communications and networking: A survey," *IEEE Communications Surveys & Tutorials*, vol. 21, no. 4, pp. 3133–3174, 2019.

[51] M. Mahmud, M. S. Kaiser, A. Hussain, and S. Vassanelli, "Applications of deep learning and reinforcement learning to biological data," *IEEE transactions on neural networks and learning systems*, vol. 29, no. 6, pp. 2063–2079, 2018.

[52] J. Duan, D. Shi, R. Diao, H. Li, Z. Wang, B. Zhang, D. Bian, and Z. Yi, "Deep-reinforcement-learning-based autonomous voltage control for power grid operations," *IEEE Transactions on Power Systems*, vol. 35, no. 1, pp. 814–817, 2019.

[53] Y. Ye, D. Qiu, M. Sun, D. Papadaskalopoulos, and G. Strbac, "Deep reinforcement learning for strategic bidding in electricity markets," *IEEE Transactions on Smart Grid*, vol. 11, no. 2, pp. 1343–1355, 2019.

[54] A. R. Mahmood, D. Korenkevych, G. Vasan, W. Ma, and J. Bergstra, "Benchmarking reinforcement learning algorithms on real-world robots," in *Conference on Robot Learning*, 2018, pp. 561–591.

[55] F. P. Such, V. Madhavan, R. Liu, R. Wang, P. S. Castro, Y. Li, J. Zhi, L. Schubert, M. G. Bellemare, J. Clune, *et al.*, "An atari model zoo for analyzing, visualizing, and comparing deep reinforcement learning agents," in *Proceedings of the 28th International Joint Conference on Artificial Intelligence*, AAAI Press, 2019, pp. 3260–3267.

[56] S. Fujimoto, D. Meger, and D. Precup, "Off-policy deep reinforcement learning without exploration," in *International Conference on Machine Learning*, 2019, pp. 2052–2062.

[57] C. Dann, G. Neumann, J. Peters, *et al.*, "Policy evaluation with temporal differences: A survey and comparison," *Journal of Machine Learning Research*, vol. 15, pp. 809–883, 2014.

[58] A. Tamar, H. Xu, and S. Mannor, "Scaling up robust mdps by reinforcement learning," *arXiv preprint arXiv:1306.6189*, 2013.

[59] A. G. Barto and S. Mahadevan, "Recent advances in hierarchical reinforcement learning," *Discrete event dynamic systems*, vol. 13, no. 1, pp. 41–77, 2003.

[60] L. Chen, B. Scherrer, and P. L. Bartlett, "Infinite-horizon offline reinforcement learning with linear function approximation: Curse of dimensionality and algorithm," *arXiv preprint arXiv:2103.09847*, 2021.

[61] D. Peteiro-Barral and B. Guijarro-Berdiñas, "A survey of methods for distributed machine learning," *Progress in Artificial Intelligence*, vol. 2, no. 1, pp. 1–11, 2013.

[62] M. Li, D. G. Andersen, J. W. Park, A. J. Smola, A. Ahmed, V. Josifovski, J. Long, E. J. Shekita, and B.-Y. Su, "Scaling distributed machine learning with the parameter server," in *11th USENIX Symposium on Operating Systems Design and Implementation (OSDI 14)*, 2014, pp. 583–598.

[63] T. Kraska, A. Talwalkar, J. C. Duchi, R. Griffith, M. J. Franklin, and M. I. Jordan, "Mlbase: A distributed machine-learning system.," in *Cidr*, vol. 1, 2013, pp. 2–1.

[64] J. Fu, A. Kumar, O. Nachum, G. Tucker, and S. Levine, "D4rl: Datasets for deep data-driven reinforcement learning," *arXiv preprint arXiv:2004.07219*, 2020.

[65] K. Arulkumaran, M. P. Deisenroth, M. Brundage, and A. A. Bharath, "Deep reinforcement learning: A brief survey," *IEEE Signal Processing Magazine*, vol. 34, no. 6, pp. 26–38, 2017.

[66] Z. Ning, P. Dong, X. Wang, M. S. Obaidat, X. Hu, L. Guo, Y. Guo, J. Huang, B. Hu, and Y. Li, "When deep reinforcement learning meets 5g-enabled vehicular networks: A distributed offloading framework for traffic big data," *IEEE Transactions on Industrial Informatics*, vol. 16, no. 2, pp. 1352–1361, 2019.

[67] P. Huang, J. Kalagnanam, R. Natarajan, D. J. Hammerstrom, R. Melton, M. Sharma, and R. Ambrosio, "Analytics and transactive control design for the Pacific Northwest Smart Grid Demonstration project," in *Proc. 2010 First IEEE International Conference on Smart Grid Communications*, Gaithersburg, MD, Oct. 2010, pp. 449–454. DOI: 10.1109/SMARTGRID.2010.5622083.

[68] S. Chen and C.-C. Liu, "From demand response to transactive energy: State of the art," *Journal of Modern Power Systems and Clean Energy*, vol. 5, no. 1, pp. 10–19, 2017.

[69] F. A. Rahimi and A. Ipakchi, "Transactive energy techniques: Closing the gap between wholesale and retail markets," *The Electricity Journal*, vol. 25, no. 8, pp. 29–35, 2012.

[70] K. Malialis, S. Devlin, and D. Kudenko, "Distributed reinforcement learning for adaptive and robust network intrusion response," *Connection Science*, vol. 27, no. 3, pp. 234–252, 2015.

[71] 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*, 2000, pp. 1057–1063.

[72] V. R. Konda and J. N. Tsitsiklis, "Actor-critic algorithms," in *Advances in neural information processing systems*, 2000, pp. 1008–1014.

[73] Z. Yang, K. Zhang, M. Hong, and T. Başar, "A finite sample analysis of the actor-critic algorithm," in *2018 IEEE Conference on Decision and Control (CDC)*, IEEE, 2018, pp. 2759–2764.

[74] K. Zhang, Z. Yang, and T. Basar, "Networked multi-agent reinforcement learning in continuous spaces," in *2018 IEEE Conference on Decision and Control (CDC)*, IEEE, 2018, pp. 2771–2776.

[75] R. S. Sutton and A. G. Barto, *Reinforcement learning: An introduction*. MIT press, 2018.

[76] V. Tadić, "On the convergence of temporal-difference learning with linear function approximation," *Machine learning*, vol. 42, no. 3, pp. 241–267, 2001.

[77] S. Bhatnagar, R. S. Sutton, M. Ghavamzadeh, and M. Lee, "Natural actor–critic algorithms," *Automatica*, vol. 45, no. 11, pp. 2471–2482, 2009.

[78] V. Borkar, "Stochastic approximation: A dynamical systems view," *Hindustan Publ. Co., New Delhi, India and Cambridge Uni. Press, Cambridge, UK*, 2008.

[79] H. J. Kushner and D. S. Clark, *Stochastic approximation methods for constrained and unconstrained systems*. Springer Science & Business Media, 2012, vol. 26.

[80] S. Levine, A. Kumar, G. Tucker, and J. Fu, "Offline reinforcement learning: Tutorial, review, and perspectives on open problems," *arXiv preprint arXiv:2005.01643*, 2020.

[81] J. Jiang and Z. Lu, "Offline decentralized multi-agent reinforcement learning," *arXiv preprint arXiv:2108.01832*, 2021.

[82] A. Kumar, A. Zhou, G. Tucker, and S. Levine, "Conservative q-learning for offline reinforcement learning," *arXiv preprint arXiv:2006.04779*, 2020.

[83] R. Agarwal, D. Schuurmans, and M. Norouzi, "An optimistic perspective on offline reinforcement learning," in *International Conference on Machine Learning*, PMLR, 2020, pp. 104–114.

[84] Y. Wu, G. Tucker, and O. Nachum, "Behavior regularized offline reinforcement learning," *arXiv preprint arXiv:1911.11361*, 2019.

[85] J. Fu, A. Kumar, M. Soh, and S. Levine, "Diagnosing bottlenecks in deep q-learning algorithms," in *International Conference on Machine Learning*, PMLR, 2019, pp. 2021–2030.

[86] A. Kumar, J. Fu, M. Soh, G. Tucker, and S. Levine, "Stabilizing off-policy q-learning via bootstrapping error reduction," *Advances in Neural Information Processing Systems*, vol. 32, pp. 11 784–11 794, 2019.

[87] S. Mou, M. Cao, and A. Morse, "Target-point formation control," *Automatica*, vol. 61, pp. 113–118, 2015.

[88] X. Chen, M. A. Belabbas, and T. Basar, "Controlling and stabilizing a rigid formation using a few agents," *SIAM Journal on Control and Optimization*, vol. 57, no. 1, pp. 104–128, 2019.

[89] J. Cortes, S. Martinez, T. Karatas, and F. Bullo, "Coverage control for mobile sensing networks," *IEEE Transactions on robotics and Automation*, vol. 20, no. 2, pp. 243–255, 2004.

[90] D. Xue, S. Hirche, and M. Cao, "Opinion behavior analysis in social networks under the influence of coopetitive media," *IEEE Transactions on Network Science and Engineering*, 2019, DOI: 10.1109/TNSE.2019.2894565.

[91] K. Zhang, Y. Liu, J. Liu, M. Liu, and T. Başar, "Distributed learning of average belief over networks using sequential observations," *Automatica*, vol. 115, p. 108 857, 2020.

[92] G. Qu, A. Wierman, and N. Li, "Scalable reinforcement learning of localized policies for multi-agent networked systems," *arXiv:1912.02906*, 2019.

[93] Y. Li, Y. Tang, R. Zhang, and N. Li, "Distributed reinforcement learning for decentralized linear quadratic control: A derivative-free policy optimization approach," *arXiv:1912.09135*, 2019.

[94] F. J. Provost and D. N. Hennessy, "Scaling up: Distributed machine learning with cooperation," in *AAAI/IAAI, Vol. 1*, Citeseer, 1996, pp. 74–79.

[95] J. Konečnỳ, H. B. McMahan, D. Ramage, and P. Richtárik, "Federated optimization: Distributed machine learning for on-device intelligence," *arXiv preprint arXiv:1610.02527*, 2016.

[96] J. Konečnỳ, H. B. McMahan, F. X. Yu, P. Richtárik, A. T. Suresh, and D. Bacon, "Federated learning: Strategies for improving communication efficiency," *arXiv preprint arXiv:1610.05492*, 2016.

[97] P. Kairouz, H. B. McMahan, B. Avent, A. Bellet, M. Bennis, A. N. Bhagoji, K. Bonawitz, Z. Charles, G. Cormode, R. Cummings, *et al.*, "Advances and open problems in federated learning," *Foundations and Trends® in Machine Learning*, vol. 14, no. 1–2, pp. 1–210, 2021.

[98] J. Qi, Q. Zhou, L. Lei, and K. Zheng, "Federated reinforcement learning: Techniques, applications, and open challenges," *arXiv preprint arXiv:2108.11887*, 2021.

[99] F. S. Melo, S. P. Meyn, and M. I. Ribeiro, "An analysis of reinforcement learning with function approximation," in *Proceedings of the 25th international conference on Machine learning*, ACM, 2008, pp. 664–671.

[100] F. Lu, P. G. Mehta, S. P. Meyn, and G. Neu, "Convex q-learning," in *2021 American Control Conference (ACC)*, IEEE, 2021, pp. 4749–4756.

[101] V. Borkar, S. Chen, A. Devraj, I. Kontoyiannis, and S. Meyn, "The ode method for asymptotic statistics in stochastic approximation and reinforcement learning," *arXiv preprint arXiv:2110.14427*, 2021.

[102] S. D. McDougle, M. J. Boggess, M. J. Crossley, D. Parvin, R. B. Ivry, and J. A. Taylor, "Credit assignment in movement-dependent reinforcement learning," *Proceedings of the National Academy of Sciences*, vol. 113, no. 24, pp. 6797–6802, 2016.

[103] T. Mesnard, T. Weber, F. Viola, S. Thakoor, A. Saade, A. Harutyunyan, W. Dabney, T. Stepleton, N. Heess, A. Guez, *et al.*, "Counterfactual credit assignment in model-free reinforcement learning," *arXiv preprint arXiv:2011.09464*, 2020.

[104] T. Jaakkola, S. Singh, and M. Jordan, "Reinforcement learning algorithm for partially observable markov decision problems," *Advances in neural information processing systems*, vol. 7, 1994.

[105] M. T. Spaan, "Partially observable markov decision processes," in *Reinforcement Learning*, Springer, 2012, pp. 387–414.

[106] 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.

[107] P. C. Heredia and S. Mou, "Distributed multi-agent reinforcement learning by actor-critic method," *IFAC-PapersOnLine*, vol. 52, no. 20, pp. 363–368, 2019.

[108] M. Zhou, Z. Liu, P. Sui, Y. Li, and Y. Y. Chung, "Learning implicit credit assignment for cooperative multi-agent reinforcement learning," *Advances in Neural Information Processing Systems*, vol. 33, pp. 11 853–11 864, 2020.

[109] Y. Du, L. Han, M. Fang, J. Liu, T. Dai, and D. Tao, "Liir: Learning individual intrinsic reward in multi-agent reinforcement learning," *Advances in Neural Information Processing Systems*, vol. 32, pp. 4403–4414, 2019.

[110] P. Mannion, S. Devlin, J. Duggan, and E. Howley, "Reward shaping for knowledge-based multi-objective multi-agent reinforcement learning," *The Knowledge Engineering Review*, vol. 33, 2018.

[111] J. Sakuma, S. Kobayashi, and R. N. Wright, "Privacy-preserving reinforcement learning," in *Proceedings of the 25th international conference on Machine learning*, 2008, pp. 864–871.

[112] G. Vietri, B. Balle, A. Krishnamurthy, and S. Wu, "Private reinforcement learning with pac and regret guarantees," in *International Conference on Machine Learning*, PMLR, 2020, pp. 9754–9764.

[113] P. Heredia, H. Ghadialy, and S. Mou, "Finite-sample analysis of distributed q-learning for multi-agent networks," in *2020 American Control Conference (ACC)*, IEEE, 2020, pp. 3511–3516.

[114] A. Rahate, R. Walambe, S. Ramanna, and K. Kotecha, "Multimodal co-learning: Challenges, applications with datasets, recent advances and future directions," *Information Fusion*, vol. 81, pp. 203–239, 2022.

[115] P. P. Liang, P. Wu, L. Ziyin, L.-P. Morency, and R. Salakhutdinov, "Cross-modal generalization: Learning in low resource modalities via meta-alignment," in *Proceedings of the 29th ACM International Conference on Multimedia*, 2021, pp. 2680–2689.

[116] L. Zhen, P. Hu, X. Peng, R. S. M. Goh, and J. T. Zhou, "Deep multimodal transfer learning for cross-modal retrieval," *IEEE Transactions on Neural Networks and Learning Systems*, 2020.

[117] G. E. Monahan, "State of the art—a survey of partially observable markov decision processes: Theory, models, and algorithms," *Management science*, vol. 28, no. 1, pp. 1–16, 1982.

[118] K. Zhang, Z. Yang, and T. Başar, "Multi-agent reinforcement learning: A selective overview of theories and algorithms," *Handbook of Reinforcement Learning and Control*, pp. 321–384, 2021.

[119] S. Guicheng and W. Yang, "Review on dec-pomdp model for marl algorithms," in *Smart Communications, Intelligent Algorithms and Interactive Methods*, Springer, 2022, pp. 29–35.

[120] F. L. Lewis and K. G. Vamvoudakis, "Reinforcement learning for partially observable dynamic processes: Adaptive dynamic programming using measured output data," *IEEE Transactions on Systems, Man, and Cybernetics, Part B (Cybernetics)*, vol. 41, no. 1, pp. 14–25, 2010.

[121] A. Rodriguez, R. Parr, and D. Koller, "Reinforcement learning using approximate belief states," *Advances in Neural Information Processing Systems*, vol. 12, 1999.

[122] S. Ross, B. Chaib-draa, and J. Pineau, "Bayesian reinforcement learning in continuous pomdps with application to robot navigation," in *2008 IEEE International Conference on Robotics and Automation*, IEEE, 2008, pp. 2845–2851.

[123] G. Singh, S. Peri, J. Kim, H. Kim, and S. Ahn, "Structured world belief for reinforcement learning in pomdp," in *International Conference on Machine Learning*, PMLR, 2021, pp. 9744–9755.

[124] T. Doan, S. Maguluri, and J. Romberg, "Finite-time analysis of distributed td (0) with linear function approximation on multi-agent reinforcement learning," in *International Conference on Machine Learning*, 2019, pp. 1626–1635.

[125] J. E. Hopcroft and R. M. Karp, "An n5/2 algorithm for maximum matchings in bipartite graphs," *SIAM Journal on computing*, vol. 2, no. 4, pp. 225–231, 1973.

[126] X. Wang, S. Mou, and B. D. Anderson, "A distributed algorithm with scalar states for solving linear equations," in *2018 IEEE Conference on Decision and Control (CDC)*, IEEE, 2018, pp. 2861–2865.

[127] F. L. Lewis and D. Vrabie, "Reinforcement learning and adaptive dynamic programming for feedback control," *IEEE Circuits and Systems Magazine*, pp. 32–50, 2009.

[128] X. Chen, M. A. Belabbas, and T. Basar, "Controlling and stabilizing a rigid formation using a few agents," *SIAM Journal on Control and Optimization*, vol. 57, no. 1, pp. 104–128, 2019.

[129] W. Suttle, Z. Yang, K. Zhang, Z. Wang, T. Basar, and J. Liu, "A multi-agent off-policy actor-critic algorithm for distributed reinforcement learning," *arXiv preprint arXiv:1903.06372*, 2019.

[130] Y. Lin, K. Zhang, Z. Yang, Z. Wang, T. Başar, R. Sandhu, and J. Liu, "A communication-efficient multi-agent actor-critic algorithm for distributed reinforcement learning," in *2019 IEEE Conference on Decision and Control (CDC)*, IEEE, 2019.

[131] M. Littman, "Markov games as a framework for multi-agent reinforcement learning," in *Machine learning proceedings 1994*, Elsevier, 1994, pp. 157–163.

[132] C. Zhang and V. Lesser, "Coordinating multi-agent reinforcement learning with limited communication," in *Proceedings of the 2013 international conference on Autonomous agents and multi-agent systems*, International Foundation for Autonomous Agents and Multiagent Systems, 2013, pp. 1101–1108.

[133] S. Kapetanakis and D. Kudenko, "Reinforcement learning of coordination in cooperative multi-agent systems," *AAAI/IAAI*, vol. 2002, pp. 326–331, 2002.

[134] M. Lauer and M. Riedmiller, "An algorithm for distributed reinforcement learning in cooperative multi-agent systems," in *In Proceedings of the Seventeenth International Conference on Machine Learning*, Citeseer, 2000.

[135] T. Kemmerich and H. Büning, "A convergent multiagent reinforcement learning approach for a subclass of cooperative stochastic games," in *International Workshop on Adaptive and Learning Agents*, Springer, 2011, pp. 37–53.

[136] J. Foerster, N. Nardelli, G. Farquhar, T. Afouras, P. H. Torr, P. Kohli, and S. Whiteson, "Stabilising experience replay for deep multi-agent reinforcement learning," in *Proceedings of the 34th International Conference on Machine Learning-Volume 70*, JMLR. org, 2017, pp. 1146–1155.

[137] J. Foerster, I. Assael, N. Freitas, and S. Whiteson, "Learning to communicate with deep multi-agent reinforcement learning," in *Advances in Neural Information Processing Systems*, 2016, pp. 2137–2145.

[138] K. Zhang, Z. Yang, H. Liu, T. Zhang, and T. Başar, "Finite-sample analyses for fully decentralized multi-agent reinforcement learning," *arXiv preprint arXiv:1812.02783*, 2018.

[139] M. J. Kearns and S. P. Singh, "Finite-sample convergence rates for q-learning and indirect algorithms," in *Advances in neural information processing systems*, 1999, pp. 996–1002.

[140] D. Shah and Q. Xie, "Q-learning with nearest neighbors," in *Advances in Neural Information Processing Systems*, 2018, pp. 3111–3121.

[141] J. Bhandari, D. Russo, and R. Singal, "A finite time analysis of temporal difference learning with linear function approximation," *arXiv:1806.02450 [cs, stat]*, Jun. 2018. arXiv: 1806.02450.

[142] S. Zou, T. Xu, and Y. Liang, "Finite-sample analysis for SARSA and q-learning with linear function approximation," *arXiv:1902.02234 [cs, stat]*, Feb. 2019. arXiv: 1902.02234.

[143] Z. Chen, S. Zhang, T. T. Doan, S. T. Maguluri, and J. P. Clarke, "Finite-time analysis of q-learning with linear function approximation," *arXiv Preprint*, May 2019. arXiv: 1905.11425 [cs, math].

[144] A. Nedic, A. Ozdaglar, and P. Parrilo, "Constrained consensus and optimization in multi-agent networks," *IEEE Transactions on Automatic Control*, vol. 55, no. 4, pp. 922–938, 2010.

[145] X. Wang, J. Zhou, S. Mou, and M. Corless, "A distributed algorithm for least squares solutions," *IEEE Transactions on Automatic Control*, 2019.

[146] D. Lee and J. Hu, "Primal-Dual Q-learning framework for LQR design," *IEEE Transactions on Automatic Control*, vol. 64, no. 9, pp. 32–50, 2019.

[147] S. Mou, J. Liu, and A. S. Morse, "A distributed algorithm for solving a linear algebraic equation," *IEEE Transactions on Automatic Control*, vol. 60, no. 11, pp. 2863–2878, 2015.

[148] Z. Sun, S. Mou, B. D. O. Anderson, and A. S. Morse, "Rigid motions of 3-d undirected formations with mismatch between desired distances," *IEEE Transactions on Automatic Control*, vol. 62, no. 8, pp. 4151–4158, 2017.

[149] C. Yan and H. Fang, "Observer-based leader-follower tracking control for high-order multi-agent systems with limited measurement information," *Proceedings of the 58th IEEE Conference on Decision and Control*, pp. 3904–3909, 2019.

[150] P. C. Heredia and S. Mou, "Distributed multi-agent reinforcement learning by actor-critic method," *The 8th IFAC Workshop on Distributed Estimation and Control in Networked Systems (NECSYS)*, pp. 363–368, 2019.

[151] T. Chen, K. Zhang, G. B. Giannakis, and T. Başar, "Communication-efficient distributed reinforcement learning," *arXiv preprint arXiv:1812.03239*, 2018.

[152] M. Tavakol, M. N. Ahmadabadi, M. Mirian, and M. Asadpour, "Distributed q-learning approach for variable attention to multiple critics," *International Conference on Neural Information Processing*, pp. 244–251, 2012.

[153] P. C. Heredia, H. Ghadialy, and S. Mou, "Finite-sample analysis of distributed q-learning for multi-agent networks," *Proceedings of American Control Conference*, pp. 3511–3516, 2020.

[154] R. Srikant and L. Ying, "Finite-time error bounds for linear stochastic approximation and TD learning," *arXiv:1902.00923 [cs, stat]*, Feb. 2019. arXiv: 1902.00923. [Online]. Available: http://arxiv.org/abs/1902.00923.

[155] J. A. Bagnell and J. Schneider, "Policy search in kernel hilbert space," 2003.

[156] S. Paternain, J. Bazerque, A. Small, and A. Ribeiro, "Stochastic policy gradient ascent in reproducing kernel hilbert spaces," *arXiv preprint arXiv:1807.11274*, 2018.

[157] S. Paternain, J. A. Bazerque, A. Small, and A. Ribeiro, "Policy improvement directions for reinforcement learning in reproducing kernel hilbert spaces," in *2019 IEEE 58th Conference on Decision and Control (CDC)*, IEEE, 2019, pp. 7454–7461.

[158] G. Lever and R. Stafford, "Modelling policies in mdps in reproducing kernel hilbert space," in *Artificial Intelligence and Statistics*, 2015, pp. 590–598.

[159] N. A. Vien, P. Englert, and M. Toussaint, "Policy search in reproducing kernel hilbert space," in *Proceedings of the Twenty-Fifth International Joint Conference on Artificial Intelligence*, AAAI Press, 2016, pp. 2089–2096.

[160] A. Koppel, S. Paternain, C. Richard, and A. Ribeiro, "Decentralized Online Learning With Kernels," *IEEE Transactions on Signal Processing*, vol. 66, no. 12, pp. 3240–3255, Jun. 2018, ISSN: 1053-587X. DOI: 10.1109/TSP.2018.2830299.

[161] A. Koppel, K. Zhang, H. Zhu, and T. Başar, "Projected stochastic primal-dual method for constrained online learning with kernels," *IEEE Transactions on Signal Processing*, vol. 67, no. 10, pp. 2528–2542, May 2019, ISSN: 1053-587X. DOI: 10.1109/TSP.2019.2907265.

[162] B. Schölkopf, R. Herbrich, and A. Smola, "A generalized representer theorem," in *International conference on computational learning theory*, Springer, 2001, pp. 416–426.

[163] A. Nedić, A. Olshevsky, and M. G. Rabbat, "Network topology and communication-computation tradeoffs in decentralized optimization," *Proceedings of the IEEE*, vol. 106, no. 5, pp. 953–976, 2018.

[164] V. Popovici, S. Bengio, and J. Thiran, "Kernel matching pursuit for large datasets," *Pattern Recognition*, vol. 38, no. 12, pp. 2385–2390, Dec. 2005, ISSN: 0031-3203. DOI: 10.1016/j.patcog.2005.01.021.

[165] J. Lian, H. Ren, Y. Sun, and D. J. Hammerstrom, "Performance evaluation for transactive energy systems using double-auction market," *IEEE Transactions on Power Systems*, vol. 34, no. 5, pp. 4128–4137, 2018.

[166] A. K. Bejestani, A. Annaswamy, and T. Samad, "A hierarchical transactive control architecture for renewables integration in smart grids: Analytical modeling and stability," *IEEE Transactions on Smart Grid*, vol. 5, no. 4, pp. 2054–2065, Jul. 2014, ISSN: 1949-3053. DOI: 10.1109/TSG.2014.2325575.

[167] I. Namatēvs, "Deep reinforcement learning on hvac control.," *Information Technology & Management Science (RTU Publishing House)*, vol. 21, 2018.

[168] T. Wei, Y. Wang, and Q. Zhu, "Deep reinforcement learning for building hvac control," in *Proceedings of the 54th Annual Design Automation Conference 2017*, 2017, pp. 1–6.

[169] L. Yu, Y. Sun, Z. Xu, C. Shen, D. Yue, T. Jiang, and X. Guan, "Multi-agent deep reinforcement learning for hvac control in commercial buildings," *IEEE Transactions on Smart Grid*, 2020.

[170] W. Zhang, J. Lian, C.-Y. Chang, and K. Kalsi, "Aggregated modeling and control of air conditioning loads for demand response," *IEEE transactions on power systems*, vol. 28, no. 4, pp. 4655–4664, 2013.

[171] Y. Sun, A. Somani, and T. E. Carroll, "Learning based bidding strategy for hvac systems in double auction retail energy markets," in *2015 American Control Conference (ACC)*, IEEE, 2015, pp. 2912–2917.

[172] B. Liu, A. Murat, and T. McDermott, "Automated control of transactive hvacs in energy distribution systems," 2020.

[173] Y. Duan, X. Chen, R. Houthooft, J. Schulman, and P. Abbeel, "Benchmarking deep reinforcement learning for continuous control," in *International Conference on Machine Learning*, 2016, pp. 1329–1338.

[174] G. Brockman, V. Cheung, L. Pettersson, J. Schneider, J. Schulman, J. Tang, and W. Zaremba, "Openai gym," *arXiv preprint arXiv:1606.01540*, 2016.

[175] P. Henderson, R. Islam, P. Bachman, J. Pineau, D. Precup, and D. Meger, "Deep reinforcement learning that matters," *arXiv preprint arXiv:1709.06560*, 2017.

[176] J.-P. Corfmat and A. S. Morse, "Decentralized control of linear multivariable systems," *Automatica*, vol. 12, no. 5, pp. 479–495, 1976.

[177] J. S. Reed and P. A. Ioannou, "Discrete-time decentralized adaptive control," *Automatica*, vol. 24, no. 3, pp. 419–421, 1988.

[178] H. Khalil and A. Saberi, "Decentralized stabilization of nonlinear interconnected systems using high-gain feedback," *IEEE Transactions on Automatic Control*, vol. 27, no. 1, pp. 265–268, 1982.

[179] R. Olfati-Saber and J. S. Shamma, "Consensus filters for sensor networks and distributed sensor fusion," in *Decision and Control, 2005 and 2005 European Control Conference. CDC-ECC'05. 44th IEEE Conference on*, IEEE, 2005, pp. 6698–6703.

[180] H. P. Moravec, "Sensor fusion in certainty grids for mobile robots," *AI magazine*, vol. 9, no. 2, p. 61, 1988.

[181] D. Reid *et al.*, "An algorithm for tracking multiple targets," *IEEE transactions on Automatic Control*, vol. 24, no. 6, pp. 843–854, 1979.

[182] Y. Bar-Shalom and X.-R. Li, *Multitarget-multisensor tracking: principles and techniques*. YBs London, UK: 1995, vol. 19.

[183] L. Schenato, B. Sinopoli, M. Franceschetti, K. Poolla, and S. S. Sastry, "Foundations of control and estimation over lossy networks," *Proceedings of the IEEE*, vol. 95, no. 1, pp. 163–187, 2007.

[184] A. Mitra and S. Sundaram, "Distributed observers for lti systems," *arXiv preprint arXiv:1608.01429*, 2016.

[185] B. Rao, H. F. Durrant-Whyte, and J. Sheen, "A fully decentralized multi-sensor system for tracking and surveillance," *The International Journal of Robotics Research*, vol. 12, no. 1, pp. 20–44, 1993.

[186] R. Olfati-Saber, "Distributed kalman filtering for sensor networks," in *Decision and Control, 2007 46th IEEE Conference on*, IEEE, 2007, pp. 5492–5498.

[187] W. Han, H. L. Trentelman, Z. Wang, and Y. Shen, "A simple approach to distributed observer design for linear systems," *arXiv preprint arXiv:1708.01459*, 2017.

[188] Y. Hong, G. Chen, and L. Bushnell, "Distributed observers design for leader-following control of multi-agent networks," *Automatica*, vol. 44, no. 3, pp. 846–850, 2008.

[189] A. Chakrabarty, S. Sundaram, M. J. Corless, G. T. Buzzard, S. H. Żak, and A. E. Rundell, "Distributed unknown input observers for interconnected nonlinear systems," in *American Control Conference (ACC), 2016*, IEEE, 2016, pp. 101–106.

[190] O. Hlinka, F. Hlawatsch, and P. M. Djuric, "Distributed particle filtering in agent networks: A survey, classification, and comparison," *IEEE Signal Processing Magazine*, vol. 30, no. 1, pp. 61–81, 2013.

[191] E. A. Wan and R. Van Der Merwe, "The unscented kalman filter for nonlinear estimation," in *Adaptive Systems for Signal Processing, Communications, and Control Symposium 2000. AS-SPCC. The IEEE 2000*, Ieee, 2000, pp. 153–158.

[192] T. A. Wenzel, K. Burnham, M. Blundell, and R. Williams, "Dual extended kalman filter for vehicle state and parameter estimation," *Vehicle System Dynamics*, vol. 44, no. 2, pp. 153–171, 2006.

[193] G. Battistelli and L. Chisci, "Stability of consensus extended kalman filter for distributed state estimation," *Automatica*, vol. 68, pp. 169–178, 2016.

[194] W. Li, Y. Jia, and J. Du, "Distributed extended kalman filter with nonlinear consensus estimate," *Journal of the Franklin Institute*, vol. 354, no. 17, pp. 7983–7995, 2017.

[195] T. Karvonen *et al.*, "Stability of linear and non-linear kalman filters," 2014.

[196] K. Xiong, H. Zhang, and C. Chan, "Performance evaluation of ukf-based nonlinear filtering," *Automatica*, vol. 42, no. 2, pp. 261–270, 2006.

[197] L. Li and Y. Xia, "Stochastic stability of the unscented kalman filter with intermittent observations," *Automatica*, vol. 48, no. 5, pp. 978–981, 2012.

[198] Y. Niu and L. Sheng, "Distributed consensus-based unscented kalman filtering with missing measurements," in *Control Conference (CCC), 2017 36th Chinese*, IEEE, 2017, pp. 8993–8998.

[199] W. Li, G. Wei, F. Han, and Y. Liu, "Weighted average consensus-based unscented kalman filtering," *IEEE transactions on cybernetics*, vol. 46, no. 2, pp. 558–567, 2016.

[200] S. Haykin, *Kalman filtering and neural networks*. John Wiley & Sons, 2004, vol. 47.

[201] R. Van Der Merwe and E. A. Wan, "The square-root unscented kalman filter for state and parameter-estimation," in *Acoustics, Speech, and Signal Processing, 2001. Proceedings.(ICASSP'01). 2001 IEEE International Conference on*, IEEE, vol. 6, 2001, pp. 3461–3464.

[202] E. A. Wan and A. T. Nelson, "Dual extended kalman filter methods," *Kalman filtering and neural networks*, vol. 123, 2001.

[203] A. T. Nelson, "Nonlinear estimation and modeling of noisy time-series by dual kalman filtering methods," *Doctor of Philosopy, Oregon Graduate Institute of Science and Technology*, 2000.

[204] T.-J. Tarn and Y. Rasis, "Observers for nonlinear stochastic systems," *IEEE Transactions on Automatic Control*, vol. 21, no. 4, pp. 441–448, 1976.

[205] K. Reif, S. Gunther, E. Yaz, and R. Unbehauen, "Stochastic stability of the discrete-time extended kalman filter," *IEEE Transactions on Automatic control*, vol. 44, no. 4, pp. 714–728, 1999.

[206] G. Battistelli and L. Chisci, "Kullback–leibler average, consensus on probability densities, and distributed state estimation with guaranteed stability," *Automatica*, vol. 50, no. 3, pp. 707–718, 2014.