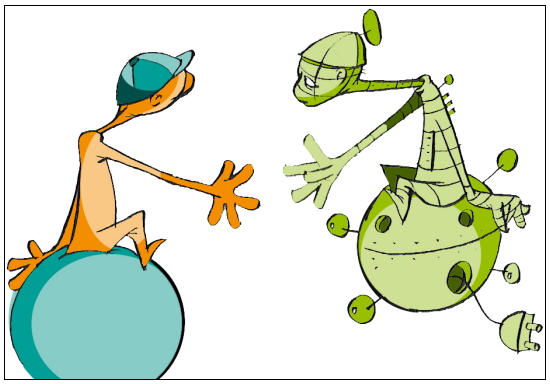
UNIVERSIDADE TÉCNICA DE LISBOA
INSTITUTO SUPERIOR TÉCNICO



Reinforcement Learning in Cooperative Navigation Tasks

Francisco António Chaves Saraiva de Melo
(Licenciado)

Dissertação para a obtenção do Grau de Doutor em
Engenharia Electrotécnica e de Computadores

Orientador: Prof. Doutora Maria Isabel Lobato de Faria Ribeiro

Júri:

Presidente: Reitor da Universidade Técnica de Lisboa

Vogais: Prof. Doutora Manuela M. Veloso, Carnegie-Mellon University
Prof. Doutor Fernando Lobo Pereira, Universidade do Porto
Prof. Doutora M. Isabel Ribeiro, Instituto Superior Técnico
Prof. Doutor Arlindo Oliveira, Instituto Superior Técnico
Prof. Doutor Pedro A. Lima, Instituto Superior Técnico
Doutor Matthijs T.J. Spaan, Instituto de Sistemas e Robótica

Novembro de 2007

UNIVERSIDADE TÉCNICA DE LISBOA
INSTITUTO SUPERIOR TÉCNICO

Reinforcement Learning in Cooperative Navigation Tasks

Francisco António Chaves Saraiva de Melo

Thesis submitted in partial fulfillment of
the requirements for the degree of
Doctor of Philosophy

November 2007

Advisor: Prof. Dr. M. Isabel Ribeiro

Thesis Committee:

Chair: Reitor da Universidade Técnica de Lisboa

Prof. Dr. Manuela M. Veloso, Carnegie-Mellon University
Prof. Dr. Fernando Lobo Pereira, Universidade do Porto
Prof. Dr. Arlindo Oliveira, Instituto Superior Técnico
Prof. Dr. Pedro A. Lima, Instituto Superior Técnico
Dr. Matthijs T.J. Spaan, Instituto de Sistemas e Robótica

Typeset in Computer Modern using TEX and LATEX 2ε

This work was partially supported by Programa Operacional Sociedade do Conhecimento (POS_C) (former Programa Operacional Sociedade de Informação – POSI) that includes FEDER funds and by the PhD grant SFRH/BD/3074/2000.

Resumo

Nesta tese aborda-se o problema da navegação robótica autónoma sob o ponto de vista da aprendizagem. Abordam-se situações nas quais um robô ou grupo de robôs tem que navegar de uma localização/configuração inicial até uma localização/configuração final. A principal contribuição da tese consiste na introdução e estudo de um conjunto de novos métodos de aprendizagem por reforço e da sua aplicação a problemas de navegação robótica. Em particular, a tese propõe métodos de aprendizagem por reforço mais gerais do que os existentes na bibliografia, que permitem implementar aprendizagem e coordenação em problemas com espaços de estado infinitos ou com observabilidade parcial, sendo este o cenário mais típico de aplicação em robótica móvel. Descrevem-se ainda os resultados obtidos ao aplicar os métodos desenvolvidos na tese a problemas concretos de navegação topológica com um/múltiplos robô(s) e conclui-se acerca das vantagens e limitações da abordagem seguida neste tipo de problemas.

Nas últimas décadas, assistiu-se a um aumento substancial do interesse na navegação de robôs autónomos. Esta tornou-se um tópico fundamental de investigação em Robótica. A tese dá uma contribuição neste abrangente campo de investigação. O trabalho que aqui se descreve divide-se em duas partes fundamentais. Consideram-se numa primeira parte situações em que um único robô se desloca num determinado ambiente, procurando navegar de uma localização inicial até uma localização final; na segunda parte considera-se então a existência de múltiplos robôs a navegar num ambiente comum. Estes problemas de navegação robótica são abordados na tese de um ponto de vista da aprendizagem, adoptando-se o formalismo usual da aprendizagem por reforço.

Na primeira parte da tese, abordam-se tarefas de navegação nas quais um único robô se deve deslocar de uma localização inicial para uma localização final. O ambiente é descrito por um mapa topológico finito, representado por um grafo no qual os nós representam estados ("locais") no ambiente e as arestas representam as ligações entre esses vários locais. O robô deverá ser capaz de explorar com sucesso

o ambiente e, através do *feedback* recebido durante a fase de exploração, aprender a desempenhar a tarefa de forma óptima. Para tal, a tese descreve o problema de navegação como um problema de decisão sequencial e descreve um conjunto de algoritmos clássicos de aprendizagem por reforço, como sejam o Q-learning e o SARSA, que são adequados para abordar problemas de navegação simples, nos quais o robô se desloca com percepção perfeita do estado.

Porém, de um ponto de vista prático, os sensores utilizados na navegação robótica apresentam geralmente erros de medida/processamento e, portanto, a localização de um robô terá sempre uma incerteza associada. Nessa situação, o robô deverá ter em conta essa incerteza no processo de decisão/escolha de acções. Na presença desta observabilidade parcial, o problema de decisão pode ser transformado num outro problema de decisão equivalente, mas com um espaço de estados infinito. A tese propõe então dois novos algoritmos de aprendizagem, designados *Approximate Q-Learning* e *Approximate SARSA*, adequados a problemas com espaço de estados infinitos. Estuda-se a convergência de ambos métodos, identificando condições nas quais esta é garantida que assentam essencialmente nas propriedades de ergodicidade do sistema e na escolha adequada de um método de aproximação. Discute-se ainda a aplicabilidade destes algoritmos na aprendizagem de tarefas de navegação com sensores imperfeitos. A primeira parte da tese termina com os resultados de simulação obtidos ao aplicar os dois métodos atrás referidos a diversos problemas de navegação robótica com observabilidade parcial.

Na segunda parte da tese, adopta-se uma abordagem semelhante, mas considerando problemas de navegação com múltiplos robôs. Em particular, analisam-se tarefas de navegação nas quais uma equipa de robôs se deve deslocar de uma configuração inicial para uma configuração final, num ambiente também descrito por um mapa topológico finito. Neste trajecto, os robôs devem evitar acidentes e outras situações indesejáveis, resultantes de má coordenação.

A tese faz uso da teoria dos jogos (em particular de jogos de Markov ou estocásticos) na modelação deste tipo de problemas de navegação. Neste contexto, discute-se o problema da selecção de pontos de equilíbrio/coordenação e propõe-se a utilização do método de *Biased Adaptive Play* (BAP). Descreve-se o algoritmo OAL (*Optimal Adaptive Learning*) e estudam-se as suas propriedades mais importantes. De seguida, a tese propõe um novo algoritmo, que designamos por CQL (*Coordinated Q-Learning*). Estes algoritmos (OAL e CQL) são aplicáveis a jogos estocásticos nos quais os jogadores têm interesses comuns, o jogo tem um conjunto finito de estados possíveis e são pois adequados para tratar problemas de navegação cooperativa quando os robôs têm percepção perfeita do estado.

Tal como na primeira parte da tese, e para abordar problemas mais realistas em que os sensores não são perfeitos, a tese propõe a utilização do algoritmo de *Approximate Q-Learning*, desenvolvido na primeira parte, a jogos estocásticos com espaços de estados infinitos. Nesta parte desenvolve-se também uma versão mais geral do método de BAP para jogos estocásticos com um espaço de estados infinito e estabelece-se a sua convergência. O algoritmo obtido da conjugação destes dois métodos, denominado por CAQL, é a contribuição final da tese, que termina apresentando os resultados deste método em diversas tarefas de navegação com múltiplos robôs, sensores centralizados e observabilidade parcial.

Resumindo, a tese propõe vários novos algoritmos (nomeadamente, *Approximate Q-learning*, *Approximate* SARSA, *Coordinated Q-learning*, *Approximate Biased Adaptive Play* e *Coordinated Approximate Q-Learning*) e desenvolve um estudo das principais propriedades de cada um deles, nomeadamente em termos de convergência e aplicabilidade a problemas de navegação com observabilidade parcial.

PALAVRAS-CHAVE: Navegação topológica, processos de Markov, jogos estocásticos, observabilidade parcial, aprendizagem por reforço, coordenação.

Abstract

In this thesis we address mobile robot navigation from a learning perspective. We address robotic navigation tasks in which a robot or group of robots must navigate from some initial location/configuration to some final location/configuration. The main contribution of the thesis is the proposal and study of several new reinforcement learning methods and their applicability to robotic navigation tasks. In particular, we propose and analyze methods that generalize those existing in the literature to address learning and coordination in problems with infinite state-spaces or with partial observability. We emphasize the fact that partially observable problems describe the typical scenario considered in mobile robot applications. We also describe the results obtained with the methods proposed in the thesis in several navigation scenarios, featuring both single-robot and multi-robot situations. We conclude by discussing the advantages and limitations of the approach in the thesis and pointing out directions for future work.

In the past decades, autonomous robot navigation has become a central topic of robotic research. In this thesis we add a contribution to this wide area of research. The work described herein is two-folded. We start by considering single-robot navigation tasks, where a single robot must navigate from an initial position to a final position; we then move to the analysis of multi-robot navigation tasks, where multiple robots must navigate in a common environment. We approach such navigation tasks from a learning perspective and adopt a reinforcement learning formalism to address the tasks at hand.

In single-robot scenarios, we address navigation tasks in which a robot must move from an initial to a final position in a finite topological map. A topological map is a graph describing the environment as a set of states (nodes in the graph) and corresponding connectivity information (the edges in the graph). The robot should be able to successfully explore its environment and use *evaluative feedback* collected during this exploration phase to learn the optimal way of completing its assigned task. We cast the navigation task as a *sequential decision problem* and describe a set of classical reinforcement learning algorithms (such as Q-learning and

SARSA) designed to handle simple tasks, where the robot moves in the environment with *perfect perception* of the state.

However, in most practical situations, the assumption of perfect perception of the state does not hold. Any localization method relying on imperfect sensor readings will exhibit some degree of uncertainty that should be taken into account in the decision process. Such *partially observable* decision problems can be re-cast as equivalent, fully-observable problems, where the state-space is now *infinite*. Therefore, we propose a set of new learning algorithms (namely, approximate Q-learning and approximate SARSA) that address tasks in *infinite* spaces. We identify a set of conditions that guarantee the convergence of such methods that essentially rely on the ergodicity properties of the process and on an adequate choice of an approximation methodology. We conclude the first part of the thesis by reporting the simulation results obtained with both mentioned algorithms in several topological navigation tasks where the assumption of perfect state perception is no longer considered.

When considering multi-robot navigation tasks, we follow a similar course of action. We consider tasks in which a team of robots must move from an initial configuration to a final configuration in a topological map, while avoiding "accidents" and other undesirable situations arising from mis-coordination. We propose the framework of Markov games as a model of interaction for the team. We discuss important issues such as equilibrium selection and propose the use of *biased adaptive play* (BAP) as a coordination mechanism. We describe the OAL algorithm (optimal adaptive learning) and contribute the new CQL algorithm (coordinated Q-learning); both methods are designed to address simple games where all players have a *common goal* and the game has a *finite set of possible states*. In other words, these algorithms are suited to address simple cooperative navigation tasks in which the robots have, once again, perfect perception of the state.

To alleviate the latter assumption, we propose the application of approximate Q-learning to games with an *infinite set of states*. We contribute an extension of BAP to this class of infinite games and establish its convergence. The method obtained by combining approximate Q-learning and approximate BAP (named *coordinated approximate Q-learning*, or CAQL) is the final contribution of the thesis. The second part concludes with the results obtained with CAQL in several multi-robot topological navigation tasks.

Summarizing, we contribute several new algorithms (*approximate Q-learning*, *approximate SARSA*, *coordinated Q-learning*, *approximate biased adaptive play* and *coordinated approximate Q-learning*) and study their main properties, such as con-

vergence and applicability to cooperative navigation problems with partial observability.

KEYWORDS: Topological navigation, Markov decision process, Markov games, partial observability, reinforcement learning, coordination.

Acknowledgements

I believe now that writing a PhD dissertation must be in many aspects similar to building a house: the project is there from the start, but only when the actual building takes place do we realize how hard it really is to fit the bricks into the idealized project.

In this building process, I am indebted to numerous people without whom I would not have been able to take this project to a satisfying conclusion.

I must start my list of acknowledgements with my PhD advisor, Professor Isabel Ribeiro. The will to go on with my studies and pursue a PhD degree was first born during her classes on system theory. More recently, throughout the last PhD years, she always encouraged me and supported me in all ups and downs of my scientific (and sometimes personal) life. Our numerous discussions were invaluable in nurturing all the ideas in this thesis. I am also grateful for all the conditions and opportunities that I was given during these years.

I must also acknowledge Professor Pedro Lima for many valuable discussions and for introducing me to discrete event systems; Professor Fernando Lobo Pereira, from Universidade do Porto, who provided many useful comments and suggestions on how to improve the work in general, maintaining it anchored to the robotic problem it is focused on.

During my brief stay at Carnegie Mellon University, I had the opportunity to interact and work with the CORAL group, lead by Professor Manuela Veloso. I can not thank Professor Manuela Veloso enough for being an extraordinary host, far surpassing any possible expectations that I could have had. I am deeply indebted for all the support, friendship, enlightening discussions, sharp and decisive suggestions and encouraging remarks that I benefitted during my stay at CMU and, at a latter stage, during the preparation of this thesis. During the three months I spent at CMU, I was able to present my work to several researchers who contributed useful suggestions. To all of them I leave my appreciation. Also, I would like to refer all the help I got from Francisco Pereira when moving in and out of Pittsburgh, and thank him for his endless patience, availability and good-will in every step of the "finding a home" process.

At ISR, I am especially grateful to Prof. João Xavier, Matthijs Spaan, Manuel Lopes and Luis Montesano for many interesting and valuable discussions that definitely contributed in one way or the other to the work presented here. I also benefitted from the helpful discussions with Prof. Carlos Rocha, Alberto Vale, Gonçalo Neto, Nelson Gonçalves and Duarte Antunes. More generally, I must gratefully acknowledge all the support from the Institute for Systems and Robotics, where I was able to learn, work and develop my research, in a stimulating environment and with

all required conditions. I must also acknowledge the Fundação para a Ciência e a Tecnologia, in their Programa Operacional Sociedade do Conhecimento (POS_C). They provided my PhD grant SFRH/BD /3074/2000 and all the necessary funding for me to proceed with my PhD studies.

Outside the scientific world, I am grateful beyond words for the extraordinary support and friendship of Bruno Nobre, Américo Barreira and Sónia, Ricardo Caxias Ferreira, Bruno and Ana Santos, Luís and Aude Pacheco, Ricardo Martins and his family. In particular, I owe Ricardo for the illustration in the cover. I would also like to thank all the support from Catarina, Sara and all my friends from Agrupamento 541 Pio XII and CVX group.

This endeavor would not be possible without the unconditional support from my family, namely my grand-mother Antónia, my parents António and Odete, my brother Nuno, my sister Teresa and her husband Miguel, my god-parents Alcindo and Alice, Lula, my uncle Armando and my aunt Belita. I have also received a great deal of support from my wife's family, José Manuel, Rosarinho, Ruy, Ana and grand-mother Maria de Lourdes.

Finally, I want to thank my wonderful wife Isabel for all the patience, love and outstanding support that nurtured me during all this time. During these last years, she was able to stand my numerous monologues about MDPs, POMDPs, norms, projections, games and equilibria without running away. She endured my absence, stress, bad mood and lack of availability during all this time. She also lent me her valuable help in writing down the .CON files describing the environments used in the simulations. This thesis and all the work in it would never have been possible without her and her support. I dedicate this work to her and to our son, Vasco.

Lisbon, PORTUGAL — Francisco Melo
November, 2007

Contents

List of Figures

List of Tables

Ad majorem Dei gloriam

Chapter 1

Introduction

In this introductory chapter we present a general motivation for the work developed in the thesis. We describe the class of robotic navigation problems to tackle and briefly present the main ideas on the approach pursued in the thesis. We describe the structure of the thesis and emphasize its main contributions.

1.1 The world of robotics and intelligent machines

In their work, researchers from all times and areas have found a way to gain knowledge about themselves and their world. The strongly debated separation between faith and science becomes shallow as the scientists are the first to recognize in their work a path to spirituality. In the words of Carl Sagan, "*science is not only compatible with spirituality; it is a profound source of spirituality.*"

The research in robotics provides a rather obvious connection between Man and his quest for knowledge, since the most obvious source of inspiration when developing autonomous/intelligent robots is still Man itself. And the last decades have witnessed the continuous birth of new and bolder fields of research, not only in robotics but actually in many fields of related technological and scientific research. Soccer-playing robots, intelligent houses, automated lawn-cutters and vacuum cleaners—where are we heading for?

* * *

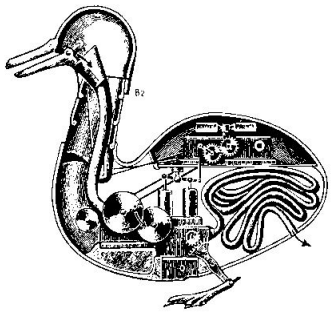


Figure 1.1: Vaucanson's duck (photo from Wikipedia Commons).

a) **b)** **c)**

Figure 1.2: The three Jaquet-Droz automata (photos from Wikipedia Commons): **a)** The Writer; **b)** The Musician; **c)** The Draughtsman.

The early works on robotics date back to the ancient Greece.[1] In 350 b.C. the Greek mathematician Archytas of Tarentum built a wooden bird (the Pigeon), whose movement was controlled by steam. Another of the first "roboticists" was Ctesibius of Alexandria, a Greek physicist and inventor who lived around the year 200 b.C. and was the creator of water clocks that included movable figures.

Later in the 15$^{\text{th}}$ century (1495), Leonardo da Vinci designed and built a mechanical device that looked like an armored knight. The device was built so that the knight would move as if there was a real person inside. This interest in automated machines steadily increased, and a great number of bright people were attracted into developing complex mechanical systems, called automata, which would mimic different living objects. This fascination reached its peak around the 18$^{\text{th}}$ century and led to the construction of several astonishing machines, presenting us with some of the finest examples of mechanical problem solving ever created.

Jacques de Vaucanson, born in France in 1709, built among other automata, a mechanical duck that quacked, ate, and defecated (Figure 1.1). Each of the moving wings of Vaucanson's Duck contained over four hundred moving parts and even today it remains something of a mystery. The original Duck disappeared.

Pierre Jaquet-Droz was a Swiss-born clock maker, who built three detailed dolls, the Writer, the Musician and the Draughtsman (Figure 1.2).These figures, presented

[1] Historical references from http://robotics.megagiant.com/history.html.

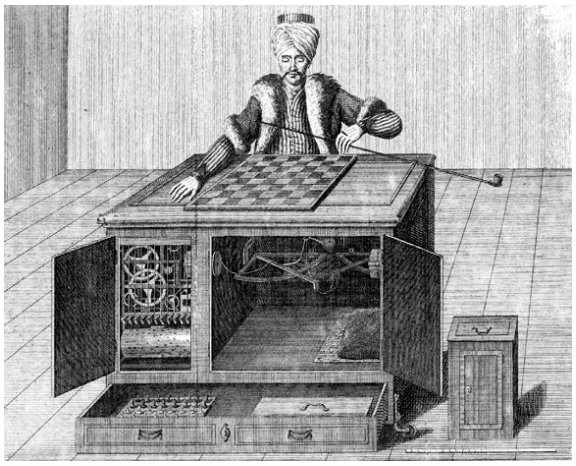


Figure 1.3: The Great Chess Automaton (photo from Wikipedia Commons).

to several kings and emperors of Europe, China, India and Japan, astonished them with their detailed craftsmanship and automation abilities. For example, the Writer was build with an input device, allowing the "user" to program several different movements.

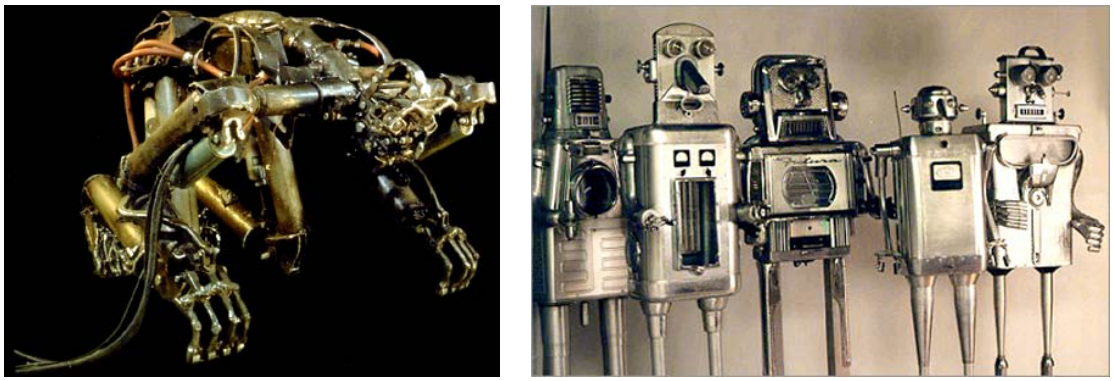
However, one can say that the most famous of all automata ever build was the Great Chess Automaton, created by the Hungarian Wolfgang von Kempelen (1734Ű1804).The Great Chess Automaton, or Turk, was built as a human-looking wooden figure dressed in Turkish clothes, whose trunk emerged out of a large wooden box filled with gears and wires (Figure 1.3). Von Kempelen claimed the Turk to be the first "thinking machine" and would challenge any volunteer from the audience to play chess against the Turk. The machine would move the pieces on its own and actually used sophisticated chess strategies, as it won most of its matches, even against experienced players. Von Kempelen traveled all around Europe, exhibiting his Turk in many cities and before many distinct audiences. It was not until 1836 that the secret of the Turk was revealed.[2]

* * *

Robotics and artificial intelligence have run a long way since the days of Von Kempelen's Turk. Nowadays, robotics encompasses countless fields, ranging from robotic manipulation to swarm robotics and it is impossible for a single researcher to grasp all the distinct fields to which research extends. However, the interest in robotics has reached out to far more than the scientific world and contributions to the field of robotics now arise from the entertainment industry as well as the artistic community. Figures 1.4 and 1.5 illustrate some examples of robotic applications in art and entertainment.

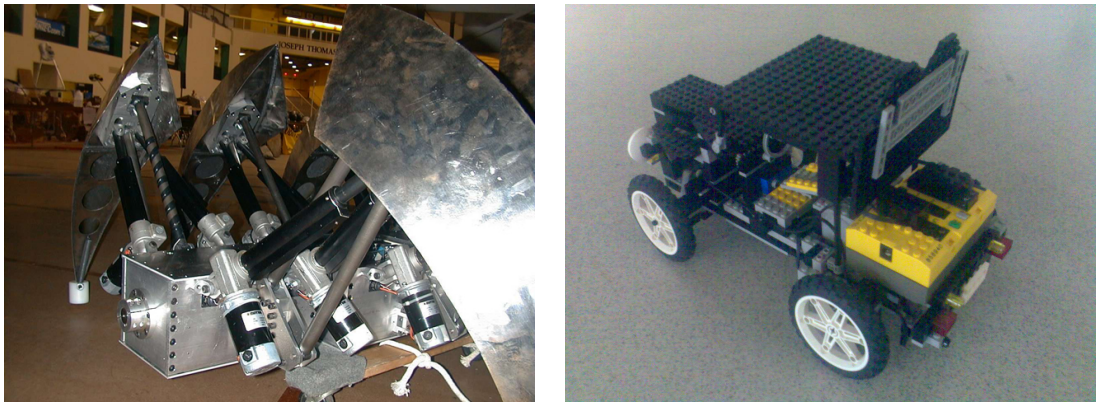
In this robotic frenzy, each individual contribution adds up to an increasingly large construction leading to the ultimate creation of an artificial intelligent creature: a humanoid.

[2]For details on the Turk, see [95] or visit the site http://www.museumofhoaxes.com/chess_auto.html.



a) **b)**

Figure 1.4: Robotics in art: **a)** Chico MacMurtie's Amorphic Robot (photo: courtesy of Amorphic Robot Works); **b)** "Robot Sculpture" by Clayton Bailey (photo: courtesy of www.ClaytonBailey.com.)



a) **b)**

Figure 1.5: Robotic applications in entertainment: **a)** BattleBot Mechadon, by Mark Setrakian, Team Sinister of Los Angeles; **b)** LEGO Mindstorm.

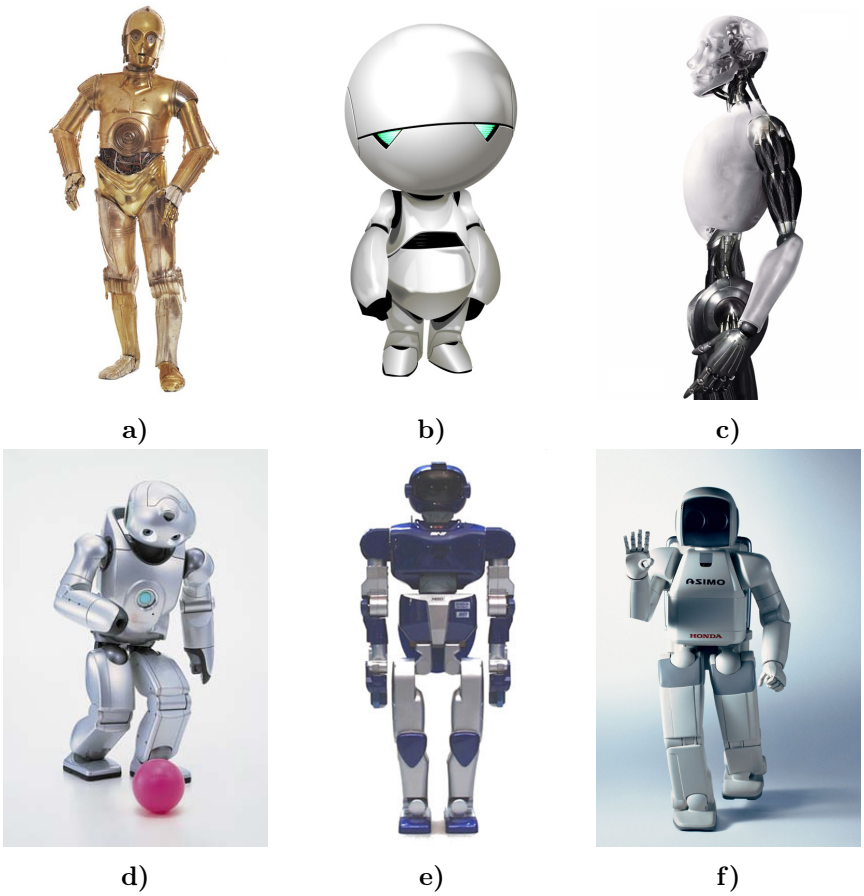
If such a justification for the interest that robotic research has raised may seem a little lyric or even simplistic, the fact is that robotic research is surely intrinsically embedded with humankind's desire to artificially "produce" one of its own. And, if some decades ago such desire lied in the realm of scientific fiction, the last decades have witnessed an astonishing development of technologies that leave room for no doubt about the near-future possibilities of robotic and artificial intelligence research.

Artificial intelligence and robotic research have provided us with tangible specimens seemingly arising from science fiction movies; Deep Thought, the gigantic intelligent computer from Douglas Adam's "The Hitchhiker's Guide to the Galaxy" series has found its counterpart in IBM's Deep Blue (Figure 1.6); and in robotics, the examples are even easier to find (Figure 1.7).

As the scenarios under which robots may operate broaden, and the possible applications of robotic technologies reach into new and demanding fields, a robot's ability to *autonomously navigate* its environment, to *explore* it and *learn* from its

a) b)

Figure 1.6: Science fiction vs. reality: **a)** Deep Thought, from "The Hitchhiker's Guide to the Galaxy" (Touchstone, 2005); **b)** IBM's Deep Blue.



a) b) c)

d) e) f)

Figure 1.7: Science fiction vs. reality: **a)** 3CPO, from the movies "Star Wars" (20th Century Fox, 1977); **b)** Marvin, the depressed robot, from the movie "The Hitchhiker's Guide to the Galaxy" (Touchstone, 2005); **c)** Sonny, from the movie "I, Robot", (20th Century Fox, 2004); **d)** SONY's QRIO; **e)** HRP-2P, by Kawada Industries; **f)** Honda's ASIMO.

experience, and to successfully *interact/cooperate* with humans and other robots become matters of central importance. In fact, these are the skills that make a

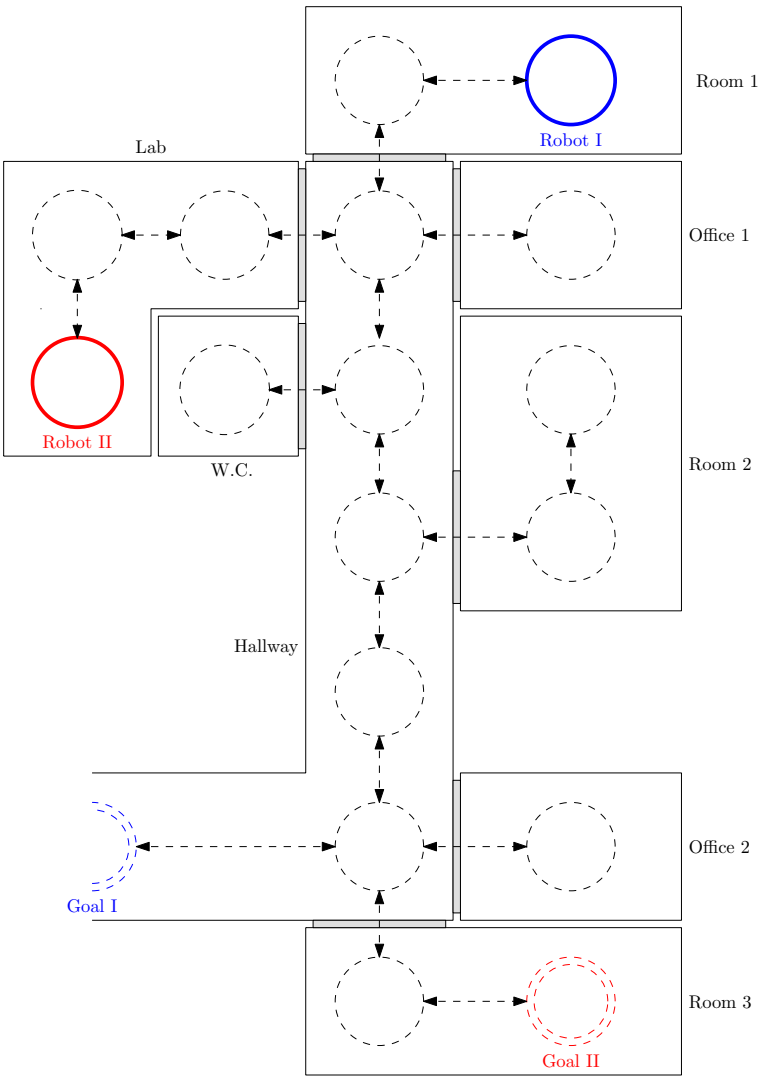


Figure 1.8: Illustration of a possible situation to address.

robot into more than a simple working tool, such as a sledge-hammer. These skills provide robots with the ability to effectively replace human operators in many tasks that jeopardize the physical integrity of the latter. In this thesis, we focus on the problems of *navigation*, *learning* and *cooperation*.

1.2 Problem statement

To describe the class of problems addressed in the thesis, we introduce a rather general example that suitably illustrates the main aspects of the navigation problems considered.

Example 1.1. Consider the scenario depicted in Figure 1.8. Two robots, I and II, are intended to complete some mission, in which they must move from

an initial configuration (marked with the bold circles) to a goal configuration (marked with the double line circles). Since the process by which the two robots navigate from the initial to the goal configuration may be too complex to program by hand, the robots instead receive *evaluative feedback* from an exterior user (*e.g.*, a human), that assigns the team with a reward/punishment for their every movement.[3] Each individual robot must learn how to coordinate with the other, avoiding situations that may lead to punishments (*e.g.*, crashing into each other) and so as to maximize the received reward, thus accomplishing the mission in the "best" possible fashion.

The two robots work as a *team* in the sense that the mission is completed only when both robots reach their goal locations. The robots have no initial knowledge on their mission and must learn how to perform it by experiencing and interacting with each other. In this process they must not only learn how to complete the specified mission, but also how to avoid undesirable situations (such as crashes) arising from mis-coordination.

We admit the robots to have a topological representation of the environment (the dashed diagram) that they use to navigate. To control their movement, each robot has available a set of high-level primitives (actions) that govern its movement. The outcome of these primitives has an associated uncertainty, due to noise in the controllers, wheel slippage, etc. The robots should take into account this uncertainty when deciding on their trajectory.

The team also has access to the security video-feed from the building, and uses this as sensory information. This sensory information provides each robot with an estimate of the configuration of the team in the environment that each robot can use to individually plan its movement.

In all of this, the robots do not know *beforehand* what the other robot will do (there is no *explicit* communication for the purpose of coordination). However, to simplify the problem, we do admit that the robots do know *a posteriori* the action chosen by the other robot (for example, they estimate it from the observed movement). ◇

This example illustrates several important aspects of the class of problems considered in the thesis. In particular,

- We consider situations where a mobile robot or group of robots must *learn* how to complete some navigation task in an environment described *topologically*;
- The robots receive *evaluative feedback* that indicates how well the team is performing. Therefore, this evaluative feedback implicitly "encodes" the specific task to be accomplished. A typical task is, for example, reaching some target location/configuration;[4]
- The robots control their movements by choosing among a finite set of possible *action primitives* with uncertain outcome;

[3] This evaluative feedback, generally provided by a human operator, is intended to "teach" the robot/robots how to perform a task without showing how to do it. This prevents the human user to have to explicitly "program" the robot/robots to perform such task.

[4] Other tasks are also possible, such as following some specified trajectory or avoiding some particular state. All these specificities are "encoded" in the rewards/punishments provided to the robots.

- The robots have access to *noisy measurements* from which they must infer their current location/configuration;

- The robots do not know beforehand the actions taken by the other robots, *i.e.*, they do not *explicitly communicate*. Coordination must *emerge* as a consequence of the interaction between the robots.

We can summarize all this by stating that the thesis addresses situations where a team of robots must *learn* how to perform a given *topological navigation* task in a *coordinated* fashion. As soon as the learning is complete, the robots will be able to perform this task optimally with no interference from a human operator.

On the other hand, it is important to refer that there are several important topics of research closely related to the ones described but which we do not address. In particular,

- We assume the topological representation of the environment is known *a priori* and do not address the problem of topological mapping. In Chapter 2 we provide several references on the topic of mapping;

- We model the navigation task as a *sequential decision process*. From this perspective, robots are seen as high-level decision-makers and we do not address the implementation of the interface between the high-level decision-maker and the low-level sensors and actuators. Once again, several references on this problem can be found in Chapter 2.

The evaluative feedback described above naturally suggests a *reinforcement learning approach* and that is the approach followed in the thesis. Our approach is, however, a "constructive" one. The thesis is organized so as to gradually build from the simpler problem with a single robot and perfect sensors to the more complex multi-robot problem with imperfect sensors. A more detailed description of the structure of the thesis is provided in the next section.

1.3 Structure of the thesis

We divided the thesis in two main parts. Part I addresses single-robot navigation tasks and Part II presents the multi-robot cooperative navigation scenario.

The structure of both parts is kept similar to facilitate the reading. In each of the two parts, an initial chapter introduces the basic concepts and notation. The subsequent chapters elaborate on the ideas presented in this introductory chapter. Simple examples are produced along the text to illustrate the application of the different concepts and methods.

Each chapter includes a small table of contents and a brief summary. Also, each chapter includes a concluding section where a more detailed summary of the chapter is presented together with some discussion on the topics addressed in it.

Finally, all symbols and acronyms used throughout the text are gathered together in one appendix, to be found in page 365.

Part I: Single-robot navigation and learning

We start in **Chapter 2** by describing several possible approaches to mobile robot navigation, while introducing the framework of (finite) Markov decision processes (MDPs). We present some simple methods to find the optimal behavior-rule (or *policy*) when all MDP parameters are known and the robot is able to unambiguously recognize its state in the environment (*i.e.*, it has full state observability). We illustrate the application of these methods in a simple example.

In **Chapter 3** we address the problem of *learning* from experience. After describing in Chapter 2 several methods that determine the optimal policy when the MDP parameters are known, we now describe several methods that learn this optimal policy from interaction with the environment, while still considering the simplifying assumption of full observability. Once again, we apply these methods to a simple example.

In **Chapter 4** we finally address the problem of learning with imperfect sensors, *i.e.*, learning in the presence of *partial state observability*. We provide new results that identify the conditions under which partial observability in MDPs can be tackled using an equivalent infinite MDP. We thus propose two new learning algorithms, (*approximate Q-learning* and *approximate* SARSA), to approximate the optimal policy in infinite MDPs. We identify the conditions under which these new algorithms converge and apply them to a simple illustrative example.

Finally, we conclude in **Chapter 5** by applying the methods developed in this first part to several benchmark navigation problems from the literature. The experimental results presented in this chapter assess the applicability of the methods in more realistic problems.

Part II: Multi-robot navigation and learning

Part II follows a similar structure to that of Part I. In **Chapter 6** we discuss several important aspects of multi-agent systems and introduce (finite) Markov games (MGs) as a suitable framework to address this class of systems. We establish a close relation between the multi-agent framework of MGs and the single-agent framework of MDPs, while emphasizing a new problem arising in the multi-agent setting: coordination.

We proceed in **Chapter 7** by addressing the problem of *learning* in multi-agent systems. We once again consider the simplifying assumption of perfect sensors and describe several methods to determine the optimal joint behavior-rule (or *strategy*). We then propose a new algorithm (*coordinated Q-learning*), that combines a well-known reinforcement learning method (*Q-learning*) with a sound coordination mechanism (*biased adaptive play*). We analyze the properties of this new method and illustrate its application in a simple example.

In **Chapter 8** we extend the approximate Q-learning algorithm from Chapter 4 to multi-agent settings. We also propose a powerful generalization of biased adaptive play to infinite MGs. We assess its convergence and combine it with approximate Q-learning to yield a new, convergent algorithm (*coordinated approximate Q-learning*). With this algorithm in hand, we address the problem of *partial observability* in multi-agent scenarios. Under the simplifying assumption of common observations, we show

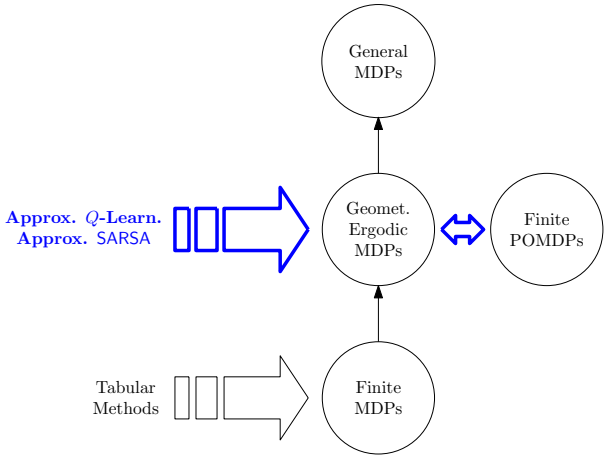


Figure 1.9: Contributions in single-agent systems (in blue). To address geometrically ergodic MDPs, we introduce approximate Q-learning and approximate SARSA. We also established a correspondence between finite POMDPs and geometrically ergodic MDPs. The vertical arrows represent inclusion relations.

that partial observability can once again be addressed by considering an equivalent infinite problem. We then apply the coordinated approximate Q-learning algorithm to this equivalent problem. We conclude the chapter with a discussion on several approaches to more general problems, where common observations are not assumed.

In **Chapter 9** we apply the methods developed in Chapter 8 to sizeable problems. The experimental results presented in this chapter assess the applicability of the methods in more realistic problems.

Finally, we conclude the thesis in **Chapter 10**. We summarize the main contributions of the thesis, discuss the applicability of the introduced methods and present some open issues that can be addressed in future research.

1.4 Contributions

As stated in Section 1.2, this thesis tackles mobile robot navigation using a reinforcement learning approach. When considering the simpler case of a single robot and perfect state perception, the topological navigation problem (*i.e.*, navigation in a topological map) can easily be formulated as a *finite* Markov decision process (MDP). We can find abundant methods in the reinforcement learning literature that adequately solve this class of decision problems. These methods are referred in Figure 1.9 as *tabular methods*.

The consideration of imperfect sensors (partial observability) poses significant difficulties even if a single robot is considered. To tackle this problem, we proposed in Chapter 4 two new algorithms, *approximate Q-learning* and *approximate SARSA*. In Theorems 4.5.2 and 4.5.3 we identified the conditions that guarantee *convergence with probability one* of both methods and provided *an interpretation for the obtained approximation*. One essential condition is the *geometric ergodicity* of the MDP under consideration.[5] Therefore, our convergence theorems extend the use of Q-learning

[5]Geometrically ergodic MDPs can be seen as "almost stationary" under adequate policies.

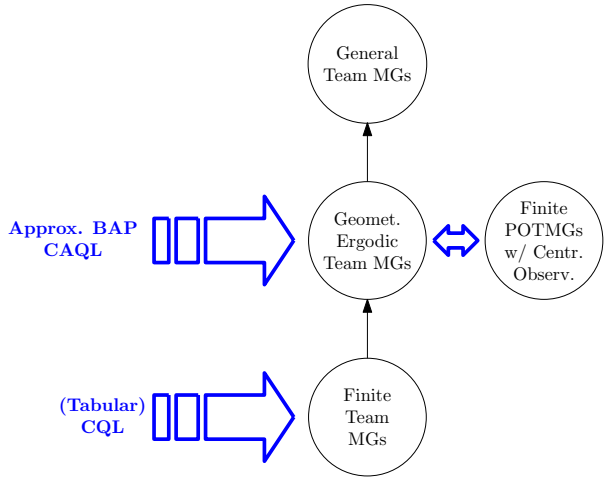


Figure 1.10: Contributions in multi-agent systems (in blue). We proposed CQL to address finite team MGs and CAQL to address geometrically ergodic team MGs. We developed approximate BAP as a coordination mechanism for geometrically ergodic team MGs and established a correspondence between finite partially observable team MGs with centralized observations and geometrically ergodic team MGs. The vertical arrows represent inclusion relations.

and SARSA to geometrically ergodic MDPs, as indicated in Figure 1.9, in blue.

To apply these algorithms to partially observable scenarios, we identified in Theorem 4.7.4 the conditions that allow us to *transform a finite, partially observable MDP (POMDP) in an equivalent fully observable, geometrically ergodic MDP*. With this new result, we are in position to apply approximate Q-learning and approximate SARSA to POMDPs. The identification of finite POMDPs with geometrically ergodic MDPs is therefore an important contribution and is also emphasized in blue in Figure 1.9.

When moving to multi-robot scenarios, cooperative navigation with perfect state perception can be formulated using *finite* team Markov games (MG). As in the single-agent case, there is a multitude of methods in the literature that adequately solve this class of decision problems. However, if many such methods are able to determine the optimal decision rule (or strategy) for this class of problems, few are able to formally guarantee that the multiple decision-makers (robots) actually coordinate in any such rule. In Chapter 7, we propose a new algorithm, dubbed as *coordinated Q-learning* (CQL). In Theorem 7.3.3 we assess that CQL not only *learns the optimal decision rule* but also that *all decision-makers coordinate in an optimal Nash equilibrium* with probability one. This is indicated in Figure 1.10 by the blue arrow between CQL and finite team MGs.

We then propose *coordinated approximate Q-learning* (CAQL), a novel algorithm that combines approximate Q-learning with *approximate biased adaptive play* (ABAP). This algorithm is suited for geometrically ergodic team Markov games, as indicated in Figure 1.10. In the process of introducing CAQL, we also develop ABAP, a novel and general coordination mechanism that can be employed in *infinite team Markov games* and *ensures coordination with probability one*. By combining this result with Theorem 4.5.2, we show that CAQL *converges with probability one*

while ensuring coordination in an optimal Nash equilibrium in an approximate game.

Finally, aiming at applying CAQL to partially observable scenarios, we introduce the concept of *centralized observations*. Under this simplified setting, we gather all the results in the thesis in Theorem 8.6.1 and identify the conditions that guarantee a *partially observable team MG with centralized observations to be equivalent to a fully observable, geometrically ergodic team MG*, as indicated by the small blue arrow in Figure 1.10. With this equivalence, we have all the requirements to apply CAQL to this class of partially observable team MGs.

To conclude, the result in Theorem 8.6.1 can be seen as a general convergence theorem. By considering restricted classes of MGs (*e.g.*, with a single player or perfect sensors), we obtain the convergence results for each of the classes considered along the thesis (finite MDPs, finite team MGs, finite POMDPs, etc.).

Therefore, we can summarize the main contribution of the thesis as being the unification provided by such convergence theorems that identify the conditions under which such distinct classes of problems as finite MDPs or partially observable team MGs can be addressed using a common methodology.

We conclude this introductory chapter by introducing some fundamental notation used throughout the thesis

1.5 Basic nomenclature

Throughout the thesis, we take all random variables to be defined in a general probability space $(\Omega, \mathcal{F}, \mathbb{P}[\cdot])$ where $\mathcal{F}$ is a countably generated σ-field on Ω. For any non-empty set $U \subset \Omega$, we denote by $\mathcal{F}(U)$ the σ-field generated by U, *i.e.*, the minimum σ-field on Ω such that $U \in \mathcal{F}(U)$. We denote by $\mathbb{E}[X]$ the expectation of the random variable X.

In general, we denote random variables using uppercase letters such as A, B or X. The corresponding sample values are denoted using lower case letters, such as a, b or x. Similarly, we denote a stochastic process by a sequence $\{X_t\}$ and the corresponding sample trajectories by $\{x_t\}$. The parameter t is assumed to belong to some (countable) index set $\mathcal{T}$. If X is a n-dimensional random vector, we may refer to its i^{th} component either as $X(i)$ or as X_i. If x is a realization of the random vector X, we also denote by $x(i)$ or x_i the corresponding i^{th} component. Vectors are taken as $n \times 1$-matrices, *i.e.*, as columns.

We write "r.v.", "r.v.s", "w.r.t." and "w.p.1" for "random variable", "random variables", "with respect to" and "with probability one". Given a r.v. X taking values in some space $\mathcal{X}$, if μ is a measure on $\mathcal{X}$ we say that a predicate $P(x)$ holds μ-almost everywhere or for μ-almost every x if $P(x)$ holds for all $x \in \mathcal{X} - \mathcal{N}$, where $\mathcal{N}$ is some μ-null set, *i.e.*, if $\mu(\mathcal{N}) = 0$.

Part I

Single Robot Navigation and Learning

Chapter 2

Topological Navigation and Markov Processes

This is an introductory chapter that provides the bridge between (single) robot navigation and reinforcement learning. In it, we introduce the fundamental concepts used throughout the thesis, setting up the required formalism to proceed with the developments in the following chapters.

We start by describing three fundamental issues arising in any navigation problem: environment representation, localization and navigation. We address each of these topics from a discrete-event perspective, finally building up to Markov decision processes and some simple solution methods. We conclude the chapter by illustrating the use of this framework in mobile robot navigation tasks, where the environment is suitably represented as a topological map.

2.1 Mobile robot navigation

The reference to "autonomous systems" may be found in a multitude of contexts, whenever a system is able to perform some task with minimum interference from a human operator. Not seldom the reference to autonomous systems appears in the context of robotic navigation, where an *autonomous robot* is able to *navigate* in some environment. Mobile robot navigation has been one of the most central topics of research from the early days of robotics. In fact, the usefulness of a mobile robot in most tasks greatly depends on its ability to move autonomously in its environment.

There is no universal definition of navigation. Under the label of navigation it is possible to find works on localization, path-planning, trajectory tracking, target interception among others. We adopt the more intuitive definition of navigation, according to which "*navigation is the process of determining and maintaining a course or trajectory to a goal location*" [90].

The classical approach to navigation models a mobile robot as an *autonomous vehicle*, irrespectively of any cognitive abilities of the robot. The movement of the robot is described using a set of differential equations arising from its kinematics and dynamics. Classical techniques such as LQG/LTR synthesis [7] or H_∞ design [233] can then be applied to synthesize a controller for the robot. In many cases, this controller can also be described by a set of differential equations, and provides a signal for the actuators of the robot, driving it to its objective. Examples of classical control applications are abundant in the literature [183, 213, 255] and illustrate the power of such methodologies.

More recently, new techniques have been proposed to address mobile robot navigation that combine the advantages of classical control methods and several ideas from computer sciences such as genetic algorithms [275] or hybrid automata [77]. These combined approaches have proven to be especially useful in systems subject to constraints of different kinds (such as non-holonomic). These methods allow, under certain conditions, a straightforward analysis of stability and robustness and have been applied to different types of mobile robots, such as omni-directional robots [331], non-holonomic robots [230] or underwater autonomous vehicles [210].

Nowadays, mobile robot navigation is addressed from widely different perspectives. These approaches range from visual-navigation strategies [184, 317, 340] to biologically inspired strategies [155, 318], genetic algorithms [115, 123] or neural networks [176].

* * *

In the definition of navigation considered above, it is implicit that the "goal location" is clearly identifiable. The most common way to do this is by "marking" it in some map. This is not such a strange procedure, since it arises from our own natural way of identifying a goal location (see Figure 2.1).

Good representations of the environment (maps) are as important as the ability to navigate in such environment and, usually, the latter greatly depends on the former.

As stated in [200, 324], maps can essentially be divided into 3 categories:

Figure 2.1: The "X" marks the spot.

- Metric maps;
- Topological maps;
- Hybrid maps.

As navigation tasks become more and more demanding, reliable maps arise as an essential element in navigation, this leading to an increasing interest in mapping algorithms. In the last decade, the SLAM problem (simultaneous localization and mapping) has raised as a field of research on its own, and it is possible to find in the literature numerous different approaches to the problem of mapping, being metric [63, 111, 218, 313, 339], topological [57, 150, 270, 325, 347] or hybrid [83, 240, 314, 315].

Nevertheless, recent years have witnessed a particular interest in the use of topological maps in navigation [217, 229, 250, 297, 324]. Topological maps "*are more suited as a qualitative world model*" [347] and allow a navigation task to be defined "*in terms that are easily related to the robot perceptual capabilities and the robot's set of possible actions*" [270]. Furthermore, "*they provide an inherently scalable localization structure which relies on robust transitions between the different regions of the map*" [9].

A topological map represents the environment as a discrete set of states (the nodes in a graph) and the transition information between such states (the edges of a graph). The definition of the states may still rely on geometric considerations but can also rely on other qualitative information or topological relations between the different states [324].

A mobile robot moving in an environment described by a topological map may be described using discrete-event systems or Markov chains. Such approach to robotic navigation comes as no novelty and there are numerous examples of application of discrete-event systems and Markov processes to mobile robot navigation [87, 116, 174, 261, 309].

In this chapter, we address topological navigation (*i.e.*, navigation relying on a topological map) of a single mobile robot. We build up from a simple discrete-event model (an untimed automaton) towards the more sophisticated Markov decision process. In this path, we address the three main questions arising in all navigation

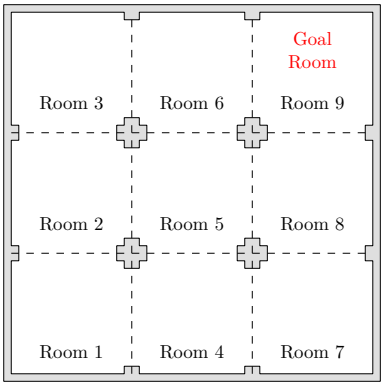


Figure 2.2: Example of an indoor environment. The dashed lines do not correspond to any physical obstacle but merely represent the separation between the rooms.

problems [267]: "where is the robot?", "where is the goal relatively to the robot?" and "how does the robot reach the goal?".

2.2 Topological maps

In this section, we describe topological maps and present a discrete-event model for the movement of a mobile robot using an untimed deterministic automaton.

2.2.1 Topological representation of the environment

For the sake of clarity, we use the following example extensively in this chapter, to help clarify the different concepts and methods introduced.

Example 2.1. Consider the indoor environment represented in Figure 2.2. A mobile robot is supposed to traverse this environment, starting in the area labeled as «Room 1» and reaching the area labeled as «Room 9». For the sake of clarity, we signaled Room 9 as the «Goal Room». Consider, for example, a situation where the environment in Figure 2.2 is a disaster scenario. The robot will provide immediate medical care to a victim in Room 9 until human help arrives.

Throughout this chapter, we gradually develop this example to illustrate the several methodologies presented. ◇

As already stated, mobile robots need suitable environmental representations to perform most tasks. It is this environmental representation together with the robot's sensory capabilities that allow a mobile robot to determine its own position as well as that of its goal.

In finding suitable environmental representation, one first solution is to attach a metric referential $\{W\}$ to the world and a second metric referential $\{R\}$ to the robot (see Figure 2.3). The coordinates of the goal position (x_{P_G}, y_{P_G}) are then defined in the world referential, and a controller can be designed to minimize some error measure such as

$$\varepsilon = \sqrt{(x_R - x_{P_G})^2 + (y_R - y_{P_G})^2}.$$

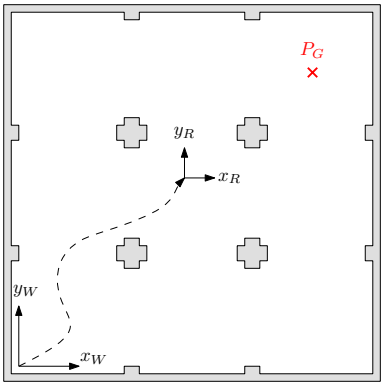
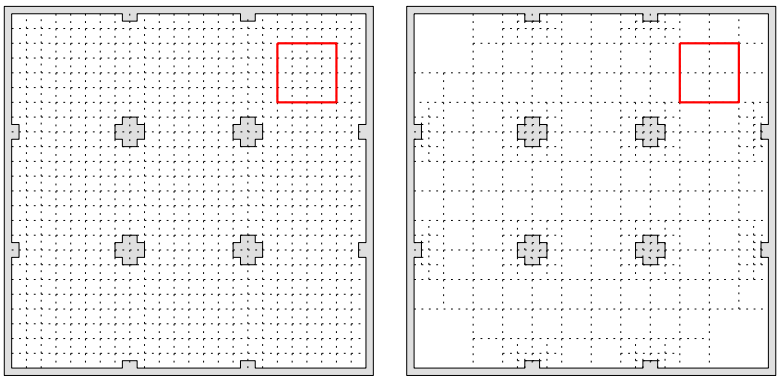


Figure 2.3: Metric referential in the environment of Figure 2.2.



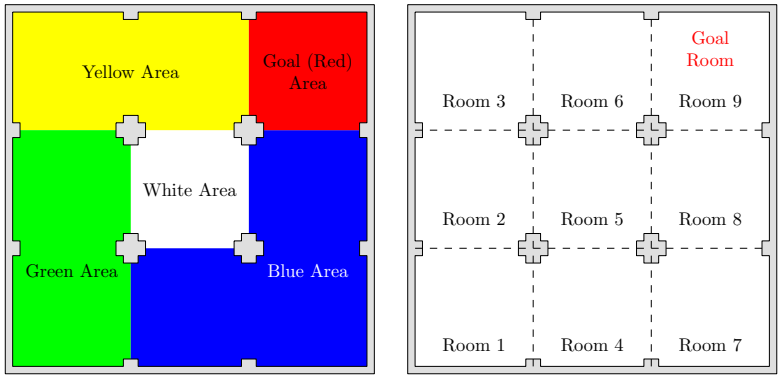
a) Regular grid **b)** Quad-tree

Figure 2.4: Two grid-based maps. In both figures, the region outlined in red represents the goal.

This method can be combined with other methods, such as potential fields, to implement obstacle avoidance [20].

However, a metric approach to navigation requires some type of representation of the obstacles/landmarks in the environment. An environment rich in obstacles/landmarks usually permits accurate localization but requires larger computational resources (both for mapping and localization) than sparse environments. A sparse environment, on the other hand, although computationally "cheap" in terms of mapping, often leads to difficult problems in localization (namely, recovering from long periods of dead-reckoning localization).

A second solution that comes to mind when deciding on an environmental representation is the use of some grid-based approach. The use of grid-based representations of the environment is not a novelty and several variants can be found in the literature [78, 297, 310, 341, 342]. In Figure 2.4, two simple grid-based maps are provided. In the map in Figure 2.4.a), a simple uniform grid divides the environment into a discrete number of regions or cells. By numbering the cells from 1 to n, the position of the robot can be referred by the number of the corresponding cell. However, this uniform grid can be computationally ineffective in environments with many small obstacles and large, free spaces. Figure 2.4.b) depicts a *quad-tree* based representation. This technique, inspired from image segmentation methods, succes-



a) Feature-based **b)** Context-based

Figure 2.5: Two possible region partitions. In the feature-based partition, *color* is used as a distinctive feature to partition the environment into five "regions". In the context-based partition, the partition relies on the ability to distinguish «Rooms», which is clearly an environment-specific ability (hence the designation "context-based").

sively divides the space into smaller regions (up to some threshold), until each region is of one of two types: free-space or obstacle. Notice that the quad-tree approach readily overcomes the ineffective space partition of regular grids, by considering a finer division in regions with small obstacles and a coarser division in large regions of free-space.

The use of grid representations for environments is one step closer to abstract environment representation. Grids provide discrete representations of the environment and are able to encode obstacles and free space. But it is with the use of topological maps that mapping reaches a new level of abstraction. As stated in [9, 112], topological maps are graph-based descriptions of the environment in which different places correspond to different states. The distinction between different places makes use of the concept of *distinctive place*. This concept plays a central role in the elaboration and use of such categorized maps, since the environment is partitioned in a discrete number of regions according to the ability of the robot to *distinguish* among the different places. This distinction may arise at different levels, either feature-based [9, 57, 325] or at a more abstract level [250, 347, 348] (see Figure 2.5).

A *topological map* can be described as a connected, directed graph $\mathcal{G} = (V, \mathcal{E})$, where V is the set of all nodes (or vertices) in the graph and $\mathcal{E}$ is a subset of $V \times V$. A pair (i, j) belongs to $\mathcal{E}$ if there is a directed edge from vertex i to vertex j in $\mathcal{G}$.

Example 2.1. (cont.) Consider once again the environment of Figure 2.2 with the partition depicted in Figure 2.5.b). The corresponding topological map is depicted in Figure 2.6. This topological map is a graph $\mathcal{G} = (V, \mathcal{E})$, where

- $V = \{1, 2, 3, \ldots, 9\}$;
- $\mathcal{E} = \{(1,2), (2,1), (1,4), (4,1), (2,3), (3,2), \ldots, (8,9), (9,8)\}$.

Each node $i \in V$ corresponds to Room i in the environment of Figure 2.5.b).

◇

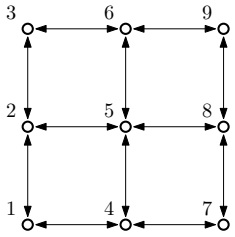


Figure 2.6: Topological map corresponding to the environment of Figure 2.5.b). We numbered the vertices in the graph according to the room numbers in Figure 2.5.b).

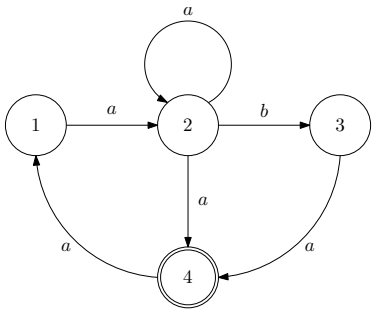


Figure 2.7: Example of a simple automaton.

Notice that a topological map is merely an environment representation and, as such, encodes no information regarding the objective of the robot. However, there is a tight relation between the movement of a mobile robot in an environment and the topological representation of that environment. That topic is further explored in the following subsection, where we introduce a discrete-event model for the movement of the robot.

2.2.2 Discrete-event models

A discrete-event system is a dynamic system with a discrete state-space in which state transitions occur at discrete instants in time. Such transitions are interpreted as *events* and the system is studied only at such event-times. In other words, in a discrete-event system, state transitions are driven by events rather then by some clock [55].

Two common models for discrete-event systems are finite-state automata and Petri nets. An automaton is basically a state-machine, or transition diagram, such as the one depicted in Figure 2.7.

We notice that a finite-state automaton such as the one in Figure 2.7 can also be represented as a directed graph. The encircled numbers represent the *states* of the automaton and the labels on the arrows represent the *events*. The state marked with a double line is referred as a *marked state* and this notation indicates that, in a sense, state 4 is a "goal state".[1] Therefore, a finite-state automaton modeling the

[1]In terms of automaton terminology, the concept of marked state gives rise to the concept of *marked language*, which is the set of all strings ending in a marked state. We can think of these marked strings as being the "desirable" strings generated by the automaton. Noticing that

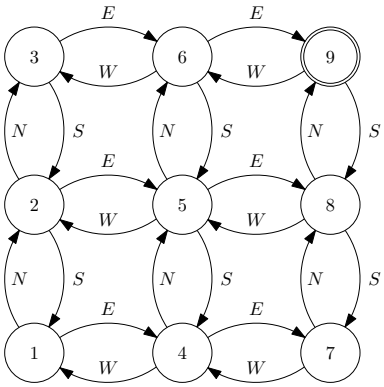


Figure 2.8: Finite-state automaton modeling the movement in the environment of Figure 2.2.

movement of a mobile robot can be obtained directly from the topological map of the environment. The automaton thus obtained is referred to as *navigation automaton* [189].

As stated in [55, 97], it is possible to derive a Markov model from a discrete-event model such as a finite-state automaton. In particular, there is an equivalence between finite-state Markov chains and a particular class of stochastic-timed automata. We briefly address that topic in the next section.

We conclude this section with an example.

Example 2.1. (cont.) Consider once again the environment of Figure 2.2 and the corresponding topological map depicted in Figure 2.6. The finite-state automaton describing the movement of the robot in the given environment is presented in Figure 2.8.

Once again, we have labeled the states from 1 to 9 as in the map of Figure 2.6. Notice, however, that we marked state 9 to emphasize that it is the goal state. On the other hand, we have labeled the transitions between states (the events) according to the direction of the corresponding movement. Therefore, we have for example labeled with N (North) the transition from state 1 to state 2 and with W (West) the transition between state 9 and state 6.

◇

2.3 Topological localization

Localization consists in answering the question "where is the robot" with respect to some representation of the environment. In the particular case of a topological map, localization consists in determining the *node* corresponding to the position of the robot in the environment.

When addressing mobile robot localization, one must necessarily consider some time reference to which localization is concerned. This is because a mobile robot

a string is nothing but a sequence of events (a trajectory), marked strings represent the desirable trajectories: those ending in a marked state.

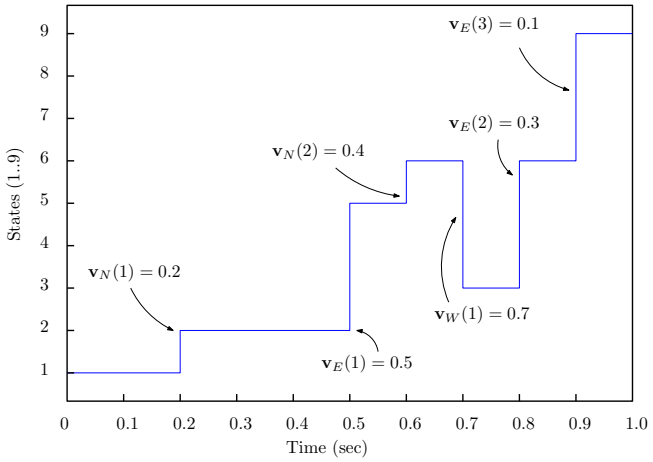


Figure 2.9: State evolution plot for the automaton of Figure 2.8 with the clock structure in (2.1).

moves and therefore its position changes with time. Therefore, untimed discrete-event models such as finite-state automata are not enough to properly address mobile robot localization.

By appending a clock structure to a discrete-event system, it is possible to analyze the temporal evolution of the state of the system. A clock structure defines the time instants in which the different events occur [55], given some reference time. In an automaton with a set of possible events $\mathcal{E} = \{e_1, \ldots, e_n\}$, a clock structure is a set of n sequences, $\mathbf{v}_{e_1}, \ldots, \mathbf{v}_{e_n}$, each corresponding to an event e_i in $\mathcal{E}$. Each sequence $\mathbf{v}_{e_i}$ determines the time intervals between two consecutive occurrences of e_i.

Example 2.1. (cont.) Consider once again the automaton in Figure 2.8. A possible clock structure is

$$\begin{aligned} \mathbf{v}_N &= \{0.2; 0.4\} \\ \mathbf{v}_S &= \emptyset \\ \mathbf{v}_E &= \{0.5; 0.3; 0.1\} \\ \mathbf{v}_W &= \{0.7\} \end{aligned} \tag{2.1}$$

where the numbers correspond to time in seconds with respect to the initial time $t = 0$. The plot of the corresponding state evolution is presented in Figure 2.9.

To better understand how the clock structure works, notice for example that $\mathbf{v}_N = \{0.2; 0.4\}$. This means that the event N occurs at time 0.2 seconds and then at time $0.2 + 0.4 = 0.6$ seconds. ◇

If the clock structure is known, it is possible to determine at each time instant, the exact state in which the robot is. A finite-state automaton as the one in Figure 2.8 endowed with a clock structure as the one in (2.1) is called a *timed automaton*.

However, in timed automata there is no uncertainty, *i.e.*, all events occur in deterministic time instants and the outcome of each event is also deterministic. In

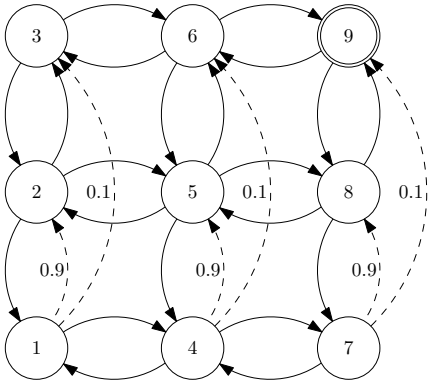


Figure 2.10: Probabilistic transition diagram obtained from the automaton in Figure 2.8 when the action N has uncertain outcome. We omitted the event labels for the sake of clarity. The dashed lines denote uncertain transitions (associated with the event N).

order to properly model the movement of a robot, uncertainty must be taken into account and a richer model is necessary.

2.3.1 Probabilistic localization

So far, we admitted that the transitions in an automaton are triggered by events belonging to some event set $\mathcal{E}$. We now further suppose that some (or all) of these events have an uncertain outcome. For example, suppose that the robot, when in state 1, moves to state 2 or 3 with probabilities 0.9 and 0.1 when event N occurs. Indicating the corresponding probability in each of the transitions of the automaton yields the transition diagram of Figure 2.10, analyzed in detail in the following example.

> **Example 2.1. (cont.)** Consider once again the automaton in Figure 2.8 and suppose that, whenever event N occurs, there is a 0.9 probability of the robot moving to the adjacent state to the north and a 0.1 probability of the robot moving *two* states to the north (see Figure 2.10). The corresponding *stochastic automaton* is represented in Figure 2.10. Notice that, if the robot is in states 2, 5 or 8 it is not possible to move two states to the north, and therefore, the transition always occurs to the adjacent northern state.
>
> A clock structure for this automaton is still a set of 4 sequences,
>
> $$\{\mathbf{v}_N, \mathbf{v}_S, \mathbf{v}_E, \mathbf{v}_W\},$$
>
> each corresponding to an event N, S, E or W. However, it is no longer possible to determine exactly which is the state of the robot at each time instant. $\diamond$

When in the presence of uncertainty, localization consists in determining the probability of being in each state at each time instant t. Probabilistic localization is a widely studied topic of research [9, 87, 89]. Several approaches to probabilistic localization have been proposed in the literature, using similar underlying principles, but applied to different models, both discrete and continuous.

Let X_t be a random variable representing the state of the automaton at time instant t and suppose that a particular event $e \in \mathcal{E}$ occurs at some time t. We denote by $\mathsf{P}_e(i,j)$ the probability of $X_{t^+} = j$ given that $X_t = i$ and event e occurred, *i.e.*,

$$\mathbb{P}\left[X_{t^+} = j \mid X_{t^-} = i, E_t = e\right] = \mathsf{P}_e(i,j), \tag{2.2}$$

where E_t is the event occurred at time t. According to this model, the location of the robot in the topological map after event e depends only on the location of the robot immediately before the occurrence of e. This dependence of the state of the robot solely on the immediate past is similar to the Markov property observable in some stochastic processes. This, however, is not a mere coincidence, as it is possible to prove that, with an adequate clock structure, a stochastic timed automata generates a random process $\{X_t\}$ that is a Markov chain (see [55, 97] for details).

By using a Markov chain to model the movement of the robot in the environment, it is no longer necessary to take into account the exact time instants at which events occur, as established in [55]. Furthermore, the use of Markov chains greatly simplifies the treatment of localization and navigation, as will soon become apparent.

2.3.2 Markov chains and localization

At this stage, we once again resort to the environment of Figure 2.2 to introduce Markov chains as a suitable model for the movement of a mobile robot in a topological map.

Example 2.2. Consider once again the environment of Figure 2.2 and suppose that a mobile robot moves in that environment, described by the automaton of Figure 2.8.

Suppose, however, that all the events N, S, E and W have an uncertain outcome. In particular, suppose that each particular event moves the robot to the immediate adjacent state in the corresponding direction with probability 0.8; it moves the robot by *two* states in the corresponding direction with probability 0.1; and leaves the robot in the same state with probability 0.1. For example, for the event S, these probabilities can be summarized in the matrix

$$\mathsf{P}_S = \begin{bmatrix} 1.0 & 0.0 & 0.0 & 0.0 & 0.0 & 0.0 & 0.0 & 0.0 & 0.0 \\ 0.9 & 0.1 & 0.0 & 0.0 & 0.0 & 0.0 & 0.0 & 0.0 & 0.0 \\ 0.1 & 0.8 & 0.1 & 0.0 & 0.0 & 0.0 & 0.0 & 0.0 & 0.0 \\ 0.0 & 0.0 & 0.0 & 1.0 & 0.0 & 0.0 & 0.0 & 0.0 & 0.0 \\ 0.0 & 0.0 & 0.0 & 0.9 & 0.1 & 0.0 & 0.0 & 0.0 & 0.0 \\ 0.0 & 0.0 & 0.0 & 0.1 & 0.8 & 0.1 & 0.0 & 0.0 & 0.0 \\ 0.0 & 0.0 & 0.0 & 0.0 & 0.0 & 0.0 & 1.0 & 0.0 & 0.0 \\ 0.0 & 0.0 & 0.0 & 0.0 & 0.0 & 0.0 & 0.9 & 0.1 & 0.0 \\ 0.0 & 0.0 & 0.0 & 0.0 & 0.0 & 0.0 & 0.1 & 0.8 & 0.1 \end{bmatrix},$$

where $\mathsf{P}_S(i,j)$ represents the probabilities defined in (2.2). The uncertainty in the transitions resulting from the events can be interpreted as arising from

external perturbations, wheel slippage, terrain slope, etc. Notice that, as expected, the limits of the environment restrict the transitions of the robot (*e.g.*, the robot cannot move south from state 1 nor two states to the south from state 2). ◇

Consider once again the r.v. X_t representing the position of the robot on the map at time instant t (the state of the robot). Let $\mathcal{X}$ be the set of all possible states of the robot, henceforth referred to as the *state-space*. As seen in the example, if $X_t = i$ and an event e occurs, the probability of moving from i to j is given by $\mathsf{P}_e(i, j)$, with $i, j \in \mathcal{X}$ (notice that we have considered this transition probability to be time-independent).

We have distinguished the transitions between states based on the events that triggered the transition. For example, we considered a transition triggered by the event N to be different from a transition triggered by S. If, instead, we only wish to identify movements of the robot, we could consider a single event, $Move$, triggering all transitions. This event would have an uncertain outcome: sometimes it would correspond to a movement to the North; sometimes it would correspond to movement to the South, and so on. Notice that this is just a change in the representation and does not impact the actual movement of the robot.

If there is a single event triggering the transitions between states, the associated transition probabilities P_{Move} can be denoted by P, omitting the implicit event $Move$. With this formulation, the state of the robot at each time instant only depends on the state in the immediate past. If we consider the robot to move at discrete instants in time, $t = 1, 2, \ldots$, this means that the state X_t at each instant t only depends on X_{t-1}. This property is known as the *Markov property*, formally described as

$$\mathbb{P}\left[X_{t+1} = j \mid \mathcal{F}_t\right] = \mathbb{P}\left[X_{t+1} = j \mid X_t = i\right] = \mathsf{P}(i, j), \tag{2.3}$$

where $\mathcal{F}_t = \{X_0 = i_0, X_1 = i_1, \ldots, X_t = i\}$ is the history of the process up to time t. Any discrete-time stochastic process verifying (2.3) is called a *homogeneous Markov chain*.[2] We henceforth omit the qualifier homogeneous when referring to Markov chains, with the understanding that all referred chains are to be taken as homogeneous unless explicitly stated otherwise.

A Markov chain is completely described by its state-space $\mathcal{X}$ and its transition probability matrix P. Therefore, we may sometimes refer to a Markov chain as a pair $(\mathcal{X}, \mathsf{P})$.

By modeling the movement of the robot as a Markov chain, the problem of localization is easy to solve using standard probability results. Suppose that, at some time instant t, the robot is in state $X_t = i$. Using simple inductive reasoning, it is easy to show that

$$\mathbb{P}\left[X_{t+k} = j \mid X_t = i\right] = \sum_{l \in \mathcal{X}} \mathsf{P}^{k-1}(i, l)\mathsf{P}(l, j) = \mathsf{P}^k(i, j), \tag{2.4}$$

[2]The designation homogeneous arises from the fact that the transition probabilities in (2.3) do not depend on the particular time instant t considered.

where $\mathsf{P}^k(i,j)$ stands for the ij^{th} element of the matrix P^k. The matrix P^k is known as the *k-step transition probability matrix*.

Finally, suppose that at time instant t the position of the robot is not exactly known but the robot is estimated to be in state i with probability $\mu(i)$, where μ is a discrete probability measure on $\mathcal{B}(\mathcal{X})$. Since $\mathcal{X}$ is finite, it is possible to represent μ as a vector. Then, at time $t+k$ the robot will be in state $j \in \mathcal{X}$ with probability

$$\begin{aligned}\mathbb{P}_\mu\left[X_{t+k}=j\right] &= \sum_{i\in\mathcal{X}} \mathbb{P}\left[X_{t+k}=j \mid X_t=i\right]\mathbb{P}\left[X_t=i\right] = \\ &= \sum_{i\in\mathcal{X}} \mathsf{P}^k(i,j)\mu(i) = \\ &= \left(\mu^\top \mathsf{P}^k\right)(j),\end{aligned}$$

where $\left(\mu^\top \mathsf{P}^k\right)(j)$ represents the j^{th} component of vector $\mu^\top \mathsf{P}^k$.

2.4 Topological navigation

So far, we introduced topological maps as environment representations for navigation and Markov chains as dynamic models for the movement of a mobile robot. Within this framework we modeled the movement of the robot as state transitions in a Markov chain. However, we have so far disregarded any control that the robot may have over its own movement.

In fact, we would expect the movement of the robot to result (at least on some degree) from some "intentional action", rather than the outcome of some event generating mechanism as considered so far.[3] Furthermore, if the robot must choose between different possible "actions" to reach its ultimate goal, it is expectable that different actions have different outcomes and contribute in different extents to the completion of the robot's mission. As such, we now introduce the framework of *controlled Markov chains*. Controlled Markov chains enrich the simpler Markov chain model by allowing the robot to control its own movement (the trajectories of the chain).

Let $\{\mathsf{P}_a\}$ be a family of stochastic matrices,[4] where a is an index taking values in some finite set $\mathcal{A}$. Let $\{A_t\}$ be some sequence in $\mathcal{A}$ and $\{X_t\}$ a stochastic process such that

$$\mathbb{P}\left[X_{t+1}=j \mid X_t=i, A_t=a\right] = \mathsf{P}_a(i,j).$$

Clearly, $\{X_t\}$ verifies the Markov property and is, therefore, a Markov chain. However, the transition probabilities for this chain now depend on the sequence $\{A_t\}$, usually called the *control process*. In the context of this thesis, we will refer to the elements in $\mathcal{A}$ as being *actions* and to the set $\mathcal{A}$ as the *action-space*.

Suppose that a mobile robot is moving in an environment modeled as a topological map and, at each time instant t, the robot is allowed to choose the value of A_t. This means that the robot is able to control (probabilistically) its movement

[3] Notice that in considering the transitions as being triggered by some event/events we were not concerned on how such events were generated.

[4] A stochastic matrix is any matrix M verifying $\sum_j M(i,j) = 1$.

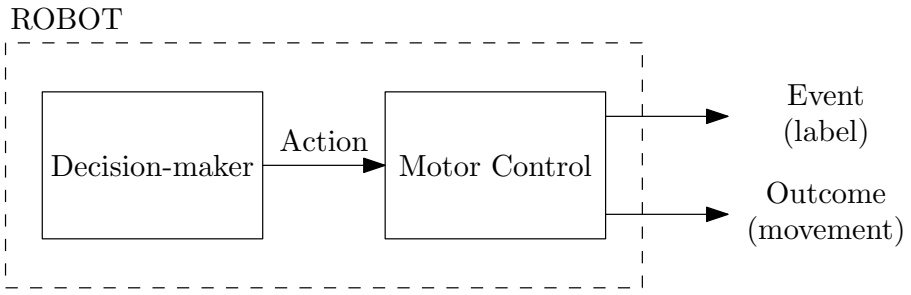


Figure 2.11: Distinction between event and action.

by choosing different values for A_t. This value may depend on the complete past history of the process and this dependency may be stochastic. Therefore, in general, the value of the A_t at time instant t is a r.v. taking values in $\mathcal{A}$. We represent a controlled Markov chain as a triplet $(\mathcal{X}, \mathcal{A}, \mathsf{P})$.

Example 2.1. (cont.) Consider yet again the environment depicted in Figure 2.2. By now, it should be clear that the movement of a mobile robot in this environment can be described by a controlled Markov chain $(\mathcal{X}, \mathcal{A}, \mathsf{P})$, where:

- As before, the state-space is $\mathcal{X} = \{1, \ldots, 9\}$;
- We now admit that the robot explicitly chooses the direction of its movement, at each time instant. Therefore, we have $\mathcal{A} = \{N, S, E, W\}$;
- Each action $a \in \mathcal{A}$ moves the robot in the corresponding direction, but there is some uncertainty in the outcome of each action. In particular, if the robot takes some action $a \in \mathcal{A}$ corresponding to one of the 4 possible directions, it will move to the adjacent state in that direction (if any) with probability p_1; it will move two states in that direction with probability p_2 (if possible) or, with a probability p_0, it will remain in the same state.

Notice that, in this model, different actions have (generally) different outcomes. The robot is now able to choose, at each time instant, the action A_t that controls its movement and it should be possible to derive a proper strategy to drive the robot to the goal. Finally, the addition of the control actions does not affect the ability of the robot to localize, and everything stated in the previous section on probabilistic localization still holds without any change.

Finally, one remark to emphasize the difference between *events* in the sense of Sections 2.2 and 2.3 and *actions*. Events can be seen as *labels* for the outcome of the *actions*. We present in Figure 2.11 a pictorial representation of this difference. ◇

As claimed in the example, the Markov chain model encompasses:

- *Localization in the environment*, as the state of the chain is related to the position of the robot in a topological map; so far we assume the robot to be able to observe such state;

- *Prediction* of the consequences of each action by means of the corresponding transition probabilities;

- *Movement control* by an adequate choice of different actions at successive time instants.

In order to be able to "program" the robot to perform a navigation task, it remains only to define a proper way to describe this task to the robot (and not necessarily how to accomplish it). For this, we resort to a reward structure to be introduced next.

2.4.1 Markov decision processes

So far, we have addressed the navigation of a mobile robot in an environment described by a topological map by using a controlled Markov chain to model the movement of the robot. The control parameter of the chain—the action—allows the robot to control its movement in the environment.

For the robot to be able to perform a desired task, it is imperative to provide it with a mechanism allowing it to choose the "correct" action at each time instant.

To this purpose, we assume that there is a deterministic function

$$r : \mathcal{X} \times \mathcal{A} \times \mathcal{X} \longrightarrow \mathbb{R}$$

assigning a *reward* $r(i, a, j)$ every time a transition from i to j occurs as a consequence of taking action a. Since the transition from i to j is a random event, we denote by $R(i, a)$ the random reward received for taking action a at state i. Clearly,

$$\mathbb{E}\left[R(i,a)\right] = \sum_{j \in \mathcal{X}} \mathsf{P}_a(i,j) r(i,a,j).$$

We assume that there is a constant $\mathcal{R} \in \mathbb{R}$ such that $|r(i, a, j)| < \mathcal{R}$ for all $i, j \in \mathcal{X}$ and all $a \in \mathcal{A}$.[5]

Formally, we define the objective of the robot as the determination of the control sequence $\{A_t\}$ that maximizes the infinite-horizon total discounted reward, defined as

$$V(\{A_t\}, i) = \mathbb{E}\left[\sum_{t=0}^{\infty} \gamma^t R(X_t, A_t) \mid X_0 = i\right], \tag{2.5}$$

where $0 \leq \gamma < 1$ is a discount-factor. As already stated, $R(i, a)$ represents a random "reward" received for taking action $a \in \mathcal{A}$ in state $i \in \mathcal{X}$. In a way, these rewards encode the *objective/objectives* of the robot (*e.g.*, move to some target state, avoid some dangerous state, follow some trajectory). By choosing its actions so as

[5]Usually, in the reinforcement learning literature, there is no assumption on the existence of a function r as the one described. However, the use of the function r greatly simplifies the notation and the presentation of the results and does not introduce a significant loss in generality. The assumption on the boundedness of r is tantamount to the standard requirement that the rewards have uniformly bounded variance.

to maximize the functional V above, the robot is also choosing its actions so as to accomplish whatever task is "encoded" in the rewards in an optimal way, as intended.

We refer to the 5-tuple $(\mathcal{X}, \mathcal{A}, \mathsf{P}, r, \gamma)$ as a *Markov decision process* (MDP). The rewards $r(i, a, j)$ are usually known as *reinforcements* and the problems modeled as MDPs are therefore known as reinforcement learning problems. Intuitively, one may think of $r(i, a, j)$ as a "prize" (or punishment, if $r(i, a, j) < 0$) for taking a given action a in state i and ending up in state j. Therefore, the rewards provide the robot with some "feedback" on how good its choice of action a was. This feedback is usually provided by some entity external to the robot (*e.g.*, a human operator).

With the reward mechanism defined by the function r and the functional V defined in (2.5), the navigation task for the robot is properly defined. The next example clarifies how rewards can be used to encode the robot's goal.

Example 2.1. (cont.) Consider once again the situation described in Example 2.1, where a mobile robot moves in the environment depicted in Figure 2.2. The navigation task consists in reaching Room 9. The Markov decision process describing the robot and task is the 5-tuple $(\mathcal{X}, \mathcal{A}, \mathsf{P}, r, \gamma)$ where

- $\mathcal{X}$, $\mathcal{A}$, P are as defined before;
- The reward function r assigns a reward of $+10$ for every transition triplet (i, a, j) such that $j = 9$ and 0 otherwise;
- γ is some discount factor such that $0 < \gamma < 1$.

Notice that the reward function above assigns a positive reward to every transition that ends in the goal state. Once the reward function is defined, the purpose of the robot is to choose its actions so at to maximize its total discounted reward,

$$V(\{A_t\}, i) = \mathbb{E}\left[\sum_{t=0}^{\infty} \gamma^t R(X_t, A_t) \mid X_0 = i\right].$$

Notice that, because of the discount factor γ, rewards that arise further in the future are "less desirable" than rewards arising in a nearer future. This implies that the optimal control sequence $\{A_t\}$ will move the robot towards the goal state *as quickly as possible*.

To further clarify this point, suppose that $p_1 = 1$ and $p_0 = p_2 = 0$, *i.e.*, the actions of the robot have a deterministic outcome. Further suppose that the robot is at state 8. It can then move to the goal state by choosing action N and receive a reward of 10. However, it can also choose for example action E and remain in state 8. In the next time step it can then choose action N and move to state 9. The expected reward until reaching the goal is 10 in the first case and 10γ in the second. Since $\gamma < 1$, the first course of action is preferable. ◇

To conclude this chapter, we briefly address the problem of determining the optimal control sequence $\{A_t\}$ given the MDP $(\mathcal{X}, \mathcal{A}, \mathsf{P}, r, \gamma)$.

2.4.2 Dynamic programming

Given an MDP $(\mathcal{X}, \mathcal{A}, \mathsf{P}, r, \gamma)$, we define the *optimal value function* V^* for each state $i \in \mathcal{X}$ as

$$V^*(i) = \max_{\{A_t\}} V(\{A_t\}, i) = \max_{\{A_t\}} \mathbb{E}\left[\sum_{t=0}^{\infty} \gamma^t R(X_t, A_t) \mid X_0 = i\right].$$

This value function is known to verify the recursive relation

$$V^*(i) = \max_{a \in \mathcal{A}} \sum_{j \in \mathcal{X}} \mathsf{P}_a(i,j)\big[r(i,a,j) + \gamma V^*(j)\big],$$

which is a form of the Bellman optimality equation. Notice $V^*(i)$ represents the total discounted reward that the robot expects to receive by starting at state $i \in \mathcal{X}$ and always acting optimally. We also define the optimal Q-values $Q^*(i,a)$ for each state-action pair $(i,a) \in \mathcal{X} \times \mathcal{A}$ as

$$Q^*(i,a) = \sum_{j \in \mathcal{X}} \mathsf{P}_a(i,j)\big[r(i,a,j) + \gamma V^*(j)\big]. \tag{2.6}$$

The function

$$Q^* : \mathcal{X} \times \mathcal{A} \longrightarrow \mathbb{R},$$

assigning to each pair $(i,a) \in \mathcal{X} \times \mathcal{A}$ the corresponding optimal Q-value, $Q^*(i,a)$, is referred henceforth as the *optimal Q-function*. Much like V^*, Q^* represents the total discounted reward that the robot expects to receive by starting at state $i \in \mathcal{X}$, *taking $a \in \mathcal{A}$ as the first action* and always acting optimally afterwards.

From the optimal Q-function it is possible to define a mapping $\delta^* : \mathcal{X} \longrightarrow \mathcal{A}$ as

$$\delta^*(i) = \arg\max_{a \in \mathcal{A}} Q^*(i,a) \tag{2.7}$$

for all $i \in \mathcal{X}$. The control process $\{A_t^*\}$ defined by $A_t^* = \delta^*(X_t)$ is optimal in the sense that

$$V^*(i) = V(\{A_t^*\}, i),$$

for all $i \in \mathcal{X}$. The mapping δ^* is known as the *optimal policy* for the Markov decision process $(\mathcal{X}, \mathcal{A}, \mathsf{P}, r, \gamma)$.[6]

More generally, a (stochastic) *policy* is a mapping $\delta_t : \mathcal{X} \times \mathcal{A} \longrightarrow [0,1]$ that generates a control process $\{A_t\}$ verifying

$$\mathbb{P}\left[A_t = a \mid X_t = i\right] = \delta_t(i,a),$$

for all t. In other words, a policy defines the probability of choosing each particular action in each state at each time instant. Clearly, since δ_t defines for each $i \in \mathcal{X}$

[6] Although we refer to δ^* as *the* optimal policy, this policy is not necessarily unique. However, all optimal policies yield the same total discounted reward and so the possible existence of multiple optimal policies does not bring any additional complication.

a probability distribution over $\mathcal{A}$, it holds that $\sum_a \delta_t(i,a) = 1$ for all t. We write $V^{\delta_t}(i)$ instead of $V(\{A_t\}, i)$, whenever the control process $\{A_t\}$ is generated by a policy δ_t and refer to V^{δ_t} as the value function associated with policy δ_t.

In the particular case where, for each state $i \in \mathcal{X}$, $\delta_t(i,a) = 1$ for some $a \in \mathcal{A}$, we say that δ_t is a *deterministic policy*. In this case, we write $\delta_t(i)$ to denote the action in $\mathcal{A}$ which is taken w.p.1 in state i. The control process $\{A_t\}$ thus generated verifies $A_t = \delta_t(X_t)$ for all t. On the other hand, if a policy δ_t does not depend on t, we say that δ_t is a *stationary policy* and denote it simply as δ.

It should be clear from (2.7) that the optimal control sequence can be obtained by determining the optimal (deterministic and stationary) policy δ^*, which in turn can be obtained from Q^*. Therefore, the problem of optimally navigating a mobile robot described by the MDP $(\mathcal{X}, \mathcal{A}, \mathsf{P}, r, \gamma)$ is "solved" once the function Q^* is known for all pairs $(i,a) \in \mathcal{X} \times \mathcal{A}$.

For the purpose of determining Q^*, we consider once again Equation (2.6). This expression suggests the following fixed-point iteration

$$Q_{k+1}(i,a) = \sum_{j \in \mathcal{X}} \mathsf{P}_a(i,j)\big[r(i,a,j) + \gamma \max_{b \in \mathcal{A}} Q_k(j,b)\big], \qquad \text{for all } i,a. \tag{2.8}$$

To show that the iterative process in (2.8) converges to the optimal Q-values, we define an operator $\mathbf{H}$ as

$$(\mathbf{H}q)(i,a) = \sum_{j \in \mathcal{X}} \mathsf{P}_a(i,j)\big[r(i,a,j) + \gamma \max_{b \in \mathcal{A}} q(j,b)\big],$$

where $q : \mathcal{X} \times \mathcal{A} \longrightarrow \mathbb{R}$ is any bounded measurable function. The operator $\mathbf{H}$ is a contraction in the sup-norm with contraction factor γ [27]. Therefore, the convergence of (2.8) is an immediate consequence of the Banach fixed-point theorem. This method and several standard variations of it are known as *value iteration* methods (VI). A good overview of these methods and their convergence properties can be found in [27]. We will make further use of the operator $\mathbf{H}$ in the following chapters, where stochastic versions of iteration (2.8) are introduced.

Example 2.1. (cont.) We finally bring Example 2.1 to a satisfying conclusion. Recall that we are considering a mobile robot moving in the environment of Figure 2.2 and modeled by the MDP $(\mathcal{X}, \mathcal{A}, \mathsf{P}, r, \gamma)$ described in the previous example. Suppose that $\gamma = 0.9$, $p_0 = 0.1$, $p_1 = 0.8$ and $p_2 = 0.1$.

We applied the VI methodology above and obtained the Q^* function represented in Table 2.1.

To provide an interpretation for the obtained Q-values, consider for example, those concerning state 9. If the robot chooses either action N or E, and since no further movement in that direction is possible, the robot will remain in state 9 and receive a reward of 10 in every time step. Thus,

$$Q^*(9,N) = Q^*(9,E) = \frac{10}{1-\gamma} = 100.$$

On the other hand, if the robot is in state 8, the optimal action is N. This means that, if the robot chooses action N, it will move to state 9 with proba-

Table 2.1: Optimal Q and value functions for the Markov decision process from the previous example. The boldface cells represent optimal values.

States	$Q^*(i,a)$				V^*
	N	S	E	W	
1	**71.7**	64.5	**71.7**	64.5	71.7
2	**79.4**	65.2	**79.4**	71.5	79.4
3	80.3	71.7	**89.2**	80.3	89.2
4	**79.4**	71.5	**79.4**	65.2	79.4
5	**88.0**	72.3	**88.0**	72.3	88.0
6	89.0	79.4	**98.9**	81.2	98.9
7	**89.2**	80.3	80.3	71.7	89.2
8	**98.9**	81.2	89.0	79.4	98.9
9	**100.0**	89.2	**100.0**	89.2	100.0

bility 0.9 and remain in the same state otherwise. This means that,

$$Q^*(8,N) = 0.9\left(10 + \gamma \underbrace{\frac{10}{1-\gamma}}_{\text{Value of state 9}}\right) + 0.1\big(0 + \gamma Q^*(8,N)\big).$$

Solving this equation readily leads to the value of 98.9.

From Q^* we can immediately obtain the optimal value function V^*, simply by considering the maximum of each line. This yields the function in the last column of Table 2.1. The optimal policy can also be obtained from Q^*. For example, in state 1, the two actions yielding maximum reward are N and E. Suppose that the robot chooses action N and moves to state 2. Again in this state, actions N and E are both optimal. If, once again, the robot chooses action N and moves to state 3 then, in state 3, the robot will choose action E and the same will occur in state 6. We further illustrate the optimal policy in the diagram of Figure 2.12. The arrows leaving a particular state correspond to the directions of the optimal actions in that state. For clarity, we have omitted the optimal actions in the goal state.

◇

2.5 Concluding remarks

To conclude this chapter, we summarize the main concepts presented herein. We briefly discuss the main ideas so far and point out the direction of the presentation in the following chapters.

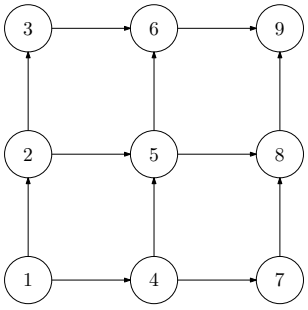


Figure 2.12: Optimal policy for the MDP in the previous example. Notice that, in this representation, all trajectories obtained by following the arrows lead to the goal state in a minimum number of steps, as expected.

2.5.1 Summary

The main purpose of this chapter was to introduce MDPs as a suitable framework for topological navigation. It is an introductory chapter, aiming at establishing the notation to be used in the forthcoming chapters and to motivate the use of MDPs to address the problem of navigating a single mobile robot in an environment described by a topological map.

We started by reviewing several bibliographical references in the topic of autonomous mobile robot navigation, enumerating the main different approaches to the problem. We defined navigation as the process of maintaining a robot's course to a goal location and divided this process into three important steps: defining a map, localizing the robot in that map and driving the robot towards the goal.

Starting by the mapping problem, we classified maps into three main categories: metric, topological and hybrid. We opted for a topological representation, valuing its scalability and high level of abstraction.

We used discrete event systems to model the movement of the robot in a topological map. In particular, we argued that Markov chains provide a suitable probabilistic framework to the problem of localization, taking into account uncertainty in the movement of the robot. Markov chains are stochastic processes in which the state at each time instant depends solely on the immediately past state.

A Markov chain is completely described by a set of transition probabilities that indicate the probability of moving between any two states in the environment. By allowing these transition probabilities to depend on a discrete parameter controlled by the agent—*the action*—we have a *controlled Markov chain*. With this set of actions controlling the transitions between the states of the Markov chain, the robot is now able to control its trajectory up to some degree of inherent uncertainty in the transitions.

The introduction of an additional reward-assigning mechanism completes the definition of a *Markov decision process*. The reward-assigning mechanism is used to define an optimality criterion for the robot to meet. In a sense, the rewards "encode" the mission of the robot.

Finally, we concluded the chapter by presenting a simple dynamic programming algorithm to compute the optimal control for the robot.

2.5.2 Discussion

MDPs arise in a multitude of contexts essentially related to reinforcement learning. Reinforcement learning differs from other learning paradigms in that the correct action is not provided to the robot, as in supervised learning, nor is the robot required to learn without any feedback on its performance, as in unsupervised learning. Instead, a *reinforcement signal* is provided, in the form of rewards. These rewards indirectly indicate which should be the proper choice of actions.

The VI method described in this chapter is a dynamic programming (DP) algorithm. It can be computed off-line and only the optimal policy needs to be implemented in the robot. As such, no apparent learning takes place in the process, for which it would seem strange to classify such approach as reinforcement learning. However, the fact is that any such problem where the robot assesses the validity of its choice of actions upon a received reinforcement is classified as a reinforcement learning problem. In this particular case, learning takes place off-line rather than as a consequence of interaction with the environment. In the next chapter we describe other methods that lead to the solution of the MDP (either of V^*, Q^* or δ^*) from direct interaction with the environment. This may be required if, for example, the robot has no explicit knowledge of the reward function.

The use of MDPs for mobile robot navigation clearly requires some other mechanism to interface with the lower level control. In fact, it is necessary to translate at some lower-level stage the information from the sensors into topological information for the robot and the high-level action commands into control signals for the robot's actuators (such as motors).

The problem of the interface between the abstract topological level and the low level sensor raw data and actuator control is approached using different perspectives in the topological navigation works cited in Section 2.1. It is an interesting topic of research since, as seen, discrete state approaches such as the one described in this chapter allow the treatment of the navigation problem at a higher level of abstraction, approaching the human process of navigation. We do not address such topic in this thesis, but several references can be found along Sections 2.1 and 2.2.

Two final remarks should be made. First of all, in this chapter we have motivated the use of the MDP framework in mobile robot navigation. In this framework, the robot must determine a map between *states*, corresponding to the positions of the robot, and *actions*. In practical situations, the robot will estimate its position (the state) from noisy sensor measurements and there will be unavoidable uncertainty in this estimation. We postpone the discussion of this issue to Chapter 3, where we explicitly address the problem of *partial observability*.

Secondly, the framework of Markov decision processes is suited for a wide range of different applications, from power systems control [73] to *e*-mail applications [329]. We adopt in the next chapters the more common computer science nomenclature and refer to the robot as an *agent*.

Chapter 3

Reinforcement Learning in Finite State-Spaces

In this chapter, we review several reinforcement learning algorithms for finite-state MDPs. Unlike the VI method in Chapter 2, the methods in this chapter do not assume any explicit knowledge of the MDP model and rely solely on the interaction of the agent with its environment.

Specifically, given a policy δ, we feature the functions V^δ, V^* and Q^* described in the previous chapter as fixed-points of corresponding operators and study how these functions can be determined by direct interaction with the environment. To this purpose, we review several methods from the literature that build a model for the underlying Markov process (model-based methods) and then use that model to determine the desired functions; we also review other methods that directly determine the desired functions without the need to estimate a model of the MDP

(model-free methods). We identify the conditions under which all these methods converge and apply them to a simple navigation example.

3.1 Reinforcement learning

Reinforcement learning is a topic of research that addresses the problem of an agent required to sequentially choose an action from a set of possible actions in order to meet some optimality criterion. In the problems addressed in the thesis, the agent is a mobile robot and the optimality criterion is met as long as the robot optimally accomplishes some navigation task. In the model of Markov processes introduced in the previous chapter, we saw that different actions may contribute differently to the accomplishment of the final mission, depending on the state (position) of the robot. Different actions also influence in different ways the temporal evolution of such state.

The methods designed to address reinforcement learning problems—commonly dubbed as reinforcement learning methods—essentially compute a mapping from the set of all possible states of the system to the set of all possible actions of the agent, defining which action should be taken in each individual state. As seen in the previous chapter, such mapping is called a *policy* and it is customary to define a utility-function, or *value-function*, estimating the practical utility of each particular policy.

Reinforcement learning methods can be divided into two main categories. The first category comprises the so-called *policy-search methods* (or actor-only methods). These algorithms maintain explicit representations of policies and update such representations using different search strategies. Examples include policy iteration [27, 295], simulated annealing [285], evolutionary algorithms [205] and others [179, 288].

The second category relies on the computation of value-functions and we refer to such methods as *value-based methods* (or critic-only methods). These methods estimate these utility functions and use them to determine the optimal policy. Examples include gradient-based search [10, 12], dynamic programming [265] or stochastic approximation. As claimed in [134], this second class of methods uses the special structure of reinforcement learning problems as constraints which decrease variance of the estimates.[1]

It is not clear if any of the two classes of methods is preferable to the other and one idea is to devise some combined approach that takes advantage of the appealing properties of both. Recent years have witnessed an increasing interest in the so-called *actor-critic methods* [10, 145, 239, 296]. In this thesis, we focus on value-based algorithms.

* * *

[1] Simple policy-search methods which (often) disregard the inherent structure of Markov processes usually present high-variance estimates. There are, however, some policy-search methods which specifically account for this variance, and produce low-variance estimates. See, for example, [2, 3].

The VI algorithm described in the previous chapter is a value-based method that computes the optimal Q-function using a dynamic programming approach. Dynamic programming algorithms like VI are computationally efficient but usually require a complete model of the Markov process [100, 170, 228].

There are other known value-based reinforcement learning methods that require little *a priori* knowledge on the Markov process. Furthermore, they exhibit a performance no worse than that obtained using dynamic programming [16]. Examples include both prediction methods (*e.g.*, *temporal difference learning* [289]) and control methods (*e.g.*, *SARSA* [263, 289] and *Q-learning* [332, 333]). The book by Sutton and Barto [295] provides a comprehensive introduction to reinforcement learning.

Such methods have been exhaustively covered in the literature [27, 295] and, under mild assumptions, have been proven to converge to the desired value-function [27, 108, 131, 280]. These proofs have enormously benefitted from the developments on the convergence theory of stochastic approximation algorithms. Central references on the topic are the books by Benveniste et al. [21] and Kushner and Yin [153].

Several reinforcement learning algorithms have been proposed in the literature to address MDPs with different optimality criteria from the one proposed in the previous chapter. For example, instead of the total discounted reward, the decision-maker could aim at maximizing the *average per step reward*, as in [4, 178, 322]. However, the total discounted reward criterion has been widely studied and it is possible to find it in numerous applications [16, 76, 107, 108]. Furthermore, in many situations, the two aforementioned criteria are equivalent [322].[2] In this thesis, we therefore restrict to the more standard total discounted reward criterion introduced in Chapter 2.

3.2 Learning and fixed-point computations

In the previous chapter, we have introduced MDPs as a framework to address single-agent navigation problems. We described an MDP as a tuple $(\mathcal{X}, \mathcal{A}, \mathsf{P}, r, \gamma)$, where $\mathcal{X}$ is the (finite) state-space, $\mathcal{A}$ is the action-space, P represents the transition probabilities for the underlying controlled Markov chain and r is the reward function. The parameter γ is a discount factor assigning more importance to immediate decisions than to those further into the future.

As in Chapter 2, in this chapter we only address problems with *finite state-spaces*. We also consider that the agent is able to unambiguously perceive its state, *i.e.*, it has *perfect state perception*. Only in the next chapter, as we introduce methods that can handle MDPs with infinite state-spaces, will we be endowed with the tools to alleviate this assumption of perfect perception. Therefore, we postpone the discussion of situations where the agent perceives its state with a certain degree of uncertainty to Chapter 4.

In this chapter, we present several reinforcement learning methods used to determine the optimal value-function for an MDP by direct interaction with the environment.

[2] Equivalence here means that both criteria yield the same optimal policy.

* * *

Recall from the previous chapter that the VI method could be used to generate a sequence $\{Q_t\}$ of estimates for the optimal Q-values. The VI iteration

$$Q_{k+1}(i,a) = \sum_{j\in\mathcal{X}} \mathsf{P}_a(i,j)\big[r(i,a,j) + \gamma \max_{b\in\mathcal{A}} Q_k(j,b)\big] \tag{3.1}$$

determines the fixed point Q^* of the operator $\mathbf{H}$ defined as

$$(\mathbf{H}q)(i,a) = \sum_{j\in\mathcal{X}} \mathsf{P}_a(i,j)\big[r(i,a,j) + \gamma \max_{b\in\mathcal{A}} q(j,b)\big] \tag{3.2}$$

for any mapping $q : \mathcal{X}\times\mathcal{A} \longrightarrow \mathbb{R}$. Given a policy δ, the corresponding value function V^δ also verifies the recursive relation

$$V^\delta(i) = \sum_{j\in\mathcal{X}}\sum_{a\in\mathcal{A}} \delta(i,a)\mathsf{P}_a(i,j)\big[r(i,a,j) + \gamma V^\delta(j)\big]$$

and is therefore the fixed point of the operator $\mathbf{T}^\delta$ defined as

$$(\mathbf{T}^\delta v)(i) = \sum_{j\in\mathcal{X}}\sum_{a\in\mathcal{A}} \delta(i,a)\mathsf{P}_a(i,j)\big[r(i,a,j) + \gamma v(j)\big]. \tag{3.3}$$

for any mapping $v : \mathcal{X} \longrightarrow \mathbb{R}$. Finally, the optimal value function V^* also verifies the recursive relation

$$V^*(i) = \max_{a\in\mathcal{A}} \sum_{j\in\mathcal{X}} \mathsf{P}_a(i,j)\big[r(i,a,j) + \gamma V^*(j)\big]$$

and is therefore the fixed point of an operator $\mathbf{T}$ defined for any mapping $v : \mathcal{X} \longrightarrow \mathbb{R}$ as

$$(\mathbf{T}v)(i) = \max_{a\in\mathcal{A}} \sum_{j\in\mathcal{X}} \mathsf{P}_a(i,j)\big[r(i,a,j) + \gamma v(j)\big]. \tag{3.4}$$

Using fixed-point iterations such as the one in (3.1), it is possible to determine (off-line) each of the three-functions V^δ, V^* and Q^*. Such DP methods are extensively covered in the literature [27, 295]. They are fast and reliable and can be used to determine any of the three functions V^δ, V^* and Q^*, for any policy δ. This is because each of the three functions V^δ, V^* and Q^* can be represented in a table of real entries.[3] Therefore, at each iteration it is possible to *individually* update the value of V^δ, V^* and Q^* at each state $i \in \mathcal{X}$ or pair $(i,a) \in \mathcal{X}\times\mathcal{A}$.

However, in order to iterate the operators $\mathbf{H}$, $\mathbf{T}$ or $\mathbf{T}^\delta$ we need to have exact knowledge of the transition probabilities P as well as of the reward function r. Such information may not always be available and it would be of interest to devise a method to determine V^δ, V^* and/or Q^* without explicit knowledge of P and r.

[3] V^* and V^δ can be represented using a $|\mathcal{X}|\times 1$ table containing the value of each state $i\in\mathcal{X}$ w.r.t. the policies δ^* and δ, respectively; Q^* can be represented using a $|\mathcal{X}|\times|\mathcal{A}|$ table containing the optimal Q-values.

Instead, we would like the agent (robot) to learn these functions—V^δ, V^* and Q^*—from direct interaction with the environment.

The importance of learning how to act from direct interaction with the environment will be particularly evident in Part II, where multiple robots move in a common environment and must learn a coordinated behavior by interacting with the environment and with each other. We also remark that, in several situations considered in the forthcoming chapters, we admit the agent (robot) to have a model of its movement, *i.e.*, it knows P. It does not, however, know its mission beforehand, *i.e.*, it does not explicitly know the reward function r.

In the remainder of the chapter we describe two possible approaches to learn the referred functions by direct interaction with the environment:

- *Model-based* (Section 3.3): We use a sample path of the MDP $(\mathcal{X}, \mathcal{A}, \mathsf{P}, r, \gamma)$ to estimate the transition probabilities P and the rewards r. We then use these estimates to determine Q^*, V^* and/or V^δ by performing fixed-point iterations as described previously;

- *Model-free* (Section 3.4): We use a sample path of the MDP and apply stochastic approximation to determine Q^*, V^* and/or V^δ.

Notice an important difference between DP methods like VI and the model-based and model-free methods to be described. The former encompass iterative algorithms that can be run *off-line*, while the latter rely on *sample paths* obtained by interaction with the process and are thus naturally *on-line*.

3.3 Model-based learning

In this section, we address *model-based learning*.[4] In model-based learning, the agent builds a model of its environment by direct interaction with it. It then uses this learnt model to determine the functions Q^*, V^* and V^δ.

Since V^* can be determined immediately from Q^* as

$$V^*(i) = \max_{a \in \mathcal{A}} Q^*(i, a),$$

we focus on methods to determine V^δ and Q^*.

The two algorithms presented in this section, ARTVI and ARTQI, are based in a common working principle: using experience gathered from direct interaction with the environment, the agent builds statistical estimates $\hat{\mathsf{P}}$ and $\hat{r}$ of the model parameters P and r. These estimated parameters are then used to perform asynchronous DP updates leading to the functions of interest, V^δ and Q^*. As already mentioned, both these methods are on-line methods (they run as the agent interacts with the environment).

This technique is known as *adaptive real time dynamic programming* (ARTDP), and was first introduced in [17, 47]. Convergence of this method and several possible variations thereof was established in [113]. The working principle in ARTDP is also

[4] Model-based learning methods are also known as *indirect methods*.

similar to the one behind the DYNA architecture [291, 292, 293, 294], the DYNA-Q architecture [236] and the prioritized sweeping algorithm [203, 204].

3.3.1 Determining V^δ

Let $(\mathcal{X}, \mathcal{A}, \mathsf{P}, r, \gamma)$ be an MDP modeling a sequential decision problem and suppose that the decision-maker follows a fixed policy δ. We wish to evaluate the value-function V^δ from direct interaction with the environment.

We first notice that, when using a fixed policy δ, it is possible to consider a non-controlled Markov chain $(\mathcal{X}, \mathsf{P}_\delta)$, where the transition probabilities P_δ are given by

$$\mathsf{P}_\delta(i,j) = \sum_{a \in \mathcal{A}} \delta(i,a) \mathsf{P}_a(i,j).$$

Since the agent follows the policy δ, its transitions are governed by the transition probabilities P_δ thus defined.

Suppose then that $\{i_t\}$ and $\{r_t\}$ are infinite sample sequences of states and rewards experienced by the agent when following policy δ. Denoting by $n_t(i,j)$ the number of transitions from state i to state j experienced up to time t, the quantity

$$\hat{\mathsf{P}}_t(i,j) = \frac{n_t(i,j)}{\sum_{k \in \mathcal{X}} n_t(i,k)} \tag{3.5}$$

is an unbiased estimator of the transition probabilities $\mathsf{P}_\delta(i,j)$ obtained from the sample trajectory $\{i_t\}$. It estimates the probability of moving to j when starting in i by determining the percentage of times that this transition occurred on the total number of visits to state i.

Incrementally, this estimator can be computed as follows. Let $n_t(i)$ denote the number of visits to state i in the finite sequence $\{i_0, \ldots, i_t\}$. Then, for each $t \in \mathcal{T}$, $\hat{\mathsf{P}}_t$ can be updated as

$$\hat{\mathsf{P}}_{t+1}(i,j) = \hat{\mathsf{P}}_t(i,j) + \frac{\mathbb{I}_i(i_t)}{n_{t+1}(i)} \big(\mathbb{I}_j(i_{t+1}) - \hat{\mathsf{P}}_t(i,j) \big), \tag{3.6}$$

where $\mathbb{I}_x(\cdot)$ is the indicator function for x, given by

$$\mathbb{I}_x(z) = \begin{cases} 1 & \text{if } z = x; \\ 0 & \text{otherwise.} \end{cases}$$

Therefore, every time instant that the agent visits state i and moves to state j, it increases the estimated transition probability $\hat{\mathsf{P}}(i,j)$, as indicated in (3.6), and decreases all other estimates $\hat{\mathsf{P}}(i,j')$ for $j' \neq j$. For $i' \neq i$, all the estimated probabilities $\hat{\mathsf{P}}(i',\cdot)$ remain unchanged. If all states are visited infinitely often (which occurs, for example, if the chain $(\mathcal{X}, \mathsf{P}_\delta)$ is ergodic), the sequence $\{\hat{\mathsf{P}}_t\}$ will converge to P_δ w.p.1.

A similar procedure can be used to estimate the function r. In this case, given

the sample sequences of states and rewards, $\{i_t\}$ and $\{r_t\}$, the iteration

$$\hat{r}_{t+1}(i,j) = \hat{r}_t(i,j) + \frac{\mathbb{I}_{(i,j)}(i_t, i_{t+1})}{n_{t+1}(i,j)}\big(r_t - \hat{r}_t(i,j)\big) \tag{3.7}$$

generates a sequence $\{\hat{r}_t\}$ converging w.p.1 to the function r_δ given by

$$r_\delta(i,j) = \mathbb{E}_\delta\left[R_t \mid X_t = i, X_{t+1=j}\right] = \frac{\sum_{a\in\mathcal{A}} \delta(i,a)\mathsf{P}_a(i,j)}{\sum_{b\in\mathcal{A}} \delta(i,b)\mathsf{P}_b(i,j)} r(i,a,j),$$

where, once again, $\mathbb{I}$ denotes the indicator function.

Finally, computing at each time instant

$$V_{t+1}(i_t) = \sum_{j\in\mathcal{X}} \hat{\mathsf{P}}_t(i_t,j)\big(\hat{r}_t(i_t,j) + \gamma V_t(j)\big), \tag{3.8}$$

we obtain a sequence $\{V_t\}$ converging to V^δ w.p.1. This iterative procedure is referred as *adaptive real time value iteration* (ARTVI) and its convergence is formally established in Subsection 3.3.3.

3.3.2 Determining Q^*

Applying ARTDP to the determination of Q^* involves additional complications that do not arise in the determination of V^δ. In particular, it is necessary to explore the state-action-space in order to ensure that no optimal actions are left unexplored, since this could eventually lead to the determination of a sub-optimal Q-function.

Let δ be a stochastic policy verifying $\delta(i,a) > 0$ for all $(i,a) \in \mathcal{X}\times\mathcal{A}$. To estimate P we modify (3.6) to

$$\hat{\mathsf{P}}_{t+1}(i,a,j) = \hat{\mathsf{P}}_t(i,a,j) + \frac{\mathbb{I}_{(i,a)}(i_t,a_t)}{n_{t+1}(i,a)}\big(\mathbb{I}_j(i_{t+1}) - \hat{\mathsf{P}}_t(i,a,j)\big), \tag{3.9}$$

where $n_t(i,a)$ denotes the number of occurrences of the pair (i,a) in the history up to time t, given by $\mathcal{H}_t = \{i_0, a_0, i_1, \ldots, a_{t-1}, i_t\}$. To estimate r, we modify (3.7) to

$$\hat{r}_{t+1}(i,a,j) = \hat{r}_t(i,a,j) + \frac{\mathbb{I}_{(i,a,j)}(i_t,a_t,i_{t+1})}{n_{t+1}(i,a,j)}\big(r_t - \hat{r}_t(i,a,j)\big), \tag{3.10}$$

where now $n_t(i,a,j)$ denotes the number of occurrences of the transition triplet (i,a,j) in the history $\mathcal{H}_t$.

Under suitable conditions yet to be specified, the estimates $\{\hat{\mathsf{P}}_t\}$ converge to P w.p.1, and the estimates $\{\hat{r}_t\}$ converge to r.[5]

To compute Q^*, we apply the fixed point iteration in (3.1) using $\hat{\mathsf{P}}_t$ and $\hat{r}_t$ instead

[5]Notice that since $r(i,a,j)$ is a deterministic function of (i,a,j), the sequence $\{\hat{r}_t\}$ converges to r *point-wise*. This is a stronger statement than w.p.1 convergence: for all $\omega\in\Omega$, $\hat{r}_t(\omega) \to r(\omega)$.

of P and r, *i.e.*,

$$Q_{t+1}(i_t, a_t) = \sum_{j \in \mathcal{X}} \hat{\mathsf{P}}_t(i_t, a_t, j)\big(\hat{r}_t(i_t, a_t, j) + \gamma \max_{b \in \mathcal{A}} Q_t(j, b)\big). \tag{3.11}$$

The sequence obtained using this expression, $\{Q_t\}$, converges to the desired Q^* w.p.1. This iterative procedure is referred as *adaptive real time Q-iteration* (ARTQI) and its convergence is formally established in Subsection 3.3.3.

* * *

We now discuss how the choice of actions of a decision-maker in an MDP changes as it learns the optimal Q-function. We henceforth refer to the policy used to *explore* the environment as the *learning policy*. The policy whose value-function or Q-function is learnt is referred to as the *learnt policy*.

In the following definition, we present the idea of *convergence in behavior* from [166].

Convergence in Behavior

A learning agent is said to *converge in behavior* if its learning policy converges to a stationary policy as $t \to \infty$.

In describing ARTQI we required that $\delta(i, a) > 0$ for all pairs $(i, a) \in \mathcal{X} \times \mathcal{A}$. The formal justification for this is provided in Theorem 3.3.2. For the moment, we just take such requirement as necessary and discuss the existence of policies that actually verify it. There are numerous policies ensuring this condition. As an example, consider the uniform policy: $\delta(i, a) = \frac{1}{|\mathcal{A}|}$, for all $(i, a) \in \mathcal{X} \times \mathcal{A}$. This uniform policy does ensure the condition above. However, it does not guarantee that the agent converges in behavior to the *optimal* policy δ^*. In other words, the uniform policy does not guarantee that, as $t \to \infty$, the agent's choice of actions converges to optimality (even though its learning policy is stationary from the start).

On the other hand, the agent could simply use the *greedy policy* w.r.t. Q_t. This is the deterministic policy given by

$$\delta_g^{Q_t}(i, a) = \begin{cases} 1 & \text{if } a = \arg\max_{b \in \mathcal{A}} Q_t(i, b); \\ 0 & \text{otherwise,} \end{cases}$$

for each $(i, a) \in \mathcal{X} \times \mathcal{A}$. Notice that this policy depends on the current estimate of Q^*. If $Q_t \to Q^*$, then $\delta_g^{Q_t} \to \delta^*$ and the agent will converge in behavior to the optimal policy. The problem with this policy is that it does not satisfy $\delta(i, a) > 0$. This, as argued before, may lead to insufficient exploration of the state-action-space $\mathcal{X} \times \mathcal{A}$ and, therefore, does not guarantee that $Q_t \to Q^*$.

To deal with such problem, Singh et al. [280] proposed the use of policies that have the *GLIE property* (greedy in the limit with infinite exploration). Given a function $Q : \mathcal{X} \times \mathcal{A} \longrightarrow \mathbb{R}$, denote by δ_g^Q the greedy policy w.r.t. Q.

GLIE Policy

Given an MDP $(\mathcal{X}, \mathcal{A}, \mathsf{P}, r, \gamma)$ and a function $Q : \mathcal{X} \times \mathcal{A} \longrightarrow \mathbb{R}$, a (non-stationary) policy δ_t is *greedy in the limit with infinite exploration* w.r.t. Q if $\delta_t(i, a) > 0$ for all $t \in \mathcal{T}$, $(i, a) \in \mathcal{X} \times \mathcal{A}$ and $\delta_t \to \delta_g^Q$ as $t \to \infty$.

An example of a GLIE policy is *Boltzmann exploration*

$$\delta_t(i, a) = \frac{e^{Q_t(i,a)/\tau_t(i)}}{\sum_{b \in \mathcal{A}} e^{Q_t(i,b)/\tau_t(i)}},$$

where the parameter $\tau_t(i)$ is known as *temperature parameter*. It is a state-dependent parameter converging to zero as $t \to \infty$, thus ensuring that $\delta_t(i, a) > 0$ for all (i, a) and that $\delta_t \to \delta^*$ as $t \to \infty$ (as long as $Q_t \to Q^*$).

More details on GLIE policies can be found in [280].

3.3.3 Convergence of model-based learning

Insofar, we have introduced two model-based learning methods to determine V^δ and Q^* (ARTVI and ARTQI). Both methods estimate the underlying MDP model as a tuple $(\mathcal{X}, \mathcal{A}, \hat{\mathsf{P}}, \hat{r}, \gamma)$ and use this estimated model to perform DP updates on the estimates for V^δ and Q^*.

We have claimed (rather informally) that both methods converge to the desired functions, but have not yet established this fact or described the exact conditions that ensure this convergence. The following statements formalize this claim.

Theorem 3.3.1. *Given a finite-state MDP $(\mathcal{X}, \mathcal{A}, \mathsf{P}, r, \gamma)$ and a stationary policy δ, the sequence of estimates $\{V_t\}$ generated by ARTVI according to* (3.8) *converges to V^δ w.p.1 for any initial estimate V_0, as long as every state $i \in \mathcal{X}$ is visited infinitely often.*

Proof See Appendix F.1. □

The condition of infinite visits to every state can be reformulated by requiring the chain $(\mathcal{X}, \mathsf{P}_\delta)$ to be irreducible (see Appendix B.10). We will see that this irreducibility assumption will also implicitly arise in the convergence algorithms of Chapter 4.

As for ARTQI, we restate Theorem 3.3.1 as follows.

Theorem 3.3.2. *Given a finite-state MDP $(\mathcal{X}, \mathcal{A}, \mathsf{P}, r, \gamma)$, the sequence of estimates $\{Q_t\}$ generated by ARTQI according to (3.11) converges to Q^* w.p.1 for any initial estimate Q_0, as long as every state-action pair $(i, a) \in \mathcal{X} \times \mathcal{A}$ is visited infinitely often. Furthermore, if ARTQI uses a GLIE policy δ_t, the agent will converge to δ^* in behavior.*

Proof See Appendix F.1. □

We conclude this section with the following remark. Reinforcement learning algorithms can be described according to many different criteria. Some common classifications include *on-line or off-line*, *model-based or model-free* (or, which is the same, direct or indirect), *synchronous or asynchronous*, *tabular or non-tabular*, *on-policy or off-policy* and other. The two algorithms described in this section (ARTVI and ARTQI) are:

- *on-line*, since learning takes place at run-time (*i.e.*, as the agent interacts with the environment);
- *model-based* or *indirect*, since the functions of interest (V^δ and Q^*) are not computed directly; instead, the algorithms build an explicit model of the MDP (namely of P and r) that is used to determine the desired functions;
- *asynchronous*, since not all components of V_t and Q_t are updated at each time step;
- *tabular*, since the algorithm is designed to handle tabular representations of V^δ and Q^*.

However, the two algorithms differ in the relation between the policy used to interact with the environment (the *learning policy*) and the value function learnt: ARTVI is an *on-policy* method, while ARTQI is an *off-policy* algorithm. This designation broadly means the following: in ARTVI the agent *is learning the value for the policy it is executing*; in ARTQI the agent *is learning the value of policy δ^* while executing a different policy*. In other words, in on-policy methods, the learning policy and the learnt policy coincide; this does not hold in off-policy methods.

3.4 Model-free learning

We now consider algorithms that directly estimate the functions V^δ and Q^* from the interaction with the environment, without recurring to any model (learnt or not). Recall that in the model-based methods in the previous section, the agent used the experience collected by interacting with its environment to estimate the model parameters $\hat{\mathsf{P}}$ and $\hat{r}$. These were in turn used to estimate either V^δ or Q^*.

In the model-free algorithms presented herein, the agent uses its experience to *directly estimate* V^δ and Q^*. It does so by considering stochastic variants of the

following, already familiar fixed-point iterations

$$V_{k+1}(i) = \sum_{j\in\mathcal{X}} \sum_{a\in\mathcal{A}} \delta(i,a)\mathsf{P}_a(i,j)\big[r(i,a,j) + \gamma V_k(j)\big] \tag{3.12}$$

and

$$Q_{k+1}(i,a) = \sum_{j\in\mathcal{X}} \mathsf{P}_a(i,j)\big[r(i,a,j) + \gamma \max_{b\in\mathcal{A}} Q_k(j,b)\big]. \tag{3.13}$$

The stochastic versions of (3.12) and (3.13) that we introduce next go under the common name of *temporal-differencing methods*, for reasons soon to become apparent. They have been exhaustively studied in the literature, some central references being [26, 27, 289]. In particular, the temporal differencing algorithm method used in the determination of Q^* is known as *Q-learning*. This algorithm was first introduced by Watkins in his PhD thesis [333]. From the first appearance of Q-learning in 1989, this algorithm has been one of the most studied reinforcement learning algorithms, leading to extensive literature and numerous variants (*e.g.*, [5, 34, 65, 68, 131, 235, 263, 280, 300, 332, 333]).

3.4.1 Determining V^δ

One can look at stochastic approximation methods as algorithmic solutions to address the problem of determining the zero of a real-valued function h when the function is not known but noise-corrupted samples of h are available at any desired point. The algorithm takes the basic form

$$\theta_{t+1} = \theta_t + \alpha_t H(\theta_t, X_t), \tag{3.14}$$

where θ_t is the current estimate of the zero, $\{\alpha_t\}$ is a sequence of positive step-sizes and $H(\theta_t, X_t)$ is a noisy sample of $h(\theta_t)$ that depends on the r.v.s X_t. In particular, considering $\{X_t\}$ to be some ergodic stochastic process,

$$h(\theta) = \mathbb{E}\left[H(\theta, X_t)\right],$$

where the expectation is taken with respect to the stationary distribution of the process $\{X_t\}$ (see Appendix D for a general overview of stochastic approximation algorithms).

Suppose now that we want to determine the fixed-point θ^* of an unknown function $\bar{h}$, *i.e.*, the point θ^* verifying

$$\bar{h}(\theta^*) = \theta^*$$

or, equivalently,

$$\bar{h}(\theta^*) - \theta^* = 0.$$

If we define the function $h(\theta) = \bar{h}(\theta) - \theta$, determining the fixed-point of $\bar{h}$ reduces to determining the zero of h. Applying (3.14), we get

$$\theta_{t+1} = \theta_t + \alpha_t H(\theta_t, X_t)$$

or, equivalently,

$$\theta_{t+1} = \theta_t + \alpha_t\big(\bar{H}(\theta_t, X_t) - \theta_t\big). \tag{3.15}$$

If θ is a vector in $\mathbb{R}^M$, the update in (3.15) can be performed component-wise as

$$\theta_{t+1}(i) = \theta_t(i) + \alpha_t\big(\bar{H}(\theta_t, X_t)_i - \theta_t(i)\big), \tag{3.16}$$

where we denoted by $\bar{H}(\theta, X)_i$ the i^{th} component of $H(\theta, X)$.

The function V^δ is the fixed point of operator $\mathbf{T}^\delta$, given by

$$(\mathbf{T}^\delta v)(i) = \sum_{j\in\mathcal{A}}\sum_{a\in\mathcal{A}} \delta(i,a)\mathsf{P}_a(i,j)\big(r(i,a,j) + \gamma v(j)\big).$$

It is possible to rewrite $\mathbf{T}^\delta$ as

$$(\mathbf{T}^\delta v)(i) = \mathbb{E}_\delta\left[r(i,a,j) + \gamma v(j)\right].$$

If we replace θ_t by V_t in (3.16) and $\bar{H}$ by a sampled version of $\mathbf{T}^\delta$, we obtain

$$V_{t+1}(i_t) = V_t(i_t) + \alpha_t(i_t)\big(r_t + \gamma V_t(i_{t+1}) - V_t(i_t)\big). \tag{3.17}$$

The quantities

$$\Delta_t = r_t + \gamma V_t(i_{t+1}) - V_t(i_t),$$

appearing in the right-hand side of (3.17), are known as *temporal differences* (TD) and measure a 1-step estimation error, *i.e.*, the difference between the current estimate (given by V_t) and a 1-step-ahead estimate (given by $r_t + \gamma V_t(i_{t+1})$). Learning methods that rely on this estimation error to perform updates are known as *temporal differencing* methods. In particular, (3.17) is known as TD(0).[6]

Similarly to the methods in Section 3.3, (3.17) updates the estimate V_t one single state at a time. However, if all states are visited infinitely often and the step-size sequence $\{\alpha_t\}$ verifies some convergence rate conditions, the sequence $\{V_t\}$ will converge to V^δ w.p.1. Notice furthermore that, in (3.17), we allowed the step-sizes to depend on the current state i_t, whose component $V_t(i_t)$ is to be updated. In fact, at each time step t, only the component $V_t(i_t)$ is updated *i.e.*, the updates of the several components of V_t are done asynchronously. Therefore, allowing the step-sizes α_t to depend on the state to be updated is a simple way of generalizing (3.16) to cope with the asynchronous update schedule.

3.4.2 Determining Q^*

It is straightforward to replicate the process leading to (3.17) to obtain an algorithm to compute Q^*. Q^* is the fixed point of the operator $\mathbf{H}$ defined in Section 3.2. This

[6] The designation TD(0) arises from the fact that this method relies only on the temporal difference Δ_t to perform the updates at each time-step t. It is possible, however, to use in each iteration the past temporal differences $\Delta_{t-1}, \Delta_{t-2}, \ldots$ weighted by a "temporal discount factor" λ, with $0 \leq \lambda \leq 1$. For a general λ the corresponding method is called TD(λ). TD(0) corresponds to the case $\lambda = 0$.

operator can be written, for each $(i,a) \in \mathcal{X} \times \mathcal{A}$, as

$$(\mathbf{H}q)(i,a) = \mathbb{E}_\delta \left[r(i,a,j) + \gamma \max_{b \in \mathcal{A}} q(j,b) \right],$$

which leads to the iteration

$$Q_{t+1}(i_t,a_t) = Q_t(i_t,a_t) + \alpha_t(i_t,a_t)\big(r_t + \gamma \max_{b \in \mathcal{A}} Q_t(i_{t+1},b) - Q_t(i_t,a_t)\big). \tag{3.18}$$

The algorithm in (3.18) is known as *Q-learning*. It is one of the most widely studied reinforcement learning methods and has led to a vast literature, as seen in the beginning of this section. Like TD(0), it is a temporal-differencing method, using the temporal differences

$$\Delta_t = r_t + \gamma \max_{b \in \mathcal{A}} Q_t(i_{t+1},b) - Q_t(i_t,a_t)$$

to perform the update on the estimate Q_t. It is also asynchronous, updating one component of Q_t at each iteration. As long as every component (each state-action pair $(i,a) \in \mathcal{X} \times \mathcal{A}$) is updated infinitely often, Q-learning will converge to Q^* w.p.1.

It should be clear that, while TD(0) is an on-policy method, Q-learning is an off-policy method. The learning policy used to sample the state-action-space only needs to ensure that all state-action pairs are visited infinitely often, but can otherwise be arbitrary: the agent will learn the function Q^* independently of the learning policy used.

There is an on-policy variant of Q-learning, introduced by Rummery and Niranjan [263] and further explored by Sutton [289], known as SARSA. SARSA stands for *state-action-reward-state-action*, since the method uses a sample 5-tuple

$$(i_t, a_t, r_t, i_{t+1}, a_{t+1})$$

to perform the update

$$Q_{t+1}(i_t,a_t) = Q_t(i_t,a_t) + \alpha_t(i_t,a_t)\big(r_t + \gamma Q_t(i_{t+1},a_{t+1}) - Q_t(i_t,a_t)\big). \tag{3.19}$$

at each time instant. Notice that in (3.19) the temporal-difference Δ_t is taken along the policy δ by using $r_t + \gamma Q_t(i_{t+1},a_{t+1})$ instead of $r_t + \gamma \max_{b \in \mathcal{A}} Q_t(i_{t+1},b)$ as the 1-step-ahead estimate. Clearly, SARSA will converge to the Q-values of the policy δ. However, Singh et al. [280] have shown that, by using a GLIE learning policy in SARSA, the algorithm converges to the optimal Q-values as long as every state-action pair is visited infinitely often. Although referring to [280] for a more detailed discussion on GLIE policies, a possible and simple way of ensuring a given policy to be GLIE is to set $\big|\delta_t(i,a) - \delta_g^{Q_t}(i)\big| \in \mathrm{O}(\alpha_t(i,a))$.[7]

[7]We adopted the symbol O in the standard asymptotic notation. Recall that for two positive functions f, g, we say that $f(t) \in \mathrm{O}(g(t))$ iff

$$\limsup_{t \to \infty} \left| \frac{f(t)}{g(t)} \right| < \infty.$$

To conclude this section, we provide the formal results establishing the convergence of the three model-free methods presented: TD(0), Q-learning and SARSA. These are standard convergence results from the RL literature.

3.4.3 Convergence of model-free learning

We now formalize the claims presented along this section on the convergence of TD(0), Q-learning and SARSA. The three results presented assess the convergence of the three described methods. The proofs can be found in Appendix F and follow the proofs in several well-known works from the literature [27, 131, 280].

Theorem 3.4.1. *Given a finite-state MDP $(\mathcal{X}, \mathcal{A}, \mathsf{P}, r, \gamma)$ and stationary policy δ, the sequence of estimates $\{V_t\}$ generated by TD(0) converges to V^δ w.p.1 for any initial estimate V_0, as long as*

$$\sum_{t \in \mathcal{T}} \alpha_t(i) = \infty; \qquad \sum_{t \in \mathcal{T}} \alpha_t^2(i) < \infty,$$

and $\alpha_t(i) = 0$ if $i \neq i_t$.

Proof See Appendix F.1. □

The convergence theorem for Q-learning is very similar.

Theorem 3.4.2. *Given a finite-state MDP $(\mathcal{X}, \mathcal{A}, \mathsf{P}, r, \gamma)$, the sequence of estimates $\{Q_t\}$ generated by Q-learning converges to Q^* w.p.1, as long as*

$$\sum_{t \in \mathcal{T}} \alpha_t(i, a) = \infty; \qquad \sum_{t \in \mathcal{T}} \alpha_t^2(i, a) < \infty,$$

and $\alpha_t(i, a) = 0$ if $(i, a) \neq (i_t, a_t)$. Furthermore, if Q-learning uses a learning policy δ_t with the GLIE property, the agent will converge in behavior to δ^* w.p.1.

Proof See Appendix F.1. □

The convergence result for SARSA requires that a GLIE policy be used for $\{Q_t\}$ to converge to Q^*. This comes as no surprise, since SARSA is an on-policy method, as seen in the previous subsection.

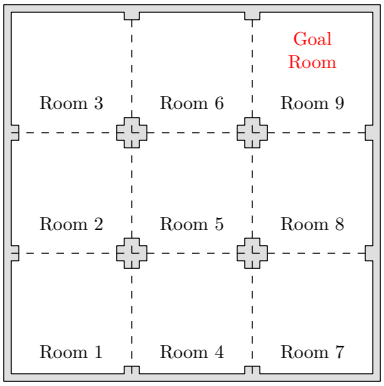


Figure 3.1: Example of an indoor environment.

Theorem 3.4.3. *Consider a finite-state MDP $(\mathcal{X}, \mathcal{A}, \mathsf{P}, r, \gamma)$ and let δ_t be a fixed learning policy with the GLIE property w.r.t. the current estimate Q_t. Then the sequence of estimates $\{Q_t\}$ generated by SARSA converges to Q^* w.p.1 for any initial estimate Q_0, as long as*

$$\sum_{t \in \mathcal{T}} \alpha_t(i,a) = \infty; \qquad \sum_{t \in \mathcal{T}} \alpha_t^2(i,a) < \infty,$$

and $\alpha_t(i,a) = 0$ if $(i,a) \neq (i_t, a_t)$. Furthermore, the agent will converge in behavior to δ^* w.p.1.

Proof See Appendix F.1. □

3.5 An illustrative example

Consider the already familiar indoor environment depicted in Figure 3.1. The navigation problem can be described by an MDP $(\mathcal{X}, \mathcal{A}, \mathsf{P}, r, \gamma)$, where

- $\mathcal{X} = \{1, \ldots, 9\}$;
- $\mathcal{A} = \{N, S, E, W\}$;
- Each action $a \in \mathcal{A}$ moves the robot in the corresponding direction. With a probability of 0.8 this movement ends up in the adjacent state (in that direction); with a probability of 0.1 the movement ends up two states ahead (once again, in the corresponding direction); finally, with a 0.1 probability, it remains in the same state;
- The reward function r assigns a reward of $+10$ for every transition triplet (i, a, j) such that $j = 9$ and 0 otherwise;
- We consider $\gamma = 0.95$.

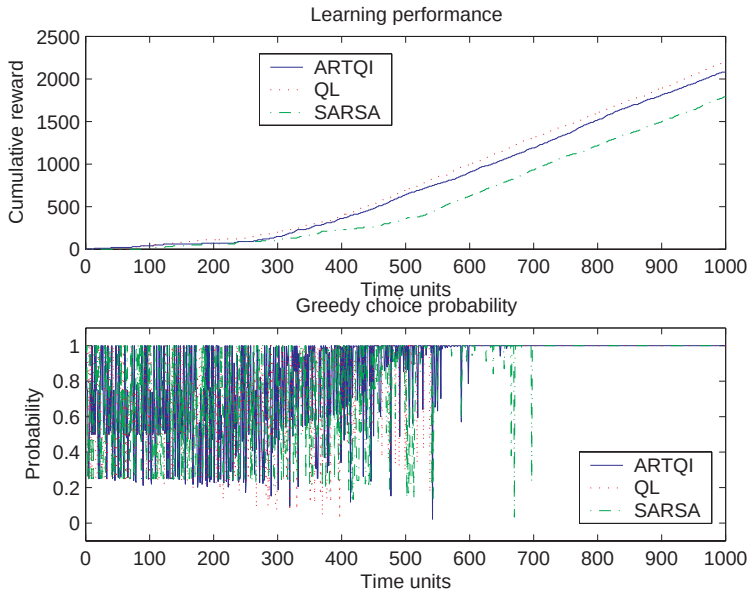


Figure 3.2: Cumulative reward and greedy choice probability during the 1,000-time-units learning period.

We now assume that, whenever the robot reaches the goal state, its position is randomly reset to any of the other 8 states, independently of the robot's action.

We applied ARTQI, Q-learning and SARSA to this MDP. The robot was allowed to explore the environment/learn during 1,000 time units and the obtained policy was then evaluated for 100 time units. During learning we used Boltzmann exploration in all three experiments.

Figure 3.2 represents the total undiscounted reward obtained by the agent during learning, as well as the evolution of the probability of choosing the greedy action. The solid blue line represents the ARTQI learner, the dashed red line represents the Q-learner and the dash-dotted green line represents the SARSA learner. Notice that all three learning algorithms present a similar learning behavior. SARSA is slightly slower, due to its on-policy updates, but the difference is not significant. Notice that all methods eventually become greedy, *i.e.*, the probability of choosing the greedy action goes to 1.

We also tested each of the learnt policies in the environment. We ran each of the three learnt policies for 100 time units and determined the total discounted reward obtained in each case. Table 3.1 represents the results obtained. We run 2,000 independent Monte-Carlo trials and present the average and standard deviation obtained using each of the methods. Notice that the three methods present a similar performance, as expected, since all three methods converge to the optimal Q-function.

For the sake of comparison, we also present the value obtained with the optimal policy, in the line labeled "VI". Clearly, the performance of all three methods levels that of VI, which means that all methods learnt the optimal policy.

We conclude this section by presenting in Figure 3.3 the policy learnt by each of the three methods. Notice that each of the three policy graphs can be obtained

Table 3.1: Comparative results for ARTQI, Q-learning and SARSA after the learning period is complete. We present the average total discounted reward and standard deviation obtained over $2,000$ independent Monte-Carlo trials.

Method	Total Disc. Reward
ARTQI	50.336 ± 5.163
Q-learning	50.373 ± 5.051
SARSA	50.382 ± 5.169
VI	50.515 ± 5.050

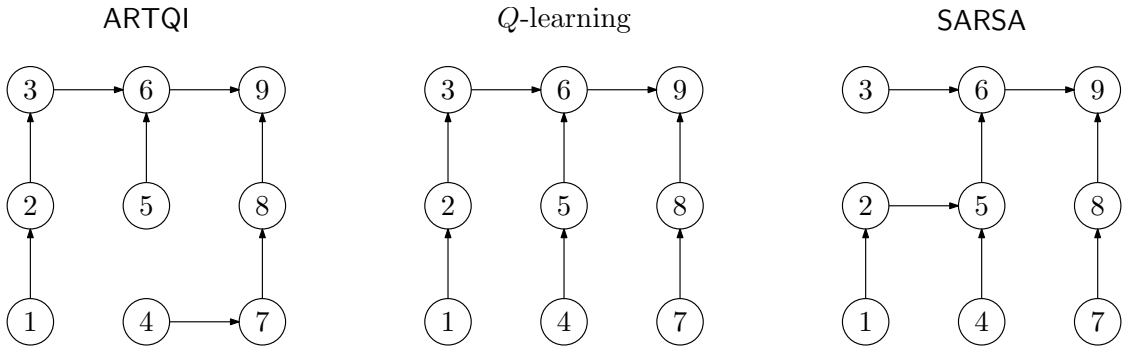


Figure 3.3: Policy graphs for the 3 learnt policies.

from the one in Figure 2.12, as expected.

3.6 Concluding remarks

To conclude this chapter, we summarize the main ideas presented so far and provide some hints on the developments yet to come.

3.6.1 Summary

In Chapter 2, we described MDPs as a framework suitable to address single-agent topological navigation problems. MDPs model situations where an agent is required to learn to perform a task without being shown how to perform it. Instead, it receives a reinforcement signal for each action it executes. This setting was applied to a topological navigation problem with success, as illustrated in a simple example.

In the present chapter, we presented several algorithms to either evaluate a given policy or learn the optimal policy for a given MDP. In these methods (and unlike the VI method described in Chapter 2), the agent has little *a priori* knowledge of the environment: it does not know the transition probabilities or rewards/penalties for each action.

We described two classes of methods. We described *model-based methods*, that estimate the transition probabilities and rewards for each action and then use these

estimates to either evaluate a given policy or to determine the optimal policy. Estimation of the parameters P and r of the MDP model relies on direct interaction with the environment.

The second class of methods is *model-free*, in that the agent does not attempt to learn the MDP parameters. Instead, the agent uses its interaction with the environment to directly evaluate a given policy or estimate the optimal Q-function. The three model-free methods presented make use of *temporal differences* to update the estimates of the function of interest.

We concluded the chapter by illustrating the use of the three methods in a simple topological navigation example.

3.6.2 Discussion

In this chapter, we presented several reinforcement learning algorithms that can be used either to evaluate a policy δ or to determine the optimal Q-values. Under suitable conditions, all the algorithms converge asymptotically to the desired value/Q-function. Furthermore, several such algorithms also ensure convergence in behavior to the optimal policy, as long as the learning policy has the GLIE property (see definition on page 45). In all this presentation, we postponed two important discussions to these concluding remarks. We address these two topics next.

Exploration vs. exploitation

In the methods described in this chapter, the agent successively refines its estimate of the function Q. This is achieved through *sampling*: as the agent visits more and more state-action pairs, the more accurately it is able to estimate the corresponding Q-values. On the other hand, while striving to abundantly visit each state-action pair, the agent is often unable to act optimally, *i.e.*, *exploit* the knowledge it has already acquired.

This balance between *exploration*—choosing non-optimal actions in order to sufficiently visit every state-action pair—and *exploitation*—acting greedily with respect to the acquired knowledge—has been an important problem discussed in the reinforcement learning literature. The exporation/exploitation tradeoff plays a fundamental role in guaranteeing convergence of many reinforcement learning methods and greatly influences the speed of such convergence.

Figure 3.4 illustrates the influence of different learning policies in the per-step expected reward. We applied Q-learning to the MDP in the example of Section 3.5.[8] For each of three learning policies the agent was now allowed to learn for 10^4 time steps. We determined the per-step reward averaged from $2,000$ independent Monte-Carlo trials. We depict the results obtained with a greedy policy, an ε-greedy policy with $\varepsilon = 0.1$ and Boltzmann exploration with a temperature parameter decaying at a rate $\mathrm{O}(1/n)$.

The greedy policy improves faster in the beginning, since it always uses all knowledge collected so far. However, due to insufficient exploration, it fails to reach the optimal performance—its plot eventually settles in a value smaller than that of the

[8] We report the results for Q-learning, but the conclusions hold in general.

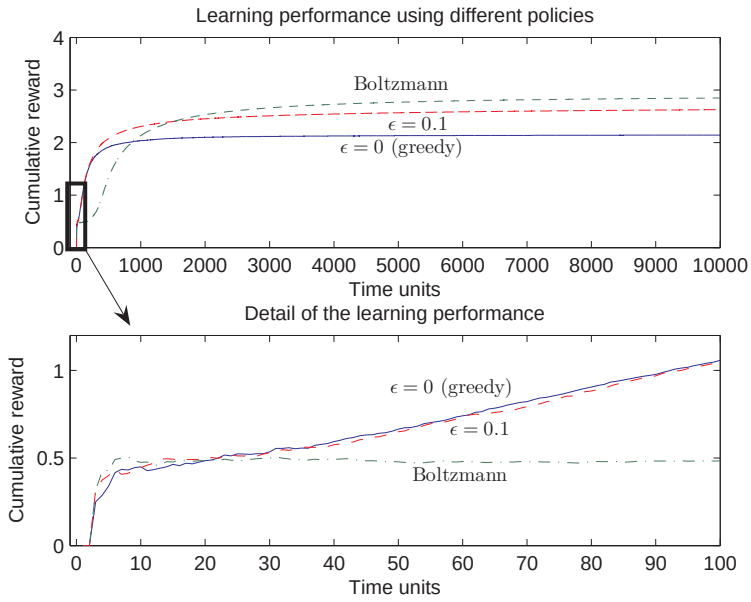


Figure 3.4: Average per-step reward using different learning policies.

other two. The ε-greedy policy chooses a random action with probability ε and behaves greedily otherwise. It exhibits a similar performance to that of the greedy policy at the beginning, but it quickly surpasses the greedy policy. However, as it always maintains a positive exploration probability, it never reaches the optimal performance: with $\varepsilon = 0.1$, the ε-greedy policy will choose a sub-optimal action one out of ten times. Finally, Boltzmann exploration is the slowest at the beginning but, due to its decaying exploration rate, it eventually outperforms the other two policies.

In order for a policy δ_t to have the GLIE property, it is important to ensure that not only $\delta_t \to \delta_g$, with δ_g the greedy policy, but also that sufficient visits to all state-action pairs are enforced. This can be ensured by properly adjusting the rate of convergence of δ_t to δ_g.[9]

The compromise between exploration and exploitation is an important topic of research in the reinforcement learning community. The need for exploration in many RL methods implies that convergence to optimality is only asymptotic. On the other hand, in non-stationary environments, persistent exploration policies can be used to track slow changes in the environment.

Several researchers have addressed the problem of efficient exploration, attempting to provide strategies that can improve the convergence in behavior of reinforcement learning algorithms. Thrun [312] surveys several possible exploration techniques, distinguishing between *directed methods* and *undirected methods*. Undirected methods use randomized action selection to explore alternative policies; directed methods use exploration information (such as trajectory statistics) to guide the exploration of alternative policies. Many other reinforcement learning works

[9] As referred in Section 3.4, one possibility consists in setting δ_t so that $\delta_t - \delta_g \in O(\alpha_t)$. We refer to [280] for details on adjusting GLIE policies.

address the problem of exploration:

- Wiering and Schmidhuber [338] address the exploration/exploitation tradeoff in model-based learning: a modified version of Moore and Atkeson's prioritized sweeping algorithm [204] is described, by introducing a new priority schedule. This algorithm is then combined with different exploration strategies such as Kaelbling's interval estimation [133].

- Kearns and Singh [138] propose the E^3 "algorithm", a model-based near-optimal reinforcement learning with ε-optimal convergence guarantees and polynomial convergence time. This work establishes the existence of an algorithm that is able to determine an ε-optimal policy with probability $1-\beta$ in a time that is polynomial in ε, $1/\beta$, $1/(1-\gamma)$, the size of the MDP and the maximal reward $\mathcal{R}$. The paper then describes E^3 (*Explicit explore or exploit*) algorithm verifying the performance bounds described.

- Several other approaches (generally referred under the name of Bayesian RL) adopt a Bayesian approach to the problem of balancing exploration and exploitation. *Bayesian Q-learning* [65] addresses the exploration/exploitation tradeoff in Q-learning. In the referred paper, the authors consider a prior on the distribution of the Q-values that is updated as the agent gathers new experience. Different action-selection rules are proposed and compared. Bayesian RL is further explored by Duff [74]. In this work, the author formulates the general Bayesian RL approach by using an extended MDP model (the *Bayes-adaptive MDP*—BAMDP) and establishes a close relation between BAMDPs and *partially observable MDPs* (POMDPs).[10]To efficiently tackle BAMDPs, the author proposes the use of Monte-Carlo methods and function approximation. Finally, Poupart et al. [242] further extend the work in [74] by describing several important facts arising from the adopted formulation. These facts lead to an efficient method to approximate the policy that optimally settles the exploration/exploitation trade-off.

Further references to the exploration/exploitation tradeoff can be found in the cited papers.

We return to the exploration/exploitation tradeoff in Chapter 10.

Model-free vs. model-based approaches

Another important issue to discuss is related with the question of whether model-free methods are preferable or not to model-based methods. This question has raised significant debate in the reinforcement learning community, with no apparent conclusions.

Proponents of model-free methods often argue that more experience is required to learn a good model than to learn a good policy; proponents of model-based methods, on the other hand, argue that the learnt model can be used to perform unlimited

[10]A POMDP is an extended MDP model that encompasses uncertain perception of the state of the environment. This class of problems is dealt with in greater detail in the next chapter.

Table 3.2: Advantages of model-free vs. model-based methods.

	Advantages
Model-free methods	• Less memory • Computationally simpler
Model-based methods	• Faster • Better use of experience

off-line updates of the estimate of Q^*, speeding up the convergence of such methods. We summarize in Table 3.2 some of the argued advantages and disadvantages of the two classes of methods.

In [137] these arguments are partially softened: Kearns and Singh establish that the error bounds and memory requirements for both Q-learning and model-based learning are basically similar. In other words, both methods require similar amount of experience to provide a ε-optimal policy and the memory requirements for both methods are also similar.[11]

Some methods try to combine the advantages of both approaches. In [293, 294], Sutton introduces DYNA as an architecture that combines learning, planning and reacting: in DYNA, the agent uses experience to improve the Q-estimates and to build a model of the environment. This model is then used to perform further updates to the Q-estimates to speed up the convergence. This framework was further explored in the DYNA-Q architecture and the prioritized sweeping algorithm.

* * *

In the presentation so far, we discussed several known methods from the literature that learn the optimal control sequence in a given environment. Such a control sequence is generated by a policy that can in turn be computed from a set of real parameters, the Q-values.

In everything presented so far, the state of the system has been considered *finite* and *fully observable*. The finiteness of the state-space can be justified by our consideration of a topological representation of the environment. Since we are focusing on topological navigation problems, topological maps provide a natural "decomposition" of the space into a finite set of nodes—the states.

On the other hand, we have implicitly assumed that the agent can access, know or perceive the true state of the system and this assumption is seldom verified in practice. Most agents and, in particular, mobile robots make use of sensorial information to *estimate* the state of the environment. Therefore, in most practical situations, the agent can only *infer* the state of the system/environment from its

[11] This, in particular, implies that model-based learning can achieve near-optimal performance with a sparse model.

observations. When this occurs, we refer to the state as having *partial observability*. In the presence of partial observability, the agent must choose its actions based not on the actual state of the process, but on the *belief of the agent about the current state of the process*.

In the next chapter we formalize the concept of *belief* as a probability vector that indicates the probability of the agent being in each particular state at each time instant. These real-valued vectors will provide the motivation to extend the methods from this chapter to problems with infinite state-spaces, as these extended methods will provide us with the tools to deal with partial observability.

Chapter 4

Generalized Reinforcement Learning

Having introduced in the previous chapters the fundamental concepts on reinforcement learning, we are now in position to present the first set of results contributed in the thesis. In this chapter, we propose new algorithms to tackle MDPs with infinite state-spaces or with partial observability.

We survey several reinforcement learning algorithms from the literature designed to address MDPs with infinite state-spaces. We analyze the convergence

properties of these methods and point out the main assumptions required to establish such convergence. We then propose two new learning algorithms that use linear function approximation. The interest of these algorithms is their extended applicability, since they include many of the existing methods as particular cases. We establish the convergence of these algorithms w.p.1. This is one of the main contributions of this chapter.

We then proceed by addressing problems with partial observability. We show that, under certain conditions, such problems can be reduced to MDPs with infinite state-spaces. We further provide a series of new results that guarantee the applicability of the methods previously developed to this class of partially observable Markov decision processes.

4.1 Learning and function approximation

The reinforcement learning algorithms described in the previous chapters require an explicit representation of the state-space. This can be a major drawback, since there are many situations of interest where the state-space is large or even infinite and, hence, unsuitable for explicit representation.

For example, consider a mobile robot moving in an environment described by a topological map and suppose that, unlike the approach considered so far, the robot is not able to determine with complete certainty its current state. Instead, it must *infer* its position in the topological map from inaccurate sensor readings. In this situation, the robot must act upon its *belief* on its current state, which translates into a probability distribution over all possible states in the topological map. The space of all possible probability distributions is a simplex of dimension $|\mathcal{X}|$, and is therefore a continuous space.[1]

Decision problems where the agent is required to learn how to act in an infinite number of situations (like MDPs with infinite state-spaces) usually require the decision-maker to be able to *generalize* its action-selection rule to situations never encountered before. Learning mechanisms that exhibit such generalization capabilities include artificial neural-networks and other general function approximation strategies. These approximations allow for compact approximate representations of the state-space or the target function. The corresponding learning methods focus on determining a low dimensional parameter that, when combined with the chosen function approximation, provides the best estimate to the desired function.

There are numerous works in the topic of generalization. In many such works, a suitable approximation architecture is proposed and then applied with one's favorite learning method [44, 290]. Encouraging results were reported, perhaps the most spectacular of which by Tesauro's Gammon player [307, 308]. In his work, Tesauro combines temporal-difference learning with neural networks, and the im-

[1] At this moment, we do not pursue the argument that the beliefs on a Markov chain are Markovian themselves, but mention that this will be formally established later in this chapter.

pressive results of his learning agent have established the applicability of function approximation in reinforcement learning problems.

Several other works provided formal analysis of convergence when RL algorithms are combined with function approximation. We refer the early works by Baird [10, 11, 12], Boyan [45, 46], Gordon [101, 103], Tsitsiklis and Van Roy [320, 321], Szepesvári [301, 302] and several others [8, 15, 28, 207, 225, 238, 244, 296].

* * *

From the first appearance of temporal-difference methods in 1988 [289], this class of algorithms has been extensively studied, and has lead to a wide literature and numerous variations. In its essence, temporal-difference learning methods are stochastic approximation algorithms, and most of the convergence proofs available in the literature make use of more or less sophisticated results from the theory of stochastic approximation algorithms.

As seen in Chapter 3 and further detailed in Appendix D, stochastic approximation algorithms can be used to determine the fixed point θ^* of an unknown map h by using the iteration

$$\theta_{t+1} = \theta_t + \alpha_t\big[H(\theta_t, X_t) - \theta_t\big], \tag{4.1}$$

where H is a "perturbed" version of h and $\{X_t\}$ is a random noise sequence. Notice that the form of (4.1) is very similar to the update rule for TD(0) in (3.17), Q-learning in (3.18) and SARSA in (3.19). In the referred methods, determining $V^\delta(i)$ or $Q^*(i,a)$ for all $i \in \mathcal{X}$ or all pairs $(i,a) \in \mathcal{X} \times \mathcal{A}$ in the case of finite state and action-spaces reduces to the determination of finite dimensional "vectors" in $\mathbb{R}^{|\mathcal{X}|}$ or $\mathbb{R}^{|\mathcal{X}|\times|\mathcal{A}|}$. Each component of such vector is the value of $V^\delta(i)$ for a state i or the optimal Q-value, $Q^*(i,a)$, for a state-action pair (i,a).

However, when addressing problems in which $\mathcal{X}$ is infinite, these algorithms can not be used as originally defined. This is particularly clear by observing the update equations (3.17), (3.18) and (3.19). In these equations, the values of V_t or Q_t are updated for each individual state i or pair (i,a). If $\mathcal{X}$ is infinite, there are infinitely many such states/pairs, and infinite memory is required to store them.

In this chapter, we describe several approaches from the literature specifically designed to handle decision problems with infinite state-spaces. In particular, we describe model-based and model-free algorithms to approximate both V^δ and Q^* that make use of different approximation techniques to deal with the infinite state-space.

Our contributions in this chapter include convergence results for Q-learning and SARSA with function approximation, that we dub as *approximate Q-learning* and *approximate SARSA*. The results presented herein extend those from several other works in the literature, where Q-learning has been shown to converge using several particular approximation strategies (such as "soft" state-space discretization [279] and interpolation [302]). In fact, many such methods can be cast as particular cases of the new method proposed here. Our results also extend those on [238] on the convergence of SARSA by providing an interpretation for the obtained limit point when the learning policy has the GLIE property. Another contribution of this chapter is the identification of several conditions that establish the applicability of approximate Q-learning and approximate SARSA in partially observable scenarios.

4.2 Infinite state-space Markov processes

When considering MDPs with infinite state-spaces, some minor modifications in the notation are mandatory. We introduce such modifications next.

Let $\mathcal{X}$ be a compact subspace of $\mathbb{R}^p$ and $\{X_t\}$ a $\mathcal{X}$-valued controlled Markov chain with a finite action-space $\mathcal{A}$. The transition probabilities for the chain are given by a probability kernel

$$\mathbb{P}\left[X_{t+1} \in U \mid X_t = x, A_t = a\right] = \mathsf{P}_a(x, U),$$

where $U \in \mathcal{B}(\mathcal{X})$. As in finite MDPs, the expected total discounted reward is given by

$$V(\{A_t\}, x) = \mathbb{E}\left[\sum_{k=0}^{\infty} \gamma^k R(X_k, A_k) \mid X_0 = x\right],$$

and r is now a measurable function such that

$$\mathbb{E}\left[R(x, a)\right] = \int_{\mathcal{X}} r(x, a, y)\mathsf{P}_a(x, dy).$$

The purpose of the agent is once again to determine the control sequence $\{A_t\}$ that maximizes $V(\{A_t\}, x)$. The optimal value function V^* now verifies the modified form of the Bellman optimality equation

$$V^*(x) = \max_{a \in \mathcal{A}} \int_{\mathcal{X}} \big[r(x, a, y) + \gamma V^*(y)\big]\mathsf{P}_a(x, dy),$$

and the optimal Q-values, $Q^*(x, a)$, are defined for each state-action pair $(x, a) \in \mathcal{X} \times \mathcal{A}$ as

$$Q^*(x, a) = \int_{\mathcal{X}} \big[r(x, a, y) + \gamma V^*(y)\big]\mathsf{P}_a(x, dy).$$

As in the finite state-space case, the optimal policy is the mapping $\delta^* : \mathcal{X} \longrightarrow \mathcal{A}$ given by

$$\delta^*(x) = \arg\max_{a \in \mathcal{A}} Q^*(x, a), \qquad \text{for all } x \in \mathcal{X}.$$

Notice that in this infinite setting the definitions of *stochastic policy*, *deterministic policy* and *stationary policy* carry on from those in a finite setting with no modification. The operators $\mathbf{T}^\delta$ and $\mathbf{H}$ introduced in Chapter 3 now take the form

$$(\mathbf{T}^\delta v)(x) = \int_{\mathcal{X}} \sum_{a \in \mathcal{A}} \big[r(x, a, y) + \gamma v(y)\big]\delta(x, a)\mathsf{P}_a(x, dy), \tag{4.2}$$

$$(\mathbf{H}q)(x, a) = \int_{\mathcal{X}} \big[r(x, a, y) + \gamma \max_{b \in \mathcal{A}} q(y, b)\big]\mathsf{P}_a(x, dy). \tag{4.3}$$

Because of the fact that $\mathcal{X}$ is now an infinite set, it is no longer possible to straightforwardly apply any of the update rules in (3.17), (3.18) and (3.19), since these stochastic approximation algorithms update the corresponding functions for each individual point in their domain (and there are infinitely many such points

now).

We will now review several related works from the literature. We will explore some of these works in greater detail further ahead in this chapter to facilitate the comparison with our own methods and to provide a global perspective on the fundamental works on RL in infinite settings.

4.3 Related work

In this section we survey several works from the literature that are related in some way to the new algorithms introduced in this chapter.

The early works by Gordon [103] and Tsitsiklis and Van Roy [320] provide convergence analysis for several RL methods using function approximation. The two referred papers portray similar results, although with a slightly different setting and focus on variations of dynamic programming using function approximation. There is also a brief discussion on how stochastic variations of these algorithms (closer in spirit to the Q-learning algorithm) can be used. These stochastic variations are essentially equivalent to the Q-learning algorithm with soft-state aggregation portrayed in [279], as pointed out in [27].

Soft-state aggregation is extensively studied in [279]. In this work, the authors propose the use of a "soft"-partition of the state-space: the state-space is split into "soft" regions (each state x belongs to region i with a probability $p_i(x)$) and an "average" Q-value $Q(i,a)$ is defined for each region-action pair. Each of these regions is then treated as a "hyper-state" and the method uses standard Q-learning updates to determine the average Q-values for each region. The function Q^* is then approximated for a state-action pair (x,a) as $Q^*(x,a) \approx \sum_i p_i(x)Q(i,a)$.

In a different work, Tsitsiklis and Van Roy [321] provide a detailed analysis of the approximate temporal difference algorithm (approximate TD) for policy evaluation. The authors provide comprehensive results regarding the convergence and/or divergence of such method when a linear function approximation is used. Given a stationary policy δ whose value function V^δ is to be estimated, a parameterized linear family $\mathcal{V}$ of functions is used to approximate V^δ. The authors establish that the sequence of estimates obtained by approximate TD closely follow those of an associated globally asymptotically stable ODE and thus converge to the unique equilibrium point of such ODE. They also provide an interpretation of the approximation obtained as the fixed point of a composite operator $\mathcal{P}_\mathcal{V}\mathbf{T}^\delta$, where $\mathcal{P}_\mathcal{V}$ is an orthogonal projection and $\mathbf{T}^\delta$ is a TD operator. The authors also provide error bounds on the obtained approximation (see Subsection 4.5.1 and [326] for further details).

Least-squares temporal difference learning methods (LSTD) depart from the work by Tsitsiklis and Van Roy and propose an alternative method to compute the same approximation [45, 46]. As established by Tsitsiklis and Van Roy [321], approximate TD-learning converges in the limit to a parameter vector θ^* that verifies a linear system of the type

$$\mathbf{A}\theta^* + \mathbf{b} = 0,$$

where the matrices $\mathbf{A}$ and $\mathbf{b}$ arise from a "stationary version" of the algorithm. LSTD estimates directly the matrices $\mathbf{A}$ and $\mathbf{b}$ from the sample trajectories of the

underlying Markov chain, thus converging to the same limiting approximation and with some extra argued advantages [46]. Although not exactly in the line of the algorithms described herein, it is important to mention LSTD as closely related with approximate TD.

Two other works are closely related with those above and are worth mentioning. In [244], the authors provide an off-policy algorithm for policy evaluation using linear function approximation. Unlike approximate TD, the algorithm in [244] uses episodic updates. This means that the process is allowed to run for a fixed number of time-steps (episode) before an update is performed. The process is then reset and another episode starts. The authors establish convergence w.p.1 of their method and provide similar error bounds to those obtained by Tsitsiklis and Van Roy [321]. In another work, Perkins and Precup [238] establish the convergence of SARSA with linear function approximation if the learning policy verifies some regularity conditions. In this chapter we provide a more complete version of the result in [238] by establishing convergence of SARSA with linear function approximation while providing an interpretation for the obtained approximation if a GLIE learning policy is used.

Szepesvári and Smart [302] address the problem of policy optimization in infinite settings. In this paper, the authors propose a version of Q-learning that approximates the optimal Q-values at a given set of sample points and then uses convex interpolation to estimate the optimal Q-function at any query point. This method, dubbed *interpolation-based Q-learning* (IBQL), uses a *spreading function* similar to the one used in multi-state Q-learning [252, 300]. As in [321], the authors establish convergence w.p.1 of the algorithm and provide an interpretation of the limit point as the fixed-point of a composite operator $\mathcal{P}\hat{\mathbf{H}}$, where $\mathcal{P}$ is a projection-like operator and $\hat{\mathbf{H}}$ can be interpreted as a modified TD-learning operator. We remark that IBQL includes Q-learning with soft-state aggregation as a particular case, by considering an adequate convex interpolator.

Atkeson et al. [8] describe the application of local weighted regression methods to estimate the model (namely the kernel P and the reward function r) in a continuous-state reinforcement learning problem and this fundamental idea is generalized in [225]. In the latter work, Ormoneit and Śaunak Sen [225] establish convergence in probability and derive the limiting distribution of the obtained approximation. They also address the bias-variance tradeoff in reinforcement learning problems.

All the described methods have been shown to converge in some sense. However, methods that allow the use of general function approximations (such as the approximate TD) only allow policy evaluation, while methods that allow policy optimization are only shown to converge with very particular architectures of function approximation. The results produced in this chapter extend the existing methods in that they allow policy optimization using more general function approximation architectures.

* * *

In Chapter 3 we described model-based and model-free methods, both for policy evaluation and policy optimization. For consistency, we will follow a similar course of

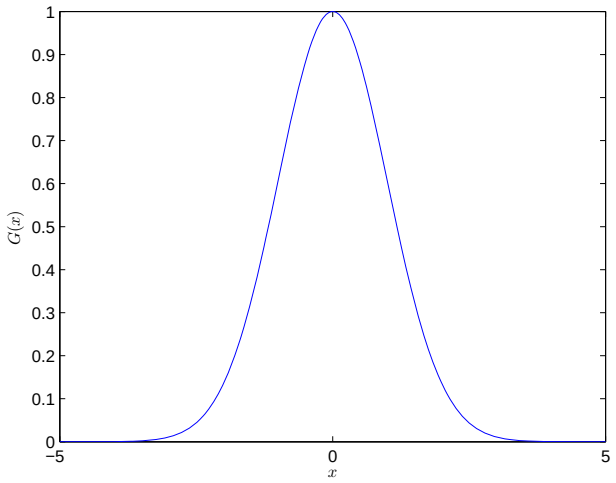


Figure 4.1: Gaussian kernel.

action in this chapter. As such, we review in greater detail the model-based method in [225] that can be used both for policy evaluation or optimization. We also review the model-free method of Tsitsiklis and Van Roy [321] for policy evaluation and finally proceed by introducing our two model-free methods for policy optimization, *approximate Q-learning* and *approximate SARSA*.

4.4 Model-based learning

In this section, we review in greater detail the *kernel-based reinforcement learning* algorithm in [225]. Kernel-based learning is a model-based reinforcement learning algorithm that, in its essence, approximates the dynamic programming operator $\mathbf{H}$ introduced in (4.3) using kernel-based methods.

The authors start by considering a *mother kernel* G that is univariate and non-negative (*e.g.*, the Gaussian kernel—see Figure 4.1). Then, given a sample history $\mathcal{H} = \{s_1, \ldots, s_m\}$ of m transition tuples $s_i = (x_i, a_i, r_i, y_i)$, where y_i is the state succeeding x_i when action a_i is taken, the operator $\mathbf{H}$ is approximated by

$$\hat{\mathbf{H}}q(x,a) = \sum_{s_i \in \mathcal{H}} \mathbf{k}(x_i, x)\mathbb{I}_a(a_i)\big[r_i + \gamma \max_{b \in \mathcal{A}} q(y_i, b)\big],$$

where $\mathbf{k}$ is a history-dependent kernel defined as

$$\mathbf{k}(x_i, x) = \frac{\mathsf{G}\left(\frac{\|x_i - x\|}{b}\right)}{\sum_{s_j \in \mathcal{H}} \mathsf{G}\left(\frac{\|x_j - x\|}{b}\right)},$$

b is a *bandwidth parameter* and $\mathbb{I}_a$ is the indicator function for the set $\{a\}$.

The authors also provide convergence guarantees for this method: under suitable conditions on the bandwidth parameter b, $\left\|\hat{\mathbf{H}}q - \mathbf{H}q\right\|_\infty$ is shown to converge to 0 in

probability as the number m of samples in the history goes to infinity. Furthermore, the authors also derive the limiting distribution of the obtained approximation $\hat{Q}^*$.

The same exact procedure can be used to determine V^δ, by sampling the history according to the policy δ and eliminating the actions from the equations. The kernels $\mathbf{k}(x_i, x)$ are then be used to estimate the operator $\mathbf{T}^\delta$ instead of $\mathbf{H}$.

The algorithm and the formal convergence result and proofs can be found in [225].

4.5 Model-free learning

We now consider algorithms that directly approximate the functions V^δ and Q^* by interacting with the environment, using a similar stochastic approximation approach as described in Section 3.4. The algorithms presented in this section use the agent's experience to estimate the fixed points of Bellman-like operators related to $\mathbf{T}^\delta$ and $\mathbf{H}$ (defined in (4.2) and (4.3)) using stochastic approximation.

4.5.1 Determining V^δ

Let $(\mathcal{X}, \mathcal{A}, \mathsf{P}, r, \gamma)$ be an MDP with infinite state-space $\mathcal{X}$, admittedly a compact subset of $\mathbb{R}^p$.[2] Given a policy δ, the corresponding value-function V^δ is to be determined. V^δ verifies the following recursive relation

$$V^\delta(x) = \int_{\mathcal{X}} \sum_{a \in \mathcal{A}} \big[r(x, a, y) + \gamma V^\delta(y) \big] \delta(x, a) \mathsf{P}_a(x, dy). \tag{4.4}$$

It is straightforward from (4.4) that the desired value function V^δ is the fixed point of $\mathbf{T}^\delta$ as defined in (4.2).

Let $\mathcal{V} = \{v_\theta\}$ be a family of functions, $v_\theta : \mathcal{X} \to \mathbb{R}$, parameterized by a finite dimensional parameter $\theta \in \mathbb{R}^M$. In particular, suppose that the family $\mathcal{V}$ is a linear space in that, if $v_1, v_2 \in \mathcal{V}$, then so does $\alpha v_1 + v_2$ for any $\alpha \in \mathbb{R}$. As such, $\mathcal{V}$ can be represented as the linear span of a set of M linearly independent functions $\xi_i : \mathcal{X} \longrightarrow \mathbb{R}$, and each $v \in \mathcal{V}$ can be written as

$$v(x) = \sum_{i=1}^{M} \xi_i(x) \theta(i) = \xi^\top(x) \theta,$$

for all $x \in \mathcal{X}$, where $\theta(i)$ is the i^{th} component of the vector $\theta \in \mathbb{R}^M$ and $\xi(x)$ is a vector in $\mathbb{R}^M$ with i^{th} component given by $\xi_i(x)$. $\mathcal{V}$ is a hyperplane of dimension M in the space of all measurable functions $v : \mathcal{X} \longrightarrow \mathbb{R}$.

Given a policy δ and the corresponding sample trajectories $\{x_t\}$, $\{a_t\}$ and $\{r_t\}$, define the sequence $\{\theta_t\}$ recursively as

$$\theta_{t+1} = \theta_t + \alpha_t \xi(x_t) \Delta_t \tag{4.5}$$

[2] In the original work, Tsitsiklis and Van Roy [321] consider $\mathcal{X}$ to be countably infinite, but this assumption was later extended to subsets of $\mathbb{R}^p$ [326]. In our presentation we admit compacity of $\mathcal{X}$ for simplicity.

where Δ_t is the temporal difference

$$\Delta_t = r_t + \gamma V_t(x_{t+1}) - V_t(x_t),$$

and $V_t(x)$ is given by

$$V_t(x) = \xi^\top(x)\theta_t.$$

Under some regularity assumptions on the MDP $(\mathcal{X}, \mathcal{A}, \mathsf{P}, r, \gamma)$ (to be specified below), the sequence $\{\theta_t\}$ generated by (4.5) converges w.p.1 to the vector θ^* such that

$$v_{\theta^*}(x) = (\mathcal{P}_\mathcal{V} \mathbf{T}^\delta v_{\theta^*})(x), \tag{4.6}$$

where $\mathcal{P}_\mathcal{V} v$ represents the orthogonal projection of v into the linear space $\mathcal{V}$. In other words, $v_{\theta^*}(x)$ is the fixed point of the combined operator obtained from the orthogonal projection $\mathcal{P}_\mathcal{V}$ and the TD operator $\mathbf{T}^\delta$.

In order to ensure that the projection operator $\mathcal{P}_\mathcal{V}$ is well defined, we consider the space of all square-measurable functions $v : \mathcal{X} \longrightarrow \mathbb{R}$ endowed with the inner product

$$\langle v_1, v_2 \rangle = \int_\mathcal{X} v_1(x) v_2(x) d\mu(x), \tag{4.7}$$

where μ is the invariant probability measure associated with the Markov chain $(\mathcal{X}, \mathsf{P}_\delta)$. Since the operator $\mathbf{T}^\delta$ is a contraction w.r.t. the norm induced by the inner-product in (4.7), the ODE associated with (4.5) has in θ^* a globally asymptotically stable equilibrium point. Standard results from stochastic approximation then ensure the convergence of $\{\theta_t\}$ w.p.1.

Tsitsiklis and Van Roy [321] also provide a bound on the error between the obtained function v_{θ^*} and the actual value-function V^δ as a function of the distance between V^δ and the hyperplane $\mathcal{V}$. This bound is given by

$$\left\| v_{\theta^*} - V^\delta \right\|_2 \leq \frac{1}{\sqrt{1-\beta^2}} \left\| V^\delta - \mathcal{P}_\mathcal{V} V^\delta \right\|_2, \tag{4.8}$$

where β is the contraction factor of the composite operator $\mathcal{P}_\mathcal{V} \mathbf{T}^\delta$.

Figure 4.2 illustrates the relation between V^δ, $\mathcal{P}_\mathcal{V} V^\delta$ and v_{θ^*}. The dashed lines represent the possible trajectories of the algorithm, converging to the limit point v_{θ^*}. The plane represents the space $\mathcal{V}$ and $\mathcal{P}_\mathcal{V} V^\delta$ is the orthogonal projection of the desired function V^δ on $\mathcal{V}$. The point v_{θ^*} described by (4.6) is the fixed point of the combined operator $\mathcal{P}_\mathcal{V} \mathbf{T}^\delta$ and corresponds to the point in $\mathcal{V}$ where $\mathbf{T}^\delta$ operates orthogonally to that hyperplane.

We now present without proof the main result from [321]. A more formal presentation of the ideas in this subsection can be found in [321, 326]. We henceforth refer to this method as *approximate TD*(0).

Theorem 4.5.1. *Let $(\mathcal{X}, \mathcal{A}, \mathsf{P}, r, \gamma)$ be a Markov decision process with compact state-space $\mathcal{X} \subset \mathbb{R}^p$ and δ a fixed stationary policy. Assume the Markov chain $(\mathcal{X}, \mathsf{P}_\delta)$*

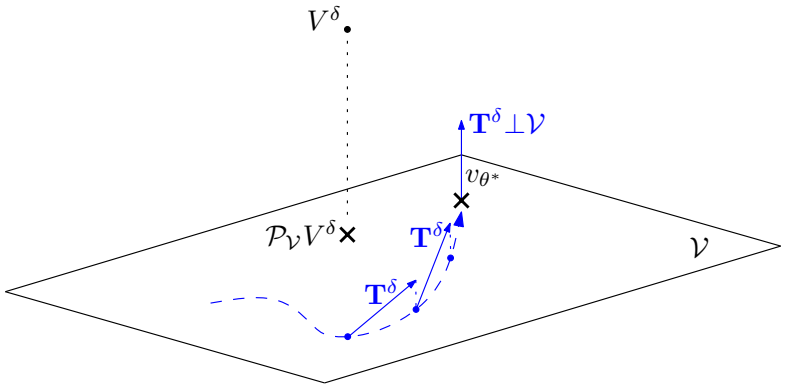


Figure 4.2: Difference between the projection $\mathcal{P}_\mathcal{V} V^\delta$ of V^δ on the space $\mathcal{V}$ and the function v_{θ^*} in (4.6). The solid blue lines represent the application of $\mathbf{T}^\delta$ and the dotted lines represent the orthogonal projection on $\mathcal{V}$, $\mathcal{P}_\mathcal{V}$. The dashed blue line represents a possible trajectory of the algorithm. Notice that, in v_{θ^*}, $\mathbf{T}^\delta$ is orthogonal to $\mathcal{V}$.

to be geometrically ergodic with invariant probability measure μ_X, where P_δ is the transition kernel for the chain obtained by following policy δ.

Let $\Xi = \{\xi_i, i = 1, \dots, M\}$ be a set of M bounded, linearly independent functions defined on $\mathcal{X}$ and taking values in $\mathbb{R}$.

Then, the following hold.

1. ***Convergence***

   For any initial condition $\theta_0 \in \mathbb{R}^M$, the algorithm

   $$\theta_{t+1} = \theta_t + \alpha_t \xi(x_t)\big(r_t + \gamma V_{\theta_t}(x_{t+1}) - V_{\theta_t}(x_t)\big) \tag{4.9}$$

   converges w.p.1 as long as the step-size sequence $\{\alpha_t\}$ verifies

   $$\sum_t \alpha_t = \infty \qquad \sum_t \alpha_t^2 < \infty.$$

2. ***Limit of convergence***

   Under these conditions, the limit θ^* of (4.9) verifies

   $$V_{\theta^*}(x) = (\mathcal{P}_\mathcal{V} \mathbf{T}^\delta V_{\theta^*})(x), \tag{4.10}$$

   where $\mathcal{P}_\mathcal{V}$ denotes the orthogonal projection operator on $\mathcal{V}$ defined by

   $$(\mathcal{P}_\mathcal{Q} V)(x) = \xi^\top(x) \Sigma^{-1} \mathbb{E}_{\mu_X}\left[\xi(z) V(z)\right].$$

   and the matrix Σ is given by

   $$\Sigma = \mathbb{E}_{\mu_X}\left[\xi(x)\xi^\top(x)\right].$$

4.5.2 Determining Q^*

We now extend the approach by Tsitsiklis and Van Roy [321] described in the previous section to accommodate *policy optimization*. The two algorithms proposed herein, dubbed *approximate Q-learning* and *approximate* SARSA, and corresponding analysis of convergence, constitute one of the fundamental contributions of this chapter.

Let $(\mathcal{X}, \mathcal{A}, \mathsf{P}, r, \gamma)$ be an MDP with infinite state-space $\mathcal{X}$, admittedly a compact subset of $\mathbb{R}^p$. We want to determine the optimal Q-function, verifying the following recursive relation

$$Q^*(x,a) = \int_{\mathcal{X}} \big[r(x,a,y) + \gamma \max_{b \in \mathcal{A}} Q^*(y,b) \big] \mathsf{P}_a(x,dy) \tag{4.11}$$

or, equivalently,

$$Q^*(x,a) = (\mathbf{H}Q^*)(x,a).$$

Once again, we consider a family of functions $\mathcal{Q} = \{Q_\theta\}$ parameterized by a M-dimensional parameter vector $\theta \in \mathbb{R}^M$. We admit $\mathcal{Q}$ to be a linear space in that, if $q_1, q_2 \in \mathcal{Q}$, then so does $\alpha q_1 + q_2$ for any $\alpha \in \mathbb{R}$. $\mathcal{Q}$ is therefore the linear span of a set of M linearly independent functions $\xi_i : \mathcal{X} \times \mathcal{A} \longrightarrow \mathbb{R}$, and each $q \in \mathcal{Q}$ can be written as

$$q(x,a) = \sum_{i=1}^{M} \xi_i(x,a)\theta(i)$$

for all pairs $(x,a) \in \mathcal{X} \times \mathcal{A}$. $\mathcal{Q}$ is a hyperplane of dimension M in the space of all measurable functions $q : \mathcal{X} \times \mathcal{A} \longrightarrow \mathbb{R}$. If $\Xi = \{\xi_i, i = 1, \ldots, M\}$ is a set of M linearly independent functions, we interchangeably use $Q(\theta)$ and Q_θ to denote the function

$$Q_\theta(x,a) = \sum_{i=1}^{M} \xi_i(x,a)\theta(i) = \xi^\top(x,a)\theta.$$

Given a policy δ and the corresponding sample trajectories $\{x_t\}$, $\{a_t\}$ and $\{r_t\}$, define the sequence $\{\theta_t\}$

$$\theta_{t+1} = \theta_t + \alpha_t \xi(x_t, a_t)\Delta_t, \tag{4.12}$$

where Δ_t is the temporal difference

$$\Delta_t = r_t + \gamma \max_{b \in \mathcal{A}} Q_{\theta_t}(x_{t+1}, b) - Q_{\theta_t}(x_t, a_t). \tag{4.13}$$

We refer to such update rule as *approximate Q-learning*.

As in Chapter 3, it is possible to devise an on-policy version of (4.13) using the alternative temporal difference

$$\Delta_t = r_t + \gamma Q_{\theta_t}(x_{t+1}, a_{t+1}) - Q_{\theta_t}(x_t, a_t). \tag{4.14}$$

We refer to the algorithm thus obtained as *approximate* SARSA.

In the remainder of this section we establish conditions under which the algorithm (4.12) converges using both (4.13) and (4.14).

4.5.3 Convergence of Q-learning with function approximation

To prove convergence of the algorithm (4.12) using the temporal difference in (4.13), we make use of a standard result from stochastic approximation, reproduced in Appendix D as Theorem D.1.1. The details of the proof can be found in Appendix F.

Let δ be a stationary policy and $(\mathcal{X}, \mathsf{P}_\delta)$ the corresponding Markov chain with invariant probability measure μ_X. Denote by $\mathbb{E}_{\mu_\delta}[\cdot]$ the expectation w.r.t. the probability measure μ_δ defined for every set $Z \times U \subset \mathcal{X} \times \mathcal{A}$ as

$$\mu_\delta(Z \times U) = \int_Z \sum_{a \in U} \delta(x,a)\mu_X(x).$$

Theorem 4.5.2. *Let $(\mathcal{X}, \mathcal{A}, \mathsf{P}, r, \gamma)$ be a Markov decision process with compact state-space $\mathcal{X} \subset \mathbb{R}^p$. Assume the Markov chain $(\mathcal{X}, \mathsf{P}_\delta)$ to be geometrically ergodic with invariant probability measure μ_X, where P_δ is the transition kernel for the chain obtained by following a stochastic policy δ verifying $\delta(x,a) > 0$ for all $a \in \mathcal{A}$ and μ_X-almost all $x \in \mathcal{X}$.*

Let $\Xi = \{\xi_i, i = 1, \ldots, M\}$ be a set of M bounded, linearly independent functions defined on $\mathcal{X} \times \mathcal{A}$ and taking values in $\mathbb{R}$. In particular, admit that $\sum_{i=1}^N |\xi_i(x,a)| \leq 1$ for all $(x,a) \in \mathcal{X} \times \mathcal{A}$.

Then, the following statements hold:

1. ***Convergence***

   For any initial condition $\theta_0 \in \mathbb{R}^M$, the algorithm

   $$\theta_{t+1} = \theta_t + \alpha_t \xi(x_t, a_t)\big(r_t + \gamma \max_{b \in \mathcal{A}} Q_{\theta_t}(x_{t+1}, b) - Q_{\theta_t}(x_t, a_t)\big) \tag{4.15}$$

   converges w.p.1 as long as the step-size sequence $\{\alpha_t\}$ verifies

   $$\sum_t \alpha_t = \infty \qquad \sum_t \alpha_t^2 < \infty.$$

2. ***Limit of convergence***

   Under these conditions, the limit θ^* of (4.15) verifies

   $$Q_{\theta^*}(x,a) = (\mathcal{P}_\mathcal{Q} \mathbf{H} Q_{\theta^*})(x,a), \tag{4.16}$$

   where $\mathcal{P}_\mathcal{Q}$ denotes the orthogonal projection operator on $\mathcal{Q}$ defined by

   $$(\mathcal{P}_\mathcal{Q} Q)(x,a) = \xi^\top(x,a)\Sigma^{-1}\mathbb{E}_{\mu_\delta}\left[\xi(z,u)Q(z,u)\right].$$

   and the matrix Σ is given by

   $$\Sigma = \mathbb{E}_{\mu_\delta}\left[\xi(x,a)\xi^\top(x,a)\right].$$

Proof See Appendix F.2. □

In the proof of the theorem, we carefully establish each of the assertions of the theorem. To prove the first assertion we write (4.15) in the form

$$\theta_{t+1} = \theta_t + \alpha_{t+1} H(\theta_t, X_{t+1}), \tag{4.17}$$

and establish that the trajectories of the algorithm closely follow those of an associated ODE. We then show independently that this associated ODE has a globally asymptotically stable equilibrium point, which in turn implies the convergence of (4.15) w.p.1.

To prove the second assertion of Theorem 4.5.2, we provide an interpretation of the equilibrium point of the ODE as the fixed point of a composite operator.

To convey a deeper insight on the conditions of the Theorem, we remark that the geometric ergodicity assumption and the requirement that $\delta(x, a) > 0$ for all $a \in \mathcal{A}$ and μ_X-almost all $x \in \mathcal{X}$ can be interpreted as a continuous counterpart to the usual condition that all state-action pairs are visited infinitely often. In fact, geometric ergodicity implies that all the regions of the state-space with positive μ_X measure are "sufficiently" visited [201], and the condition that $\delta(x, a) > 0$ ensures that, at each state, every action is "sufficiently" tried.

On the other hand, geometric ergodicity guarantees that the Markov chain $(\mathcal{X}, \mathsf{P}_\delta)$ converges exponentially fast to stationarity, and thus the analysis of convergence of the algorithm can be conducted in terms of a "stationary version" of it.[3]

We also notice that the requirements on the basis functions ξ_i simply guarantee (in a rather conservative way) that no two functions ξ_i lead to "colliding updates", as in the known counter-example by Baird [11].[4]

4.5.4 Convergence of SARSA with function approximation

Convergence of the algorithm in (4.14) is essentially a consequence of Theorem 4.5.2 and is summarized in the following result.

Theorem 4.5.3. *Let $(\mathcal{X}, \mathcal{A}, \mathsf{P}, r, \gamma)$ be a Markov decision process with a compact state-space $\mathcal{X} \subset \mathbb{R}^p$. Let $(\delta_\theta)_t$ be a fixed learning policy with the GLIE property w.r.t. the estimate $Q_\theta(x, a) = \xi^\top(x, a)\theta$. Suppose, furthermore, that $\delta_t(x, a) > 0$ and that there is a constant $C > 0$ such that for each $t \in \mathcal{T}$ and each $(x, a) \in \mathcal{X} \times \mathcal{A}$*

$$|(\delta_\theta)_t(x, a) - (\delta_{\theta'})_t(x, a)| \leq C \|\theta - \theta'\|. \tag{4.18}$$

[3] In particular, exponential convergence to stationarity ensures that the analysis of the trajectories of the sequence $\{\theta_t\}$ can be performed in terms of the trajectories of an associated ODE, as mentioned above.

[4] Suppose, for example, that two basis functions ξ_1 and ξ_2 were used, with $\xi_1 = -\xi_2$. Then, the corresponding updates would always have opposite signs and the contribution of these functions in the final approximation would always be zero.

Assume the Markov chain $(\mathcal{X}, \mathsf{P}_\theta)$ to be geometrically ergodic with invariant probability measure μ_X^θ, for every θ. We denote by P_θ the transition probabilities for the chain obtained by following policy $(\delta_\theta)_t$, with fixed θ.

Let $\Xi = \{\xi_i, i = 1, \ldots, M\}$ be a set of M bounded, linearly independent functions defined on $\mathcal{X} \times \mathcal{A}$ and taking values in $\mathbb{R}$. In particular, admit that $\sum_{i=1}^N |\xi_i(x, a)| \leq 1$ for all $(x, a) \in \mathcal{X} \times \mathcal{A}$.

Then, the following hold.

1. ***Convergence***

   For any initial condition $\theta_0 \in \mathbb{R}^M$, the algorithm

$$\theta_{t+1} = \theta_t + \alpha_t \xi(x_t, a_t)\big(r_t + \gamma Q_{\theta_t}(x_{t+1}, a_{t+1}) - Q_{\theta_t}(x_t, a_t)\big) \tag{4.19}$$

   converges w.p.1 as long as the step-size sequence $\{\alpha_t\}$ verifies

$$\sum_t \alpha_t = \infty \qquad \sum_t \alpha_t^2 < \infty.$$

2. ***Limit of convergence***

   Under these conditions, the limit θ^* of (4.15) verifies

$$Q_{\theta^*}(x, a) = (\mathcal{P}_{\mathcal{Q}} \mathbf{H} Q_{\theta^*})(x, a), \tag{4.20}$$

   where $\mathcal{P}_{\mathcal{Q}}$ denotes the orthogonal projection operator on $\mathcal{Q}$ defined w.r.t. the measure $\mu_X^{\theta *}$.

Proof See Appendix F. □

We conclude this section by presenting the following immediate corollary of Theorems 4.5.2 and 4.5.3.

Corollary 4.5.4. *Let $(\mathcal{X}, \mathcal{A}, \mathsf{P}, r, \gamma)$ be an MDP with compact state-space $\mathcal{X} \subset \mathbb{R}^p$. The sequence $\{\theta_t\}$ generated by (4.15) converges to a limit point θ^* verifying*

$$Q_{\theta^*}(x, a) = (\mathcal{P}_{\mathcal{Q}} \mathbf{H} Q_{\theta^*})(x, a), \tag{4.21}$$

as long as the conditions of Theorem 4.5.2 hold. Furthermore, if the agent uses a GLIE learning policy $(\delta_\theta)_t$ verifying the conditions of Theorem 4.5.3, the agent will converge in behavior to the greedy policy w.r.t. Q_{θ^*}.

Before concluding this section, we analyze the practical implications of the conditions stated in the previous theorems in terms of Markov decision processes. A more general discussion on the practical implications (in terms of robotic systems)

of the several conditions required by the different convergence theorems along the thesis is postponed to Chapter 10.

First of all, the conditions listed are sufficient, meaning that it is possible that convergence may occur even if some (or all) of the conditions fail to hold. On the other hand, apart from the standard condition on the step-size sequence $\{\alpha_t\}$, Theorem 4.5.2 requires two essential conditions to ensure convergence: one on the function set Ξ and the other on the Markov chain.

The condition on the functions in Ξ depends only on the implementation, and is therefore, not too restrictive. Given a set of bounded, linearly independent functions, a simple normalization process ensures these functions to verify $\sum_{i=1}^{M} |\xi_i(x,a)| \leq 1$ for all $(x,a) \in \mathcal{X} \times \mathcal{A}$.

The only "worrisome" condition is the one related with the geometric ergodicity of the Markov chain obtained from the learning strategy. Geometric ergodicity of a Markov chain basically implies that the chain quickly converges to a stationary behavior and we can, therefore, study any properties of the stationary chain by direct sampling. This definition is, however, far from rigorous and we refer to Appendix B for a formal definition.

Geometric ergodicity can be established using several possible approaches (see Theorems B.8.2 through B.8.4 in Appendix B). In Section 4.7 we apply a simple procedure to establish geometric ergodicity of a continuous chain obtained from a related discrete chain. This approach requires the chain $(\mathcal{X}, \mathsf{P}_\delta)$ to verify the *Feller property*, stating that the transition kernel P_δ is "well-behaved", in that the transition probabilities do not abruptly change around each point in $\mathcal{X}$.[5] As long as this property holds, it is relatively simple to verify geometric ergodicity.

It so happens that, in the applications envisioned herein, the Feller property is a rather fair assumption, and geometric ergodicity is not an unreasonable requirement for convergence. In a very broad sense, as already stated, this requirement is closely related to the requirement of sufficient exploration in the classical RL convergence results.

4.6 Two illustrative examples

We now analyze two continuous versions of the example used in the previous chapters.

Example 4.1. Consider the indoor environment depicted in Figure 4.3. A mobile robot is intended to navigate from the bottom-left corner to the *goal region*, signaled with the bold, red line. The environment is a 1×1 square and the state of the robot at each time instant is represented as a pair $(\mathbf{x}, \mathbf{y})$ of coordinates.[6] The coordinates of the corner in the goal region are $(1, 1)$.

[5] In other words, considering two sufficiently close states $x, y \in \mathcal{X}$, the probability of moving from x to some open set $O \in \mathcal{B}(\mathcal{X})$ is not too different from the probability of moving from y to O. We once again refer to Appendix B for a formal definition.

[6] We use boldface symbols $\mathbf{x}$ and $\mathbf{y}$ to denote the physical coordinates of the robot to distinguish from the symbols x and y used to denote generic elements of the state-space $\mathcal{X}$.

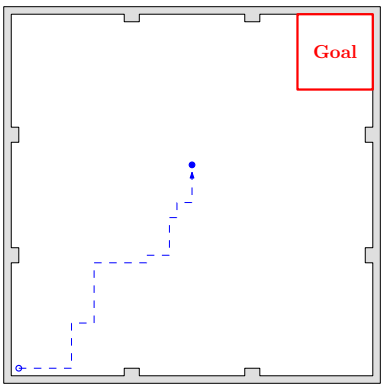


Figure 4.3: Indoor environment for Examples 4.1 and 4.2.

The navigation problem can be described by an MDP $(\mathcal{X}, \mathcal{A}, \mathsf{P}, r, \gamma)$, where

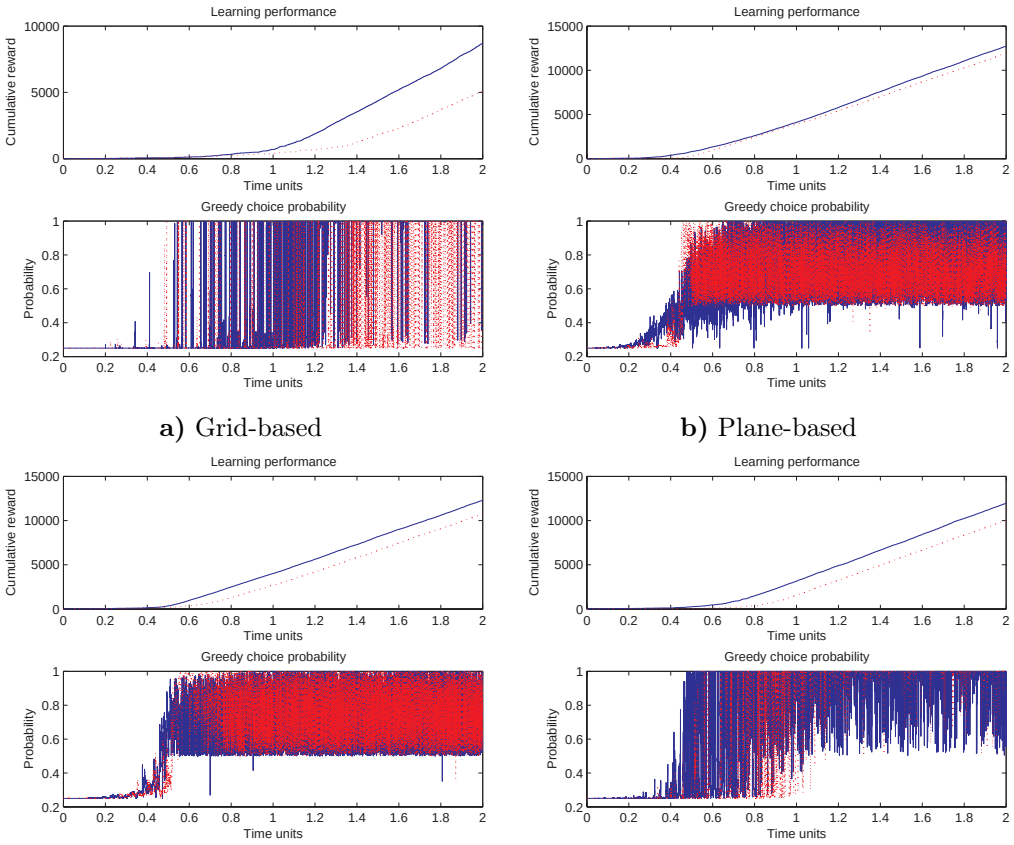
- $\mathcal{X} = [0; 1] \times [0; 1]$;
- $\mathcal{A} = \{N, S, E, W\}$;
- Each action $a \in \mathcal{A}$ moves the robot in the corresponding direction of a random amount taking values uniformly between 0 and 0.3;
- The reward function r assigns a reward of $+10$ for every transition triplet (x, a, y) such that $\|y - (1; 1)\|_\infty < 0.1$ and 0 otherwise;
- We consider $\gamma = 0.95$.

When the agent reaches the goal region, its position is randomly reset to any point in the room, independently of the agent's action.

We applied approximate Q-learning and approximate SARSA to this MDP. The robot was allowed to explore and learn during 2×10^4 time steps, and the obtained policy was then evaluated for 100 time steps. During learning we used Boltzmann exploration in all experiments.

We ran four experiments, each using a different set of basis functions, to be found in page 370. In Figure 4.4 we depict the total reward obtained during learning in each of the four experiments. In Table 4.1 we present the total discounted reward obtained during the 100 time-steps test period.

In each of the plots of Figure 4.4, the solid blue line represents the Q-learner and the dashed red line represents the approximate SARSA learner. For each of the 4 experiments (corresponding to a different set of basis functions) we depicted the total reward obtained by the robot during learning and the probability of choosing the optimal action. Expectedly, as this probability approaches 1, the performance of the robot should improve, since the robot explores much less and abundantly exploits the information collected during the learning. This is easily seen from the plots by observing the slope of the learning curve. Another important aspect is that, in all 4 examples, the final slope of the curve is similar for both approximate Q-learning and approximate SARSA. Noticing that the slope of the curve is a "rough" indicator of the performance of the robot, we can conclude from the plots in Figure 4.4 that in all 4 experiments both methods converged to a similar performance. This is expected because, as seen in Theorems 4.5.2 and 4.5.3, both approximate



a) Grid-based **b)** Plane-based

c) Exp-based **d)** Kernel-based

Figure 4.4: Example 4.1: Cumulative reward and greedy choice probability during the 2×10^4-time-units learning period. The solid blue line represents the results with approximate Q-learning and the dashed red line represents the results with approximate SARSA.

Q-learning and approximate SARSA converge to the same approximation, if using the same set of basis functions. Finally, just like in the finite-state case, approximate SARSA is slightly slower, due to its on-policy updates.

On the other hand, there are some slight differences between the plots of the different experiments. This is due to the representational power of the chosen set of basis functions. Although the methods are guaranteed to converge, these guarantees do not imply anything on the quality of the approximation. Therefore, a set of basis functions that allows a more accurate representation of the optimal Q-function will learn faster and exhibit better final performance. Although the differences observed with the 4 sets of functions used are not significant (the final slope of the learning curve is approximately the same in all 4 plots), other sets of basis functions could possibly lead to much sharper differences.

We also tested each of the learnt policies in the environment. We ran each of the learnt policies for 100 time units and determined the total discounted reward obtained in each case. Table 4.1 represents the results obtained. We have run $2,000$ independent Monte-Carlo trials and present the average and standard deviation obtained using each of the methods. Once again, both

Table 4.1: Example 4.1: Comparative results for approximate Q-learning and approximate SARSA after the learning period is complete. We present the average total discounted reward and standard deviation obtained over $2,000$ independent Monte-Carlo trials.

Method	**Q-learning**			**SARSA**		
Grid-based	8.474	$\pm$	1.728	8.576	$\pm$	1.706
Plane-based	8.221	$\pm$	1.751	8.188	$\pm$	1.700
Exp-based	8.212	$\pm$	1.716	8.134	$\pm$	1.701
Kernel-based	8.554	$\pm$	1.687	8.573	$\pm$	1.710

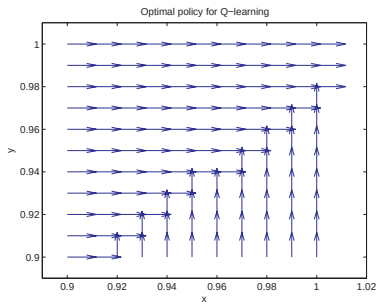


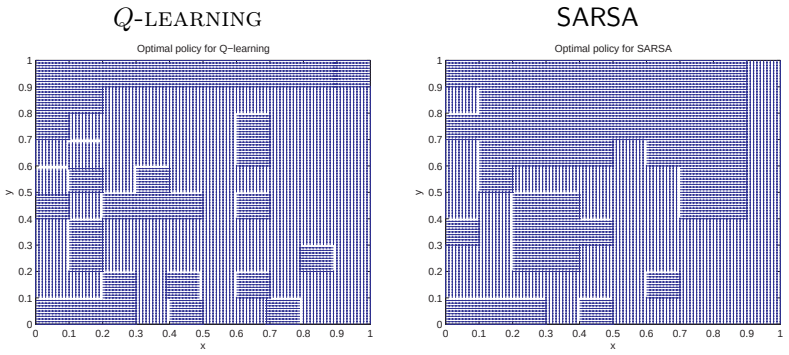
Figure 4.5: Detail of the policy obtained using Q-learning with a plane-based approximation (detail of Figure 4.6).

methods present a similar performance, since they are both expected to converge to a similar approximation of the optimal Q-function. Also, the difference in performance when comparing the different sets of basis functions are not significant.

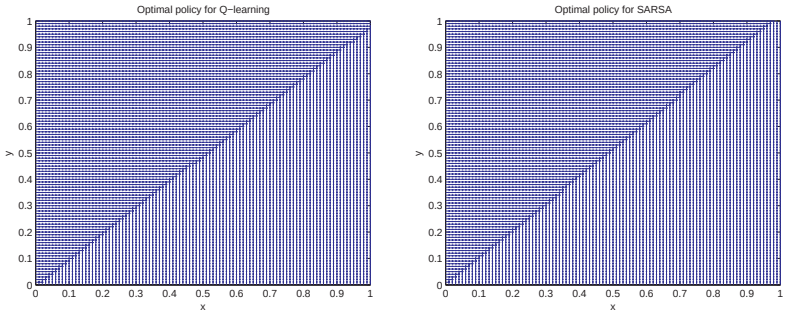
We present in Figure 4.5 a detail of the learnt policy around the goal region for approximate Q-learning in one of the experiments. The complete policy is plotted in Figure 4.6 for both approximate Q-learning and approximate SARSA, for the 4 sets of basic functions. The detail in Figure 4.5 should help to clarify the different patterns observed in the policy representations of Figure 4.6, by comparison.

Notice that, although the policy representations corresponding to approximate Q-learning and approximate SARSA are not always coincident, they correspond to similar optimal behavior. This can be seen if we interpret the directional lines in Figure 4.6 as *flux lines* describing the movement of the robot. Then, starting at any position in the 1×1 square, all policies will drive the robot to the goal region. The differences between Q-learning and SARSA are easily understood if we realize that they have very different sampling methods.[7] The finite learning time and the different sampling method account for the differences between the learnt policies (different trajectories, rewards and updates).

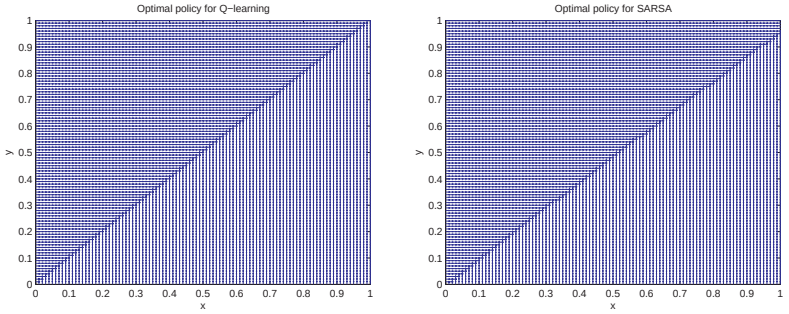
[7]The fact that Q-learning is an off-policy method and SARSA is an on-policy method necessarily implies that they follow different learning policies and, therefore, sample the state-space in distinct ways.



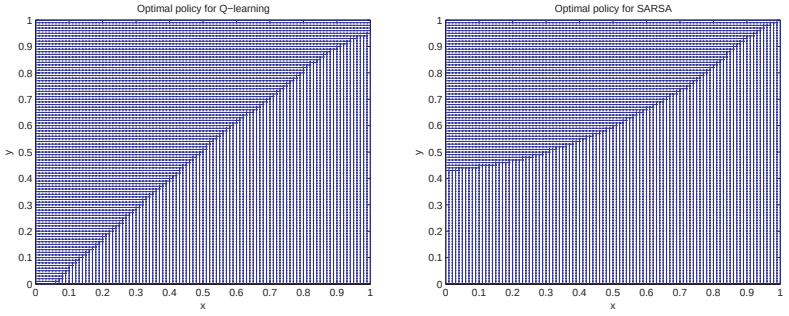
a) Grid-based approximation;



b) Plane-based approximation;

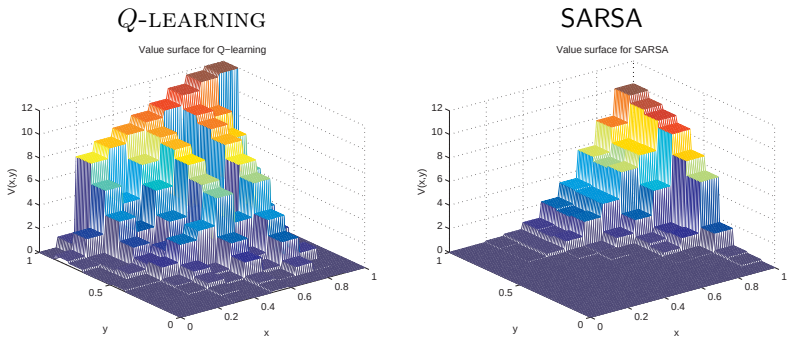


c) Exp-based approximation;

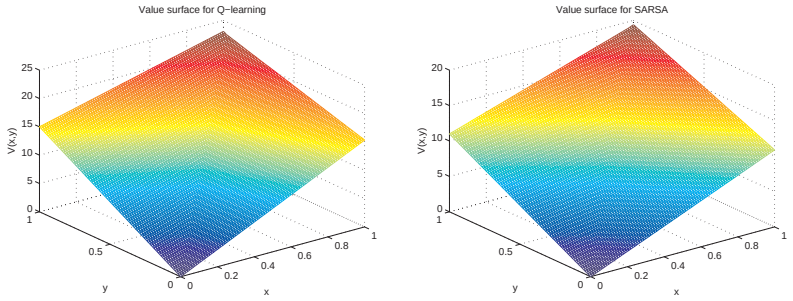


d) Kernel-based approximation;

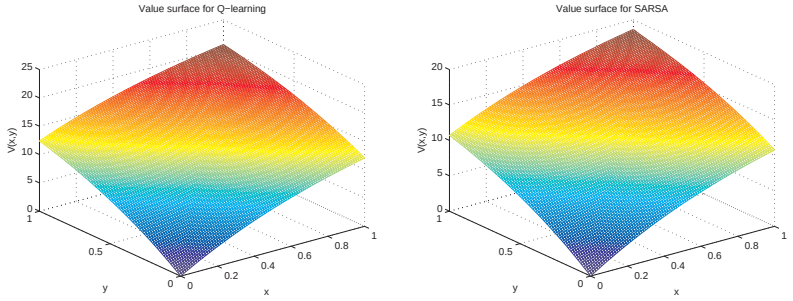
Figure 4.6: Example 4.1: Policies learnt by approximate Q-learning and approximate SARSA for different sets of basis functions.



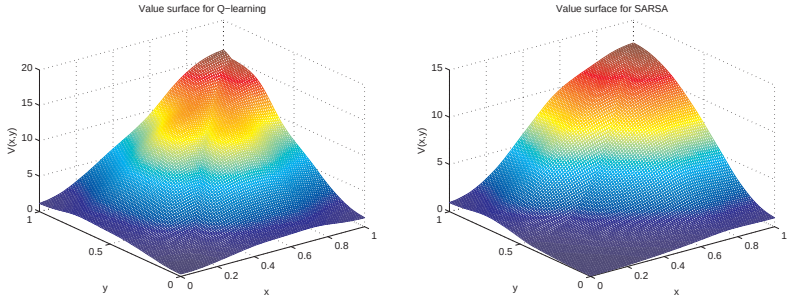
a) Grid-based approximation;



b) Plane-based approximation;

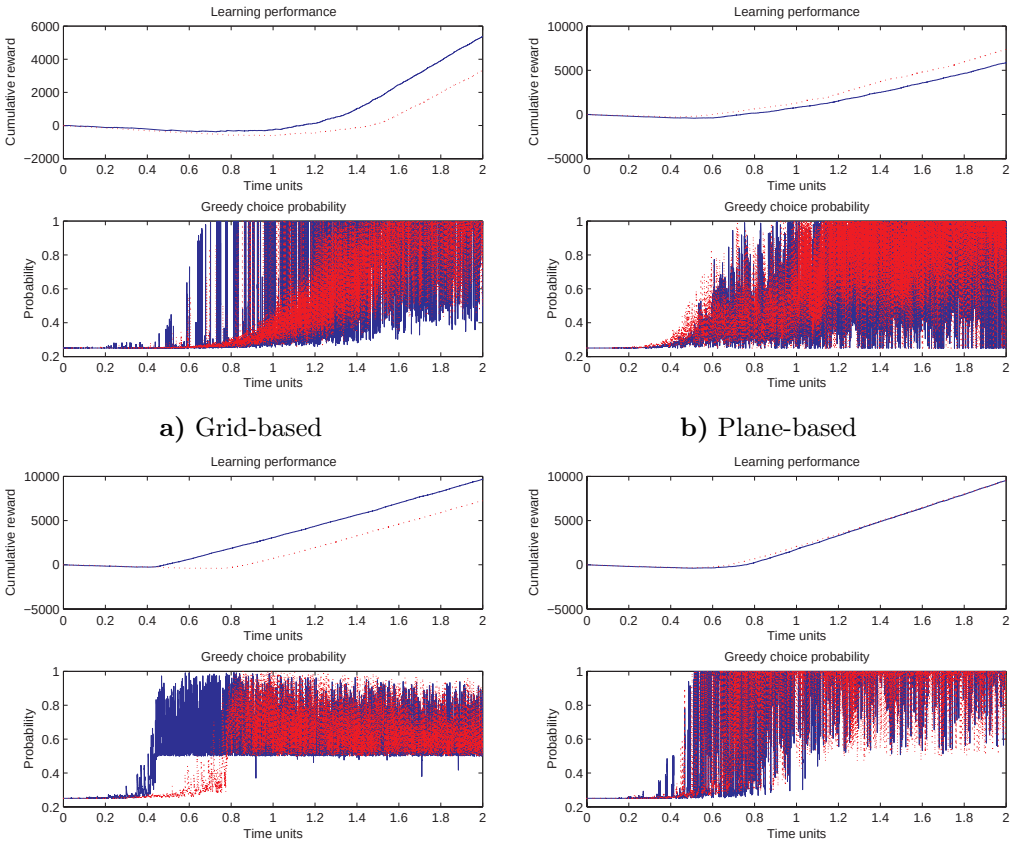


c) Exp-based approximation;



d) Kernel-based approximation;

Figure 4.7: Example 4.1: Value-functions learnt by approximate Q-learning and approximate SARSA using different sets of basis functions.



a) Grid-based **b)** Plane-based

c) Exp-based **d)** Kernel-based

Figure 4.8: Example 4.2: Cumulative reward and greedy choice probability during the 2×10^4-time-units learning period. The solid blue line represents the results with approximate Q-learning and the dashed red line represents the results with approximate SARSA.

On the other hand, the different patterns observed in Figure 4.6 when comparing the different approximations used are clearly due to the differences between the basis functions chosen. Each basis function defines a surface over the 1×1 square and the different patterns in Figure 4.6 arise from the intersections of those surfaces. To further clarify this point, we present in Figure 4.7 the surfaces representing the obtained value functions for Q-learning and SARSA using the different sets of basis functions. ◇

Example 4.2. To further explore the application of the algorithms, we repeated all tests in a modified version of the previous problem. In this modified version, we penalized the agent with a reward of -0.1 every time-step, except on the goal region, in which we kept the same reward of $+10$. Notice that this does not affect the optimal policy, but it does affect the value function. The robot was once again allowed to explore and learn during 2×10^4 time steps and the obtained policy was then evaluated for 100 time units. The results are presented in Figure 4.8 and Table 4.2.

Table 4.2: Example 4.2: Comparative results for approximate Q-learning and approximate SARSA after the learning period is complete. We present the average total discounted reward and standard deviation obtained over $2,000$ independent Monte-Carlo trials.

Method	**Q-learning**			**SARSA**		
Grid-based	6.223	$\pm$	1.799	6.641	$\pm$	1.707
Plane-based	6.476	$\pm$	1.665	6.712	$\pm$	1.754
Exp-based	6.616	$\pm$	1.710	6.506	$\pm$	1.671
Kernel-based	6.714	$\pm$	1.718	6.606	$\pm$	1.660

Figure 4.8 represents the total reward obtained by the robot during the learning period. Once again, the solid blue line represents the approximate Q-learner and the dashed red line represents the approximate SARSA learner. Because of the similarity between this problem and the one considered previously, all comments on the former problem apply to the current one.

However, the simple inclusion of the negative payoff leads to some interesting effects that are worth commenting. In the previous example, the robot only received feed-back upon reaching the goal, where it would get a payoff of $+10$. During the learning period, due to the required exploration, it would be possible for the robot to move about the environment for a long period of time without getting any reward. This means that the updates of both methods would rely on the information gathered around the goal region. Therefore, the differences between the functions learnt by approximate Q-learning and approximate SARSA were justified merely by the different frequencies with which each method could eventually visit this region.

In this problem, *all regions* of the state-space contribute with information used in the updates. This means that the differences between the *trajectories* of both methods during learning will have a larger influence on the learnt value functions (in finite time). However, from Figure 4.8 we can conclude that, in spite of these differences, both methods were still able to reach a similar performance.

When tested in the environment for 100 time units, the total discounted reward obtained in each case is presented in Table 4.2. We have run $2,000$ independent Monte-Carlo trials and present the average and standard deviation observed using each of the methods. As we had concluded from observing Figure 4.8, the results in Table 4.2 also confirm that both methods present a similar performance. Notice the difference between the results in Tables 4.1 and 4.2: because of the negative rewards at each time-step, even the optimal policy will lead to an inferior payoff, when compared with the one observed in Example 4.1.

To conclude this section we present in Figures 4.9 and Figure 4.10 the learnt policy and value function for the current problem. We emphasize, once again, the much sharper differences between the policy/value-function learnt when using approximate Q-learning and those learnt using approximate SARSA. Nevertheless, as can easily be confirmed from Figure 4.9, both algorithms still

manage to learn an optimal policy.

◇

4.7 Partial observability

In this section we finally address the problem of *partial observability*. So far, we considered that the agent was able to unambiguously perceive its current state and act accordingly. However, it is a fact that in most real situations autonomous agents have limited sensing capabilities and are not able to completely determine the current state of their surroundings. Instead, they have access to a set of noisy measurements from which they must infer the value of that state. When this happens, agents are said to have *partial observability*.

For example, if we consider the particular case of mobile robot navigation, actual robots rely on sensorial data to determine their state/location in the environment. Any procedure relying on sensor data should account for the noise in the measurements and it is reasonable to admit partial observability when modeling the navigation of a mobile robot. In problems with partial observability, the decision-maker must act based on its observations and past history. This information can be used to estimate the actual state, but this estimate is generally accurate only up to some degree of uncertainty.

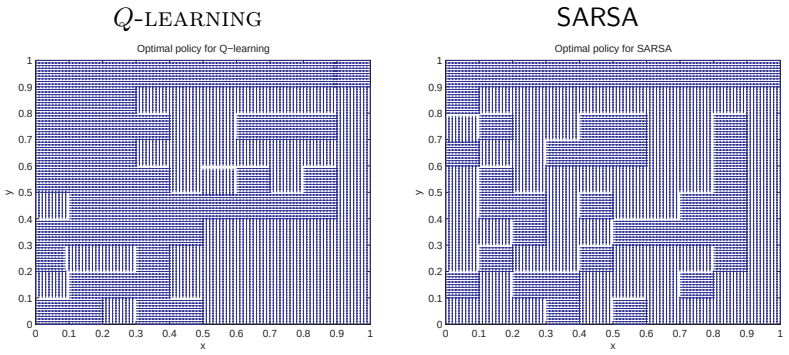
The uncertainty arising from partial observability is a complex problem to deal with in learning situations. The most widely used algorithms in the reinforcement learning literature assume that the learning agent is able to determine the state of the system unambiguously. When this assumption fails, all those methods present severe limitations and yield, in general, poor policies.

The extension of reinforcement learning methods from fully observable environments to to their partially observable counterparts is far from easy. In fact, although partial observability allows more realistic modeling in many problems, this added modeling ability comes at a significant cost in complexity, since even algorithms using complete models of the environment are often untractable [164, 175, 177, 206].

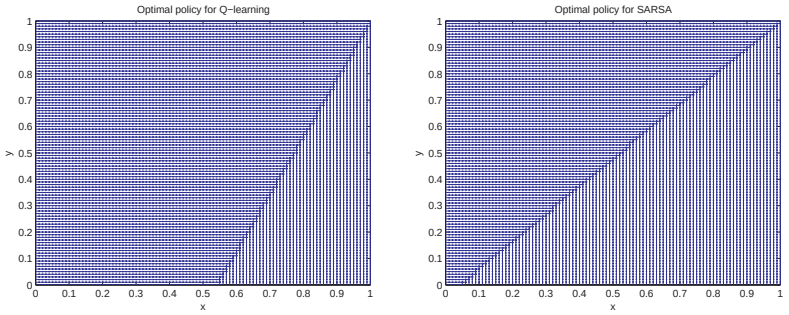
In spite of all this, there are still numerous examples in which the straightforward application of RL algorithms to partially observable scenarios led to impressive results, examples for which those methods were not primarily designed [173]. Such successes gave rise to a significant effort to extend reinforcement learning algorithms into the world of partial observability.

Partial observability in mobile robot navigation

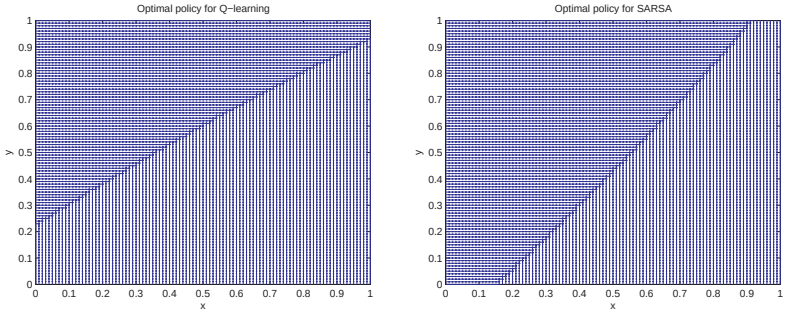
In Chapter 2 we justified the use of topological maps in part for its scalability: topological maps represent an environment as a (usually finite) graph. Also, as seen in Chapter 2, this representation gives rise to a finite-state MDP model for the navigation of the robot. But in the previous sections we have introduced several methods to approximate V^δ and Q^* in MDPs with infinite state-spaces. It would seem that topological navigation would relieve us from having to resort to these more elaborate algorithms, which require more stringent assumptions than those in Chapter 3.



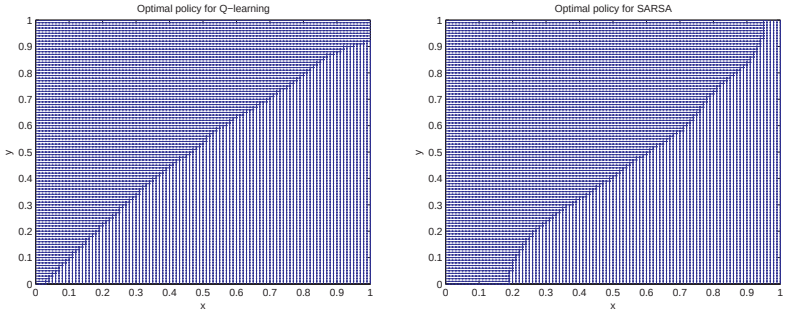
a) Grid-based approximation;



b) Plane-based approximation;

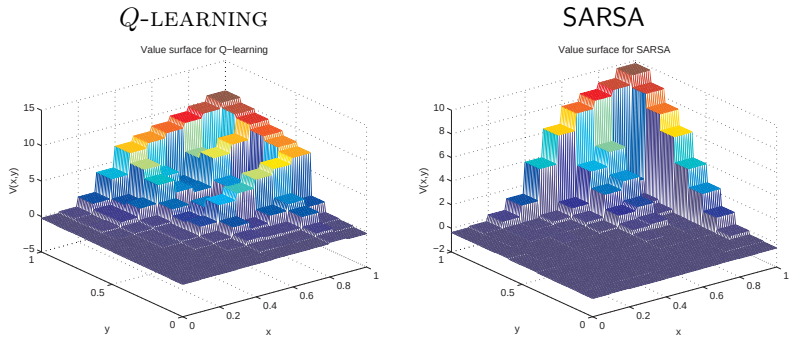


c) Exp-based approximation;

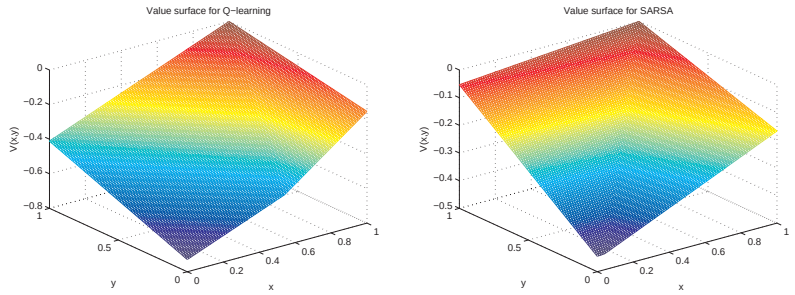


d) Kernel-based approximation;

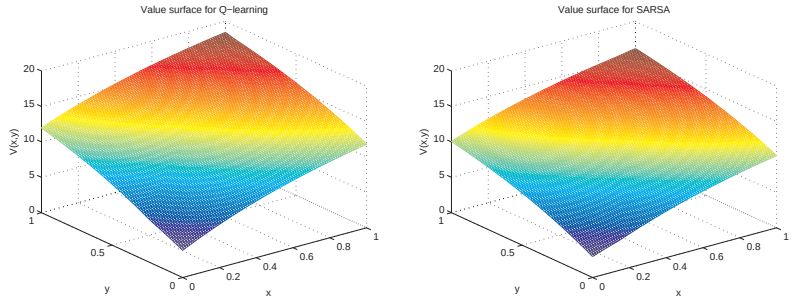
Figure 4.9: Example 4.2: Policies learnt by approximate Q-learning and approximate SARSA for different sets of basis functions.



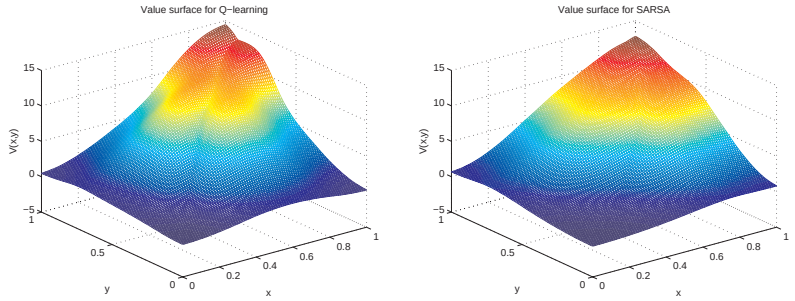
a) Grid-based approximation;



b) Plane-based approximation;



c) Exp-based approximation;



d) Kernel-based approximation;

Figure 4.10: Example 4.2: Value-functions learnt by approximate Q-learning and approximate SARSA for different sets of basis functions.

On the other hand, as already stated, in all the Markov processes considered so far it is implicitly assumed that the state X_t of the process is accessible to the agent, *i.e.*, the agent has *full observability*. In fact, the very definition of policy introduced implies that an agent chooses its actions *as a function of the state of the process.*

However, the assumption of full observability is seldom verified in practice. Mobile robots interacting in the real world are able to perceive the state of the process (*i.e.*, their location in the environment) by means of their sensors, which are often subject to errors and noise. This means that any such perception of the state has necessarily some inherent uncertainty and the model provided by the framework of MDPs does not account for such uncertainty.

In the remainder of the chapter, we discard the full observability assumption. We describe the framework of partially observable Markov chains and partially observable MDPs (POMDPs), while analyzing some fundamental consequences of the loss of observability in terms of the theory of Markov chains.

We re-cast a partially observable finite state-space Markov chain as a fully observable infinite state-space Markov chain. This leads to a central result that describes the stability/ergodicity properties of this associated fully-observable Markov chain in terms of the stability/ergodicity/observability properties of the original partially-observable chain.

This result establishes the applicability of the methods developed in this chapter to topological navigation problems that include partial observability, extending the Markov decision process model used so far.

4.7.1 Partial observability and internal state

In the theory and methods described so far in this chapter, we have considered general, infinite state-space Markov processes. We now return to the less general situation addressed in Chapter 3. In particular, the Markov processes in the remainder of the chapter admittedly have a *finite* state-space $\mathcal{X}$. As will soon become apparent, the introduction of partial observability, even in this finite setting, immediately leads back to the infinite state-space scenario. Nevertheless, and unless is explicitly stated, we admit $\mathcal{X}$ to be finite.

Let $(\mathcal{X}, \mathsf{P})$ be a *finite* state-space Markov chain. Let $\mathcal{Z}$ be a finite set of *possible observations* and suppose that, at each time instant, the state X_t of the chain is not observable. Instead, a random measurement Z_t is "issued" that depends on the state X_t of the chain according to an observation probability given by

$$\mathbb{P}\left[Z_t = z \mid X_t = i\right] = \mathsf{O}(i, z), \tag{4.22}$$

Notice that, as indicated in (4.22), $\mathsf{O}(i, \cdot)$ is a discrete (probability) measure on $\mathcal{Z}$, since $\mathcal{Z}$ is finite. The function O determines the probability of making a particular observation $z \in \mathcal{Z}$ as a function of the current state X_t and is referred as the *observation probability function.* Notice that, since both $\mathcal{X}$ and $\mathcal{Z}$ are admitted finite, P and O can be represented by matrices.

A *partially observable Markov chain* (also known as a *hidden Markov model*) is defined as a 4-tuple $(\mathcal{X}, \mathcal{Z}, \mathsf{P}, \mathsf{O})$, where $\mathcal{X}$ and $\mathcal{Z}$ are the state and observation spaces and P and O are the transition and observation probability matrices.

Let π_t be a discrete probability measure on $\mathcal{X}$ conveying the probability distribution of the state X_t over the set $\mathcal{X}$ at time instant t. We write $X_t \sim \pi_t$ to denote the fact that the r.v. X_t is distributed according to π_t. Since $\mathcal{X}$ is assumed finite, π_t is a vector with i^{th} component given by

$$\pi_t(i) = \mathbb{P}\left[X_t = i \mid \mathcal{F}_t\right], \tag{4.23}$$

where $\mathcal{F}_t$ represents the history of the process up to time t.

Suppose, now, that at time instant t the chain is in state $i \in \mathcal{X}$ with probability $\pi_t(i)$ and a transition occurs, governed by the transition probabilities in P. Suppose, furthermore, that an observation $Z_{t+1} = z$ is made at time instant $t+1$. Some explicit computations lead to

$$\begin{aligned}
&\mathbb{P}\left[X_{t+1} = j \mid Z_{t+1} = z, X_t \sim \pi_t\right] = \\
&= \frac{\mathbb{P}\left[X_{t+1} = j, Z_{t+1} = z \mid X_t \sim \pi_t\right]}{\mathbb{P}\left[Z_{t+1} = z \mid X_t \sim \pi_t\right]} = \\
&= \frac{\mathbb{P}\left[Z_{t+1} = z \mid X_{t+1} = j, X_t \sim \pi_t\right]\mathbb{P}\left[X_{t+1} = j \mid X_t \sim \pi_t\right]}{\sum_{k\in\mathcal{X}}\mathbb{P}\left[Z_{t+1} = z \mid X_{t+1} = k, X_t \sim \pi_t\right]\mathbb{P}\left[X_{t+1} = k \mid X_t \sim \pi_t\right]}.
\end{aligned}$$

Using the Markov property, this leads to

$$\begin{aligned}
&\mathbb{P}\left[X_{t+1} = j \mid Z_{t+1} = z, X_t \sim \pi_t\right] = \\
&= \frac{\sum_{i\in\mathcal{X}}\mathbb{P}\left[Z_{t+1} = z \mid X_{t+1} = j\right]\mathbb{P}\left[X_{t+1} = j \mid X_t = i\right]\mathbb{P}\left[X_t = i \mid X_t \sim \pi_t\right]}{\sum_{i,k\in\mathcal{X}}\mathbb{P}\left[Z_{t+1} = z \mid X_{t+1} = k\right]\mathbb{P}\left[X_{t+1} = k \mid X_t = i\right]\mathbb{P}\left[X_t = i \mid X_t \sim \pi_t\right]} = \\
&= \frac{\sum_{i\in\mathcal{X}}\mathsf{O}(j,z)\mathsf{P}(i,j)\pi_t(i)}{\sum_{i,k\in\mathcal{X}}\mathsf{O}(k,z)\mathsf{P}(i,k)\pi_t(i)} = \\
&= \frac{\sum_{i\in\mathcal{X}}\pi_t(i)\mathsf{P}(i,j)\mathsf{O}(j,z)}{\sum_{i,k\in\mathcal{X}}\pi_t(i)\mathsf{P}(i,k)\mathsf{O}(k,z)}.
\end{aligned}$$

As such, given the observation $Z_{t+1} = z$, the update rule for π_t is given by

$$\pi_{t+1}(j) = \Pi(\pi_t, z)_j = \frac{\sum_{i\in\mathcal{X}}\pi_t(i)\mathsf{P}(i,j)\mathsf{O}(j,z)}{\sum_{i,k\in\mathcal{X}}\pi_t(i)\mathsf{P}(i,k)\mathsf{O}(k,z)}. \tag{4.24}$$

It is known that the vector π_t is a sufficient statistic for X_t [51] and it "records" all observations made so far in a single probability vector.[8] Notice, furthermore, that π_{t+1} depends exclusively on the parameters P and O of the process and on the *previous probability vector* π_t, which implies that, in this sense, *it verifies the Markov property.*

As such, consider the stochastic process $\{\pi_t\}$ where π_t is as defined by (4.23).[9] Suppose that $|\mathcal{X}| = n$. Then, since each π_t is a probability vector, it lies in $\mathbb{R}^n$ and,

[8] A *sufficient statistic* for X_t means that no probability distribution on $\mathcal{X}$ is more "accurate" than π_t given the whole history of the process up to time t.

[9] Notice that the randomness in π_t is due to the dependence on the observations Z_t which are random variables.

in particular, in the $n-1$-dimensional simplex $\mathbb{S}^n$ defined as

$$\mathbb{S}^n = \left\{ x \in \mathbb{R}^n \mid \sum_{i=1}^{n} x_i = 1 \right\}.$$

As a subset of $\mathbb{R}^n$, $\mathbb{S}^n$ is endowed with all the metric and topological properties of $\mathbb{R}^n$. In particular, considering the usual topology in $\mathbb{R}^n$, $\mathbb{S}^n$ is a compact, separable and metrizable topological space and these properties permit the utilization of several results from the theory of Markov chains on topological spaces.

Given any partially observable Markov chain $(\mathcal{X}, \mathcal{Z}, \mathsf{P}, \mathsf{O})$, we can define an equivalent fully-observable Markov chain $(\mathbb{S}^n, \bar{\mathsf{P}})$, where $n = |\mathcal{X}|$ and the kernel $\bar{\mathsf{P}}$ is given, for any $\pi \in \mathbb{S}^n$ and any set $U \in \mathcal{B}(\mathbb{S}^n)$, by

$$\bar{\mathsf{P}}(\pi, U) = \sum_{z \in \mathcal{Z}} \sum_{i,j \in \mathcal{X}} \pi(i) \mathsf{P}(i,j) \mathsf{O}(j,z) \mathbb{I}_U(\Pi(\pi, z)),$$

where $\Pi(\pi, z)$ is as defined in (4.24) and $\mathbb{I}_U(\pi)$ is the indicator function for the set U, defined as

$$\mathbb{I}_U(\pi) = \begin{cases} 1 & \text{if } \pi \in U; \\ 0 & \text{otherwise.} \end{cases}$$

The state π_t of the new chain is now a vector in $\mathbb{S}^n$ and can be interpreted as an *internal state* of the agent, tracking the state X_t of the original chain. Since the i^{th} coordinate of π_t describes the *belief* that the underlying state of the chain is $X_t = i$, it is common to refer to π_t as the *belief-state* of *belief-vector* at time t and that designation shall also be adopted henceforth.

4.7.2 Geometric ergodicity in associated Markov chains

We have just established that a partially observable Markov chain $(\mathcal{X}, \mathcal{Z}, \mathsf{P}, \mathsf{O})$ can be recast as a fully observable Markov chain $(\mathbb{S}^n, \bar{\mathsf{P}})$ with an infinite state-space. We have also emphasized the fact that $\mathbb{S}^n$ is a subset of $\mathbb{R}^n$ and, consequently, it is a compact, separable and metrizable topological subspace of $\mathbb{R}^n$ with the usual topology.

We proceed the study of partially observable stochastic chains by defining the properties required from the chain $(\mathcal{X}, \mathcal{Z}, \mathsf{P}, \mathsf{O})$ to ensure the stability of the trajectories of the chain $(\mathbb{S}^n, \bar{\mathsf{P}})$. In particular, we aim at providing conditions on P and/or O ensuring that the fully observable, infinite state-space chain $(\mathbb{S}^n, \bar{\mathsf{P}})$ is *geometrically ergodic*. This result will prove of great use when addressing partially observable Markov decision processes, in the next subsection. In fact, this class of Markov processes can also be recast as a fully observable, infinite state-space counterpart to which the methods of Section 4.5 can be applied.

The new results derived herein constitute the second fundamental contribution of this chapter and provide all the necessary means to conclude the analysis of convergence of the reinforcement learning methods in partially observable scenarios.

* * *

In this subsection we establish the geometric (in fact, uniform) ergodicity of the chain $(\mathbb{S}^n, \bar{\mathsf{P}})$ by means of three distinct lemmas, in which we separately identify the conditions under which

- $(\mathbb{S}^n, \bar{\mathsf{P}})$ is ψ-irreducible;
- $(\mathbb{S}^n, \bar{\mathsf{P}})$ is aperiodic;
- $\mathbb{S}^n$ is petite.[10]

Once these facts are properly established, geometric ergodicity of the chain $(\mathbb{S}^n, \bar{\mathsf{P}})$ follows trivially from Theorems B.4.2 and B.8.4 in Appendix B.

The three facts above are established formally in the following results.

Lemma 4.7.1. *Let $(\mathcal{X}, \mathcal{Z}, \mathsf{P}, \mathsf{O})$ be a partially observable Markov chain. Then, if $(\mathcal{X}, \mathsf{P})$ is irreducible and there is an observation $z \in \mathcal{Z}$ and a state $i^* \in \mathcal{X}$ such that, for all $i \in \mathcal{X}$,*

$$\mathsf{O}(i, z) = \begin{cases} 1 & \text{if } i = i^*; \\ 0 & \text{otherwise,} \end{cases}$$

the Markov chain $(\mathbb{S}^n, \bar{\mathsf{P}})$ is ψ-irreducible.

Proof See Appendix F. □

Lemma 4.7.2. *Let $(\mathcal{X}, \mathcal{Z}, \mathsf{P}, \mathsf{O})$ be a partially observable Markov chain verifying the conditions of Lemma 4.7.1. Then, if $(\mathcal{X}, \mathsf{P})$ is aperiodic, so is $(\mathbb{S}^n, \bar{\mathsf{P}})$.*

Proof See Appendix F. □

Lemma 4.7.3. *Let $(\mathcal{X}, \mathcal{Z}, \mathsf{P}, \mathsf{O})$ be a partially observable Markov chain. Then, if the conditions of Lemmas 4.7.1 and 4.7.2 are met, the state-space $\mathbb{S}^n$ of the chain $(\mathbb{S}^n, \bar{\mathsf{P}})$ is petite.*

Proof See Appendix F. □

Now, Lemma 4.7.1 through 4.7.3 together with Theorems B.4.2 and B.8.4 lead to the following concluding result.

[10] Formal definitions of ψ-irreducibility, aperiodicity and petiteness in Markov chains can be found in Appendix B.

Theorem 4.7.4. *Let $(\mathcal{X}, \mathcal{Z}, \mathsf{P}, \mathsf{O})$ be a partially observable Markov chain. Suppose that the chain $(\mathcal{X}, \mathsf{P})$ is irreducible and aperiodic (and therefore ergodic). Suppose, furthermore, that there is an observation $z \in \mathcal{Z}$ and a state $i^* \in \mathcal{X}$ such that, for all $i \in \mathcal{X}$,*

$$\mathsf{O}(i, z) = \begin{cases} 1 & \text{if } i = i^*; \\ 0 & \text{otherwise.} \end{cases}$$

Then, the Markov chain $(\mathbb{S}^n, \bar{\mathsf{P}})$ is geometrically ergodic.

Remark: We notice that the observability condition in this theorem is less stringent than it may appear. It simply requires that there is at least *one* identifiable state, *i.e.*, one state that the agent can unambiguously identify. This is often the case in many useful applications. For example, in robotic navigation tasks the *goal* of the robot is usually an identifiable state. ◇

4.7.3 POMDPs and associated MDPs

So far, we have addressed partial observability in Markov chains. Partially observable MDPs arise naturally from partially observable Markov chains by considering a control process and a reward function.

Therefore, a *partially observable Markov decision process* (POMDP) is a tuple $(\mathcal{X}, \mathcal{A}, \mathcal{Z}, \mathsf{P}, \mathsf{O}, r, \gamma)$. The parameters $\mathcal{X}$, $\mathcal{A}$, P, r and γ are as defined in Chapter 2 and define the "underlying MDP". Grossly stated, a POMDP is an MDP with partial observability as featured by the pair $(\mathcal{Z}, \mathsf{O})$. The set $\mathcal{Z}$ is the finite observation-space and O is the observation probability function: for each action $a \in \mathcal{A}$, $\mathsf{O}_a(i, z)$ represents the observation probability

$$\mathsf{O}_a(i, z) = \mathbb{P}\left[Z_{t+1} = z \mid X_{t+1} = i, A_t = a\right].$$

We emphasize the fact that the observations Z_{t+1} depend on the action A_t *in the previous time instant*. This dependence simply adds greater generality to the POMDP model. It allows, for example, to include in the POMDP model perceptual effects arising from changes in orientation of a robot, active sensing, etc.

From a POMDP we can derive a fully observable MDP with infinite state-space by reasoning as in Subsection 4.7.1. Consider the belief state π_t, each component of which indicates the probability of being in each state $i \in \mathcal{X}$ at time t. This means that the belief π_t is a vector in $\mathbb{R}^n$ ($n = |\mathcal{X}|$) with i^{th} component given by

$$\pi_t(i) = \mathbb{P}\left[X_t = i \mid \mathcal{F}_t\right].$$

If the agent takes an action $A_t = a$, the updated belief at time $t + 1$ given the observation $Z_{t+1} = z$ is given by an expression equivalent to (4.24). Explicitly, the

updated belief state π_{t+1} is

$$\pi_{t+1}(j) = \Pi_a(\pi_t, z) = \frac{\sum_{i\in\mathcal{X}} \pi_t(i)\mathsf{P}_a(i,j)\mathsf{O}_a(j,z)}{\sum_{i,k\in\mathcal{X}} \pi_t(i)\mathsf{P}_a(i,k)\mathsf{O}_a(k,j)}. \tag{4.25}$$

Apart from the index a, (4.25) and (4.24) are essentially equivalent.

For this associated infinite-state MDP, we define the transition probability kernels $\bar{\mathsf{P}}_a$ as

$$\bar{\mathsf{P}}_a(\pi, U) = \sum_{z\in\mathcal{Z}} \sum_{i,j\in\mathcal{X}} \pi(i)\mathsf{P}_a(i,j)\mathsf{O}_a(j,z)\mathbb{I}_U(\Pi_a(\pi,z)),$$

where $\Pi_a(\pi, z)$ is defined in (4.25). Define the reward $\bar{r}$ associated with a transition (π, a, π') as

$$\bar{r}(\pi, a, \pi') = \sum_{i,j\in\mathcal{X}} \pi(i)\mathsf{P}_a(i,j)r(i,a,j).$$

With $\bar{\mathsf{P}}$ and $\bar{r}$, we obtain a fully observable MDP $(\mathbb{S}^n, \mathcal{A}, \bar{\mathsf{P}}, \bar{r}, \gamma)$ associated with the POMDP $(\mathcal{X}, \mathcal{A}, \mathcal{Z}, \mathsf{P}, \mathsf{O}, r, \gamma)$, where $\mathbb{S}^n$ denotes once again the $n-1$-dimensional probability simplex.

> Remark: Notice that the reward as defined above corresponds to the expected immediate reward for being in each state i with probability $\pi(i)$ and taking action a. It does not depend on the belief π' after the transition. This means that the rewards in the associated MDP $(\mathbb{S}^n, \mathcal{A}, \bar{\mathsf{P}}, \bar{r}, \gamma)$ do not depend on the observations Z_t, which is true. ◇

In the new MDP $(\mathbb{S}^n, \mathcal{A}, \bar{\mathsf{P}}, \bar{r}, \gamma)$, it is straightforward to define a deterministic policy δ as a mapping

$$\delta : \mathbb{S}^n \longrightarrow \mathcal{A}$$

that determines an action for each belief-state in $\mathbb{S}^n$. A stochastic policy is defined analogously, by assigning to each belief-state a probability distribution over $\mathcal{A}$.

As in Chapter 2, given a policy δ, the corresponding value-function V^δ is

$$V^\delta(\pi) = \mathbb{E}_\delta\left[\bar{r}(\pi, a, \pi') + \gamma V^\delta(\pi')\right].$$

The optimal value function V^* verifies the Bellman equation

$$V^*(\pi) = \max_{a\in\mathcal{A}} \mathbb{E}\left[\bar{r}(\pi, a, \pi') + \gamma V^\delta(\pi')\right],$$

and the optimal Q-function verifies

$$Q^*(\pi, a) = \mathbb{E}\left[\bar{r}(\pi, a, \pi') + \gamma \max_{b\in\mathcal{A}} Q^*(\pi', b)\right].$$

More intuitive and well-known expressions for these functions can readily be

obtained by replacing $\bar{\mathsf{P}}$ and $\bar{r}$ by the corresponding definitions, yielding

$$V^{\delta}(\pi) = \sum_{i,j\in\mathcal{X}} \sum_{a\in\mathcal{A}} \delta(\pi,a)\pi(i)\mathsf{P}_a(i,j) \left[r(i,a,j) + \gamma \sum_{z\in\mathcal{Z}} \mathsf{O}_a(j,z) V^{\delta}(\Pi_a(\pi,z)) \right];$$

$$V^{*}(\pi) = \max_{a\in\mathcal{A}} \sum_{i,j\in\mathcal{X}} \pi(i)\mathsf{P}_a(i,j) \left[r(i,a,j) + \gamma \sum_{z\in\mathcal{Z}} \mathsf{O}_a(j,z) V^{*}(\Pi_a(\pi,z)) \right];$$

$$Q^{*}(\pi,a) = \sum_{i,j\in\mathcal{X}} \pi(i)\mathsf{P}_a(i,j) \left[r(i,a,j) + \gamma \sum_{z\in\mathcal{Z}} \mathsf{O}_a(j,z) \max_{b\in\mathcal{A}} Q^{*}(\Pi_a(\pi,z),b) \right].$$

With the definition of V^{δ}, V^{*} and Q^{*} we complete the presentation of the theoretical framework needed to apply approximate Q-learning and approximate SARSA to POMDPs. In the remainder of the section, we review some bibliographical references on POMDPs and, in the next section, we conclude the chapter with an illustrative example.

4.7.4 Related work

There are numerous works in the literature surveying partially observable Markov decision problems. Cassandra [52] presents a comprehensive introduction to this class of problems. More developed approaches can be found in [2, 51]. Murphy [209] surveys several algorithms designed to handle partially observable scenarios; Hasinoff [120] reviews model-free reinforcement learning methods that address partial observability and Hauskrecht [121] analyzes several value-based methods for problems with partial observability.

The methods specifically developed to address decision problems in partially observable scenarios can grossly be divided into three main classes, namely:

(A) *Exact methods* that seek to determine the exact optimal policy or a corresponding value-function from which the optimal policy can be determined. Exact methods include several dynamic-programming-based algorithms (such as the witness algorithm [163] or incremental pruning [54]) and others [37, 346].

(B) *Methods for particular problems.* The methods in this class are designed to address specific problems, exhibiting particular properties. A few examples can be found in [19, 109, 187, 264, 271].

(C) *Approximate methods* that provide sub-optimal solutions. This is, by far, the widest and most diverse of all three classes and includes numerous algorithms supported by either formally justified approximations [15, 18, 30, 119, 122, 199, 219, 231, 284] or by more heuristic reasoning [48, 93, 195, 262, 309, 311].[11]

[11] The list of bibliographic references provided does not intend to be exhaustive in any way, but merely to convey an idea of the huge amount of work addressing partial observability in reinforcement learning problems. Many of the references provided herein contain themselves further references for the interested reader.

In most of these methods, the agent maintains some internal state that is mapped to a corresponding action according to some policy. This internal state provides the agent with some record of the past history of actions/observations that allows to "overcome" the problem of partial observability. There are also some works in the literature which simply disregard partial observability and treat the observations as actual state measurements [132, 173, 237]. This approach, however simple it may be, can lead to arbitrarily poor policies, as argued in [278].[12]

Clearly, of the three classes of methods described above, the methods in class C are those more widely applicable. Approximate Q-learning and approximate SARSA (seen as POMDP solutions methods) naturally fall within this category.

Our methods also exhibit several other interesting properties as POMDP solution methods. First of all, both our methods are *belief-based*, since both methods determine policies as mappings from belief-vectors to actions. In robotic terms, a belief vector provides an estimate on the position of the robot in the environment, taking into account the uncertainty arising from the observations. Therefore, computing/updating a belief-vector corresponds to *localizing* the robot and is tantamount to the so-called *Markov localization* [87, 88, 89]. Therefore, beliefs arise naturally in robotic applications and have been successfully applied in several real-world navigation tasks (see references above).

On the other hand, the value function V^* for the MDP $(\mathbb{S}^n, \mathcal{A}, \bar{\mathsf{P}}, \bar{r}, \gamma)$ is known to have specific properties. In particular, it is convex and, in some situations, piecewise linear.[13] These properties can be taken into consideration when choosing the basis functions in our algorithms. For example, we can use the projections of the belief-vector in each of its components as basis function, yielding an approximation to the optimal value function that will be necessarily piecewise linear. That is the approach followed in the next chapter.

4.8 An illustrative example

We return to the problem introduced in Chapter 2 and consider the indoor environment in Figure 4.11. A mobile robot must navigate to the signaled goal room. At each time instant, the robot can move in any of the four possible directions: North (N), South (S), East (E) and West (W). An internal sensor allows the robot to determine when the goal room is reached. The robot has no other sensors available.

This navigation problem can be modeled as a POMDP $(\mathcal{X}, \mathcal{A}, \mathcal{Z}, \mathsf{P}, \mathsf{O}, r, \gamma)$, where

- $\mathcal{X}$, $\mathcal{A}$, P and r are as in the example of Chapter 3;
- $\mathcal{Z} = \{\emptyset, Goal\}$;

[12] A more detailed argument on the limitations of reactive/memoryless policies can be found in [164].

[13] In a general POMDP, a policy is called *finitely transient* if the corresponding value function is piecewise linear. We abusively say that a POMDP is finitely transient if the optimal policy is finitely transient. V^* is piecewise linear iff the POMDP is finitely transient [52].

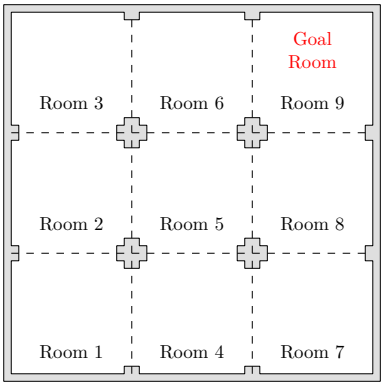


Figure 4.11: Example of an indoor environment.

- O represents the observation probability function. For $i = 1, \ldots, 8$, $\mathsf{O}(i, \emptyset) = 1$. For $i = 9$, $\mathsf{O}(i, Goal) = 1$;
- $\gamma = 0.95$.

Whenever the agent reaches the goal state, its position is randomly reset to any of the other 8 states, independently of the agent's action.

Notice that, since the robot is only able to observe the goal location, the set of possible observations has only two elements: $\emptyset$, corresponding to the null observation, and *Goal*, the observation corresponding to the goal state. The goal state clearly verifies the conditions of Lemma 4.7.1 and the algorithms in Section 4.5 can be applied with guaranteed convergence.

We applied approximate Q-learning and approximate SARSA to the associated MDP, defined as the tuple $(\mathbb{S}^n, \mathcal{A}, \bar{\mathsf{P}}, \bar{r}, \gamma)$. We used a simple set of 9×4 functions, each associated with one component of the belief-vector and one action, *i.e.*,

$$\Xi = \{\xi_{1,N}, \ldots, \xi_{9,N}, \xi_{1,S}, \ldots, \xi_{9,W}\}$$

with each $\xi_{i,a}$ given by

$$\xi_{i,a}(\pi, b) = \pi(i)\mathbb{I}_a(b).$$

This corresponds to a total of 9×4 scalar parameters $\theta(i, a)$, one for each pair (i, a). In the fully observable case, the learnt parameters corresponded to the actual optimal Q-values, one parameter for each state-action pair (i, a). Using the set of basic functions described above, the number of learnt parameters is the same as in the fully observable case: for each state $i \in \mathcal{X}$ and action $a \in \mathcal{A}$ there is one function $\xi_{i,a} \in \Xi$ and a corresponding parameter $\theta(i, a)$.

As in the fully observable case, the robot was allowed to explore/learn during $1,000$ time units and the obtained policy was then evaluated for 100 time units. We use a similar learning time in the fully and partially observable scenarios to allow a better comparison of our results in the presence of partial observability with those obtained with full observability. During learning we used Boltzmann exploration in all experiments.

Figure 4.12 represents the total reward obtained by the robot during learning. The solid blue line represents approximate Q-learning and the dashed red line rep-

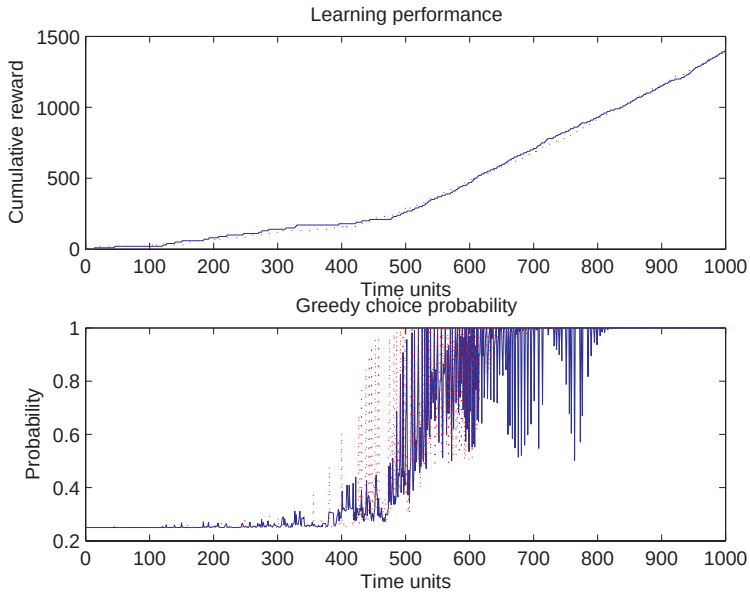


Figure 4.12: Cumulative reward and greedy choice probability during the 1,000-time-units learning period, using a set of 36 basis functions. The solid blue line corresponds to approximate Q-learning and the dashed red line corresponds to approximate SARSA.

Table 4.3: Comparative results for approximate Q-learning, approximate SARSA and incremental pruning (labeled as "Optimal") after the learning period is complete. We used a set of 36 belief-based basis functions. We present the average total discounted reward and standard deviation obtained over 2,000 independent Monte-Carlo trials.

Method	**Total Disc. Reward**
Approx. Q-learning	41.680 ± 4.941
Approx. SARSA	41.639 ± 4.846
Optimal	41.651 ± 4.982

resents approximate SARSA. Notice that both learning algorithms present a similar learning behavior. Notice that no significant difference between the methods is perceivable.

We also tested each of the learnt policies in the environment. We ran each learnt policy for 100 time units and determined the total discounted reward obtained in each case. We have run 2,000 independent Monte-Carlo trials and present in Table 4.3 the average and standard deviation obtained using each of the methods. For comparison, we also present the optimal results, obtained using the incremental pruning algorithm [54]. Notice that the two learning methods present a similar performance, as expected, since both converge to the same approximation. The remarkable thing is that *both methods exhibit optimal performance* in the given environment.

Approximate Q-learning when used with the set of functions described in the

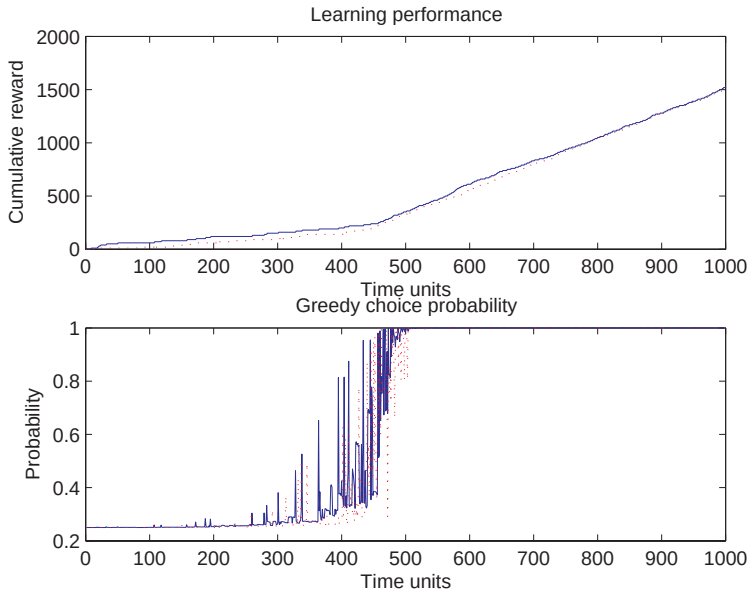


Figure 4.13: Cumulative reward and greedy choice probability during the 1,000-time-units learning period, using a set of 16 basis functions. The solid blue line corresponds to approximate Q-learning and the dashed red line corresponds to approximate SARSA.

Table 4.4: Comparative results for approximate Q-learning and approximate SARSA using a kernel-based approach, after the learning period is complete. We used a set of 16 kernel-based basis functions. We present the average total discounted reward and standard deviation obtained over 2,000 independent Monte-Carlo trials.

Method	Total Disc. Reward
Approx. Q-learning	41.513 ± 4.866
Approx. SARSA	41.580 ± 4.945

previous test corresponds to the *linear-Q* algorithm introduced in [169]. In this work, the author reports similar results to those in Table 4.3 in a similar problem (the 4×4 grid problem).

To further explore the applicability of our proposed methods, we ran a second set of tests using a kernel-based approximation with only 4×4 basis functions. The results are presented in Figure 4.13 and Table 4.4.

Notice that, by comparing the results in Table 4.4 with the optimal results in Table 4.3, we conclude that the algorithms also attain optimal performance using this second set of basis functions. However, instead of learning $9 \times 4 = 36$ parameters, we are able to use a more compact representation and learn only $4 \times 4 = 16$ parameters. This representation is even more compact that the one used in the optimal, fully observable MDP in Chapter 3, that also learnt 36 parameters (the Q-values).

We conclude by summarizing the main conclusions from this example. First of all, using a very simple belief-based approximation we are able to reproduce the performance obtained with an exact method (incremental pruning). Furthermore,

we are able to still attain optimal performance even when using a more restricted set of basis functions. Finally, our experimental results agree with those provided in [169]. The added value in our approach arises from the established convergence guarantees of our method, whose application in [169] was only empirically motivated.

4.9 Concluding remarks

To conclude this chapter, we summarize the main ideas presented so far. We discuss the applicability of the reinforcement learning framework introduced in the two previous chapters to single-robot navigation problems. We also hint on how this framework can be extended to multi-agent settings, a topic to be developed in the second part of this thesis.

4.9.1 Summary

In Chapters 2 and 3, we described Markov decision processes as a suitable framework to address single-agent topological navigation problems. We discussed several known methods from the literature that *learn* the optimal control sequence in a given environment. Such control sequence is generated by a *policy* that can, in turn, be computed from a set of real parameters, the *Q-values*.

However, in the two referred chapters, two essential assumptions were made:

- The state-space $\mathcal{X}$ was finite;
- At each time instant, the agent (robot) was able to perceive its state X_t with no uncertainty.

If the use of topological maps provides an argument in favor of the first assumption, the second assumption is seldom verified in real application scenarios.

In this chapter we introduced generalizations of the methods in Chapter 3 specifically designed to alleviate the previous assumptions. In particular, we extended Q-learning and SARSA, combining both methods with linear function approximation. The new methods thus obtained, dubbed as *approximate Q-learning* and *approximate SARSA*, can be applied in problems where the state-space is infinite. We established convergence w.p.1 of both methods and provided an interpretation of the obtained approximation as the fixed-point of a Bellman-like operator.

We then addressed the problem of *partial observability*. A decision problem is said to have partial observability if it violates the second of the two assumptions above. We described a method to reformulate this class of problems as fully observable problems with infinite state-spaces and proposed the application of approximate Q-learning and approximate SARSA to such equivalent problems.

We concluded the chapter by illustrating the use of the proposed methods in a familiar topological navigation example.

4.9.2 Discussion

In this chapter, we presented several reinforcement learning algorithms that can be used either to evaluate a policy δ or to determine the optimal Q-values. These meth-

ods differ from those in the previous chapter in their capability to cope with problems with infinite state-spaces. Under suitable conditions, identified in Theorems 4.5.2 and 4.5.3, all the algorithms converge asymptotically to an approximation of the desired function. Similarly to the algorithms in the previous chapter, the algorithms presented herein also ensure convergence in behavior to the optimal policy w.r.t. the learnt approximation, as long as the learning policy has the GLIE property.

In all this presentation, we postponed the discussion of several important issues to these concluding remarks. We address these issues next.

Quality of approximation and error bounds

In Section 4.5 we introduced approximate Q-learning and approximate SARSA and established the convergence of both methods to a fixed-point of a Bellman-like operator. This operator resulted from the composition of the Bellman operator $\mathbf{H}$ (defined in (4.3)) and the orthogonal projection $\mathcal{P}_{\mathcal{Q}}$ on a finite dimensional hyperplane $\mathcal{Q}$. This approach is similar to the one pursued by Tsitsiklis and Van Roy [321] with the main difference that the method in the referred work is a policy evaluation method, while the methods introduced in this chapter allow policy optimization (by approximating the optimal function Q^*). However, unlike [321], we are unable to provide error bounds for the obtained approximation.

Our methods converge to the function $Q(\theta^*)$ verifying the fixed-point recursion in (4.16). This means that, if the set of functions Ξ chosen is such that $Q^* \in \mathcal{Q}$, both methods will actually converge to Q^*. However, as the distance between Q^* and $\mathcal{Q}$ increases, the performance of the methods will (expectedly) degrade. This means that the functions in the chosen linear space $\mathcal{Q}$ provide poor approximations of the desired function and there are no practical guarantees on its usefulness (in terms of the corresponding greedy policy).

The algorithm in [321] suffers from a similar disadvantage. However, the error bounds derived in that work are reassuring in that they state that the performance of approximate TD(0) "gracefully" degrades as the distance between V^δ and $\mathcal{V}$ increases. For our methods we have no such error bounds available and, as such, cannot guarantee such "graceful" degradation in performance. Ensuring similar bounds as those in [321] would require significant changes in the method and would impose more stringent restrictions on the possible approximation architectures. We present in Appendix E an alternative method for which error bounds can be derived.

In any case, when using function approximation, the functions should be chosen so as to include all available information regarding the true function to be estimated. Van Roy [326] briefly addresses the problem of how to choose "adequate" basis functions. The interpolation method by Szepesvári and Smart [302] suggests a natural choice of basis functions that is guaranteed to converge to the optimal Q-function as the number of sample points increases. We refer to the concluding remarks in Chapter 10 for further references on this topic.

Projections and norms

In Theorem 4.5.2, several conditions are listed that ensure the convergence of the methods addressed therein. We briefly review those conditions:

- We require the sampled Markov chain to be *geometrically ergodic*, a condition that simply ensures that the average obtained from the sample trajectories is close to the expectation w.r.t. the corresponding stationary measure;

- We require the step-size sequence $\{\alpha_t\}$ to verify the customary summability conditions. This is a standard condition from stochastic approximation that ensures that the temporal differences used in the updates are properly averaged over time;

- We require the functions in Ξ to be bounded and linearly independent. In particular, we require that $\sum_i |\xi_i(x,a)| \leq 1$ for all (x,a).

We now focus on the last condition and analyze its implications.

We start by noticing that, if the set of functions Ξ is not linearly independent, we can simply remove the linearly dependent functions. Since the linear space spanned by the set of functions remains unchanged when the linearly dependent functions are removed, the generalization capabilities of the algorithm do not improve by adding linearly dependent functions.

On the other hand, the fact that the functions ξ_i are bounded together with the fact that the rewards r are bounded, ensures that the iterations θ_t of the algorithm remain bounded. This is a central condition when analyzing the trajectories of the algorithm by means of the associated ODE.

Finally, w.r.t. the condition $\sum_i |\xi_i(x,a)| \leq 1$, several observations are in order. First of all, Tsitsiklis and Van Roy [321] make use of the fact that the TD operator $\mathbf{T}^\delta$ is a contraction in the 2-norm to establish the convergence of approximate TD(0). In fact, the orthogonal projection $\mathcal{P}_\mathcal{V}$ is non-expansive in this norm (since it is naturally defined in the corresponding inner-product space) and the combined operator $\mathcal{P}_\mathcal{V}\mathbf{T}^\delta$ is a contraction in the 2-norm. This single fact is fundamental in guaranteeing the existence of the corresponding fixed-point, the convergence of the algorithm and the referred error bounds.

The fact that the operator $\mathbf{H}$ used in Q-learning is a contraction in the sup-norm makes it hard to define a "projection" operator that is non-expansive in the sup-norm and projects any function to $\mathcal{Q}$.[14] To avoid this difficulty, we impose the condition that $\sum_i |\xi_i(x,a)| \leq 1$ for all x, a. When studying the associated ODE $\dot{\theta}_t = h(\theta_t)$ we are able to decompose h into two functions h_1 and h_2 such that $h(\theta) = h_1(\theta) - h_2(\theta)$. Since one of these functions is a contraction and the other is a non-expansion, we are able to establish convergence w.p.1 of approximate Q-learning without requiring the combined operator $\mathcal{P}_\mathcal{Q}\mathbf{H}$ to be contractive.

Finally, the condition above is not too restrictive, in that a simple normalization can guarantee its verification. Furthermore, several known approximation strategies trivially verify such condition (*e.g.*, discretization [266], "soft"-discretization [279] or convex interpolation [302]).

[14] Szepesvári and Smart [302] define a projection-like operator that is shown to be non-expansive in the sup-norm. In Appendix E we use a similar projection-like operator to establish the convergence and error bounds of a sample-based Q-learning algorithm. The proof of convergence of both these methods immediately follows from the arguments above.

Applicability to partially observable scenarios

In Section 4.7, we applied approximate Q-learning and approximate SARSA to partially observable Markov decision processes by reformulating finite POMDPs as a fully observable MDPs with infinite state-space. Tracking the state of an associated MDP reduces to tracking the belief-state π_t that can easily be computed from the original POMDP parameters as in (4.25).

Notice that tracking the belief-state, *i.e.*, maintaining an updated belief-state at each time instant $t \in \mathcal{T}$, implies that the agent must have a model of the POMDP. In particular, it requires the knowledge of the parameters P and O. This is less general than the approach adopted in most reinforcement learning methods, where no model of the system is assumed.

In our mobile robot navigation problem, this corresponds to stating that the robot has a model (at least a probabilistic model) of its dynamic behavior and sensors, but not on the mission. This is a reasonable assumption in many cases, and allows us to deal with partial observability. Also, as will be seen in the next chapter, our methods are able to tackle complex navigation problems, with hundreds of states, problems which are computationally untractable with exact methods.

Finally, notice that the overall conditions required to ensure convergence of the methods in this chapter in partially observable scenarios are extremely similar to the requirements for convergence in fully observable scenarios. In fact, as seen in Chapter 3, convergence of Q-learning requires infinite visits to each state-action pair, this condition being verified if the chain is ergodic. Convergence in partially observable scenarios simply requires one extra condition: that at least one state is identifiable. In navigation problems (and in several other benchmark problems from the POMDP literature) this condition is always verified.

* * *

With the methods presented in this chapter, we conclude the theoretical study of single-agent topological navigation problems, from a learning perspective. In the next chapter we conclude this first part of the thesis by applying this methodology to several different problems of increasing complexity.

In the second part of the thesis, we extend the study developed so far to multi-agent systems. We introduce Markov games (or stochastic games) as a multi-agent extension of Markov decision processes. The extreme similarities between the two frameworks are evident: a Markov decision process can be seen as a Markov game with a single player.

Our approach to multi-agent problems is similar to the one pursued in this first part: we describe several known reinforcement learning algorithms for Markov games and extend these algorithms to problems with infinite state-spaces. We then address the problem of partial observability, far more complex when there are multiple decision-makers than in single-agent problems.

Chapter 5

Results on Single Robot Navigation

We introduced in Chapter 4 the necessary methods to address general single-robot topological navigation problems. In this chapter we apply this framework and methods to several navigation problems.

We describe the general robot model used and the POMDP framework arising from this model. We present several complex navigation problems and apply our reinforcement learning algorithms to these problems. We verifies that the necessary convergence conditions are met in each of the considered problems and evaluate the performance of our methods in such complex scenarios.

5.1 Introductory remarks

In this chapter, we propose the application of the methodology developed in the previous chapters to more elaborate problems. In the previous chapter, we already evaluated the performance of these methods on an extremely small toy problem.

In that problem, we dealt with a POMDP with only 9 states, 2 observations and 4 possible actions. This simple example helped us to evaluate the performance of the proposed methodology and compare it with the performance of the optimal solution. However, in that problem, the optimal policy was very simple and easily implemented even with few observations. As such, it provides us with little information on how useful the methods developed so far can be in actual topological navigation problems.

On the other hand, determining optimal solutions is computationally too expensive for large POMDPs (with more than a few states) and this prevents us from conducting a similar comparison in larger, more complex problems. In other words, the performance of approximate methods (such as those proposed in the previous chapter) cannot be compared with that of the actual optimal policy when considering large environments.

The set of problems addressed in this chapter (some of which have already been addressed in other works), constitute benchmarks in the POMDP literature [51] and seek to experimentally assess the usefulness of this model and corresponding solution methods in relatively large-sized problems. Although no comparison with the optimal policy is possible, we can use the fully observable solution as a performance upper bound, knowing however that this bound will seldom be tight.[1]

We consider a fully autonomous mobile robot navigating in several large and complex environments and apply the algorithms described in the previous chapter to "control" the decision-making process. The main goal of the chapter is to illustrate the successful application of reinforcement learning methods in mobile robot navigation with partial observability.

It is important to refer once again that, since the optimal solution for these problems is unavailable, we compare the performance of our methods with that of an omniscient robot—one that is able to observe the actual state of the process at all times. The policy implemented by such omniscient robot can readily be obtained by any of the methods described in Chapters 2 and 3. Clearly, if our results are close to those of the omniscient robot, we know that they must also be close to optimal.

In this chapter, we start by describing the applications considered. We describe the environments, the robot and derive the POMDP model describing each navigation problem. We then describe the experiments conducted and the results obtained with approximate Q-learning and approximate SARSA using a simple, linear approximation. We briefly comment on these results and compare the performance of our methods with that of the omniscient robot.

5.2 The experimental setup

In this section, we describe the scenarios, the mobile robot and the obtained POMDP model for the different experiments.

[1] The optimal policy in the presence of partial observability can, in general, be much worse than the optimal fully observable policy, as shown in [278].

5.2.1 The scenarios

We consider six possible scenarios of application, four of which have been used as benchmarks in several POMDP works [51]. Specifications for these environments can also be found on-line at Cassandra's POMDP page.[2] In this page it is also possible to find several references to other works where the results of several methods in these benchmark problems are reported and compared.

All scenarios are simplified representations of existing environments, namely

- ISR is a coarse representation of the 7^{th} floor of the ISR building, at IST;
- MIT is Cassandra's simplified model of the 8^{th} floor of MIT's Computer Science building;
- PENTAGON was inspired by the Pentagon building, with an inner and outer hallway;
- CIT is Cassandra's coarse model of the 4^{th} floor of the CIT building, Brown University;
- SUNY represents the 1^{st} floor of the Computer Science building, SUNY at Stony Brook;
- CMU represents the 7^{th} floor of the Computer Science buiding, at Carnegie-Mellon University.

The several scenarios are sketched in Figures 5.1 through 5.6. Each of the cells in the figures is represented as a different node in the corresponding topological map. The number of nodes in the topological representation of the six environments is summarized in the second column of Table 5.1. The cells in dark represent rooms and the remaining cells represent corridors. The passages between rooms and corridors (*i.e.*, doorways) are narrower than those between adjacent corridor cells. The cells marked with a bold border and an arrow correspond to initial/goal states for the robot, as detailed in the continuation. In particular, blue states correspond to initial states while red states correspond to goal states.

We consider that, in each cell, the robot can be in one of four possible orientations, according to the four compass directions: N, S, E and W. The number of states for the corresponding POMDP models is summarized in the third column of Table 5.1.

5.2.2 The robot

We address the problem of a mobile robot moving in the environments depicted in Figures 5.1 through 5.6. In each of the environments, the robot must reach a target state that it is able to unambiguously identify upon reaching it. In this goal state, the orientation of the robot is irrelevant.

For the purposes of our experiments we admit that some low-level control ensures motor control, obstacle avoidance, etc., so that we can focus on the navigation

[2] http://www.cs.brown.edu/research/ai/pomdp/

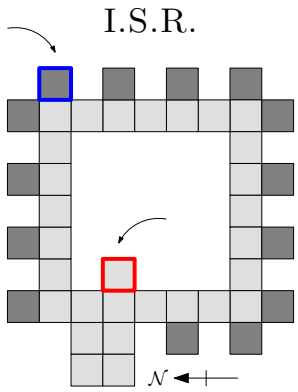


Figure 5.1: Topological representation of the ISR environment with 43 nodes.

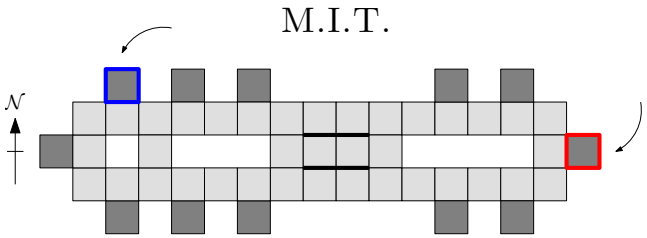


Figure 5.2: Topological representation of Cassandra's MIT environment with 49 nodes.

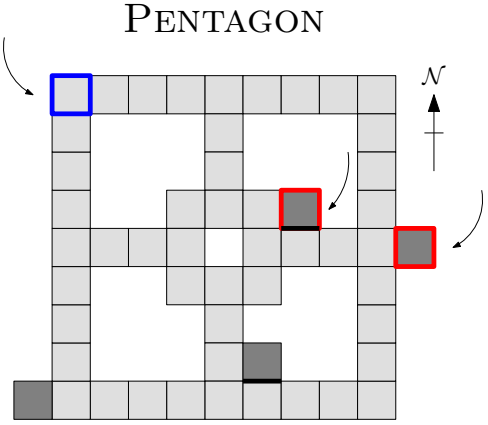


Figure 5.3: Topological representation of Cassandra's PENTAGON environment with 52 nodes.

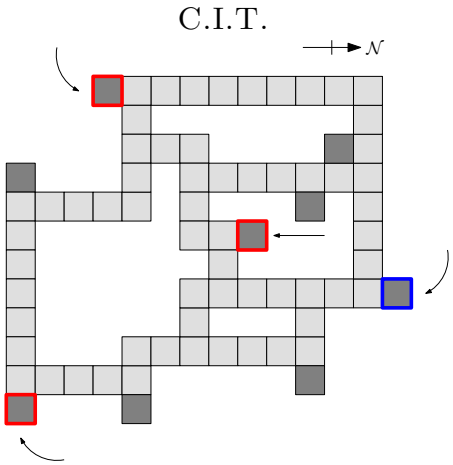


Figure 5.4: Topological representation of Cassandra's CIT environment with 70 nodes.

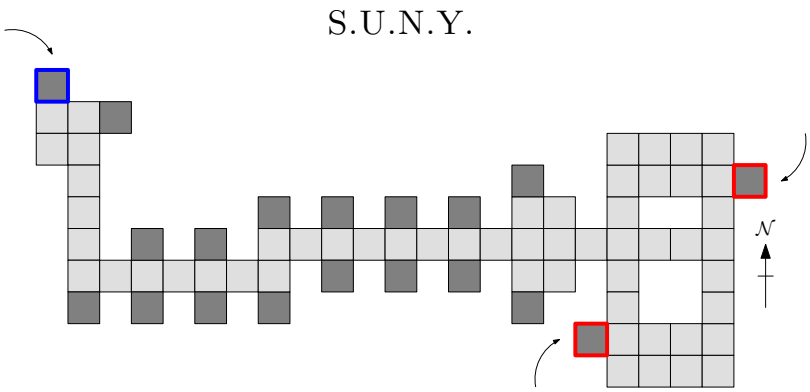


Figure 5.5: Topological representation of Cassandra's SUNY environment with 74 nodes.

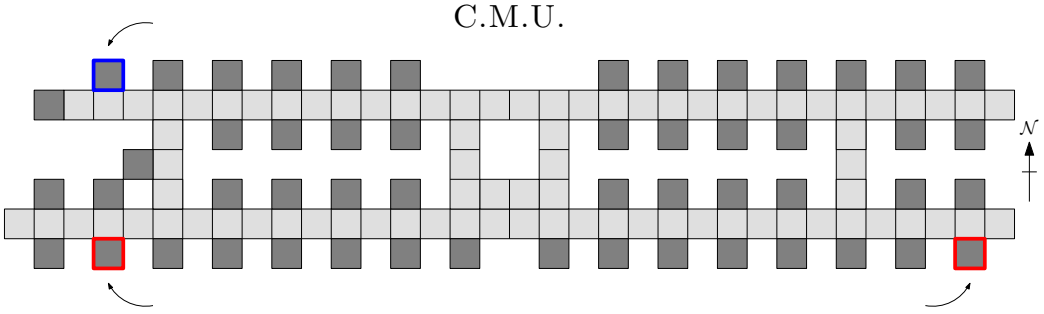


Figure 5.6: Topological representation of the CMU environment with 133 nodes.

Table 5.1: Nodes in the topological representations of the ISR, MIT, PENTAGON, CIT, SUNY and CMU environments.

Environment	# Nodes	# States
ISR	43	169
MIT	49	193
PENTAGON	52	205
CIT	70	277
SUNY	74	293
CMU	133	529

Table 5.2: Outcome probabilities for the different actions.

Action	Outcome		
Turn left	No action 0.05	Turn left 0.90	Turn back 0.05
Turn right	No action 0.05	Turn right 0.90	Turn back 0.05
Move forward		No action 0.10	Move forward 0.90

problem from a higher level of abstraction, as intended.[3] At each time instant, the robot has 3 available actions: turn left, turn right and move forward.[4] The outcome of each of these actions is not deterministic, in that each action can lead to an "unexpected" effect. For example if the robot takes the action "Turn Left", it may happen that the orientation does not change or that, instead, the robot turns back. This uncertainty is summarized in Table 5.2, where each action is described in terms of the possible outcomes. In the case above, the action "Turn Left" succeeds with a probability of 0.9 and has an unexpected outcome with a probability 0.1 (be it a 180^o turn or no turn at all).

The robot has 3 onboard sensors, one on each side of the robot and one on the front of the robot (see Figure 5.7). To simplify the problem, we consider that the sensor data received by the decision-maker is already processed. We admit the processed data from each of the sensors to take one of three possible values:

- *Wall*, indicating that there is a wall next to the sensor;

[3]We once again refer the discussion in Chapter 2 on the interface between the high-level and the low-level control of the robot.

[4]Clearly, the consideration of only the 3 referred actions implies that the robot will be non-holonomic.

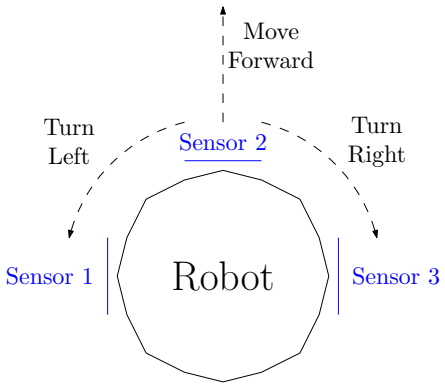


Figure 5.7: The robot, its sensors and actions.

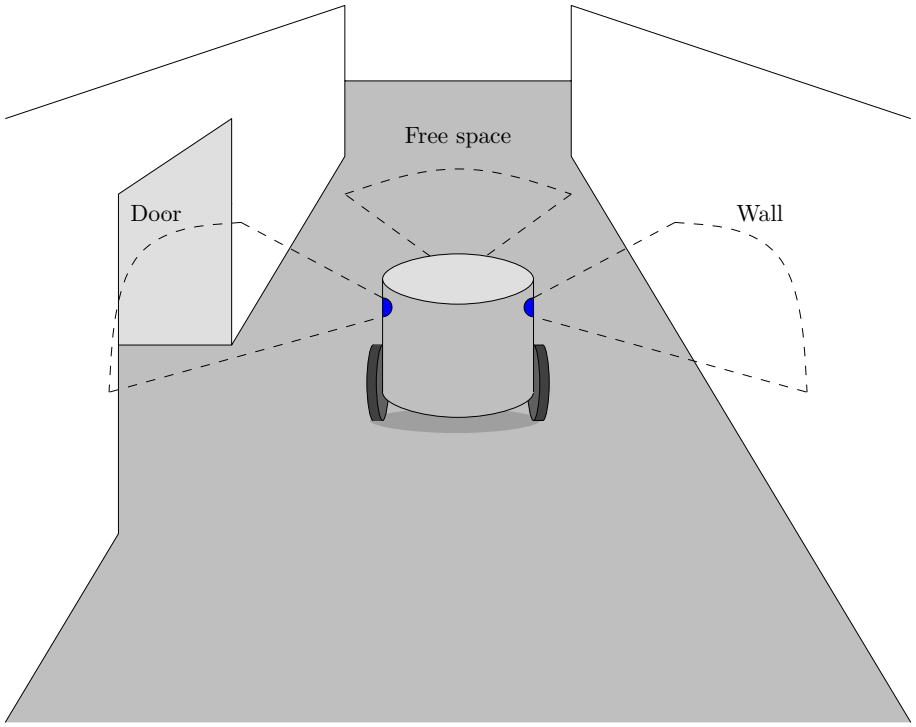


Figure 5.8: Possible observations.

- *Door*, indicating that there is a doorway in or out of a room, next to the sensor; and
- *Free space* indicating that there is a hallway in front of the sensor (see Figure 5.8 for an illustration).

Also, when the robot reaches the goal, it receives a specific signal, which also constitutes a possible observation.

Observations are also prone to errors (arising both during the measuring and/or the processing phases) and there is a non-zero probability of a misclassification in the observations. The agent has a total of 28 possible observations (3 for each of the 3 sensors and the extra *Goal* observation). Table 5.3 summarizes the probabilities

Table 5.3: Observation probabilities.

"True" observation	**Outcome**		
Wall	Wall	Door	Free
	0.90	0.05	0.05
Door	Wall	Door	Free
	0.15	0.70	0.15
Free space	Wall	Door	Free
	0.03	0.07	0.90
Goal		Goal	
		1.00	

governing the errors in the observations of the robot (for each sensor). For example, suppose that there is a wall next to one of the sensors. Once processed, the sensor will provide the observation "Wall" with a 0.9 probability. It will observe a "Door" with a 0.05 probability and "Free space" with a 0.05 probability.

Notice that the probabilities summarized in Tables 5.2 and 5.3 correspond to the *dynamic model* and the *sensor model* of the robot. The process by which these models can be derived is discussed in several works in the literature, such as [53, 274, 277]. The values used correspond to those in [51] so as to facilitate comparison. We adhered to the same experimental setting since this will allow us to compare our results with those reported in the referred work.

Finally, the robot receives a reward of +10 every time it reaches the goal state and 0 every other time step. We consider a discount factor $\gamma = 0.95$.

5.2.3 The experiments

We used a simulator to generate state transitions, observations and immediate rewards in the several environments described above. For each experiment, the starting state is chosen according to Table 5.4 and the initial belief state corresponds to a uniform distribution over all non-goal states. Once the robot reaches the goal state, its position is randomly reset, this time to any of possible states in the environment (except the goal) and the belief is also reset to the initial uniform distribution.

For all experiments, we allowed the learning algorithms to explore the environment for 10^5 time steps. This long learning period is to make sure that the robot has enough time to explore all its action-space and all the *belief-space* (which is a continuous space with a dimensionality corresponding to the number of states in the underlying MDP). We also conducted a series of trials with each learnt policy to evaluate its performance. In all experiments, a single trial consisted of a truncated trajectory of 250 simulated steps starting from the initial state. The immediate re-

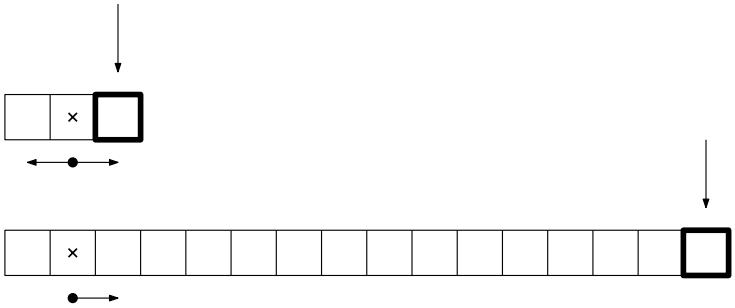


Figure 5.9: Comparison of two policies in environments of different sizes.

wards were appropriately discounted and then added to yield a sample of the total discounted reward. As in all experimental results reported along the thesis, this was repeated for 2,000 independent Monte-Carlo trials and the results reported are the averages over all trials.

In each trial we also verify whether the robot is able to successfully reach the goal state within the 250 time steps. We thus determine the percentage of successful trials, in which the robot was able to reach the goal. This percentage provides us with a second performance measure assessing the usefulness of the learnt policy in terms of the navigation capabilities of the robot.

We emphasize the utility of this second performance measure: in the large scenarios considered, the percentage of "successful missions" conveys a better indication on the performance of the robot than the total discounted reward. To understand why this is so, consider the following very simple example.

Example 5.1. Consider the two environments in Figure 5.9. In both situations, a robot starts at the state marked with a "×" and must reach the state with a bold border and marked with an arrow. Upon reaching the goal state, the robot receives a reward of +10 and remains there forever. For simplicity, we assume that the transitions are deterministic. The discount factor considered is $\gamma = 0.95$.

In the situation depicted in the upper part of Figure 5.9, the robot just follows a random policy, moving randomly either to the left or to the right. Some simple computations lead to the conclusion that the value of the initial state is, in this case, 8.33.

In the situation depicted in the lower part of Figure 5.9, the robot follows the optimal policy, *i.e.*, always moves to the right towards the goal state. However, due to the length of the environment, the value of the initial state is 4.88. Notice the difference in the value of the initial state from the situation where the robot follows a random policy in a small environment.

However, suppose now that we evaluate the percentage of times that the robot is able to reach the goal within 14 steps. In this situation, the robot following the optimal policy will reach the goal 100% of the times. The robot following a random policy has a non-zero probability of not reaching the goal state and, therefore, its performance will be inferior. ◇

Table 5.4: Set of experiments with a single robot.

Experiment	Description	Figure
ISR_1	*Starting state*: blue; *Goal state*: red	5.1
MIT_1	*Starting state*: blue; *Goal state*: red	5.2
PENTAGON_1	*Starting state*: blue; *Goal state*: upper red	5.3
PENTAGON_2	*Starting state*: blue; *Goal state*: lower red	5.3
CIT_1	*Starting state*: blue; *Goal state*: rightmost red	5.4
CIT_2	*Starting state*: blue; *Goal state*: upper red	5.4
CIT_3	*Starting state*: blue; *Goal state*: lower red	5.4
SUNY_1	*Starting state*: blue; *Goal state*: upper red	5.5
SUNY_2	*Starting state*: blue; *Goal state*: lower red	5.5
CMU_1	*Starting state*: blue; *Goal state*: leftmost red	5.6
CMU_2	*Starting state*: blue; *Goal state*: rightmost red	5.6

This simple example clearly shows the importance of also using the second performance measure in our experiments. And, although not immediately evident, it also provides a rationale for considering trials of 250 time-steps: not only it allows us to compare our results with those in other works in the literature [51] but also allows a reasonable time for the robot to reach the goal in any of the environments even if the robot "gets lost" along the way.

Back to our experiments, in each trial the robot starts at the state marked in blue in Figures 5.1 through 5.6 and must reach a particular goal state, marked in red. Table 5.4 describes the goal state for each of the experiments conducted, as well as the reference figure.

We applied approximate Q-learning and approximate SARSA to the MDP obtained from the original POMDP. Recall that both methods should use a set of basis functions verifying

$$\sum_i |\xi_i(x,a)| \leq 1$$

for all $(x,a) \in \mathcal{X} \times \mathcal{A}$, if convergence is to be guaranteed. The beliefs π_t arise as a natural set of basis functions in the MDP $(\mathbb{S}^n, \mathcal{A}, \bar{\mathsf{P}}, \bar{r}, \gamma)$, since $\sum_i \pi_t(i) = 1$ by definition. Therefore, for this set of experiments, we use a set of belief-based basis functions, as in the example in the previous chapter. In particular, we use as basis functions

$$\Xi = \{\xi_{i,a}, i = 1, \ldots, |\mathcal{X}|, a = 1, \ldots, |\mathcal{A}|\},$$

with each $\xi_{i,a}$ given by

$$\xi_{i,a}(\pi, u) = \pi(i)\mathbb{I}_a(u),$$

for all $u \in \mathcal{A}$ and $\pi \in \mathbb{S}^n$. Recall that $\mathbb{I}_a(u)$ denotes the indicator function for the set $\{a\} \subset \mathcal{A}$. With this set of basis functions, approximate Q-learning corresponds to the *linear-Q* algorithm described by Littman et al. [169]. Furthermore, the parameter vector learnt by the algorithm has the same dimension as the Q-functions for the

underlying MDP and we used the so-called Q_{MDP}-values to initialize the parameter vector and speed the learning process.[5] We used Boltzmann exploration to ensure a suitable exploration/exploitation tradeoff.

5.3 Experimental results

We present in Figures 5.10 and 5.11 the cumulative reward obtained during the learning period. The total discounted reward obtained by the learnt policy and the percentage of successful missions for approximate Q-learning and approximate SARSA are summarized in Tables 5.5 and 5.6, respectively.

5.3.1 Discussion

We now discuss the results of the experiments summarized in Figures 5.10 and 5.11 and in Tables 5.5 and 5.6 .

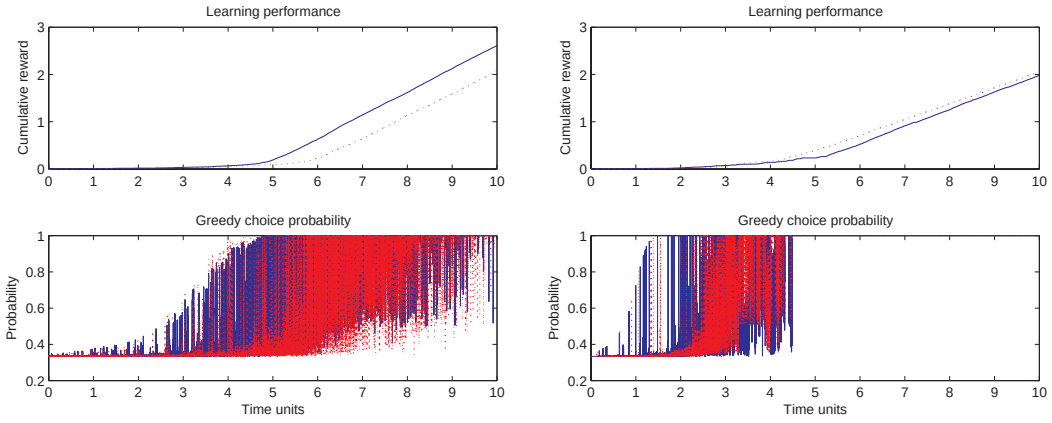
We start by comparing approximate Q-learning and approximate SARSA. From Figures 5.10 and 5.11, we conclude that both algorithms exhibit similar performance, as expected. This is clear by observing that the learning curves for approximate Q-learning (solid blue line) and approximate SARSA (dotted red line) are essentially similar (*i.e.*, both curves stabilize in a similar slope). Also, when comparing the performance obtained with the learnt policies (summarized in Tables 5.5 and 5.6) no significant difference between the two methods is observed.[6] Since the POMDPs for all scenarios verify the conditions of Theorem 4.7.4 and therefore the associated fully observable MDPs verify the conditions of applicability of approximate Q-learning and approximate SARSA, both methods are expected to converge, as they do. Furthermore, because we used the same basis functions with both methods, the limit of convergence with both methods is, once again as expected, the same.

To assess the usefulness of the learnt policies, we provide in Table 5.7 the performance of the optimal policy in the underlying fully observable MDP. Recall that exact methods are of little use in environments with the dimensions of those used in the experiments and the optimal fully observable policy provides an upper bound on the performance of our methods.

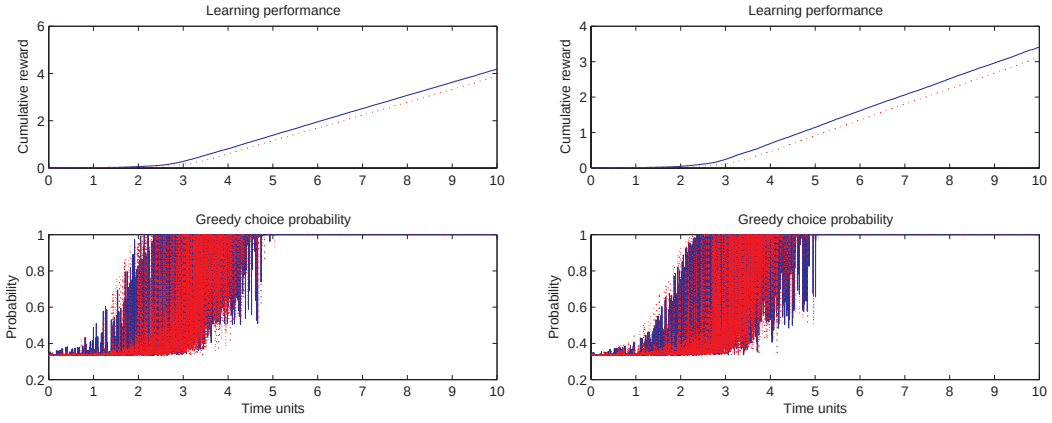
As expected, the optimal fully-observable policy is always able to reach the goal: since the state is fully observable there is no ambiguity on the robot's position in the environment and the robot can therefore choose the optimal action and move efficiently towards the goal. On the other hand, the results described in Tables 5.5 and 5.6 correspond to situations in which the robot can only infer its position from the noisy observations and from the past history. As we argued already, in most

[5] The Q_{MDP} values correspond to the optimal Q-values in the underlying MDP.

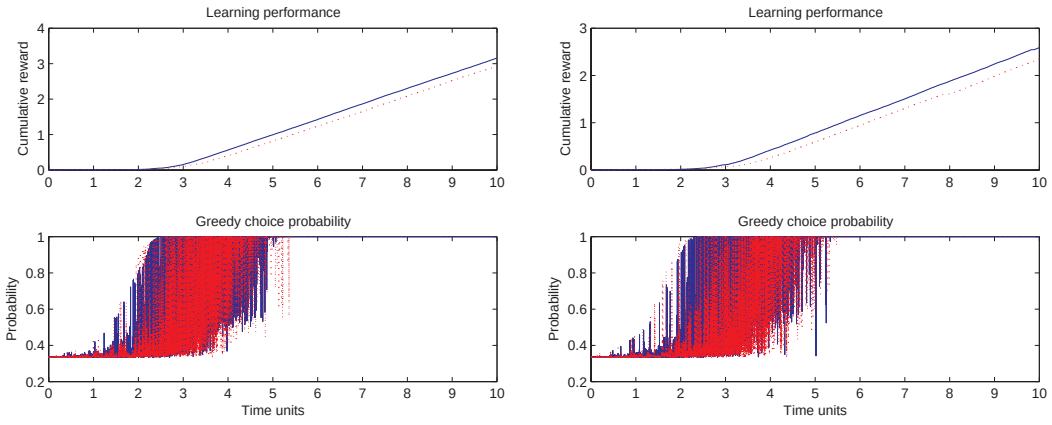
[6] Once again we remark that the convergence guarantees for both methods are asymptotic. This implies that, since the agent only learns for a finite number of steps—10^5—there will be some differences in the learnt approximations. These differences arise from the different ways by which each method samples the state-space, as argued in Chapter 4. In the finite learning period, the difference in the learning policies is not yet completely averaged out and is more clearly noticeable in the larger environments. However, and as seen in the results, even with a finite number of training steps, the differences in performance are not very significant. Further discussion can be found in Chapter 10.



a) ISR$_1$; **b)** MIT$_1$;

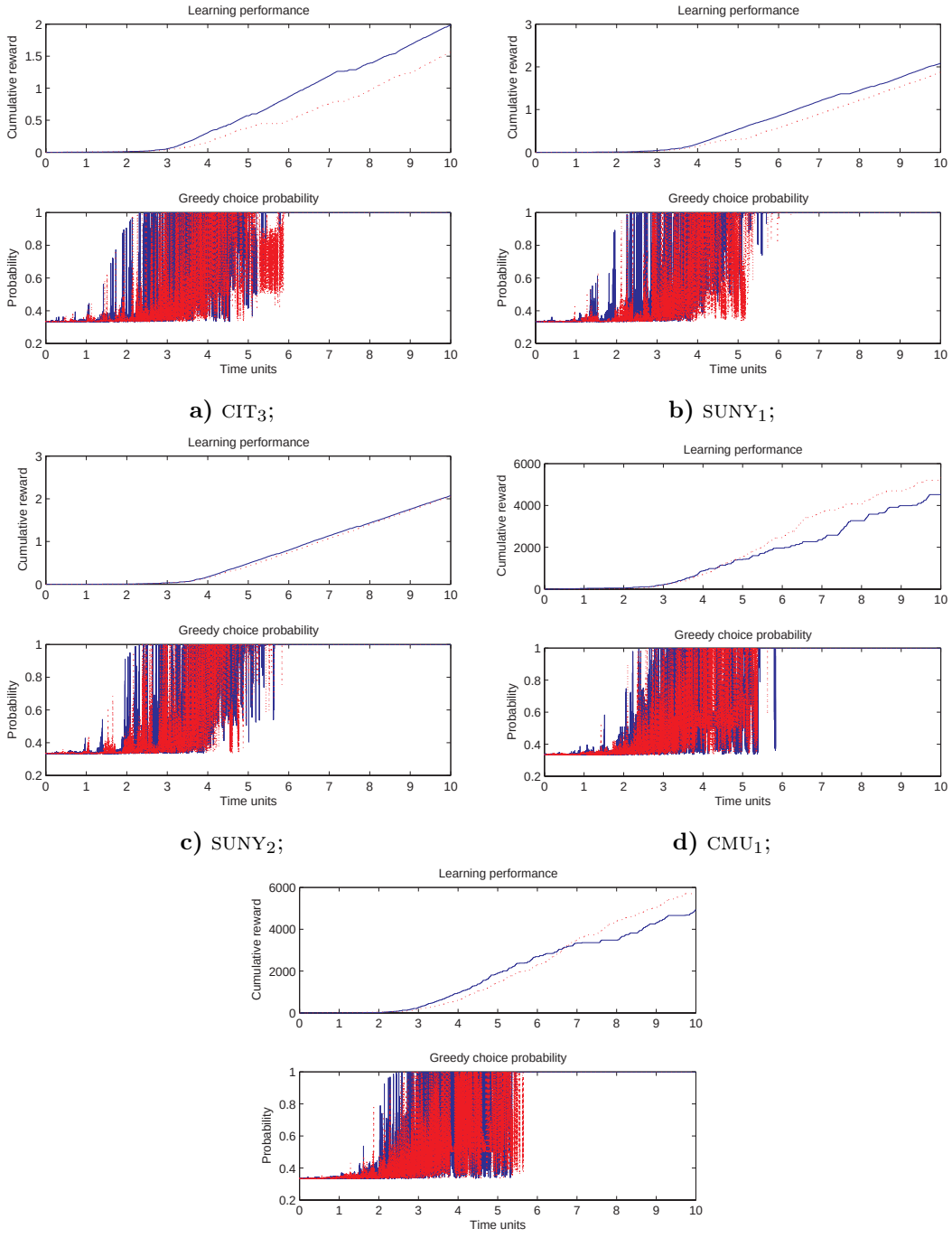


c) PENTAGON$_1$; **d)** PENTAGON$_2$;



e) CIT$_1$; **f)** CIT$_2$;

Figure 5.10: Cumulative reward and greedy choice probability during the 10^5-time-units learning period. The solid blue line represents the results with approximate Q-learning and the dashed red line represents the results with approximate SARSA.



a) CIT_3; **b)** SUNY_1; **c)** SUNY_2; **d)** CMU_1; **e)** SUNY_2.

Figure 5.11: Cumulative reward and greedy choice probability during the 10^5-time-units learning period (continued). The solid blue line represents the results with approximate Q-learning and the dashed red line represents the results with approximate SARSA.

Table 5.5: Total discounted reward and percentage of successful missions in the 11 experiments using approximate Q-learning after the learning period is complete. We present the average total discounted reward and standard deviation obtained over 2,000 independent Monte-Carlo trials.

Experiment	Total Disc. Reward	Success %
ISR_1	3.960 ± 1.491	100.00 %
MIT_1	3.983 ± 1.039	100.00 %
PENTAGON_1	7.478 ± 2.050	100.00 %
PENTAGON_2	4.962 ± 2.018	100.00 %
CIT_1	4.372 ± 1.211	100.00 %
CIT_2	4.078 ± 1.069	99.95 %
CIT_3	1.484 ± 0.525	99.55 %
SUNY_1	0.726 ± 0.541	99.90 %
SUNY_2	1.016 ± 0.350	99.85 %
CMU_1	2.889 ± 1.089	99.30 %
CMU_2	0.800 ± 0.460	84.30 %

Table 5.6: Total discounted reward and percentage of successful missions in the 11 experiments using approximate SARSA after the learning period is complete. We present the average total discounted reward and standard deviation obtained over 2,000 Monte-Carlo trials.

Experiment	Total Disc. Reward	Success %
ISR_1	3.759 ± 1.476	99.95 %
MIT_1	3.952 ± 1.129	99.95 %
PENTAGON_1	7.411 ± 2.131	100.00 %
PENTAGON_2	4.817 ± 1.771	99.95 %
CIT_1	4.187 ± 1.215	100.00 %
CIT_2	4.080 ± 1.071	99.95 %
CIT_3	1.562 ± 0.581	99.85 %
SUNY_1	0.898 ± 0.527	99.95 %
SUNY_2	1.162 ± 0.410	99.80 %
CMU_1	3.178 ± 1.784	97.25 %
CMU_2	0.630 ± 0.333	85.20 %

of the environments there is strong *perceptual aliasing*: many states "look alike", and the robot must often disambiguate its position before heading to the goal state. Therefore, it is expectable that the robot will take longer to reach the goal and collect the reward. This is indeed so and the total discounted reward observed with the learnt policies is somewhat smaller than the total discounted reward obtained with the fully observable policy (Table 5.7).

Also notice the differences in the total reward observed in experiments on the same environment, *e.g.*, between CIT_1 and CIT_3 or CMU_1 and CMU_2. These differences

Table 5.7: Total discounted reward and percentage of successful missions in the 11 experiments using the optimal policy for the underlying MDP. We present the average total discounted reward and standard deviation obtained over 2,000 independent Monte-Carlo trials.

Experiment	Total Disc. Reward	Success %
ISR_1	11.210 ± 2.076	100.00 %
MIT_1	6.083 ± 1.096	100.00 %
PENTAGON_1	12.519 ± 2.075	100.00 %
PENTAGON_2	9.009 ± 1.672	100.00 %
CIT_1	9.472 ± 1.636	100.00 %
CIT_2	5.630 ± 1.140	100.00 %
CIT_3	4.411 ± 0.912	100.00 %
SUNY_1	2.164 ± 0.537	100.00 %
SUNY_2	1.852 ± 0.470	100.00 %
CMU_1	6.412 ± 1.330	100.00 %
CMU_2	1.493 ± 0.321	100.00 %

are justified by 2 main factors: the distance to the goal and the perceptual aliasing in the environment. The distance to the goal, as seen in Example 5.1, greatly affects the total discounted reward received by the robot, because the longer it takes the robot to reach the goal, the more discounted its reward will be. Perceptual aliasing also affects the total discounted reward, although not as directly. In fact, in environments where the perceptual aliasing is more pronounced, the robot will "get lost" more often, therefore taking longer to reach the goal and consequently receiving rewards that are more severely discounted. By inspecting the environment representations in Figures 5.1 through 5.6 it is clear to see that the differences in the total discounted reward are either due to the distance (*e.g.*, in PENTAGON or CMU) or perceptual aliasing (*e.g.*, in CIT).

In spite of all this, the success of the robot in the navigation missions is nearly flawless, using either of the two learning algorithms. In all experiments, the agent was able to reach the goal approximately 90% of the times, this success rate being closer to 100% in all but the largest problems. We also emphasize the fact that performance was measured upon 250-time-step runs, and 7 of the test scenarios have more than 250 states. This implies that the success rate does not arise from a "lucky guess" on the choice of actions or by exhaustive exploration of the state-space, since in most scenarios this would not guarantee that the goal state is ever visited.

We conclude this discussion by referring that similar results have been reported in [51]. In the referred work, the authors apply the Linear-Q algorithm, which is nothing more than approximate Q-learning with the set of basis functions used in this chapter. However, the approach in [51] lacked theoretical support in terms of convergence and obtained approximation, and the Linear-Q algorithm was validated only experimentally, by applying it to several benchmark problems such as MIT, PENTAGON, CIT and SUNY. The results in Chapter 4 of this thesis provide the

lacking theoretical support for the approach in [51]. Furthermore, the results in this chapter also replicate those in the aforementioned paper using a different approach, namely approximate SARSA.

5.4 Concluding remarks

We conclude the chapter with a broad overview of all ideas introduced in this first part of the thesis. We briefly review the main contributions of this part and discuss the general applicability of the RL methodology to topological navigation tasks. We also hint on how this framework can be extended to multi-agent settings, a topic to be developed in the second part of the thesis.

5.4.1 Summary

In this chapter we tested the algorithms described in the previous chapters in several large benchmark problems from the literature.

In particular, we considered large navigation problems where a mobile robot with limited sensing capabilities must navigate from an initial position to a target position. The robot does not know beforehand its target position and must learn how to reach it as it navigates in the environment.

We applied approximate Q-learning and approximate SARSA to this set of problems and verified that, in all situations, the robot is able to learn its path to the target position, exhibiting a nearly-perfect performance (*i.e.*, the robot is able to reach the goal close to 100% of the test runs).

5.4.2 Discussion

The results presented in this chapter leave no doubts on the applicability of reinforcement learning methods to robot navigation tasks. We were able to apply the methods derived within this first part of the thesis to sizeable problems from the literature. Furthermore, the methods present a remarkable performance even in the presence of strong perceptual aliasing.

We further remark that although the use of approximate reinforcement learning in partially observable scenarios is not new (see, for example, [51, 122, 169]), our approach contributes novel convergence guarantees, non-existent in partially observable settings.

However, we should also point out that robot navigation tasks are generally well-structured. For example, in robot navigation tasks it is not unreasonable to assume that the robot has a probabilistic model of its behavior. The existence of a distinctive state (such as a goal state) is also a reasonable assumption. Furthermore, robot navigation tasks exhibit some locality in the transitions, *i.e.*, the robot cannot perform arbitrary "jumps" between the states in the environment. This means that if the robot is in state i at time instant t, only the states around i are likely to be visited at time $t + 1$. This simplifies belief tracking and certainly contributes for the fact that, as seen in the results presented in this chapter, a simple linear

approximation seems to be powerful enough to yield satisfactory results in this class of problems.

All these features, peculiar to robot navigation tasks, probably make this class of problems particularly amenable to the methods described herein and contribute to the success observed in the experiments. They justify (in part) the fact that such a simple approximation architecture as the one used displays such a remarkable performance.

When addressing general decision problems, and in the presence of partial observability, it is not likely that such a simple approximation architecture will work, especially in problems where perceptual aliasing requires *information gathering* decisions.[7] As an example, in [231] the authors describe a problem that specifically requires information gathering decisions and the optimal value function can not be properly approximated as the linear combination of a set of basis functions. Although the convergence results presented in the previous chapter guarantee that approximate Q-learning and approximate SARSA do not oscillate or diverge as long as the problem under consideration is reasonably well structured, there is no guarantee on the usefulness of the obtained policy. As argued in Chapter 4, the quality of the obtained solution depends on a great deal on the set of basis functions used in the approximation. Therefore, care must be exerted upon choosing these functions, taking into consideration all useful information about the problem.

* * *

In the second part of the thesis, we turn to multi-agent systems. We address problems in which a *group of robots* must navigate in a common environment from an initial configuration to a final configuration. We essentially follow a similar course of action to the one in this first part, considering the model provided by Markov games as the natural extension of MDPs to multi-agent settings.

Once the framework is established, we proceed by surveying several well-known reinforcement learning methods from the literature. We discuss important problems such as coordination and equilibrium selection and how these problems can be tackled from a learning perspective. We then address problems with infinite state-spaces and discuss partial observability in multi-agent settings. As will become clear from that discussion, partial observability is far more complex when there are multiple decision-makers interacting in a common environment than it is in single-agent problems.

[7] We refer to the discussion in [51, 231].

Part II

Multi-Robot Navigation and Learning

Chapter 6

Cooperative Navigation and Markov Games

Much like Chapter 2 in the first part of the thesis, this is an introductory chapter that provides the liaison between multi-robot navigation and the reinforcement learning framework. We describe Markov games, the multi-agent "version" of MDPs, and generalize the fundamental concepts from Part I to multi-robot settings.

Since we are interested in cooperative multi-robot systems, we go over three fundamental issues that must be addressed in any cooperative multi-agent systems: the task to be addressed, the mechanism of cooperation and the performance measure for the team. In considering Markov games as a multi-agent extension of MDPs, we establish this framework to settle the first two of the three fundamental issues above. Finally, we address the problem of coordination and equilibrium selection, thus settling the remaining of the three issues.

6.1 Multi-robot systems

In the first part of this thesis, we addressed single-robot navigation problems from a learning perspective. Given a mobile robot moving in an environment described as a topological map, the robot must learn how to execute a predefined navigation task by interacting with the environment. We used POMDPs as a modeling framework and successfully applied RL methods to address these problems.

In this second part of the thesis, we move to more elaborate navigation problems in which *a team of mobile robots* must *cooperatively interact* to reach *a common goal*. We assume that there is no explicit way for any of the robots to know *before-hand* what the other robots will do. In other words, if cooperation is to arise from the interaction of the multiple robots, it cannot be supported by communication. This assumption is referred as *absence of explicit communication*.

The motivation to extend the work presented so far to multi-robot scenarios can be found in many different successful applications of multi-robot systems [50]. As argued by Cao et al. [50], it is not hard to establish a justification for the study of multi-robot systems (MAS):

- Many missions found in practice are inherently too complex (or even impossible) for a single robot to execute;
- The use of multiple robots may lead to a substantial improvement in the overall performance;
- The use of multiple robots with simple motor/sensorial capabilities is often more cost-effective and fault-tolerant than the use of a single complex and powerful robot;
- "Cooperative mobile robotics can possibly yield insights into fundamental problems in the social sciences (organization theory, economics, cognitive psychology), and life sciences (theoretical biology, animal ethology)".[1]

The study of multi-robot systems constitutes a discipline on its own and addresses those many situations in practice in which information, sensing and/or actuation are inherently distributed.[2] After all, a single robot, independently of its sensorial and motor capabilities, is necessarily spatially limited.

* * *

When approaching general multi-agent systems (MAS) from a cooperative point-of-view, it is fundamental that "cooperation" is properly defined. There are numerous definitions of cooperation, such as "the association of persons or businesses for common, usually economic, benefit" or "joint action toward a common end or purpose".[3]

[1] Cited from [50], Section 1, lines 14-18.

[2] See the discussion on distributed artificial intelligence in the work by Gasser [92].

[3] Definitions from http://dictionary.reference.com/.

Even in the multi-agent literature, explicit definitions of cooperation are seldom found and vary according to the application in consideration. Definitions of cooperation include "joint behavior directed toward some goal in which there is common interest or reward" [14], "some form of interaction, often based on communication" [186] and "joint action so as to create a progressive result, such as increasing performance or time saving" [245]. All these definitions, though providing different perspectives on cooperation, are bound by two common concepts: *interaction of multiple agents* with *mutual benefit resulting from it*. This will be the general definition of cooperation that we consider in this thesis.

In the literature on cooperative MAS, it is possible to find a wide range of applications in which cooperation is advantageously implemented. Examples include task decomposition/allocation [35, 36], distributed learning and knowledge [92, 287], management of distributed information [75], correctness and fault-tolerance on multi-agent systems [149], etc. In these different approaches, the attention of the research community is focused on the three "fundamental" issues for cooperation [50]: the *task* to accomplish, the *mechanism of cooperation* and the *performance* of the multi-agent system. The different ways in which MAS address each of these issues permit the definition of a taxonomy that groups MAS in different classes. We refer to [50] for a detailed classification of cooperative MAS.

In this thesis we are interested in addressing cooperative robotics from a learning perspective. In terms of the fundamental issues referred above, we want a team of robots to *learn the task to perform* and the mechanism of cooperation to *emerge* from the mutual interaction among the different robots and the environment. As already stated, we assume that there is no *explicit communication* between the agents. This assumption emphasizes the need for a cooperation mechanism that does not rely on communication but, instead, arises from the (silent) interaction between the robots. We will return to this problem in Section 6.4.

Nevertheless, it is important to refer several examples in the literature in which communication plays a central role in the learning process. If we assume that communication between agents is free and unlimited, then multiagent decision processes reduce to single agent decision processes [246, 247]. Even if communication is not free, it may be used advantageously to decrease the overal complexity of the multi-agent decision problem [212].

In [256, 257, 258], an algorithm is proposed to assess in which situations communication may bring the agents actual benefits, taking into consideration communication constraints such as communication cost or bandwidth limitations. In [91], multiple interacting agents learn a *common ontology* for communication, so that all assign the same meaning to the exchanged messages. In this paper, the agents learn to assign a common meaning to a set of symbols, so that they can effectively communicate and use this communication to act cooperatively in a joint task. This work thus features *communication* as one of the purposes of the learning process.

In another work, Ohko et al. [224] approach the problem of *communication reduction*, in which a group of communicating agents learns how to optimize the load of communication between them. Learning leads the agents to avoid broadcasting in favor of directed communication, thus reducing the overall communication traffic and consequently decreasing the load of communication.

Matarić [185] discusses how communication can improve the learning process. In particular, communication is featured as a tool to reduce the undesirable effects of locality in the learning of cooperative behaviors. In fact, each robot learns from the *local* information it gathers in each location it visits and, as argued in [185], communication can facilitate *global* cooperative behaviors to emerge. The author explores the use of communication from two different perspectives: communication as sensing [185, Sec. 3] and communication as reinforcement [185, Sec. 4].

In the line of [185], Tan [305] addresses the different uses of communication in learning, and compares the use of communication to the sharing of sensations, learning episodes and learnt policies.

However, as argued in [272], communication is often slow and expensive and should be used with care. Agents relying too much on communication may fail to operate when communication fails and are more prone to suffer from misinterpretations arising from faulty communication. In [117], the problem of distributed knowledge is addressed and the authors establish that common knowledge is unattainable, which may harden the design of distributed protocols (such as cooperation protocols) for a group of communicating agents.

Furthermore, as argued in [75, Sec. VI], communication in MAS can affect the overall performance of the system by introducing a "distraction". In fact, effective communication requires that each agent determines *which information* should be sent to *which agent* at *what time*, taking into consideration numerous factors. Communication can consume resources otherwise applicable in the execution of the task without necessarily introducing significant improvements in the overall performance. Tsitsiklis and Athans [319] discuss the difficult problem of deciding when the use of communications is beneficial in a distributed decision-making system.

* * *

In this chapter, we address topological navigation of a group of mobile robots assuming that the robots cannot communicate their actions so as to ensure cooperation. We consider navigation tasks in which the robots must move from an initial configuration to a final configuration in a topological map, while avoiding mutual collisions and other undesirable situations. We describe an adequate generalization of MDPs that explicitly accounts for the existence of multiple decision-makers. We address three fundamental questions arising in cooperative MAS [50]: "what is the task?", "how is the performance of the group inferred?" and "what is the mechanism for cooperation?"

The presentation in this second part of the thesis will closely follow that of Part I. Once the basic tools to address multi-robot problems are laid down in this chapter, we move in Chapter 7 to the description of several simple solution methods. We leave to Chapter 8 the treatment of more complex problems. Only in this chapter will we address situations with partial observability, adopting an approach in all aspects similar to the one followed in Chapter 4 for single-agent scenarios.

6.2 Topological navigation with multiple robots

We consider the problem of a group of robots moving from an initial configuration to a final configuration. The goal of the *group* is to *jointly attain* the final configuration. Even if this goal amounts to each robot reaching its corresponding position, we are interested in addressing situations in which the robots *act as a team* and jointly strive to reach the goal of the team (*i.e.*, reaching the goal configuration).

In this section we assess how the co-existence of multiple robots in a common environment carries additional difficulties that do not arise in single-robot situations (such as resource sharing or colliding interests). We propose the use of Markov games (also known as stochastic games) to explicitly account for the existence of multiple, independent decision-makers that must interact to accomplish their missions.

6.2.1 Navigation and distributed control

Suppose that a group of N mobile robots is to accomplish some navigation task in a common environment. As in Chapter 2, we assume a topological representation of the environment. In the case of multi-robot systems, the use of topological maps offers further advantages besides the ones argued in Chapter 2. Consider, for example, a group of heterogeneous robots navigating in a common environment. The use of a topological representation of the environment can convey a *common* and *unified* representation of the environment. Topological maps are *high-level representations* of the physical environment where the robots move and the mechanism "translating" the sensorial information of a robot into its location in the topological map may differ from one robot to another, while sharing the same topological representation of the environment.

We emphasize once again that, as stated in Chapter 1, robots are to be understood as high-level decision-makers. We do not address the problem of the interface between the high-level decision-making and the low-level sensors and actuators. We also disregard the problem of partial state observability in this chapter and postpone the discussion of that problem until Chapter 8.

To introduce the Markov game framework, we start by considering a group of robots where the decision-making process is *centralized*. This situation, even if accounting for the position/movement of multiple robots, is not significantly different from the one considered in single-agent scenarios. Therefore, we can determine a "composite" MDP, describing the movement of the group of robots in the environment. We illustrate this in the following example.

Example 6.1. We return to our recurrent example and consider once again the indoor environment represented in Figure 6.1, where now we account for the existence of *two* mobile robots.

The two robots (I and II) must traverse this environment, robot $k = \mathrm{I}, \mathrm{II}$ starting in the area labeled as «Start Room k» and striving to reach the area labeled as «Goal Room k». The movement of each robot can be modeled independently using an automaton as in Chapter 2. The automata for this example are depicted in Figure 6.2

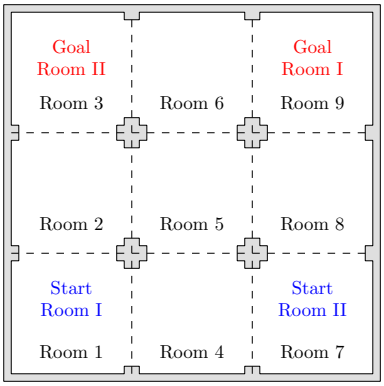


Figure 6.1: Example of an indoor environment.

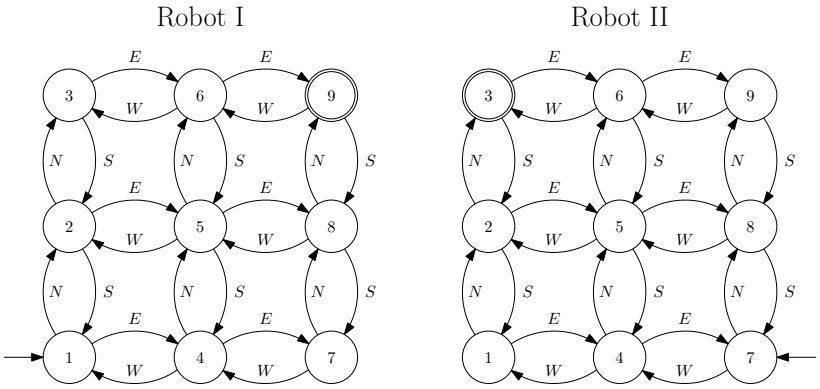


Figure 6.2: Finite-state automata modeling the independent movement of each of two robots in the environment of Figure 6.1.

The two automata in Figure 6.2 can be combined to yield a 81-state automaton describing the joint movement of the two robots in the environment. In this composite automaton, the state of the team describes the positions of *both* robots in the environment; the transitions between the states of the automaton correspond to joint movements of the robots. We can take advantage of the automaton representation to get an MDP describing the joint, controlled movement of both robots and the goal for the team. If the state of robot k, with $k = \mathrm{I}, \mathrm{II}$ at time t is given by X_t^k, we denote the state of the group as a pair $X_t = (X_t^{\mathrm{I}}, X_t^{\mathrm{II}})$, taking any of 81 possible states: $(1,1)$, $(1,2)$, $\ldots$, $(1,9)$, $\ldots$, $(9,9)$. We denote the set of individual states for each of the two robots by $\mathcal{X}^k$, $k = \mathrm{I}, \mathrm{II}$ and the set of all joint states by $\mathcal{X}$.

At each time instant, each robot has 4 actions available, namely North (N), South (S), East (E) and West (W); the action taken by a robot k at time t is denoted as A_t^k. There are 16 possible joint actions: (N,N), (N,S), (N,E), (N,W), $\ldots$, (W,W); we denote by A_t the joint action $A_t = (A_t^{\mathrm{I}}, A_t^{\mathrm{II}})$. Finally, we admit the movement of each robot to be independent. This means that if $\mathsf{P}_{a^k}^{(k)}(i^k, j^k)$ denotes the probability of robot k moving from state i^k to state j^k when the individual action a^k is taken,[4] the probability of moving from the joint state $(i^{\mathrm{I}}, i^{\mathrm{II}})$ to the joint state $(j^{\mathrm{I}}, j^{\mathrm{II}})$ given the combined action

[4]We write $\mathsf{P}^{(k)}$ instead of P^k to avoid confusion with the k-step transition kernel P^k.

$(a^{\mathrm{I}}, a^{\mathrm{II}})$ is

$$\mathbb{P}\left[X_{t+1} = (j^{\mathrm{I}}, j^{\mathrm{II}}) \mid X_t = (i^{\mathrm{I}}, i^{\mathrm{II}}), A_t = (a^{\mathrm{I}}, a^{\mathrm{II}})\right] = \mathsf{P}^{(\mathrm{I})}_{a^{\mathrm{I}}}(i^{\mathrm{I}}, j^{\mathrm{I}})\mathsf{P}^{(\mathrm{II})}_{a^{\mathrm{II}}}(i^{\mathrm{II}}, j^{\mathrm{II}}). \tag{6.1}$$

We denote the transition probability in (6.1) by $\mathsf{P}_a(i, j)$, where $i = (i^{\mathrm{I}}, i^{\mathrm{II}})$, $j = (j^{\mathrm{I}}, j^{\mathrm{II}})$ and $a = (a^{\mathrm{I}}, a^{\mathrm{II}})$.[5] Finally, given a transition triplet (i, a, j), with $i, j \in \mathcal{X}$ and $a \in \mathcal{A}$, if $j = (9, 3)$ then both agents receive a reward of $+20$.

Summarizing, this multi-robot navigation task can be modeled using an MDP $(\mathcal{X}, \mathcal{A}, \mathsf{P}, r, \gamma)$, where

- $\mathcal{X} = \mathcal{X}^{\mathrm{I}} \times \mathcal{X}^{\mathrm{II}} = \{(1, 1), \ldots, (9, 9)\}$;
- $\mathcal{A} = \mathcal{A}^{\mathrm{I}} \times \mathcal{A}^{\mathrm{II}} = \{(N, N), \ldots, (W, W)\}$;
- P and r represent the transition probabilities and reward function and are defined above.

◇

Remark: We make a parenthesis to clarify the notation introduced in the previous example. In the symbols i^k, j^k, a^k appearing in the example, the superscript k should not be seen as a label but as part of the symbol. In other words, to index the actions in $\mathcal{A}^{\mathrm{I}}$ we use the symbol a^{I} and similarly for $\mathcal{A}^{\mathrm{II}}$. We use the general notation $(a^{\mathrm{I}}, a^{\mathrm{II}})$ to refer to *any* combination of the elements of $\mathcal{A}^{\mathrm{I}}$ and $\mathcal{A}^{\mathrm{II}}$ and *not only to those where the action of both robots is the same*; therefore, the symbols a^{I} and a^{II} should be interpreted as consisting of two distinct indices (like m and n). The superscripts are meant only to explicit the agent to which each index refers. On the other hand, and unless if explicitly stated, a simple symbol such as i, j or a should be interpreted as a tuple $i = (i^k)$, $j = (j^k)$ or $a = (a^k)$. We will use this notation extensively throughout the remainder of the thesis. ◇

We notice that, in the previous example, the actions taken by one robot can be chosen independently of the position of the other robot. In other words, if each robot performs its "individual" task while completely disregarding the existence of a second robot in the environment, the group of robots will still perform optimally. This means that the task described is *decoupled*: the optimal centralized controller can be implemented in each robot by ignoring the presence of other agents in the environment. The robots can act independently without any concern on cooperation or any other interaction.

However, in the general situation, this decoupling may not be possible. To see this, consider the more elaborate example below.

[5] We are implicitly considering that the movement of each robot is independent of the movement of the other.

Example 6.2. Consider the exact same navigation problem featured in Example 6.1. Suppose, however, that both robots receive a reward of -10 every time they end up in the same room. This can be interpreted, for example, as a "crash penalty", designed to prevent the robots from wandering too close to one another and eventually crash. In terms of the reward function, this translates in setting $r(i, a, j) = -10$ if $j^{\mathrm{I}} = j^{\mathrm{II}}$.

Notice that the "crash penalty" is chosen smaller (in absolute value) than the reward obtained for reaching the goal. This means that the robots will try to avoid moving to the same room, but only if this does not preclude them from reaching the goal. The purpose of this is to prevent situations in which both robots "freeze" in some safe state to avoid crashing. We also remark that the crash penalty does not actually prevent the robots from crashing. The purpose of this penalty is, instead, to "discourage" them from moving into situations where crashes may occur.

This problem can be modeled using an MDP $(\mathcal{X}, \mathcal{A}, \mathsf{P}, r, \gamma)$ where, now,

- $\mathcal{X}$, $\mathcal{A}$ and P are as in Example 6.1;
- The reward function r assigns a reward of $+20$ for every transition triplet (i, a, j) such that $j = (9, 3)$, -10 for every transition triplet (i, a, j) such that $j^{\mathrm{I}} = j^{\mathrm{II}}$ and 0 otherwise.

We can apply any of the techniques described in Chapters 2 and 3 to determine an optimal policy for this MDP. By following such a policy, we have the guarantee that the group of robots will perform optimally in this scenario. This optimal (joint) policy dictates an individual policy for each of the robots in the group that depends *not only on the position of that robot but also on the position of the other robot.* This was not the case in the previous example, where each robot could choose its actions independently of the position of the other robot. ◇

In our study of MDPs, we established the existence of an optimal value function V^* for each MDP $(\mathcal{X}, \mathcal{A}, \mathsf{P}, r, \gamma)$. From V^*, an optimal policy δ^* can be determined. However, this policy needs not to be unique. In single-agent problems, the existence of multiple optimal policies is of no concern, since the decision-maker can, at each time instant, decide which optimal policy to follow. However, in multi-agent scenarios, the existence of multiple policies can lead to problems. To illustrate this, suppose that in Example 6.2, the robots are in the joint position $(1, 3)$ and the optimal action can be either $a = (E, N)$ or $b = (N, W)$. A centralized decision-maker as featured in the example simply chooses a or b. However, if there are two distinct decision-makers, two scenarios are possible:

- The agents are able to communicate. If that is the case, one of the agents can simply communicate "Choose action a". Both agents then comply to this choice of actions by choosing, respectively, actions E and N, yielding the desired combined action a.
- The agents are unable to communicate. If this is the case, it may happen that one of the agents, say agent A, decides upon the optimal action a, and executes the corresponding action E. Agent B, on the other hand, may decide upon the optimal action b, executing the corresponding action W. The combined

action is (E, W), which not only is not an optimal joint action but will grant them a penalty with great probability.

In the problems addressed in this thesis we consider the different robots as independent decision-makers with no ability to explicitly communicate. In this multi-agent setting, cooperation will occur under the form of *coordination*: the different robots have to *agree* on a particular policy, overcoming the problem of multiple optimal policies described above. Since MDPs model *single agent* decision problems, we will need to extend the MDP model to include multiple decision makers. That is carried out next.

6.2.2 Markov games

In the MDP framework, sequential decision problems are modeled irrespectively of the existence of multiple deciding agents. By solving an MDP, we are able to determine the optimal policy or policies for the particular problem under consideration. However, if multiple deciding robots exist, we are left with the difficulty of enforcing the different robots to follow one such particular optimal policy. Furthermore, there may be situations in which the different robots have colliding objectives that should be "encoded" using distinct reward functions.[6] An example of one such situation is a two-robot evade-pursuit game, where one of the robots must capture the other robot that in turn must avoid being captured.

Although later on we focus on groups of robots with a common joint goal, we now describe a framework that extends the MDP model to cope with multiple agents with possibly different rewards.[7]

A *Markov game* (MG), also known as *stochastic game*, is represented as a tuple $\left(N, \mathcal{X}, (\mathcal{A}^k), \mathsf{P}, (r^k), \gamma\right)$, where N is the set of robots interacting in a given environment,[8] $\mathcal{X}$ is the combined state-space of the N-robot system and $\mathcal{A} = \times_{k=1}^{N} \mathcal{A}^k$ is the cartesian product of the individual action-spaces $\mathcal{A}^k$. P is the transition probability matrix, defining the transition probabilities

$$\mathsf{P}_a(i, j) = \mathbb{P}\left[X_{t+1} = j \mid X_t = i, A_t = a\right].$$

As in the MDP framework, X_t is the state of the system at time t; $A_t = (A_t^1, \ldots, A_t^N)$ is the combined action of the group of robots and i and j are generic elements of $\mathcal{X}$. The transition probability matrix P defines a controlled Markov chain $(\mathcal{X}, \mathcal{A}, \mathsf{P})$, even though the choice of the action profile A_t at each time instant is distributed, *i.e.*, each robot $k \in N$ independently chooses the corresponding individual component A_t^k in A_t. The function $r = (r^1, \ldots, r^N)$ represents the combined reward, r^k being the individual reward for the k^{th} robot.

[6] As remarked in Chapter 2, the reward function "encodes" the objective of the robot/agent.

[7] A group of robots having a common joint goal simply means that they all receive the same reward and not that they must go to the same state. In Example 6.2 the robots had a joint goal, which was to *reach the configuration* $(9, 3)$.

[8] In the remainder of the thesis we use the same symbol – N – to represent both the *set* of all decision-makers in a game and the *total number* of decision-makers. It is our belief that this slight abuse of notation adds no confusion to the presentation.

Markov games were introduced in the 1950s by Lloyd S. Shapley [273]. This framework actually appeared before the MDP framework, but can still be understood as a generalization of Markov decision processes to multi-agent domains. From a game-theoretic point-of-view, MGs are the multi-state counterparts of strategic games.[9] We henceforth adopt the customary game theoretic nomenclature and refer to the decision-makers in a MG as the *players* of the game.

We emphasize once again two important aspects of the Markov game framework:

- The decision-making process is *distributed*, in that each player chooses its individual action with no knowledge of the actions chosen by the other players;
- As defined, Markov games allow for each player to have a different reward function, thus modeling situations where the players may have different interests.

In MGs the decision process is distributed and the different agents can have different/conflicting interests; therefore, there is no individual policy which is optimal for one agent independently of the policies chosen by other agents. The concept of optimal policy as the policy that maximizes the total discounted reward must be adapted to the multi-agent framework. In MAS, the corresponding concept is that of *Nash equilibrium*, to be developed in the following section.

6.3 Optimality and equilibria

Before introducing the main concepts in this section, it is important that the notation used henceforth be properly clarified. We adopt the standard game-theoretic nomenclature and concepts, presented in Appendix C.

At each time instant $t \in \mathcal{T}$ each player $k \in N$ chooses an *individual action* from its individual action set $\mathcal{A}^k$. The combined action of all players is called an *action profile* or *joint action* and takes values in $\mathcal{A}$. A joint action $a \in \mathcal{A}$ results from N individual actions and takes the form $a = (a^1, \ldots, a^N)$. The tuple

$$a^{-k} = (a^1, \ldots, a^{k-1}, a^{k+1}, \ldots, a^N)$$

is a *reduced joint action*, *i.e.*, a joint action where the individual action of player k was omitted. We abusively write

$$a = (a^{-k}, a^k)$$

to indicate that the individual action of player k in the joint action a is a^k.

In a MG, the purpose of each player is to maximize its individual total discounted reward, defined as in MDPs by

$$V^k(\{A_t\}, i) = \mathbb{E}\left[\sum_{t=0}^{\infty} \gamma^t R^k(X_t, A_t) \mid X_0 = i\right]$$

[9] See Appendix C for a brief review on game theory.

with $i \in \mathcal{X}$ and $R^k(i,a)$ the random reward received by player k for taking action a in state i.[10]

An *individual strategy* for player k is a (time-dependent) mapping σ_t^k defining a probability distribution over the set $\mathcal{A}^k$ for each state $i \in \mathcal{X}$. We say that player k follows strategy σ_t^k when playing a game $\left(N, \mathcal{X}, (\mathcal{A}^k), \mathsf{P}, (r^k), \gamma\right)$ if it chooses each action $a^k \in \mathcal{A}^k$ with probability $\sigma_t^k(i,a^k)$ at state $i \in \mathcal{X}$ and time $t \in \mathcal{T}$. If, for each state $i \in \mathcal{X}$, there is an action $a^k \in \mathcal{A}^k$ such that $\sigma_t^k(i,a^k) = 1$, the strategy σ_t^k is said to be a *pure strategy*, and a *mixed strategy* otherwise. A time-independent strategy is said to be a *stationary strategy* and denoted by omitting the time index, *i.e.*, σ^k.

The tuple $\sigma_t = (\sigma_t^1, \ldots, \sigma_t^N)$ is a *joint strategy* or *strategy profile*. Joint strategies are the multi-agent counterparts to single-agent policies in MDPs. If all individual strategies in a strategy profile σ_t are pure, σ_t is a *pure* strategy profile and *mixed* otherwise. Similarly, if all individual strategies in σ_t are stationary, we say that such a strategy profile is *stationary*.

A *reduced strategy profile* or *reduced joint strategy* is a tuple

$$\sigma_t^{-k} = (\sigma_t^1, \ldots, \sigma_t^{k-1}, \sigma_t^{k+1}, \ldots, \sigma_t^N),$$

and, as with actions, we write

$$\sigma_t = (\sigma_t^{-k}, \sigma_t^k)$$

to indicate that the individual strategy of player k in the joint strategy σ_t is σ_t^k. The *support* of a strategy σ_t^k in state $i \in \mathcal{X}$ is the set of all actions $a^k \in \mathcal{A}^k$ such that $\sigma_t^k(i,a^k) > 0$, *i.e.*, those actions that have positive probability of being played by player k in state i.

> Remark: We should emphasize the difference between a *policy* and a *strategy*. The former refers to a behavior-rule for a single individual, while the latter refers to a joint behavior-rule for a group. As defined, a strategy comprises several individual behavior-rules, one for each individual in the group. We could refer these individual behavior-rules as policies but will instead use the designation of *individual strategy* to stress the fact that it is part of a joint strategy. Therefore, we always use the designation of *policy* when referring to single-agent behavior-rules in single-agent problems. In multi-agent problems we always adopt the designation of strategy, classified as *joint* or *individual* whenever is necessary to clarify if it refers to multiple individuals or to a single individual. ◇

As in MDPs, we write $(V^{\sigma_t})^k(i)$ instead of $V^k(\{A_t\}, i)$, whenever the control sequence $\{A_t\}$ is generated according to the strategy profile σ_t, and refer to $(V^{\sigma_t})^k$ as being the *value function for player k* associated with the strategy profile σ_t.

Given a MG $\left(N, \mathcal{X}, (\mathcal{A}^k), \mathsf{P}, (r^k), \gamma\right)$, suppose that all players except player k follow a reduced, stationary and deterministic strategy σ^{-k}. If this is the case,

[10] In the general situation where each robot $k \in N$ has a different reward function r^k, each robot will maximize a different function V^k and hence the use of the superscript k.

the world as perceived by player k can be represented as an MDP, and there must be an optimal individual pure stationary strategy $(\sigma^k)^*$ that maximizes the total discounted reward for this player. The corresponding value function V^* verifies

$$V^*(i) = \sum_{j \in \mathcal{X}} \mathsf{P}_a \big[r^k(i,a,j) + \gamma V^*(j) \big], \tag{6.2}$$

where the action profile a is given by

$$a = (\sigma^{-k}(i), (\sigma^k)^*(i))$$

where we denoted by $\sigma^k(i)$ and $\sigma^{-k}(i)$ the unique individual and reduced actions in the supports of σ^k and σ^{-k}. The strategy $(\sigma^k)^*$ is the *best response* of player k to the reduced strategy σ^{-k}.

The following definition generalizes this concept of best response.

Best Response

Given a MG $\big(N, \mathcal{X}, (\mathcal{A}^k), \mathsf{P}, (r^k), \gamma\big)$ and a reduced joint strategy σ_t^{-k}, a *best response strategy* of player k to the reduced strategy σ_t^{-k} is an individual strategy $(\sigma_t^k)^*$ verifying

$$\left(V^{(\sigma_t^{-k}, (\sigma_t^k)^*)}\right)^k (i) \geq \left(V^{(\sigma_t^{-k}, \sigma_t^k)}\right)^k (i),$$

for all states $i \in \mathcal{X}$ and all individual strategies σ_t^k.

In other words, $(\sigma_t^k)^*$ is a best response strategy of player k to a reduced joint strategy σ_t^{-k} if player k can not expect to improve its total discounted reward by choosing any other individual strategy.

The concept of best response leads to the concept of Nash equilibrium, which in a way represents the idea of optimality in MGs.

Nash Equilibrium

Given a MG $\big(N, \mathcal{X}, (\mathcal{A}^k), \mathsf{P}, (r^k), \gamma\big)$, a *Nash equilibrium* is a joint strategy σ_t^* such that, for all $k \in N$, $(\sigma_t^*)^k$ is the best response of player k to the reduced strategy $(\sigma_t^*)^{-k}$, *i.e.*,

$$\left(V^{\sigma_t^*}\right)^k (i) \geq \left(V^{((\sigma_t^*)^{-k}, \sigma_t^k)}\right)^k (i),$$

for all $i \in \mathcal{X}$ and all $k \in N$.

A Nash equilibrium is therefore a strategy profile such that no player expects any improvement in its total discounted reward by unilaterally changing its individual strategy.

The notion of Nash equilibrium is due to John F. Nash, whom introduced this concept in his works on game-theory in the 1950s [214, 215, 216]. A Nash equilibrium always exists in every Markov game, but its usefulness greatly depends on all players adopting such particular strategy. If a single player follows a different strategy, that particular player will generally perform worse than if it adheres to the Nash equilibrium. However, if *several* players follow non-equilibrium strategies, their individual performance may actually improve when compared to the Nash equilibrium. As such, Nash equilibria are only optimal in the best response sense.

We refer the reader to Appendix C, where concepts such as Nash equilibrium and best response are illustrated with simple examples.

6.4 Coordination and equilibrium selection

In the previous section we introduced MGs as models for sequential interaction of multiple decision-makers. We claimed that Nash equilibria can be understood as the multi-agent "version" of optimal behavior.[11]

In MGs we model the movement of the group of robots as state transitions in a Markov chain. The purpose of each robot is to maximize its individual total discounted reward. This reward structure inherent to MGs not only defines the task, but also provides a performance measure on the group of robots. With the model of MGs we are therefore equipped to answer two of the three fundamental questions referred in Section 6.1: "what is the task?" and "how is the performance of the group inferred?".

In this section we focus on the mechanism of cooperation. We start by noticing that MGs actually provide a general model that can be used in many sequential decision problems with multiple decision-makers. In particular, it encompasses situations where the different decision-makers may have *different* and even *conflicting* goals. In this thesis, we are interested in *cooperative* navigation and therefore we restrict our attention to a particular class of Markov games, known as *team Markov games*, where the individual goals are replaced by a team goal.

6.4.1 Team games

There is a particular class of games in which all players have a common joint goal. In such games, whatever joint strategy works for one player also works for the other players.

The idea of "common goal" is translated in the Markov game model by the statement that *all players share the same reward function*, *i.e.*,

$$r^1 = r^2 = \ldots = r^N = r.$$

[11] There are some works that question the use of equilibria as "optimal" strategies in multi-agent decision problems [243, 276]. However, it will soon become apparent that equilibria are indeed the best working option in the particular class of problems addressed in this thesis.

Such games are known as *team games* or *fully cooperative games* and have the property that all such games have at least one *stationary coordinated (Nash) equilibrium* [166].[12]

A coordinated equilibrium is a Nash equilibrium σ_t^* such that, for each player k, $(V^{\sigma_t^*})^k$ is maximal:

$$\left(V^{\sigma_t^*}\right)^k(i) = \max_{\sigma_t} \left(V^{\sigma_t}\right)^k(i),$$

for all states $i \in \mathcal{X}$. Furthermore, if a game has a coordinated equilibrium, it is always possible to find one pure, stationary, coordinated equilibrium (see Appendix C). This result is similar to the one produced in Chapter 2 regarding the existence of a stationary deterministic optimal policy for any MDP. In fact, by looking at team MGs (TMGs) as MDPs with distributed decision-making, the existence of a pure strategy equilibrium for the former is a consequence of the existence of a deterministic optimal policy for the latter.

However, the existence of at least one coordinated equilibrium strategy does not imply its uniqueness and when multiple coordinated equilibria exist, we may face coordination problems, as illustrated next.

Example 6.2. (cont.) Consider once again the navigation problem in Example 6.2. We previously used an MDP to describe the problem, considering a centralized decision-maker. If we now consider independent decision-makers, we can model the exact same problem using a TMG $\left(N, \mathcal{X}, (\mathcal{A}^k), \mathsf{P}, r, \gamma\right)$, where:

- $N = \{\mathrm{I}, \mathrm{II}\}$ is the set of players;
- $\mathcal{X}$, $\mathcal{A}$ and r are as defined in Example 6.1;
- The transition probabilities are defined by a kernel P given by

$$\mathsf{P}_a(i,j) = \mathsf{P}^{(\mathrm{I})}_{a^{\mathrm{I}}}(i^{\mathrm{I}}, j^{\mathrm{I}})\mathsf{P}^{(\mathrm{II})}_{a^{\mathrm{II}}}(i^{\mathrm{II}}, j^{\mathrm{II}})$$

  and the kernel $\mathsf{P}^{(k)}$ defines the single-robot transition probabilities (we consider the same individual transition probabilities used in Example 2.1);
- We consider $\gamma = 0.95$.

As stated above, a TMG has at least one pure, stationary, coordinated equilibrium but there may exist multiple such equilibria. In this particular example, there are several equilibria, two of which are summarized in Table 6.1. Each entry in this table corresponds to a joint action $(a^{\mathrm{I}}, a^{\mathrm{II}})$ and each of the two 9×9 block corresponds to one coordinated equilibrium. Figure 6.3 illustrates some of the instances of Table 6.1 (corresponding to the colored elements).

Consider, for example, the joint state $(1, 7)$. In this situation, two actions are possible: according to the equilibrium in the upper block of Table 6.1, (E, N) is an optimal joint action; according to the lower block of Table 6.1, (N, W) is also an optimal joint action. Both these actions are represented in Figure 6.3: the former is represented in the solid lines departing from states 1

[12] A coordinated Nash equilibrium is a Pareto-optimal Nash equilibrium, *i.e.*, a Nash equilibrium that attains the maximum possible payoff for all players.

Table 6.1: Multiple coordinated equilibrium strategies. Each 9×9 block corresponds to one coordinated equilibrium. Each entry corresponds to a joint action $(a^{\mathrm{I}}, a^{\mathrm{II}}) \in \mathcal{A}$. The text in blue/red corresponds to the strategies depicted in Figure 6.3. The framed cells correspnd to a mis-coordinated trajectory as described in the main text

State	1	2	3	4	5	6	7	8	9
1	(E,N)	(E,N)	(E,N)	(N,N)	(E,W)	(E,W)	$\boxed{(E,N)}$	(E,W)	(E,W)
2	(E,N)	(E,N)	(E,N)	(E,W)	(E,N)	(E,W)	(E,W)	(N,W)	(E,W)
3	(E,N)	(E,W)	(E,N)	(E,W)	(E,W)	(E,W)	(E,W)	(E,W)	(E,N)
4	(N,N)	(N,N)	(N,N)	$\boxed{(E,N)}$	(E,N)	(E,W)	(N,N)	(E,W)	(E,W)
5	(N,N)	(N,N)	(N,N)	(N,W)	$\boxed{(E,N)}$	(E,W)	(N,W)	(N,W)	(E,W)
6	(E,N)	(E,N)	(E,N)	(E,W)	(E,W)	(E,W)	(E,W)	(E,W)	(E,W)
7	(N,N)	(N,N)	(N,N)	(N,W)	(N,N)	(N,W)	(N,W)	(N,W)	(N,W)
8	(N,N)	(N,N)	(N,N)	(N,W)	(N,N)	(N,W)	(N,W)	(N,W)	(E,W)
9	(N,N)	(N,N)	(S,S)	(N,N)	(N,N)	(N,W)	(N,W)	(N,W)	(N,W)
1	(E,N)	(E,N)	(E,W)	(N,N)	(E,W)	(E,W)	$\boxed{(N,W)}$	(E,W)	(E,W)
2	(E,N)	(E,N)	(E,W)	(E,W)	(E,N)	(E,W)	(E,W)	(N,W)	(E,W)
3	(E,N)	(E,W)	(E,W)	(E,W)	(E,W)	(E,W)	(E,W)	(E,W)	(E,E)
4	(E,N)	(E,N)	(E,N)	$\boxed{(N,W)}$	(E,W)	(E,W)	(N,N)	(E,W)	(E,W)
5	(E,N)	(E,N)	(E,N)	(E,W)	$\boxed{(N,W)}$	(E,W)	(E,W)	(N,W)	(E,W)
6	(E,N)	(E,N)	(E,W)	(E,W)	(E,W)	(E,W)	(E,W)	(E,W)	(E,W)
7	(N,N)	(N,N)	(N,W)	(N,W)	(N,N)	(N,W)	(N,W)	(N,W)	(N,W)
8	(N,N)	(N,N)	(N,W)	(N,W)	(N,W)	(N,W)	(N,W)	(N,W)	(E,W)
9	(E,N)	(E,N)	(N,S)	(E,N)	(E,N)	(E,W)	(E,W)	(E,W)	(E,W)

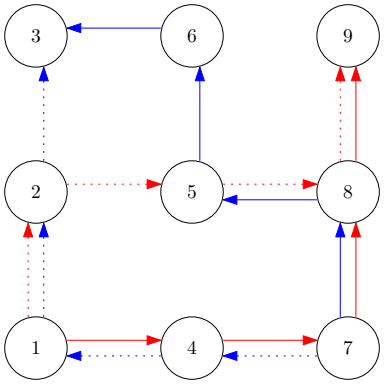


Figure 6.3: Representation of two possible "joint trajectories" using the strategies in Table 6.1. One trajectory is depicted using a solid line and the other using a dotted line

and 7 and the latter in the dotted lines departing from the same two states. The red lines correspond to Robot I and the blue lines correspond to Robot II.

Taking a closer look at the situation where the two robots start in Rooms 1 and 7, there are two optimal actions, as outlined in Table 6.1 and depicted in Figure 6.3: either (E, N) or (N, W).

However, since both strategies are optimal, each robot is equally likely to choose any of the two. In particular, it is possible that Robot I chooses the strategy in the upper block of Table 6.1 and robot II the strategy in the lower block, leading to the combined action (E, W). This action is not only not optimal, but will provably lead both robots to Room 4, with the subsequent penalty of -10. However, once in room 4, the problem persists: there are two different optimal joint actions and the robots may once again choose a different strategy, ending up in the combined action (N, N), which in turn provably leads both robots to Room 5 and again to the same problem. Finally, once in Room 6, there is a unique optimal strategy and the problem is finally solved. In all this process, the robots where penalized in $3 \times (-10)$ and this could easily be avoided using coordination. ◇

This example illustrates two important points. First, TMGs like the one used to describe the navigation problem above are suitable models to incorporate *the common goal* of the group of robots, while explicitly accounting for the existence of multiple decision-makers. Furthermore, they possess the pleasant property of having a pure strategy coordinated equilibrium, which is an adequate solution concept for this class of problems.

Secondly, and on the other hand, the use of TMGs does not yet ensure that the group of robots is able to choose an optimal strategy if the process of decision-making is distributed.[13] The problem described in the previous example is known as an *equilibrium selection problem* in the game theory literature or as a *coordination problem* in the MAS literature.

We consider the problem of coordination next.

[13]Optimal strategy in a TMG should henceforth be understood as a coordinated equilibrium.

6.4.2 Equilibrium selection

As illustrated in the previous subsection, the existence of multiple optimal strategies in MAS usually requires some sort of coordination in action selection. In the context of MAS that we adopt here, this problem is referred to as a *coordination problem* [36].

The solution to the coordination problem provides the answer to the third fundamental question in MAS, regarding the mechanism of cooperation: cooperation occurs in the form of *coordination*. If the robots are able to coordinate in an equilibrium strategy they will achieve the optimal performance and will fail to do so otherwise. This means that a coordinated action choice will result in an improved performance for the team and thus the *mechanism of cooperation* coincides in our setting with the *mechanism of coordination*.

Solving a coordination problem in TMGs requires that coordination is explicitly addressed. To this purpose, one of the following usually holds:

- The players are able to communicate with each other and use communication to "agree" upon the optimal joint strategy;
- The players adhere to a predefined coordination protocol (such as coordination graphs [110]) or social/lexicographic conventions [84, 99]) to decide among different optimal joint strategies;
- Each player infers from past observations or assumes from prior knowledge which is the optimal joint strategy that its colleagues are most likely to follow and hence determines the optimal individual strategy to choose [59, 323, 330].

In this thesis we focus on the third of the above set of coordination strategies. We are interested in ensuring coordination as a result of interaction among the players; as we cope with TMGs from a learning perspective, we require the players to *learn how to play* by playing the game. Coordination should then *emerge* from interaction among the several players as they play the game, instead of relying on a predefined coordination protocol. This makes the third of the previously listed possibilities the most adequate for the class of applications considered herein.

There are different works in the literature addressing the problem of emerging coordination in multi-agent systems. *Joint-action learners* [59] use *fictitious play* to estimate the strategies followed by the other players in team games. This estimate on the other players' strategies is then used to choose a best response strategy. In games with the fictitious play property, fictitious play is known to converge *in beliefs* to a Nash equilibrium, although not necessarily to a coordinated equilibrium (see Appendix C for details).[14] Other recent results have established the convergence of a variation of fictitious play for independent learners [161].

Several variations of the fictitious play principle have been proposed to ensure convergence to a coordinated equilibrium. *Adaptive play* [345] is a variation of fictitious play that sub-samples the history of past-plays. This sampled-history is

[14] As a side note, Uther and Veloso [323] use the same fictitious play principle to address adversarial environments, where not all players have common interests.

Table 6.2: Comparative results for a coordinated and an uncoordinated teams of two robots after the learning period is complete. We present the average total discounted reward and standard deviation obtained over 2,000 independent Monte-Carlo trials. For the purpose of comparison, we also present the results obtained with an optimal, centralized controller.

Method	Total Disc. Reward
Without Coordination	26.951 ± 8.824
With Coordination	28.709 ± 7.621
Optimal	29.057 ± 7.561

then used to estimate the other players' strategy in a fashion similar to fictitious player. *Biased adaptive play* [330] further extends the idea behind adaptive play. The advantage of biased adaptive play over simple adaptive play is that the former actually converges to a coordinated equilibrium in any TMG, unlike the latter, whose convergence guarantees limit to weakly acyclic repeated games.

Lauer and Riedmiller [156] propose another strategy to ensure coordination. In this work, each player optimistically assumes that all players behave greedily. As shown in [156], this approach converges to an optimal Nash equilibrium even if the joint actions are not observable, as long as the transitions are deterministic.[15] Posterior works [136, 157] address non-deterministic settings.

The following example illustrates the use of a coordination strategy in our familiar navigation problem.

Example 6.2. (cont.) Consider once again the indoor environment represented in Figure 6.1. Suppose that we have the same exact navigation task described in the previous example, modeled using a TMG $\left(N, \mathcal{X}, (\mathcal{A}^k), \mathsf{P}, r, \gamma\right)$ as described therein. As seen in the previous example, there are several coordinated equilibria and, in the presence of non-communicating decision makers, a coordination problem arises.

We solved for the optimal Q-function and tested the behavior of the robots in the environment. We ran 2,000 independent Monte-Carlo trials, each trial consisting of 10 time-steps. In each trial, a group of two robots moved about in the environment, following a greedy strategy with respect to the determined Q-function. We used a trial interval of only 10 time-steps to emphasize the difference between the coordinated and uncoordinated teams; the longer the trial interval is, the less noticeable the difference becomes.

Table 6.2 compares the total discounted reward obtained with a team with no coordination with that obtained with a team using a simple coordination mechanism, biased adaptive play (see Chapter 7 and Appendix C for further details). For the purpose of comparison, we also present the results obtained with an optimal centralized decision-maker.

[15] Even if the actions are not directly observable, the deterministic transitions and the greedy play assumption actually provide all information regarding the actions played.

As expected, the coordinated team clearly outperforms the uncoordinated team (both in terms of discounted reward or total reward). Furthermore, the performance of the coordinated team is similar to that of the optimal centralized controller, also as expected. This means that, with *coordination*, we are able to successfully ensure *cooperation* between the robots.

To further clarify the process by which cooperation is attained, we remark that the team using the coordination mechanism was allowed to interact for $5,000$ time-steps prior to engaging in the test trials. During this $5,000$ training period, the robots were allowed to randomly explore their action repertoire, while estimating the strategy that the other robot may be following. This $5,000$ time-step training period allowed the robots to learn how to coordinate in each of the 81 possible joint states. During the learning period, the robots randomly explored the environment using a Boltzmann distributed exploration probability.

Figure 6.4.a) displays the learning performance of the coordinated robots during the training period of $5,000$ time-steps. Figure 6.4.b) displays the average cumulative reward obtained by the coordinated and uncoordinated robots. The solid blue line represents coordinated group and the dashed green line represents the uncoordinated group. For the purpose of comparison, we also plot in red the performance of a group of robots with a centralized, optimal controller. Notice that the instant in time when coordination starts to emerge is clearly visible in the plot of Figure 6.4.a). Also, as seen in Figure 6.4.b), the robots start to coordinate just as their action choice becomes greedy. $\diamond$

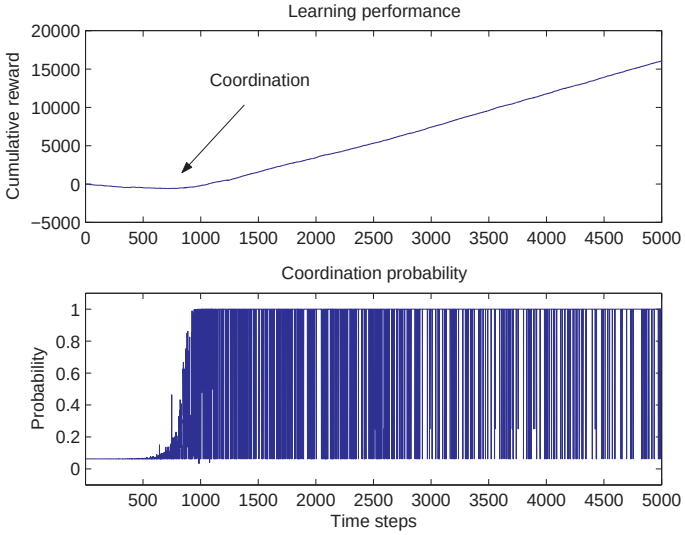
Example 6.3. We now consider the existence of *three* robots in the environment represented in Figure 6.5.

We consider the same navigation task described in the previous example, with the due modifications arising from the existence of a third robots. As with the 2-robot problem, we can model this problem using a TMG $\left(N, \mathcal{X}, (\mathcal{A}^k), \mathsf{P}, r, \gamma\right)$, where:

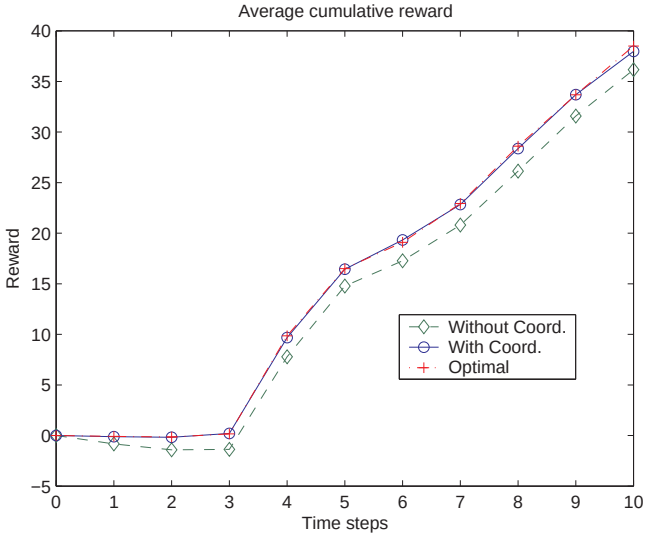
- $N = \{\mathrm{I}, \mathrm{II}, \mathrm{III}\}$ is the set of players;
-
- $\mathcal{X} = \mathcal{X}^{\mathrm{I}} \times \mathcal{X}^{\mathrm{II}} \times \mathcal{X}^{\mathrm{III}} = \{(1,1,1), \ldots, (9,9,9)\}$;
- $\mathcal{A} = \mathcal{A}^{\mathrm{I}} \times \mathcal{A}^{\mathrm{II}} \times \mathcal{A}^{\mathrm{III}} = \{(N,N,N), \ldots, (W,W,W)\}$;
- The transition probabilities are defined by a kernel P given by

$$\mathsf{P}_a(i,j) = \mathsf{P}^{(\mathrm{I})}_{a^{\mathrm{I}}}(i^{\mathrm{I}}, j^{\mathrm{I}}) \mathsf{P}^{(\mathrm{II})}_{a^{\mathrm{II}}}(i^{\mathrm{II}}, j^{\mathrm{II}}) \mathsf{P}^{(\mathrm{III})}_{a^{\mathrm{III}}}(i^{\mathrm{III}}, j^{\mathrm{III}})$$

  and the kernel $\mathsf{P}^{(k)}$ defines the single-robot transition probabilities (we consider the same individual transition probabilities used in Example 2.1);
- $r(i,a,j) = 20$ if $j = (9,3,1)$, $r(i,a,j) = -10$ if $j^{\mathrm{I}} = j^{\mathrm{II}}$ or $j^{\mathrm{I}} = j^{\mathrm{III}}$ or $j^{\mathrm{II}} = j^{\mathrm{III}}$ or $j^{\mathrm{I}} = j^{\mathrm{II}} = j^{\mathrm{III}}$ and $r(i,a,j) = 0$, otherwise.
- We consider $\gamma = 0.95$.



a) Cumulative reward and greedy choice probability during the 5,000 time-steps learning period;



b) Average cumulative reward obtained during the 10 time-step trials.

Figure 6.4: Performance of the coordinated and uncoordinated teams of 2 robots.

As before, in the presence of non-communicating decision makers, a coordination problem arises. We solved for the optimal Q-function and tested the behavior of the robots in the environment. As in the previous example, we ran $2,000$ independent Monte-Carlo trials, each trial consisting of 10 time-steps.

Table 6.2 compares the total discounted reward obtained with a team with no coordination with that obtained with a team using a simple coordination mechanism, biased adaptive play (see Chapter 7 and Appendix C for further details).

As in the two-robot situation, the coordinated team clearly outperforms the uncoordinated team (both in terms of discounted reward or total reward) and is able to attain optimal performance. In this larger scenario, with 729 states, the the team using coordination was allowed to interact for 5×10^4

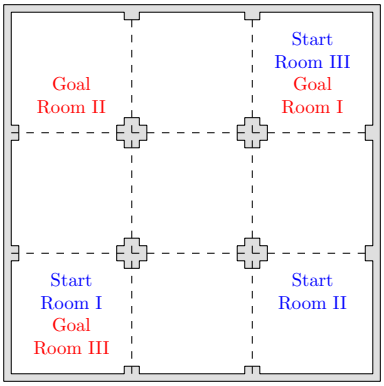


Figure 6.5: Indoor environment with 3 robots.

Table 6.3: Comparative results for a coordinated team and an uncoordinated team of three robots after the learning period is complete. We present the average total discounted reward and standard deviation obtained over 2,000 independent Monte-Carlo trials. For the purpose of comparison, we also present the results obtained with an optimal, centralized controller.

Method	**Total Disc. Reward**
Without Coordination	19.320 ± 8.942
With Coordination	22.662 ± 8.593
Optimal	23.361 ± 7.641

time-steps prior to engaging in the test trials. As before, this training period allowed the robots to achieve coordination in each of the 729 possible joint states. Notice furthermore that, due to the small environment and to the presence of three robots, "accidents" and mis-coordinations are more prone to occur than in the two-robot situation, which makes the use of a coordination mechanism even more desirable.

To further illustrate the optimality of the learnt coordinated strategy, we present in Figure 6.6 the average cumulative reward obtained by the coordinated and uncoordinated teams of robots after learning. As in the 2-robot scenario, the solid blue line represents coordinated group, the dashed green line represents the uncoordinated group and the red line represents the performance of a group of robots with a centralized, optimal controller. ◇

6.5 Concluding remarks

To conclude this chapter, we summarize the main ideas presented so far. We establish a close parallel between the development in this part of the thesis and that worked in Part I. We then point out the main ideas still to be presented in the remainder of the thesis.

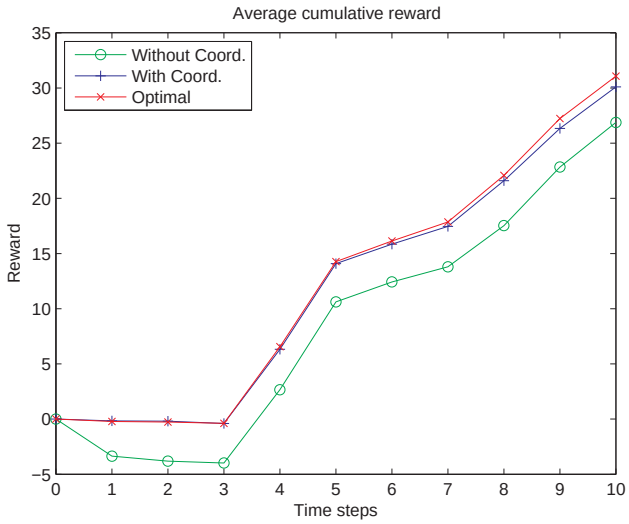


Figure 6.6: Performance of the coordinated and uncoordinated teams of 3 robots.

6.5.1 Summary

The main purpose of this chapter was to extend the MDP framework to situations where multiple decision-makers co-exist in a common environment. We introduced the main concepts to be dealt with in the second part of the thesis; we established the main differences in notation from the first part, where single-agent problems were addressed. Finally, we motivated the use of MGs as a model to address navigation problems where a set of robots moves in a common environment described topologically.

With this purpose in mind, we briefly reviewed several bibliographical references on MAS. We focused on *cooperative MAS*, where a group of agents interacts so as to improve the overall performance of the group as a whole. We described some of the problems inherent to this class of systems, as well as several issues that should be addressed in this context. Three fundamental issues immediately arise: how to define the *task*; how to measure the *performance* of the group; and how to define the *mechanism of cooperation*.

The framework of MGs provides an appealing solution to two of these issues: its reward structure readily solves the problem of task definition and performance evaluation. Furthermore, the profound relation between MGs and MDPs is an appealing feature; an intuitive understanding of the main ideas involved in the former can be gained from our previous study of the latter. In the context of MGs we reviewed fundamental concepts such as those of best response and Nash equilibrium.

We then focused on *team games*, where all decision-makers share a common joint goal. We introduced the concept of *coordinated equilibrium* and assessed the existence of such equilibria in all TMGs. We described the problem of *equilibrium selection* and referred several methods from the literature to address this problem (*fictitious play*, *adaptive play* and *biased adaptive play*).

We concluded by establishing that, with one such method, a team of robots is

able to coordinate, improving the overall performance of the team and conveying the mechanism for cooperation in the class of problems discussed in this thesis.

6.5.2 Discussion

Markov games arise in a multitude of contexts essentially related with multi-agent sequential decision making problems. Its appearance preceded that of MDPs and extended the existing idea of strategic games to sequential problems. Clearly, its great generality allows for far more applications than those portrayed here.

It is immediate that a MG $\left(N, \mathcal{X}, (\mathcal{A}^k), \mathsf{P}, (r^k), \gamma\right)$ with $N = 1$ reduces to a standard MDP. This fact renders this class of models amenable to reinforcement learning methods. The rewards r^k, usually referred as *payoffs* in the game theoretic nomenclature, provide the reinforcement signal for the different decision-makers, and Nash equilibria provide possible solutions.

Of course, Nash equilibria are only optimal in the best response sense, and its usefulness greatly depends on all agents agreeing to such a strategy. Given this fact, and as argued by Powers and Shoham [243], one may question the usefulness of striving to determine and follow a given equilibrium strategy without knowing whether the other players follow such equilibrium strategy. In [276], the authors discuss several possible research alternatives on multi-agent decision problems, addressing the particular topic of "optimality". In [243] they advance an algorithm that addresses non-equilibrium optimality.

In the context of team games addressed in the thesis, coordinated equilibria arise as the natural optimality concept. Nevertheless, even in this particular context it is important that the decision-makers do not blindly comply to one such equilibrium. Instead, it is desirable that each decision-maker be able to play a best response strategy with respect to the others' strategies, and hence the advantage of the coordination method described in this chapter. In the presence of robots with limitations or in the event that one of the robots is damaged (and has available only a reduced action set), our method is still able to adapt and choose the best response to the strategies of the others.

We would also like to comment on the absence of explicit communication between the different robots, as considered herein. Although no *explicit* communication is assumed, the fact that there is an all-knowing reward mechanism that depends on the actions of all robots is, in a sense, an *implicit* communication mechanism. The same can be argued about the ability of each robot to *know* the actions of all other robots *a posteriori*. This assumption is assumed in all coordination strategies described in this chapter (*e.g.*, fictitious play, adaptive play) and will be further commented on Chapter 8. We also postpone to Chapter 8 the discussion of *partial state observability*.

Chapter 7

Reinforcement Learning in Finite Markov Games

In this chapter, we proceed the study of multi-agent decision making by extending the methods in Chapter 3 to multi-agent settings. We also introduce several important tools that will play a central role in the developments of this second part of the thesis.

The chapter starts by strengthening the relation between TMGs and MDPs. This will allow us to immediately derive multi-agent variants of several RL methods described in Chapter 3. This also sets the necessary background to tackle the problem of simultaneous coordination and learning and will lead to the introduction of two algorithms, OAL and CQL, the latter of which is the first novel contribution in this second part of the thesis.

7.1 Learning in multi-agent systems

In the last decades, we have witnessed an increasing interest from the RL community in extending the powerful existing algorithms from single-agent to multi-agent scenarios.

The impressive success of RL in different applications have confirmed the applicability of this class of methods and sustain the claim that RL is a privileged framework to study and explore machine learning in sequential decision problems.[1] This has boosted the research in the topic and led to a number of different approaches. There are several complementary surveys on the subject of learning in MAS. We refer the surveys by Sen and Weiß [272], Stone and Veloso [287], Bowling and Veloso [39] and Yang and Gu [343], just to name a few.

Reinforcement learning methods such as the ones in Chapters 3 and 4, where an agent learns how to act by interacting with the environment, often require that same environment to be *stationary*. Such requirement can be formalized in terms of the ergodicity of the Markov chain, an assumption present in many convergence results.[2] A learning agent interacting with an environment where other learning agents exist can ignore them and treat them as part of the environment. However, in many situations, this violates the ergodicity assumption and algorithms such as the ones in Chapters 3 and 4 may not converge. Even if convergence is attained, the learnt policy can be unsatisfactory. The only solution to this problem is to explicitly consider the existence of multiple players, extending the MDP solution methods to multi-agent settings.

Littman [165] proposed the Minimax-Q algorithm as a possible application of Q-learning to zero-sum MGs.[3] Hu and Wellman [127, 128, 129] later proposed Nash-Q, a modified version of Littman's Minimax-Q algorithm that can be applied to general-sum MGs. They established convergence of their method under quite stringent conditions, as argued in [38, 166]. The restricted applicability of Nash-Q led to the development of Friend-or-Foe Q-learning (FF-Q), proposed by Littman [160, 167]. FF-Q requires less stringent assumptions than Nash-Q, while retaining its convergence properties in several classes of MGs.

Claus and Boutilier [59] proposed joint-action learners (JAL) as a suitable learning algorithm for repeated team games and a similar principle was used by Uther and Veloso [323], where opponent modeling was used to address adversarial scenarios. These algorithms are supported on the concept of *fictitious play*, as referred in the previous chapter.

Gradient-based learning strategies are analyzed with detail in Singh et al. [281] and an interesting result is established regarding the convergence of the average payoffs to those of a Nash equilibrium. Bowling and Veloso [40, 41, 42, 43] proposed a policy-based learning method which applies a policy hill-climbing strategy with

[1] See the page maintained by Satinder Singh and Michael Littman on reinforcement learning in http://neuromancer.eecs.umich.edu/cgi-bin/twiki/view/Main/SuccessesOfRL and references therein.

[2] For discrete-state MDPs, the assumption that the agent is able to visit every state-action pair infinitely often arises as a consequence of ergodicity.

[3] Zero-sum Markov games are a special class of Markov games in which two competing agents receive symmetric rewards for each action executed. See Appendix C for details.

varying learning step, using the principle of "win or learn fast" (WoLF-PHC).

Many other works on multi-agent learning can be found in [35, 36, 56, 154, 260, 334, 335, 336].

In the line of the approach presented in the first part of the thesis, we address MAS from a learning perspective. As seen in Chapter 6, MDPs find in TMGs its multi-agent counterparts and one would expect the methods described in Chapter 3 to be extendable to TMGs. In this chapter we see that this is often so. We emphasize the fact that the main difference between TMGs and MDPs lies on the process by which actions are chosen and the methods in Chapter 3 can thus be extended to multi-agent settings in a straightforward way. We also address the problem of simultaneous coordination and learning.

Our novel contribution in this chapter is related with this last problem of simultaneous coordination and learning. We propose a new algorithm, *coordinated Q-learning*, and establish its convergence w.p.1. As will become apparent, coordinated Q-learning extends the results from the literature by combining a model-free learning method (Q-learning) with biased adaptive play, a coordination mechanism. The successful combination of both methods yields a new, powerful method to address multi-agent learning and coordination in the absence of explicit communication.

7.2 Learning the game

In the remainder of the chapter we admit that all players are able to observe *a posteriori* the joint action played by the team at each time instant. This means that the only difference between TMGs and MDPs lies on the *process* by which the (joint) actions are chosen. In particular, a Markov game in which the players adhere to a strategy σ is equivalent to an MDP where a centralized decision-maker follows σ (now understood as a policy).

In addressing learning in multi-agent scenarios, we consider two distinct problems: *learning the game* and *learning to coordinate*. While learning the game, we focus on determining either the optimal Q-function or the optimal value function. Once any of these functions is known, the agents are able to determine the optimal strategy/strategies and, if necessary, deal with the problem of coordination (*i.e.*, learn to coordinate).

In this section, we focus on the particular problem of learning the game. Similarly to the development in Chapter 3, we start with model-based approach and then proceed with a model-free approach. Notice that, while decision-making in Markov games is a distributed process, learning the game adds no significant differences from the single-agent situation. Therefore, the methods described in Chapter 3 extend without change to multi-agent scenarios. Moreover, convergence of these extensions is a straightforward corollary of the convergence for the original methods.

At this point it is important to remark that, given a fixed strategy in a TMG, the *evaluation of the corresponding value function is no different from the evaluation of the value function regarding a fixed policy in an MDP*. In fact, apart from the fact that in TMGs any strategy σ is implemented in a distributed manner, no other significant difference arises from the existence of multiple agents. Therefore, a TMG

reduces to an ordinary MDP and the evaluation of a fixed joint strategy σ is nothing but the evaluation of σ (now understood as a policy) for the MDP.

Furthermore, in strategy evaluation there is no such thing as a coordination problems, since each decision-maker simply has to implement its individual strategy, independently of the plays of the other decision-makers. Therefore, as already stated, strategy evaluation in TMGs reduces to policy evaluation in MDPs and we do not address strategy evaluation in this second part of the thesis. We refer to the first part of the thesis for more details on this topic.

7.2.1 Model-based learning

In this subsection, we extend *model-based learning* to multi-agent scenarios. As seen in Section 3.3, model-based algorithms build a model of the environment from experience and then use this learnt model to determine the function Q^*.

Given a TMG $\big(N, \mathcal{X}, (\mathcal{A}^k), \mathsf{P}, r, \gamma\big)$, let σ be a stochastic joint strategy verifying $\sigma^k(i,a) > 0$ for all $(i,a) \in \mathcal{X} \times \mathcal{A}$. Let $\hat{\mathsf{P}}_t^k$ denote player k's estimate of P at time instant t. The update rule for $\hat{\mathsf{P}}_t^k$ is

$$\hat{\mathsf{P}}_{t+1}^k(i,a,j) = \hat{\mathsf{P}}_t^k(i,a,j) + \frac{\mathbb{I}_{(i,a)}(i_t,a_t)}{n_{t+1}(i,a)}\big(\mathbb{I}_j(i_{t+1}) - \hat{\mathsf{P}}_t^k(i,a,j)\big), \tag{7.1}$$

where $n_t(i,a)$ denotes the number of visits to the state-action pair (i,a) in the history up to time t, given by $\mathcal{H}_t = \{i_0, a_0, i_1, \ldots, a_{t-1}, i_t\}$, and $\mathbb{I}$ is the indicator function. Similarly, let $\hat{r}_t^k$ denote player k's estimate of r at time instant t. The update rule for $\hat{r}_t^k$ is

$$\hat{r}_{t+1}^k(i,a,j) = \hat{r}_t^k(i,a,j) + \frac{\mathbb{I}_{(i,a,j)}(i_t,a_t,i_{t+1})}{n_{t+1}(i,a,j)}\big(r_t - \hat{r}_t^k(i,a,j)\big), \tag{7.2}$$

where now $n_t(i,a,j)$ denotes the number of occurrences of the transition triplet (i,a,j) in the history $\mathcal{H}_t$. We remark that we are considering a team Markov game, where all agents share the reward function. The symbols $\hat{\mathsf{P}}^k$ and $\hat{r}^k$ denote the *estimated transition matrix* and *estimated reward function* maintained by player k. However, since the processes $\{X_t\}$ and $\{A_t\}$ are fully observable to all players, all estimates $\hat{\mathsf{P}}_t^k$ and $\hat{r}_t^k$ will be coincident and we can, therefore, drop the superscript k. We will show that, under suitable conditions, the sequences $\{\hat{\mathsf{P}}_t\}$ and $\{\hat{r}_t\}$ converge w.p.1 to P and r, respectively.

To compute Q^*, and as in Chapter 3, each agent applies the corresponding fixed-point iteration using $\hat{\mathsf{P}}_t$ and $\hat{r}_t$ instead of P and r:

$$Q_{t+1}(i_t,a_t) = \sum_{j\in\mathcal{X}} \hat{\mathsf{P}}_t(i_t,a_t,j)\big(\hat{r}_t(i_t,a_t,j) + \gamma \max_{b\in\mathcal{A}} Q_t(j,b)\big), \tag{7.3}$$

The sequence $\{Q_t\}$ will converge to the desired function Q^* w.p.1, as stated in the following result.

Theorem 7.2.1. *Given a finite-state TMG* $\big(N, \mathcal{X}, (\mathcal{A}^k), \mathsf{P}, r, \gamma\big)$*, the sequence of estimates* $\{Q_t\}$ *generated by* (7.3) *converge to* Q^* *w.p.1 for any initial estimate* Q_0*, as long as every state-action pair* $(i, a) \in \mathcal{X} \times \mathcal{A}$ *is visited infinitely often.*

Proof See Appendix F.3. □

It is important to emphasize once again that *all players* maintain *coincident estimates* $\hat{\mathsf{P}}$ and $\hat{r}$ of P and r and thus compute the same Q-function. This Q-function will later be used for coordination and hence the importance that all players have coincident estimates.

7.2.2 Model-free learning

We now consider the problem of estimating the function Q^* from direct interaction with the environment, without recurring to any model (learnt or not). The method analyzed herein is a simple multi-agent extension of the Q-learning algorithm described in Chapter 3.

Given a TMG $\big(N, \mathcal{X}, (\mathcal{A}^k), \mathsf{P}, r, \gamma\big)$, let σ be a stochastic joint strategy verifying $\sigma^k(i, a) > 0$ for all $(i, a) \in \mathcal{X} \times \mathcal{A}$ and $k \in N$. Recall from Chapter 3 the update rule for Q-learning:

$$Q_{t+1}(i_t, a_t) = Q_t(i_t, a_t) + \alpha_t(i_t, a_t)\big(r_t + \gamma \max_{b \in \mathcal{A}} Q_t(i_{t+1}, b) - Q_t(i_t, a_t)\big). \tag{7.4}$$

Once again, each player is able to observe the sequences $\{X_t\}$ and $\{A_t\}$ and therefore maintains an estimate Q_t of Q^* that is updated using the rule in (7.4). Clearly, all players perform the same update at every time instant and thus maintain coincident estimates. Just like in the single-agent case, the sequence $\{Q_t\}$ will converge to Q^* w.p.1, as long as every state-action pair $(i, a) \in \mathcal{X} \times \mathcal{A}$ is visited infinitely often. This is stated in the following result.

Theorem 7.2.2. *Given a finite-state TMG* $\big(N, \mathcal{X}, (\mathcal{A}^k), \mathsf{P}, r, \gamma\big)$*, the sequence of estimates* $\{Q_t\}$ *generated using multi-agent Q-learning converge to* Q^* *w.p.1 for any initial estimate* Q_0*, as long as*

$$\sum_{t \in \mathcal{T}} \alpha_t(i, a) = \infty; \qquad \sum_{t \in \mathcal{T}} \alpha_t^2(i, a) < \infty,$$

and $\alpha_t(i, a) = 0$ *if* $(i, a) \neq (i_t, a_t)$*.*

Proof See Appendix F.3. □

To conclude this section, we remark that in both methods introduced we established only convergence w.p.1 of the estimates $\{Q_t\}$ to the optimal Q-function.

However, we did not address *convergence in behavior*. In particular, we are interested in extending the concept of GLIE strategy to multi-agent problems. Since decision-making is now a distributed process, it is necessary to ensure that none of the players "becomes greedy" before sufficient exploration has been conducted. It is also essential that, once the greedy strategies are attained, the joint strategy arising from the individual greedy strategies is itself greedy. In the next section we further comment on the topic of GLIE strategies, as it is deeply related with the problem of coordination.

7.3 Learning to coordinate

So far in this chapter, we focused on the problem of *learning the game*. We described model-free and model-based algorithms that can be used to learn the optimal Q-function and thus determine the optimal joint strategies. We established convergence of these methods from the convergence theorems in Chapter 3.

However, we have not yet addressed the problem of coordination. This is particularly important in defining, for example, a GLIE strategy: even if all players follow a greedy individual strategy, there is no guarantee that the combined joint strategy will be greedy. As such, and so as to clarify the use of GLIE strategies in multi-agent settings, it is necessary to ensure some coordination mechanism.

In a team of players endowed with a coordination mechanism, the simple idea of *optimal strategy* is replaced by that of *coordinated optimal joint strategy*; the idea of *greedy strategy* is replaced by that of *greedily-coordinated joint strategy*. And a GLIE strategy is a strategy that is greedily-coordinated in the limit while still ensuring infinite exploration.

The two algorithms presented in this section make extensive use of GLIE strategies to ensure sufficient exploration. And, in the limit, both algorithms become greedily-coordinated, *i.e.*, all players converge in behavior to an optimal joint strategy. These two algorithms arise from combining the methods in Section 7.2 with *biased adaptive play* (BAP), a coordination mechanism introduced in [330] that ensures coordination w.p.1. The first algorithm thus obtained, dubbed *optimal adaptive learning* (OAL), was introduced by Wang and Sandholm [330] and combines model-based learning with BAP. The second algorithm is dubbed *coordinated Q-learning* (CQL) and constitutes the main novelty in this chapter. CQL combines a model-free learning algorithm with the same BAP coordination mechanism. Notice that the only difference between OAL and CQL lies on the learning method: OAL is a model-based method while CQL is a model-free method. The qualities of each method lead back to the model-free vs. model-based discussion in Chapter 3. We conclude the section by establishing two important properties of both algorithms, namely *rationality* and *convergence*.

7.3.1 Coordination and learning

Let $\left(N, \mathcal{X}, (\mathcal{A}^k), \mathsf{P}, r, \gamma\right)$ be a TMG and let Q^* be the corresponding optimal Q-function. The function Q^* defines, at each state $i \in \mathcal{X}$, a team strategic game

$\Gamma_i^* = \big(N, (\mathcal{A}^k), Q^*(i,\cdot)\big)$.[4] We refer to the team strategic games Γ_i^* by *state-game*. A coordinated equilibrium in this state-game Γ_i^* corresponds to an optimal joint strategy in the overall Markov game, for the particular state i in consideration [36].

Since BAP plays a central role in the present and the following chapters, we now briefly describe this coordination mechanism in greater detail.

Biased adaptive play

Biased adaptive play is a coordination mechanism introduced by Wang and Sandholm [330] that is, in its essence, similar to adaptive play [345]. There are, however, two main differences between BAP and adaptive play. First of all, by introducing a biasing mechanism, BAP is guaranteed to converge in a broader class of games. Secondly, by constructing an auxiliary game, BAP converges only to optimal equilibria.

Consider a team strategic game $\Gamma = \big(N, (\mathcal{A}^k), r\big)$ and suppose that the players repeatedly engage in this game, adapting their plays based on the history of the most recent plays.

In order to ensure convergence to optimal equilibria, BAP constructs an auxiliary game whose only equilibria are the optimal equilibria in the original game. This *virtual game* thus eliminates all suboptimal Nash equilibria and simplifies the coordination process by considering only those equilibria that are optimal. This virtual game is a team strategic game $VG = \big(N, (\mathcal{A}^k), r_{VG}\big)$ constructed from Γ by considering the modified payoff function r_{VG} given by

$$r_{VG}(a) = \begin{cases} 1 & \text{if } a \in \arg\max\limits_{b \in \mathcal{A}} r(b); \\ 0 & \text{otherwise.} \end{cases}$$

Notice that, in fact, every Nash equilibrium in this new game corresponds to an optimal equilibrium in the original game. Therefore, if the players are able to coordinate in a Nash equilibrium in the game VG, they will have coordinated in a coordinated equilibrium in the original game Γ, as desired. We denote the set of Nash equilibria in VG by $D = \{a \in \mathcal{A} \mid r_{VG}(a) = 1\}$.

Let $\mathcal{H}_t = \{a_1, \ldots, a_t\}$ denote the history of all past plays of Γ up to the t^{th} play. Denote by H_t the m most recent plays in $\mathcal{H}_t$, *i.e.*,

$$H_t = \{a_{t-m+1}, \ldots, a_{t-1}, a_t\}.$$

For any $0 < K \leq m$, a *K-sample from H_t* is a set of K plays randomly drawn without replacement from H_t. We denote a K-sample from H_t as $K(H_t)$.

When choosing its action at time instant $t \geq m$, each player k draws a K-sample $K^k(H_t)$ *independently of all other players* and checks if

1. There is a joint action $a^* \in D$ such that all actions $a \in K^k(H_t)$ verify $a^{-k} = (a^*)^{-k}$;
2. There is at least one action $a^* \in D$ such that $a^* \in K^k(H_t)$.

[4] A team strategic game is a team Markov game with a single state. See Appendix C for details.

Condition 1 simply states that, according to the K-sample of player k, all players have coordinated in the reduced action $(a^*)^{-k}$. Condition 2 states that, in the recent past, at least one optimal equilibrium a^* has been played and, furthermore, that the reduced action $(a^*)^{-k}$ in Condition 1 is part of this equilibrium.

If both these conditions hold, then player k is lead to believe that all players have coordinated except, eventually, player k itself and, therefore, plays the corresponding action $(a^*)^k$ in the most recent optimal equilibrium in the K-sample.

Otherwise, player k uses the K-sample to estimate the expected payoff of each individual action $a^k \in \mathcal{A}^k$ as

$$EP_t^k(a^k) = \sum_{a^{-k} \in \mathcal{A}^{-k}} r_{VG}\big((a^{-k}, a^k)\big) \frac{n_K(a^{-k})}{K},$$

where $n_K(a^{-k})$ denotes the number of times that the reduced action a^{-k} appears in the K-sample $K^k(H_t)$. Notice that EP_t^k simply estimates from the K-sample the average strategy of the other players. Then, player k chooses its action randomly from the best response set

$$BR_t^k = \left\{ a^k \mid a^k = \arg\max_{b^k \in \mathcal{A}^k} EP_t(b^k) \right\}.$$

Coordinating while learning the game

It is possible to apply BAP as defined above to each state-game Γ_i^* as long as the payoff function Q^* is known. However, in our current approach, the players do not know beforehand the optimal Q-function and BAP must be adapted so as to cope with the simultaneous learning of the game. We describe such adaptation next. Before describing how BAP can be modified to accommodate the simultaneous learning of Q^*, we introduce the following definition.

Given a function $Q : \mathcal{X} \times \mathcal{A} \longrightarrow \mathbb{R}$ and any $\varepsilon > 0$, a joint action $a^* \in \mathcal{A}$ is *ε-optimal w.r.t. Q at some state $i \in \mathcal{X}$* if

$$Q(i, a) \geq \max_{a \in \mathcal{A}} Q(i, a) - \varepsilon.$$

We denote the set of all ε-optimal actions w.r.t. Q at state i by $\mathbf{opt}_Q^\varepsilon(i)$ or, if Q is clear from the context, $\mathbf{opt}^\varepsilon(i)$.

Let $\big(N, \mathcal{X}, (\mathcal{A}^k), \mathsf{P}, r, \gamma\big)$ be a TMG. Suppose that Q^* is not known but, instead, an estimate Q_t of Q^* is available to each player. Define the estimate state-game at state i as $\Gamma_i = \big(N, (\mathcal{A}^k), Q_t(i, \cdot)\big)$. We now apply BAP to each estimate state-game Γ_i, considering a virtual game VG_t instead of VG and proceeding as described above. The game VG_t is a team strategic game obtained from Γ_i by considering the

reward function

$$r_t(a) = \begin{cases} 1 & \text{if } a \in \mathbf{opt}^{\varepsilon_t}(i); \\ 0 & \text{otherwise.} \end{cases}$$

The convergence of VG_t to VG is immediate, since the sequence $\{Q_t\}$ converges to Q^*, as seen in Section 7.2. By using a suitable decreasing sequence $\{\varepsilon_t\}$, we can ensure that VG_t also converges to VG and BAP still coordinates in an optimal Nash equilibrium. This statement is formally established in [330] and, in a more general setting, in Subsection 7.3.4.

We are now in position to combine BAP with the learning algorithms from Section 7.2. Combining BAP with model-based learning will lead to the *optimal adaptive learning* (OAL) algorithm, by Wang and Sandholm [330] that we describe next. Afterwards, we combine BAP with multi-agent Q-learning, this leading to the *coordinated Q-learning* (CQL) algorithm, the main novelty in this chapter. We describe CQL in Subsection 7.3.3.

We remark that the main difference between OAL and CQL lies on the method used to learn the game. While OAL relies on ARTQI and is, therefore, a model-based approach, CQL relies on Q-learning, being a model-free approach. Therefore, we expect both methods to be essentially equivalent. The discussion on the qualities of each method lead back to the model-free vs. model-based discussion in Chapter 3 and, therefore, we do not pursue such discussion here.

7.3.2 Optimal adaptive learning

Optimal adaptive play was proposed and analyzed by Wang and Sandholm [330], combining model-based learning and BAP. The use of the model-based learning algorithm from Section 7.2 ensures that the estimates Q_t converge to the optimal Q-function, as long as there is sufficient exploration. BAP, in turn, guarantees that all players converge to an optimal Nash equilibrium in the limit.

Figure 7.1 summarizes the pseudo-code of optimal adaptive learning for one player. We divided the algorithm in 3 main blocks: initialization, learning the game and learning coordination.

In the initialization, OAL uniformly initializes the transition matrix estimate $\hat{\mathsf{P}}^k$ and sets the reward estimates $\hat{r}$ to zero.

In learning the game, OAL implements the updates described in Section 7.2. Basically, it uses the information from the observed transition (X_t, A_t, X_{t+1}) to update the model parameters $\hat{\mathsf{P}}_t$ and $\hat{r}_t$ and uses the updated parameters to perform one VI iteration, improving the estimate Q_t. It also updates the ε_t parameter to ensure that VG_t converges to VG at an adequate speed, by means of the real function $B(N_t)$ in line 11. For the time being, we take $B(n)$ as any user-defined real function going to zero as $n \to \infty$. Further ahead we will provide more details on this function.

Finally, in learning coordination (lines 4 through 6), OAL simply implements BAP as described above: it builds the virtual game VG_t from Q_t (line 5a), determines a K-sample by means of the function K-$\mathsf{sample}(K, H_t(i))$ (line 5c) and then determines the best response action if no coordination has been attained (*i.e.*, if the two BAP conditions in Subsection 7.3.1 are not met). The symbol $n_h(a^{-k})$ denotes

Initialization:

1: **Set** $t = 0$ and $\varepsilon_t = \varepsilon_0$;

2: **For all** (i, a) **set** $n_t(i, a) = 1$, $\hat{\mathsf{P}}_t(i, a, j) = \frac{1}{|\mathcal{X}|}$, $\hat{r}_t(i, a, j) = 0$ and $Q_t(i, a) = 0$;

3: **Set** $\mathbf{opt}^{\varepsilon_t}(i) = \mathcal{A}$ and $D = \mathcal{A}$;

Learning coordination: Given current state X_t

4: **If** $t \leq m$, randomly select an action

5: **else** with GLIE exploitation probability $(1 - p_t)$ **do**

a. **Update** VG_t as

$$VG_t(X_t, a) = \begin{cases} 1 & \text{if } a \in \mathbf{opt}^{\varepsilon_t}(X_t); \\ 0 & \text{otherwise;} \end{cases}$$

b. **Set** $D = \{a \mid VG_t(X_t, a) = 1\}$;

c. **Set** $h = K\text{-}\mathsf{sample}(K, H_t(X_t))$;

d. **For all** $a^k \in \mathcal{A}^k$, **set**

$$EP_t(a^k) = \sum_{a^{-k} \in \mathcal{A}^{-k}} VG_t\big(X_t, (a^{-k}, a^k)\big) \frac{n_h(a^{-k})}{K};$$

e. **Set** $BR_t(X_t) = \big\{a^k \mid a^k = \arg\max_{b^k \in \mathcal{A}^k} EP_t(b_k)\big\}$;

f. **If** conditions 1 and 2 of Subsection 7.3.1 are met, choose the most recent joint action in $h \cap D$;

g. **else** randomly choose an action in $BR_t(X_t)$;

6: **And** with exploration probability p_t randomly select an action;

Learning the game: Given current transition triplet (X_t, A_t, X_{t+1})

7: **Set** $n_t(X_t, A_t) = n_t(X_t, A_t) + 1$;

8: **Update** $\hat{\mathsf{P}}_t$ and $\hat{r}_t$ according to (7.1) and (7.2);

9: **Update** Q_t according to (7.3);

10: **Set** $t = t + 1$ and $N_t = \min_{i,a} n_t(i, a)$;

11: **If** $\varepsilon_t \geq \varepsilon_0 B(N_t)$,

a. **Set** $\varepsilon_t = \varepsilon_0 B(N_t)$;

b. **For all** i, **set** $\mathbf{opt}^{\varepsilon_t}(i) = \{a \mid Q_t(i, a) \geq \max_b Q_t(i, b) - \varepsilon_t\}$

Figure 7.1: The OAL algorithm for one player.

the number of times that the reduced action a^{-k} appears in the K-sample h.

Notice also that OAL implements GLIE exploration by means of the exploration probabilities p_t. This ensures sufficient exploration, essential to the convergence of Q_t to the optimal Q-function. We also remark that the algorithm in Figure 7.1 is presented in the adequate order to perform the computation, different from the order in the analysis above.

7.3.3 Coordinated Q-learning

In this subsection, we introduce the *coordinated Q-learning* algorithm (CQL). As OAL, CQL also tackles the problem of simultaneous learning and coordination and is the main contribution of this chapter. With CQL, we extend the ideas in OAL to model-free learning. The interest of this new algorithm will become evident in the next chapter, as its extension to TMGs with infinite Markov games is immediate. Under suitable exploration, multi-agent Q-learning ensures the estimates Q_t to converge to the optimal Q-function and BAP guarantees convergence of the players' strategies to an optimal Nash equilibrium. We once again emphasize that the only difference between CQL and OAL *resides on the process of learning the game*.

Figure 7.2 summarizes the pseudo-code of coordinated Q-learning (CQL) for one player. Once again, notice that the main difference between CQL and OAL lies in the block that learns the game (lines 7 through 10): CQL learns the game using the model-free approach in Subsection 7.2.2, while OAL uses the model-based approach in Subsection 7.2.1. The remainder of the algorithm is similar to OAL and we refer to the corresponding analysis for an overview.

Two important observations are in order. First of all, notice that both OAL and CQL consider a GLIE learning strategy. This can be confirmed by checking lines 5 and 6 in both algorithms. As will soon become apparent, this is an essential assumption to ensure simultaneous learning and coordination in both algorithms. The GLIE strategy is implemented by considering a sequence $\{p_t\}$ converging to zero that indicates the exploration probability at each time instant $t \in \mathcal{T}$. Thus, when *exploring*, each player randomly chooses its individual action; when *exploiting*, each player chooses its individual actions using biased adaptive play. As long as $p_t \to 0$ at an adequate rate, sufficient exploration is guaranteed, and the strategy has the GLIE property.

A second observation is related to the function $B(N_t)$, appearing in both algorithms. This function is user-defined and determines the rate at which ε_t goes to zero (or, equivalently, the rate at which suboptimal actions are eliminated from the virtual game VG_t). As already stated in Subsection 7.3.1, it is important that this function ensures the convergence of the sequence $\{\varepsilon_t\}$ to zero to be slower than that of VG_t to VG (or, which is equivalent, of Q_t to Q^*). This requirement is properly formalized next.

7.3.4 Convergence and coordination

This subsection complements the convergence results on Section 7.2 by establishing *convergence in behavior* for the OAL and CQL algorithms in Figures 7.1 and 7.2 (convergence of Q_t was established in Theorems 7.2.1 and 7.2.2).

Initialization:

1: **Set** $t = 0$ and $\varepsilon_t = \varepsilon_0$;

2: **For all** (i, a) **set** $n_t(i, a) = 1$ and $Q_t(i, a) = 0$;

3: **Set** $\mathbf{opt}^{\varepsilon_t}(i) = \mathcal{A}$ and $D = \mathcal{A}$;

Learning coordination: Given current state X_t

4: **If** $t \leq m$, randomly select an action

5: **else** with GLIE exploitation probability $(1 - p_t)$ **do**

a. **Update** VG_t as

$$VG_t(X_t, a) = \begin{cases} 1 & \text{if } a \in \mathbf{opt}^{\varepsilon_t}(X_t); \\ 0 & \text{otherwise;} \end{cases}$$

b. **Set** $D = \{a \mid VG_t(X_t, a) = 1\}$;

c. **Set** $h = K\text{-}\mathsf{sample}(K, H_t(X_t))$;

d. **For all** $a^k \in \mathcal{A}^k$, **set**

$$EP_t(a^k) = \sum_{a^{-k} \in \mathcal{A}^{-k}} VG_t\big(X_t, (a^{-k}, a^k)\big) \frac{n_h(a^{-k})}{K};$$

e. **Set** $BR_t(X_t) = \big\{a^k \mid a^k = \arg\max_{b^k \in \mathcal{A}^k} EP_t(a_k)\big\}$;

f. **If** conditions 1 and 2 of Subsection 7.3.1 are met, choose the most recent joint action in $h \cap D$;

g. **else** randomly choose an action in $BR_t(X_t)$;

6: **And** with exploration probability p_t^k randomly select an action;

Learning the game: Given current transition triplet (X_t, A_t, X_{t+1})

7: **Set** $n_t(X_t, A_t) = n_t(X_t, A_t) + 1$;

8: **Update** Q_t^k according to (7.4), with $\alpha_t(i, a) = \frac{1}{n_t(i,a)}$;

9: **Set** $t = t + 1$ and $N_t = \min_{i,a} n_t(i, a)$;

10: **If** $\varepsilon_t \geq \varepsilon_0 B(N_t)$,

a. **Set** $\varepsilon_t = \varepsilon_0 B(N_t)$;

b. **For all** i, **set** $\mathbf{opt}^{\varepsilon_t}(i) = \big\{a \mid Q_t(i, a) \geq \max_b Q_t^k(i, b) - \varepsilon_t\big\}$

Figure 7.2: The CQL algorithm for one player.

Consider a sequence $\{Q_t\}$ generated by one of the algorithms in Figures 7.1 or 7.2. Since in both algorithms the players learn the game independently of the coordination mechanism, the convergence results in Section 7.2 hold for both OAL and CQL. Therefore, the sequence $\{Q_t\}$ converges to Q^* w.p.1. Suppose now that there is a function $\mathsf{rate} : \mathbb{N} \longrightarrow \mathbb{R}$ such that, w.p.1,

$$\|Q_t - Q^*\| \leq K_0 \, \mathsf{rate}(N_t), \tag{7.5}$$

where K_0 is some positive constant. The following result is a generalization of Lemma 6 in [330]. The function $B : \mathbb{N} \longrightarrow \mathbb{R}$ defines the rate of convergence of the sequence $\{\varepsilon_t\}$ to zero in the both OAL and CQL.

Lemma 7.3.1. *Consider a TMG* $\big(N, \mathcal{X}, (\mathcal{A}^k), \mathsf{P}, r, \gamma\big)$. *Let* Λ_T *be the event that, for* $t > T$, $VG_t = VG$ *for an agent following a learning algorithm for which* (7.5) *holds, where* VG_t *and* VG *are the virtual games obtained from* Q_t *and* $Q*$ *as detailed above. If* $B(N_t)$ *decreases monotonically to zero and*

$$\lim_{t\to\infty} \frac{\mathsf{rate}(N_t)}{B(N_t)} = 0,$$

then $\lim_{t\to\infty} \mathbb{P}\left[\Lambda_T\right] = 1$.

Proof See Appendix F. □

The previous result states that each of the virtual games VG_t defined from the state-games Γ_i converges to the virtual game VG obtained from Γ_i^*. As established in [36], the combination of the coordinated equilibria for each Γ_i corresponds to a coordinated equilibrium for the original TMG. On the other hand, since in OAL and CQL the players admittedly follow a GLIE strategy, every state is visited infinitely often.[5] Therefore, to establish convergence in behavior of both OAL and CQL to an optimal Nash equilibrium, it suffices to show that they converge to one such equilibrium in all state-games Γ_i, $i \in \mathcal{X}$. The following two results establish this fact.

Theorem 7.3.2. *Let* $\big(N, \mathcal{X}, (\mathcal{A}^k), \mathsf{P}, r, \gamma\big)$ *be a TMG with N players. Suppose that the following conditions hold:*

1. *The players follow a GLIE strategy as described in the algorithm of Figure 7.1;*
2. *The function* $B(N_t)$ *decreases monotonically to zero and verifies*

$$\lim_{t\to\infty} \frac{\sqrt{\frac{\log\log(N_t)}{N_t}}}{B(N_t)} = 0$$

[5] The very definition of GLIE strategy implies that all state-action pairs (in particular all states) are visited infinitely often.

3. *The lengths m and K of the history H_t and the K-sample h verify $m \geq K(N+2)$.*

Then, the sequence of estimates $\left\{Q_t^k\right\}$ generated by OAL converges to Q^* w.p.1. Furthermore, all players in N converge in behavior to a common coordinated Nash equilibrium w.p.1.

Proof See Appendix F. □

Theorem 7.3.3. *Let $\left(N, \mathcal{X}, (\mathcal{A}^k), \mathsf{P}, r, \gamma\right)$ be a TMG with N players. Suppose that the following conditions hold:*

1. *The players follow a GLIE strategy as described in the algorithm of Figure 7.2;*
2. *The function $B(N_t)$ decreases monotonically to zero and verifies*

$$\lim_{t\to\infty} \frac{\sqrt{\frac{\log\log(N_t)}{N_t}}}{B(N_t)} = 0 \tag{7.6}$$

3. *The lengths m and K of the history H_t and the K-sample h, m and K, verify $m \geq K(N+2)$;*
4. *The sequence of step-sizes $\{\alpha_t\}$ is given by*

$$\alpha_t(i,a) = \begin{cases} \frac{1}{n_t(i,a)} & \text{if } (i,a) = (i_t, a_t); \\ 0 & \text{otherwise.} \end{cases}$$

Then, the sequence of estimates $\{Q_t\}$ generated by CQL converges to Q^* w.p.1. Furthermore, all players in N converge in behavior to a common coordinated Nash equilibrium w.p.1.

Proof See Appendix F. □

One final and important remark is in order. In the previous section, we described the Q-learning update mechanism (later used in CQL) to learn the function Q^*. One could question if an on-policy update mechanism (such as SARSA) could be used to learn the game, replacing the Q-learning update in CQL.

The answer to this question is negative. As seen in Chapter 3, SARSA converges to the optimal Q-function only if a GLIE strategy is used for learning; if any other policy is used, SARSA converges to the corresponding Q-values. In CQL, the bound (7.6) on the function $B(N_t)$ implies that coordination occurs only when the estimates Q_t are "sufficiently close" to the true function Q^*. Using a SARSA-like update, the

estimates Q_t approach Q^* only as the learning strategy approximates the greedy strategy (while still ensuring sufficient exploration). These are two incompatible requirements and, therefore, it is not generally possible to use an on-policy update mechanism such as SARSA with BAP.

To conclude this section, we assess two interesting properties of OAL and CQL.

7.3.5 Convergence and rationality

We have described two methods to estimate the optimal Q-functions for team Markov games. The players of the game can then use these estimates to decide on the "correct" line of action. Like the methods described in Part I, these algorithms can be classified as *value-function based methods*. Several known reinforcement learning methods for multi-agent scenarios also fall in this category, such as Minimax-Q, Nash-Q, FF-Q and several others [166]. These algorithms have been shown to converge to the a desired solution under suitable conditions [129, 167, 168].

However, as argued in Chapter 6, the existence of multiple equilibria poses difficult problems for learning algorithms, even if the optimal Q-function is accurately learnt. As pointed out by Bowling and Veloso [39], if a game has multiple equilibria, the optimal strategy for a given player depends on the strategies of the other players and opponent-independent algorithms such as Minimax-Q, Nash-Q or FF-Q convey no guarantees on the optimality of the chosen strategy. As argued in [166], coordination mechanisms must include some sort of dependence on the strategies of other players and some assumptions must be made about how these other agents will behave.

Baring these ideas in mind, Bowling and Veloso [41] introduce two properties that we would expect from any "intelligent" learner. Such properties are *rationality* and *convergence*.

Rationality

In a MG $\left(N, \mathcal{X}, (\mathcal{A}^k), \mathsf{P}, (r^k), \gamma\right)$, player k is said to be a *rational learner* if it converges to a strategy which is a best response to the strategies of its opponents, given that the other players converge to stationary strategies.

In other words, if all other players follow a stationary strategy (or converge to one), a rational player converges to a *best response strategy*. For example, if the other players follow some sub-optimal strategy, a rational player should be able to take advantage of this fact.

Convergence

In a MG $\big(N, \mathcal{X}, (\mathcal{A}^k), \mathsf{P}, (r^k), \gamma\big)$, player k is said to be a *convergent learner* if it converges to a stationary strategy. This convergence occurs even against players using some class of learning algorithms.

In other words, a convergent learner is able to converge even if the other players in the game also adapt their strategies. This does not happen, for example, in situations where all players keep adapting their strategies to the other players without ever converging.

As argued in Bowling and Veloso [41], many multi-agent RL algorithms possess one of such properties, but not both. For example, Minimax-Q, Nash-Q and FF-Q are convergent but not rational, since they will learn some sort of equilibrium independently of the strategy of the opponents. On the other hand, fictitious play algorithms such as Claus and Boutilier's JAL [59] are rational but not convergent, since the players are only able to play pure strategies.

We now proceed with the analysis of both OAL and CQL with respect to both of these properties. We focus on CQL, although the conclusions hold for both methods.

Let $\Gamma = \big(N, \mathcal{X}, (\mathcal{A}^k), \mathsf{P}, r, \gamma\big)$ be a TMG with $N = 2$ and $|\mathcal{X}| = 1$. We consider this simplified scenario for the sake of clarity, remarking however that the conclusions hold for any number of players and a state-space of any finite dimension.[6]

We start with rationality. Suppose that Player 2 follows a stationary strategy σ^2 and Player 1 follows the CQL algorithm. Player 1 will then learn the following Q-values:

$$Q^1(a^1) = r(a^1, a^2_{\sigma^2}) + \gamma \max_{b^1 \in \mathcal{A}^1} Q^1(b^1),$$

where $a^2_{\sigma^2}$ is Player 2's action according to strategy σ^2. When building the virtual game and sampling from the past history of plays, Player 1 will find that the set of optimal actions consists of Player 1's best response strategies to σ^2 and, therefore, Player 1 will coordinate w.p.1 in a best response strategy to σ^2.

This leads to the following result.

Proposition 7.3.4. *Let $\big(N, \mathcal{X}, (\mathcal{A}^k), \mathsf{P}, r, \gamma\big)$ be a TMG and suppose that, upon a reordering of the player set, all players $2, \ldots, N$ converge to a stationary strategy σ^{-1}. Then, Player 1 following either OAL or CQL will converge to a best response strategy σ^1 to σ^{-1}. In other words, OAL and CQL are both* rational algorithms.

Consider now that Player 2 is a learning agent. In particular, suppose that Player 2 also follows either the OAL or the CQL algorithm. Then, convergence arises as an immediate corollary of Theorems 7.3.2 and 7.3.3, that we summarize in the following proposition.

[6] Since we assume infinite exploration, we can simplify the analysis of the behavior of the algorithms in the whole game to the analysis in a simple state-game. Furthermore, by considering all players $\{2, \ldots, N\}$ as a single *generalized player*, we see that the case of N players easily reduces to the case where $N = 2$.

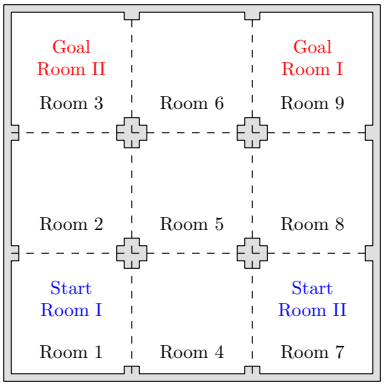


Figure 7.3: Example of an indoor environment.

Proposition 7.3.5. *Let* $\left(N, \mathcal{X}, (\mathcal{A}^k), \mathsf{P}, r, \gamma\right)$ *be a TMG and suppose that all players follow either OAL or CQL. Then, w.p.1, all players converge to a coordinated equilibrium. In other words, OAL and CQL are both* convergent algorithms *in self-play.*

Remark: In the previous proposition we considered convergence in *self-play*. However, convergence of OAL and CQL extends to a broader class of learning algorithms. For example, algorithms such as Nash-Qor FF-Qwill converge to stationarity w.p.1 when playing against OAL or CQL. Therefore, from Proposition 7.3.4, so will OAL and CQL. ◇

7.4 An illustrative example

Consider once again the indoor environment from Example 6.2 and repeated in Figure 7.3 for commodity. The corresponding multi-robot navigation problem can be described by an TMG $\left(N, \mathcal{X}, (\mathcal{A}^k), \mathsf{P}, r, \gamma\right)$ where

- $N = \{\mathrm{I}, \mathrm{II}\}$ is the set of players;
- $\mathcal{X} = \mathcal{X}^{\mathrm{I}} \times \mathcal{X}^{\mathrm{II}}$, where $\mathcal{X}^k = \{1, \ldots, 9\}$ for $k = \mathrm{I}, \mathrm{II}$;
- $\mathcal{A}^k = \{N, S, E, W\}$ for $k = \mathrm{I}, \mathrm{II}$;
- The transition probabilities are defined by a kernel P given by

$$\mathsf{P}_a(i, j) = \mathsf{P}^{(\mathrm{I})}_{a^{\mathrm{I}}}(i^{\mathrm{I}}, j^{\mathrm{I}})\mathsf{P}^{(\mathrm{II})}_{a^{\mathrm{II}}}(i^{\mathrm{II}}, j^{\mathrm{II}})$$

and the kernel $\mathsf{P}^{(k)}$ defines the single-robot transition probabilities (we consider the same individual transition probabilities used in Example 2.1);

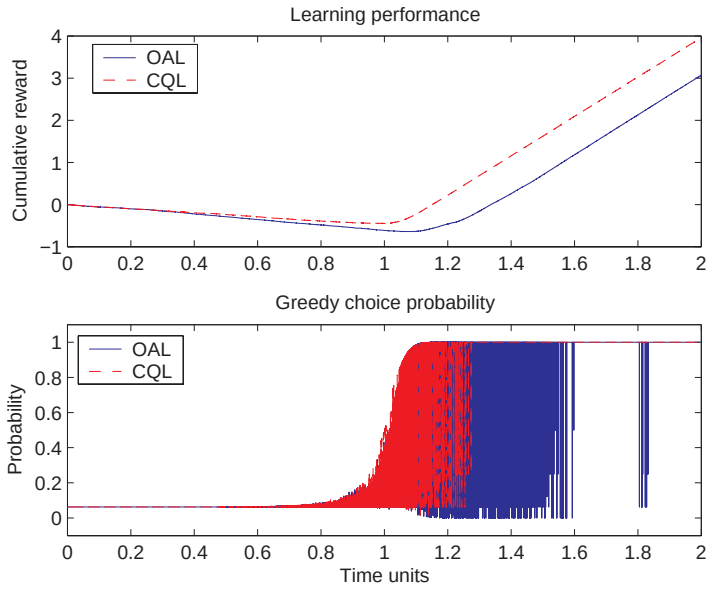


Figure 7.4: Cumulative reward and greedy choice probability during the 2×10^5-time-units learning period.

- The reward function r assigns a reward of $+20$ for every transition triplet (i, a, j) such that $j = (9, 3)$, -10 for every transition triplet (i, a, j) such that $j^{\mathrm{I}} = j^{\mathrm{II}}$ and 0 otherwise;

- We considered $\gamma = 0.95$.

We applied OAL and CQL to this TMG. The group of robots is allowed to explore/learn during 2×10^5 time units and the obtained strategy is then evaluated for 100 time units. During learning, each robot explores with a Boltzmann distributed exploration probability given by

$$p_t = \frac{e^{EP_t(a^k)}}{\sum_{b^k \in \mathcal{A}} e^{EP_t(b^k)}},$$

where a^k is the action prescribed by BAP and p_t and EP_t are as defined in Section 7.3.

Figure 7.4 represents the total undiscounted reward obtained by the groups of robots during learning, as well as the evolution of the probability of choosing the greedy action. The solid blue line corresponds to the group using OAL and the dashed red line corresponds to the group using CQL. Notice that both learning algorithms present a similar learning behavior, evidenced by the similar slope of the learning curve in the rightmost half of the plot. Also notice that, as expected, the probability of choosing the greedy action goes to 1.

We also tested each of the learnt strategies in the environment. We ran both learnt strategies for 100 time units and determined the total discounted reward obtained in each case. Table 7.1 represents the results obtained. We have run $2,000$ independent Monte-Carlo trials and present the average and standard deviation

Table 7.1: Comparative results for OAL and CQL after the learning period is complete. We present the average total discounted reward and standard deviation obtained over 2,000 independent Monte-Carlo trials.

Method	Total Disc. Reward
No coordination	79.739 ± 11.229
OAL	84.543 ± 9.537
CQL	84.034 ± 9.934
Optimal	85.039 ± 9.783

obtained using each of the methods. Notice that both OAL and CQL present a similar performance, as expected, since both methods are expected to converge to the optimal strategy.

For the purpose of comparison, we also present the results obtained with an optimal, centralized controller and by a non-coordinated group of robots. Direct comparison immediately leads to the conclusion that both groups running OAL and CQL clearly outperform the non-coordinated group. Also, both these groups were able to attain optimal performance, which implies that both algorithms yielded convergence to the optimal strategy.

7.5 Concluding remarks

To conclude this chapter, we summarize the main ideas presented so far. We briefly suggest how this framework can be extended to address problems with infinite state-spaces, a topic to be developed in the next chapter.

7.5.1 Summary

In Chapter 6 we described the framework of MGs as a multi-agent counterpart to MDPs. We successfully applied this framework to a simple multi-agent navigation example and outlined several important issues that arise in the presence of multiple decision-makers, such as the *equilibrium selection* problem.

In the present chapter, we started by extending some methods introduced in Chapter 3 to multi-agent situations. By casting TMGs as generalized MDPs, we presented multi-agent versions of ARTQI and Q-learning. These methods can be be used to learn the optimal Q-functions for TMGs, as long as every state-action pair is visited infinitely often.

We then addressed the problem of *equilibrium selection* and combined *biased adaptive play* with the two aforementioned learning methods. The two algorithms thus obtained, OAL and CQL, were shown to converge to the optimal Q-function while ensuring the corresponding players to converge in behavior to a coordinated equilibrium. Finally, we establishing both OAL and CQL to be *rational* and *convergent* in the sense of [41].

We concluded the chapter by illustrating the use of both methods in a familiar topological navigation example.

7.5.2 Discussion

In this chapter, we presented several RL algorithms that can be used to determine the optimal Q-values and ensure coordination in an optimal strategy. Under suitable conditions (identified in Theorems 7.3.2 and 7.3.3), the described algorithms converge asymptotically to the desired Q-function, while ensuring convergence in behavior to a coordinated optimal joint strategy, as long as the learning strategy has the GLIE property.

We now revisit two issues that arose along our presentation and whose detailed discussion was postponed to these concluding remarks. In particular, we elaborate on the topics of strategy evaluation (briefly addressed in Section 7.2) and on-policy methods and coordination (briefly addressed in Section 7.3).

Strategy evaluation in multi-agent systems

In MAS, strategy evaluation can be discussed from the point of view of *one agent* (individual strategy evaluation) or from the point of view of the *group of agents*.

We start by analyzing individual strategy evaluation from one agent's point-of-view. Consider a general Markov game $\left(N, \mathcal{X}, (\mathcal{A}^k), \mathsf{P}, (r^k), \gamma\right)$ and suppose that a given agent $k \in N$ adheres to some fixed individual strategy σ^k. As argued in Chapter 6 it is generally not possible to evaluate the individual strategy σ^k independently of the reduced strategy σ^{-k} followed by the remaining players. Nevertheless, it is possible to evaluate the *worst case performance* of policy σ^k, and assign it a value V^{σ^k}, defined for each state $i \in \mathcal{X}$ as

$$V_{\min}^{\sigma^k}(i) = \min_{\sigma^{-k}} V^{(\sigma^{-k}, \sigma^k)}(i).$$

This measure of performance is useful, for example, in games where the other players have goals that potentially collide with those of player k. By choosing an individual strategy $(\sigma^k)^*$ that maximizes the worst case performance over all possible individual strategies σ^k, player k is able to ensure a total cumulative reward that is no less than $V^{(\sigma^k)^*}$.

The Minimax-Q algorithm [165] uses this concept of worst-case performance to apply Q-learning to zero-sum Markov games. In a more general setting, FF-Q also applies a similar idea to distinguish between *friendly players* and *opposing players*. FF-Q applies a Q-learning update using the idea of worst case performance of the friendly players with respect to the opponent players. More details on these methods can be found in [160, 166, 167].

To analyze strategy evaluation from the point-of-view of the team of agents, consider a MG $\left(N, \mathcal{X}, (\mathcal{A}^k), \mathsf{P}, (r^k), \gamma\right)$ and suppose that the N decision makers adhere to a fixed strategy σ, stochastic or not. The process $\{X_t\}$ will evolve according to

the transition probabilities

$$\mathbb{P}\left[X_{t+1} = j \mid X_t = i\right] = \sum_{a \in \mathcal{A}} \sigma(i,a)\mathsf{P}_a(i,j) = \mathsf{P}_\sigma(i,j),$$

independently of the fact that the action-choice probabilities in σ are "generated" in a distributed fashion. Therefore, there is no conceptual difference between the chain $(\mathcal{X}, \mathsf{P}_\sigma)$ and the chains $(\mathcal{X}, \mathsf{P}_\delta)$ analyzed in Chapter 3 in the context of policy evaluation. Therefore, the policy evaluation methods described in Chapter 3 (such as ARTVI or TD(0)) can be applied exactly as described therein. Each player $k \in N$ independently estimates the corresponding value-function $(V^\sigma)^k$ and the exact same convergence results apply.

It should now be clear that policy evaluation and joint strategy evaluation can be reduced to a unique problem. Therefore, as remarked in Section 7.2, we have presented no strategy evaluation methods in this chapter, as any such methods would reduce to those in Chapter 3.

On-strategy methods and coordination

We now discuss in further detail the possibility of using on-strategy updates while ensuring coordination.

As seen in Chapter 3, on-policy methods learn the value/Q-function associated with the policy used to interact with the environment. Examples of such methods include ARTVI, TD(0) and SARSA. As seen in Chapter 3 and 4, we can straightforwardly apply these methods to perform policy evaluation. However, if an on-policy method is to be used to determine the optimal Q-function, the learning policy must eventually become greedy. This is easily seen from the definition of optimal Q-function: it corresponds to a policy that is greedy with respect to its associated Q-function.

In MAS, however, decision-making is a distributed process. Even if each player in a TMG adopts a greedy strategy, there is no guarantee that the resulting joint strategy is greedy. Therefore, coordination must be ensured before greedily-coordinated action-choice can be implemented.

If coordination can be guaranteed even without complete knowledge of the game (for example, using pre-defined conventions), then on-strategy learning methods can be used to determine the optimal Q-function. As in the single-agent case, a GLIE strategy must be used, but coordination does not depend on the learning of the game and can be implemented independently.

If coordination is meant to emerge as the players learn the game itself, there is the problem described in Section 7.3: while the game is being learnt, there are no guarantees that the players will coordinate in an optimal strategy; on the other hand, while the players do not coordinate in an optimal strategy, the learning process will not converge to the optimal Q-values.

To illustrate this fact, we repeat in Figure 7.5 the results obtained in the example of Section 7.4, where we have included the performance of CQL using a SARSA update instead of the Q-learning update described in Figure 7.2. Table 7.2 summarizes the corresponding numerical results.

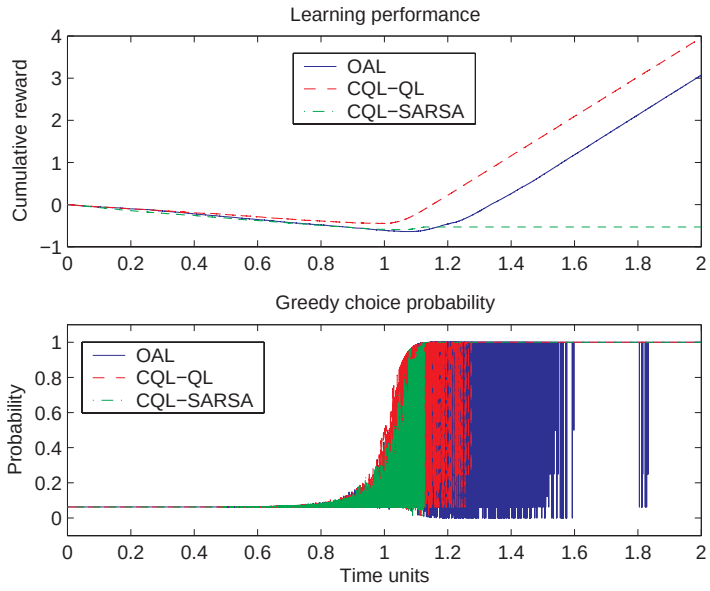


Figure 7.5: Cumulative reward obtained during the 2×10^5-time-units learning period.

Table 7.2: Comparative results for OAL, CQL with Q-learning update, CQL with SARSA update and uncoordinated Q-learning, after the learning period is complete. We present the average total discounted reward and standard deviation obtained over 2,000 independent Monte-Carlo trials.

Method	**Total Disc. Reward**
No coordination	79.739 ± 11.229
OAL	84.543 ± 9.537
CQL (Q-learning update)	84.034 ± 9.934
CQL (SARSA update)	35.856 ± 27.772

We refer to the original CQL algorithm as described in Figure 7.2 as CQL-QL (coordinated Q-learning with a Q-learning update) and to the on-policy version as CQL-SARSA (coordinated Q-learning with a SARSA update).

It is more than evident that the resulting strategy is sub-optimal as are the estimated Q-values: the performance of the team trained using the SARSA update is far inferior to that of the remaining algorithms. Also, the obtained performance is far worse than that obtained without any coordination mechanism. These results allow us to draw two fundamental conclusions:

- The poor performance cannot be justified only by mis-coordinations. If that were the case, we would expect the performance obtained to be closer to that of the uncoordinated algorithm, which is not the case;

- This, in turn, leads to the conclusion that the team converges to some suboptimal strategy and learns the corresponding Q-values.

We can summarize this as follows: CQL with a SARSA update converges to a function Q^σ corresponding to some strategy σ and the agents coordinate in the corresponding policy σ. This policy, however, is generally sub-optimal, as seen in the results from Table 7.2.

* * *

In the next chapter we introduce the final theoretical contributions of the thesis. We follow a similar methodology to the one in this chapter to extend the RL methods from Chapter 4 to MGs with infinite state-spaces. This will require some modification of the BAP coordination mechanism, since we can no longer rely on previous visits to a state to ensure coordination in that state (since there are infinitely many). Therefore, we introduce a similarity function that will allow the players to coordinate by relying on previous visits to *similar states*. This will convey yet another idea on how to build a suitable set of basis functions to use in the function approximation architecture.

Chapter 8

Reinforcement Learning in Infinite Markov Games

In this chapter, we bring out the final theoretical contributions of the thesis, by extending the work in the previous chapter to TMGs with infinite state-spaces. The developments in this chapter build up to two new algorithms and a final result that wraps all contributions in the thesis in a nice, unified result.

We start by briefly going through the main modifications in the TMG framework arising from the fact that the state-space is now considered infinite. We present a straightforward generalization of the convergence theorems in Chapter 4 to multi-agent settings. We address the problem of coordination in this class of games, this leading to the approximate BAP coordination mechanism. We finally produce the most general method in the thesis, coordinated approximate Q-learning, the main contribution of the chapter.

We conclude the chapter with some discussion on the applicability of this framework to problems with partial observability.

8.1 Introduction

In the second part of the thesis we have addressed MAS from a game-theoretic perspective. We adopted TMGs as a model to study the interaction between multiple learning agents and described a coordination mechanism (BAP) that ensures coordination on an optimal strategy. In all this, we admitted several simplificative assumptions:

- The TMG has a finite state-space;
- Each player is able to perceive the current state of the game unambiguously;
- Each player is also able to observe (*a posteriori*) the actions played by all other players.

Once again, the use of topological maps provides an argument in favor of the first assumption. However, the same cannot be said of the two remaining assumptions, as few real decision problems actually verify both assumptions, if any.

In this chapter we discuss how each of these assumptions can be alleviated. The chapter is organized as follows: In Section 8.2 we survey pertinent references and discuss the modifications to fundamental concepts such as Nash equilibria and coordination arising from the fact that we are now considering TMGs with infinite state-spaces (that we henceforth refer as *infinite TMGs*).

We then consider separately the problems of *learning the game* and *learning to coordinate* in infinite games, in a development similar to that in Chapter 7. The problem of *learning the game* is addressed in Section 8.3, where we extend approximate Q-learning to multi-agent settings. In Section 8.4 we address the problem of *learning to coordinate*. We modify BAP to cope with the infinite state-spaces, this leading to the *approximate BAP* coordination mechanism. We then combine ABAP with approximate Q-learning to obtain the *coordinated approximate Q-learning* (CAQL) algorithm. In Section 8.5 we apply CAQL to a simple navigation problem with multiple robots and infinite state-space.

We then discuss the problem of *partial observability* in Section 8.6. We start by considering a simplified setting, where the observations are shared by all agents

(centralized observations). With this simplification, we are able to replicate the arguments in Chapter 4 and convert partially observable problems in equivalent fully observable problems with infinite state-spaces, to which CAQL can be applied. We conclude Section 8.6 by discussing two models from the literature (Dec-POMDPs and I-POMDPs) that address general partially observable problems.

We conclude this chapter in Section 8.7 by applying CAQL to a simple navigation problem with multiple robots and partial observability.

8.2 Infinite state-space Markov games

As discussed in the previous chapters, single-agent RL algorithms have been widely applied (with adequate adaptations) to multi-agent decision problems. It is possible to find numerous multi-agent variations of Q-learning [104, 307], DYNA-Q [334], ARTQI [330] and others [41, 61, 281]. Some approaches consider simplifying assumptions such as deterministic transitions [156], common payoffs [59, 330] or constant-sum payoffs [165].

However, most multi-agent RL research focuses on MGs with small/finite state-spaces. A curious fact that is worth mentioning is that Samuel's pioneer works in machine learning, back in the 1950-60's, already described a multi-agent application with a huge state-space [268, 269]. Also, the impressive results in generalization obtained by Tesauro's backgammon player also feature learning in a game with a huge state-space [307, 308]. Both authors addressed learning in large, competitive decision problems and resort to approximation mechanisms to attain some level of generalization. However, both authors consider learning as an individual process. Few additional contributions to the topic of generalization in multi-agent systems can be found ever since.[1]

Bowling and Veloso [40] apply a combination of the WoLF (win-or-learn-fast) learning strategy with a function approximation mechanism to MGs with large or infinite state-spaces. They experimentally validate this combined method by applying it to the Goofspiel game, which has about 10^{11} states when using an ordinary card deck. Kok et al. [143] describe the application of coordination graphs to multiagent scenarios with continuous domain. Singh et al. also refer the interest of applying the gradient ascent techniques described in [281] to games with infinite state-spaces.

In this chapter, we contribute another step in the research of solution methods for infinite MG, by applying the approximate Q-learning algorithm from Section 4.5 to this class of problems. We also extend the BAP coordination mechanism to infinite TMGs. The combination of both methods will lead to the *coordinated approximate Q-learning algorithm*. As will soon become apparent, some of the valuable aspects of this new method are its sound convergence properties and broad applicability.

[1] We remark that in path-planning applications, several multi-robot *planning* algorithms are available. In such applications, the robots act in a usually infinite configuration space [158, 328].

8.2.1 Markov games with infinite state-spaces

Recall from Chapter 4 that an infinite MDP is a tuple $(\mathcal{X}, \mathcal{A}, \mathsf{P}, r, \gamma)$ where $\mathcal{X}$ is a compact subset of $\mathbb{R}^p$, $\mathcal{A}$ is a finite set of actions, P is a kernel defining the transition probabilities

$$\mathbb{P}\left[X_{t+1} \in U \mid X_t = x, A_t = a\right] = \mathsf{P}_a(x, U)$$

and $r : \mathcal{X} \times \mathcal{A} \times \mathcal{X}$ is the reward function assigning the decision-maker a reward $r(x, a, y)$ every time a transition from x to y occurred under action a. We defined the optimal value function V^* as verifying the modified Bellman optimality equation

$$V^*(x) = \max_{a \in \mathcal{A}} \int_{\mathcal{X}} \big[r(x, a, y) + \gamma V^*(y)\big] \mathsf{P}_a(x, dy).$$

and the optimal Q-values $Q^*(x, a)$ as

$$Q^*(x, a) = \int_{\mathcal{X}} \big[r(x, a, y) + \gamma V^*(y)\big] \mathsf{P}_a(x, dy).$$

We now consider a set of N independent decision-makers (players), each player k with a repertoire $\mathcal{A}^k$ of individual actions and define $\mathcal{A} = \times_{k=1}^{N} \mathcal{A}^k$. The tuple $(\mathcal{X}, \mathcal{A}, \mathsf{P}, r, \gamma)$ is equivalent to the one considered above except for the fact that $\mathcal{A}$ is now a cartesian product of the individual action-spaces $\mathcal{A}^k$. As in Chapter 7, the tuple $\big(N, \mathcal{X}, (\mathcal{A}^k), \mathsf{P}, r, \gamma\big)$ thus obtained is an *infinite team Markov game*. The main differences from TMGs considered in the previous chapters lie on the fact that $\mathcal{X}$ is now a compact subset of $\mathcal{X}$, not necessarily finite, and P is a transition probability kernel instead of a transition probability matrix.

As in Chapter 7, the close relation between TMGs and MDPs will allow us to adapt approximate Q-learning to multi-agent scenarios.

8.2.2 Equilibria

The existence of stationary Nash equilibria in general-sum MGs with infinite (uncountable) state-spaces is a topic of ongoing research in the game theory community, as argued in [223], and remains an open question for all but a few particular classes of games.

In the class of games considered here, the problem of establishing the existence of equilibria is greatly simplified: the fact that the action set is finite, the rewards are bounded and the game is fully cooperative (*i.e.*, all players share the same joint goal) greatly facilitates the task of establishing the existence of a stationary Nash equilibrium.

Before proceeding any further, we should remark that concepts such as individual action/strategy, joint action/strategy, reduced action/strategy, pure strategy, mixed strategy or stationary strategy carry from those in Chapter 6 without any change. The concepts of best-response and Nash equilibrium also carry without any change from those in Chapter 6. We can now introduce the following result.

Theorem 8.2.1. *Every team Markov game* $\Gamma = \big(N, \mathcal{X}, (\mathcal{A}^k), \mathsf{P}, r, \gamma\big)$*, with finite action-space* $\mathcal{A}$ *and compact state-space* $\mathcal{X} \subset \mathbb{R}^p$*, has a pure stationary Nash equilibrium.*

Proof See Appendix F. □

One final remark to emphasize that not only does at least one equilibrium always exist in the conditions of Theorem 8.2.1, but also at least one such equilibria is a coordinated equilibrium. This is immediate from the proof of Theorem 8.2.1 that can be found in Appendix F.

Coordination and equilibrium selection

As in finite TMGs, the existence of at least one coordinated equilibrium does not imply its uniqueness. In fact, many such games possess multiple equilibria and when that is the case we are, once again, faced with a coordination problem.

As in the finite case, it is necessary to consider some coordination mechanism to ensure coordination. Even if all players know the game, one must still devise some specific mechanism to ensure that, in the presence of multiple equilibria, all players commit to the same equilibrium. As in Chapter 6, we are interested in an adaptive mechanism that ensures coordination to *emerge* as the players repeatedly play the game one and yet another time. We address the problem of *learning to coordinate* in Section 8.4.

8.3 Learning the game

In this section, we address the problem of learning/approximating the optimal Q-function in TMGs with infinite state-spaces. As in Chapter 7, we look at TMGs as MDPs with a distributed decision-making process. This means that we can apply the approximate learning algorithms from Chapter 4 to TMGs with infinite state-spaces, as long as the conditions listed therein are verified.

Before proceeding any further, three important remarks are in order. First of all, as in finite Markov games, strategy evaluation is conceptually equivalent to policy evaluation in MDPs. Therefore, we do not address strategy evaluation and refer to the methods in Chapter 4, which can readily be adapted to handle multi-agent scenarios.

Secondly, unlike all previous chapters, we focus on model-free learning algorithms and establish the applicability of the approximate Q-learning algorithm to multi-agent problems. Later, as we extend BAP to infinite settings, it will become apparent the natural connection between the two methods and a combined method will naturally arise. The combination of this extended version of BAP with model-based learning methods such as kernel-based RL is less immediate and would require additional work that we do not pursue here. It is, however, an interesting topic to address in future research.

Finally, the approximate SARSA algorithm, being an on-strategy algorithm, exhibits the disadvantages discussed in the previous chapter when coordination is concerned. Since our ultimate goal is to address coordination problems, we refrain from discussing its applicability to multi-agent settings, referring to the discussion in Chapter 7 on on-strategy methods.

8.3.1 Approximate Q-learning updates in infinite Markov games

Let $\Gamma = \big(N, \mathcal{X}, (\mathcal{A}^k), \mathsf{P}, r, \gamma\big)$ be a TMG with infinite state-space $\mathcal{X}$, admittedly a compact subset of $\mathbb{R}^p$. We want to determine the optimal Q-function for this game, verifying the following recursive relation

$$Q^*(x,a) = \int_{\mathcal{X}} \big[r(x,a,y) + \gamma \max_{b\in\mathcal{A}} Q^*(y,b)\big]\mathsf{P}_a(x,dy). \tag{8.1}$$

The recursive relation in (8.1) is nothing but a multi-agent version of the one in (4.11) (see Chapter 4, page 69), where the only difference lies on the structure of the underlying action-space. This means that we can apply the approximate Q-learning method to approximate Q^* by looking at Γ as an MDP with a distributed decision-maker.

We briefly review the approximate Q-learning algorithm. Consider a linear family of functions $\mathcal{Q} = \{Q_\theta\}$ parameterized by a finite-dimensional vector $\theta \in \mathbb{R}^M$. Since $\mathcal{Q}$ is the linear span of a set of M linearly independent functions $\xi_i : \mathcal{X}\times\mathcal{A} \longrightarrow \mathbb{R}$, each $Q_\theta \in \mathcal{Q}$ can be written as

$$Q_\theta(x,a) = \sum_{i=1}^{M} \xi_i(x,a)\theta(i) = \xi^\top(x,a)\theta.$$

Let $\{x_t\}$, $\{a_t\}$ and $\{r_t\}$ be sample trajectories obtained from the TMG Γ by following some joint strategy σ. Then, given any initial parameter θ_0, we define the sequence $\{\theta_t\}$ recursively as

$$\theta_{t+1} = \theta_t + \alpha_t \xi(x_t,a_t)\Delta_t,$$

where Δ_t is the temporal difference

$$\Delta_t = r_t + \gamma \max_{b\in\mathcal{A}} Q_{\theta_t}(x_{t+1},b) - Q_{\theta_t}(x_t,a_t).$$

The only difference between applying approximate Q-learning to MDPs or to TMGs lies on the fact that, in TMGs, the action sequence $\{A_t\}$ is generated in a distributed fashion by the N players in the game. This does not affect in any way the convergence of the algorithm and the sequence θ_t will converge w.p.1 to a limit θ^* such that the corresponding function $Q(\theta^*)$ verifies

$$Q(\theta^*) = \mathcal{P}_{\mathcal{Q}}\mathbf{H}Q(\theta^*),$$

where $\mathcal{P}_{\mathcal{Q}}$ is the orthogonal projection into $\mathcal{Q}$ and $\mathbf{H}$ is defined in (4.3).

8.3.2 On convergence

We start by suitably reformulating Theorem 4.5.2, encompassing the framework of MGs.

Let σ be a stationary joint strategy and $(\mathcal{X}, \mathsf{P}_\sigma)$ the corresponding Markov chain with invariant probability measure μ_X. Denote by $\mathbb{E}_{\mu_\sigma}[\cdot]$ the expectation w.r.t. the probability measure μ_σ defined for every set $Z \times U \subset \mathcal{X} \times \mathcal{A}$ as

$$\mu_\sigma(Z \times U) = \int_Z \sum_{a \in U} \sigma(x,a)\mu_X(x).$$

Theorem 8.3.1. *Let $\big(N, \mathcal{X}, (\mathcal{A}^k), \mathsf{P}, r, \gamma\big)$ be a team Markov game with compact state-space $\mathcal{X} \subset \mathbb{R}^p$. Assume the Markov chain $(\mathcal{X}, \mathsf{P}_\sigma)$ to be geometrically ergodic with invariant probability measure μ_X, where P_σ is the transition kernel for the chain obtained when all players follow a stochastic strategy σ verifying $\sigma(x,a) > 0$ for all $a \in \mathcal{A}$ and μ_X-almost all $x \in \mathcal{X}$.*

Let $\Xi = \{\xi_i, i = 1, \dots, M\}$ be a set of M bounded, linearly independent functions defined on $\mathcal{X} \times \mathcal{A}$ and taking values in $\mathbb{R}$. In particular, admit that $\sum_{i=1}^N |\xi_i(x,a)| \leq 1$ for all $(x,a) \in \mathcal{X} \times \mathcal{A}$.

Then, the following hold.

1. ***Convergence***

   For any initial condition $\theta_0 \in \mathbb{R}^M$, the algorithm

   $$\theta_{t+1} = \theta_t + \alpha_t \xi(x_t, a_t)\big(r_t + \gamma \max_{b \in \mathcal{A}} Q_{\theta_t}(x_{t+1}, b) - Q_{\theta_t}(x_t, a_t)\big) \tag{8.2}$$

   converges w.p.1 as long as the step-size sequence $\{\alpha_t\}$ verifies

   $$\sum_t \alpha_t = \infty \qquad \sum_t \alpha_t^2 < \infty.$$

2. ***Limit of convergence***

   Under these conditions, the limit θ^* of (8.2) verifies

   $$Q_{\theta^*}(x,a) = (\mathcal{P}_\mathcal{Q} \mathbf{H} Q_{\theta^*})(x,a),$$

   where $\mathcal{P}_\mathcal{Q}$ denotes the orthogonal projection operator on $\mathcal{Q}$ defined by

   $$(\mathcal{P}_\mathcal{Q} Q)(x,a) = \xi^\top(x,a)\Sigma^{-1}\mathbb{E}_{\mu_\sigma}\left[\xi(z,u)Q(z,u)\right].$$

   and the matrix Σ is given by

   $$\Sigma = \mathbb{E}_{\mu_\sigma}\left[\xi(x,a)\xi^\top(x,a)\right].$$

Proof See Appendix F.4. □

For our purposes, we will also need a multi-agent version of Corollary 4.5.4, that we provide next.

Corollary 8.3.2. *Let* $\left(N, \mathcal{X}, (\mathcal{A}^k), \mathsf{P}, r, \gamma\right)$ *be a TMG with compact state-space* $\mathcal{X} \subset \mathbb{R}^p$ *and let* σ_θ *be a* θ*-dependent joint strategy such that, for all pairs* (x, a)*,* $\sigma_\theta(x, a) > 0$ *and*

$$|\sigma_\theta(x, a) - \sigma_{\theta'}(x, a)| \leq C \left\|\theta - \theta'\right\| \tag{8.3}$$

for some constant $C > 0$ *independent of* x *and* a*. Assume that the Markov chain obtained by following the strategy* σ_θ*, denoted as* $(\mathcal{X}, \mathsf{P}_\theta)$*, is geometrically ergodic for each* θ*, with invariant probability measure* μ_X^θ*.*

Then, all assertions of Theorem 8.3.1 hold as long as the conditions on Ξ *and on the sequence* $\{\alpha_t\}$ *are verified.*

The two previous results establish the convergence of approximate Q-learning in TMGs. This constitutes the first step in building an extension of the methods in Chapter 7 to TMGs with infinite state-spaces. We next proceed with the problem of coordination.

8.4 Learning to coordinate

As approximate Q-learning handles the problem of *learning the game*, we now address the problem of *learning to coordinate*. We discuss *convergence in behavior*, as we extend BAP to cope with TMGs with infinite state-spaces.

8.4.1 Biased adaptive play in infinite Markov games

Recall the BAP mechanism described in Chapter 7. This coordination mechanism uses incomplete samples from the history of past plays to estimate the average strategies of the players in the game. These estimates can then be used to choose a best response strategy, as long as the game is known.

In standard BAP, coordination at some given state requires that such state be visited a sufficient number of times to ensure that (i) adequate action sampling can take place and (ii) there is sufficient time to attain coordination. The successive visits to each state provide each player with a sample of the other players' strategies in that particular state and hence the requirement that every state be visited infinitely often.

Formally, the condition of infinite visits amounts to requiring the underlying Markov chain to be irreducible (every state is "visitable") and recurrent (each "visitable" state is visited infinitely often). In the infinite state-space case, we require the underlying Markov chain to be ψ-irreducible (all but a negligible part of the state-

space is "visitable") and Harris recurrent (every "visitable" region of the state-space is visited infinitely often), but we discuss this further ahead.[2]

Consider a TMG with a compact state-space $\mathcal{X} \subset \mathbb{R}^p$. Due to the infinite nature of the state-space, it is generally impossible to ensure that any particular state in the state-space is visited infinitely often. Therefore, we cannot apply standard BAP as described in Chapter 7 to infinite state-space setting. The reason behind this impossibility is easy to grasp: BAP relies on past plays of the game at each state, and there is the possibility that a particular state has never been visited before, no matter for how long the game has already been played.

In adapting BAP to cope with infinite state-spaces, coordination at each state should rely not only in past visits to that particular state but should also use the information provided by plays in *several nearby states*. The intuition behind this idea can be easily clarified. A player k can no longer use the past history at a particular state x to infer the other players' strategy in that state, since there is the possibility that it was never visited before. Instead, player k will assume that *the strategy of the other players in the states close to x does not change significatively*. If this assumption holds, player k can use the past history at nearby states to estimate the strategy of the other players *at state x*.

To implement this idea, we make use of a *similarity function* ϕ, indicating whether two states x and y in $\mathcal{X}$ are "similar". As will soon become apparent, the use of such approximation mechanism suitably adapts BAP to TMGs with infinite state-spaces while ensuring coordination in all but a negligible part of the state-space.

8.4.2 Approximate coordination

We now describe *approximate biased adaptive play* (ABAP) and establish its convergence w.p.1. To this purpose, we focus on the coordination process by looking at a TMG from a particular point of view that allows us to disregard several technicalities concerning the underlying Markov chain and facilitates the proof of our convergence result.

Let $\Gamma = \big(N, \mathcal{X}, (\mathcal{A}^k), \mathsf{P}, r, \gamma\big)$ be a team Markov game with compact state-space $\mathcal{X} \subset \mathbb{R}^p$ and finite joint action-space $\mathcal{A}$. Let Q^* be the optimal Q-function for Γ and define, for each $x \in \mathcal{X}$, the team matrix game $\Gamma_x^* = \big(N, (\mathcal{A}^k), Q^*(x, \cdot)\big)$. To introduce and analyze ABAP, we resort to an auxiliary process $\{Y_t\}$ evolving in $\mathcal{X}$. We assume this process $\{Y_t\}$ to be a ψ-irreducible and Harris recurrent Markov chain.[3]

At each time instant t, N players engage in the repeated game $\Gamma_{Y_t}^*$ where Y_t is the state of the auxiliary process $\{Y_t\}$ at time t. The sole purpose of the agents is to coordinate in an optimal policy in each state-game Γ_x^*; the agents have no knowledge otherwise on the Markov game Γ or on the auxiliary process $\{Y_t\}$, and consider the

[2]For a formal definition of ψ-irreducibility and Harris recurrence we refer to Appendix B.

[3]As mentioned above and further detailed in Appendix B, a chain is ψ-irreducible if all but a negligible part of the state-space is "visitable". It is Harris recurrent if every "visitable" region of the state-space is visited infinitely often.

payoffs $Q^*(x,\cdot)$ at different state-games Γ_x^* to be independent.

This technical artifice allows us to discard the effect of the joint actions of the agents on the state evolution of the Markov game. The agents merely visit the states in $\mathcal{X}$ along the trajectories of $\{Y_t\}$ and coordinate in each visited stage-game Γ_x^*. The effects of the agents' actions on the state evolution of the game is addressed later on.

Consider now some p-norm to be defined in $\mathcal{X}$. We define a *similarity function* as any continuous function $\phi : \mathcal{X} \times \mathcal{X} \longrightarrow [0,1]$ verifying:

a) $\phi(x,y) = 1$ if and only if $\|x-y\|_p = 0$;

b) $\phi(x,y) \to 0$ as $\|x-y\|_p \to \infty$;

c) $\phi(x,y) = \phi(y,x)$ for all $x, y \in \mathcal{X}$;

d) If $\|x-y\| < \|x-z\|$ then $\phi(x,y) > \phi(x,z)$.

Because of c and d, similarity functions will generally be defined as inverse functions of the distance between the two arguments. An example of one possible similarity function is

$$\phi(x,y) = \frac{1}{1+\|x-y\|}.$$

The similarity function provides a criterion that defines when two states x and y are "similar".

Consider the past history up to time t,

$$\mathcal{H}_t = \{(y_0,a_0),(y_1,a_1),\ldots,(y_{t-1},a_{t-1})\},$$

where the sequence $\{y_t\}$ is a sample trajectory of the process $\{Y_t\}$ and each joint action a_t corresponds to that chosen by the agents in game $\Gamma_{y_t}^*$. At each time instant $t \in \mathcal{T}$, each player determines the similarity $\phi(y_i, Y_t)$ between the current state Y_t and each state y_i occurring in $\mathcal{H}_t$. It then chooses m occurrences from this history so as to maximize the corresponding similarity. The sample set thus obtained, denoted as $S_m(Y_t, \mathcal{H}_t)$, contains the m elements in $\mathcal{H}_t$ maximizing the total similarity with Y_t,

$$\sum_{i=1}^{m} \phi(Y_t, y_{t_i}).$$

We remark that a particular state $x \in \mathcal{X}$ may occur in $S_m(Y_t,\mathcal{H}_t)$ more than once. On the other hand, if two occurrences y_{t_i} and y_{t_j} verify

$$\phi(Y_t, y_{t_i}) = \phi(Y_t, y_{t_j})$$

and only one such occurrence must be chosen, then the most recent one should be picked (*e.g.*, if $t_j > t_i$ above, then y_{t_j} would be chosen). We also notice that, due to the ψ-irreducibility and Harris recurrence of the Markov chain, given any state $x \in \mathcal{X}$ and a corresponding neighborhood U with positive ψ-measure, there is a time T_0 such that, w.p.1, $S_m(x, H_t) \subset U$ for $t > T_0$.

Once the set $S_m(Y_t, \mathcal{H}_t)$ is determined, the corresponding m plays can now be used to draw a K-sample and proceed as in standard BAP.

The following proposition establishes the convergence of ABAP.

Theorem 8.4.1. *Let $\{Y_t\}$ be a Markov chain evolving on $\mathcal{X}$ as described above. In particular, assume that the chain is ψ-irreducible and Harris recurrent. Let N be a set of players engaging in the coordination game described above and following ABAP. Suppose that the function Q^* is continuous in x in all but a ψ-null set of states. Then the players in N coordinate in an optimal Nash equilibrium w.p.1 in ψ-almost every state in $\mathcal{X}$, as long as the conditions for convergence of standard BAP are met.*

Proof See Appendix F. □

Notice that Theorem 8.4.1 is somewhat more restrictive than its finite counterpart, as it requires ψ-a.e. continuity of Q^*. However, this condition simply ensures that the function Q^* is relatively "well-behaved", so that coordination at a given point x can be achieved by observing the past plays in points "sufficiently close" to x.

In the remainder of this section we combine the learning of the game described in the previous section with approximate biased adaptive play. This is achieved in a similar way to that pursued in Chapter 7: Theorem 8.4.1 arises as a consequence of the convergence of BAP and Lemma F.3.3, used in the proof of Theorem 7.3.3, also has a straightforward extension to Markov games with infinite state-spaces.

8.4.3 Convergence in behavior

In this subsection, we contribute the last new algorithm in the thesis, which we refer as *coordinated approximate Q-learning* (CAQL). As anticipated in the previous subsection, this algorithm combines approximate Q-learning and ABAP. With sufficient exploration, CAQL guarantees that the estimates Q_{θ_t} converge to a suitable approximation Q_{θ^*} while guaranteing convergence of the players' strategies to an optimal Nash equilibrium w.r.t. Q_{θ^*}.

The basic procedure of CAQL is as follows. At each time instant $t \in \mathcal{T}$, each player $k \in N$ determines the set $S_m(X_t, \mathcal{H}_t)$ using the similarity function ϕ and draws a K-sample h from $S_m(X_t, \mathcal{H}_t)$. This K-sample is used to determine the expected payoff of each action $a^k \in \mathcal{A}^k$ w.r.t. the virtual game VG_t obtained from Q_{θ_t} and the corresponding best response action $(a^k)^*$, unless if the two BAP conditions described in Chapter 7 are met. For commodity, we repeat such conditions here:

1. There is a joint action $a^* \in D$ such that, for all actions $a \in h$, $a^{-k} = (a^*)^{-k}$; and

2. there is at least one action $a^* \in D$ such that $a^* \in h$.

In the conditions above, D is the set of all ε-optimal actions w.r.t. Q_{θ_t} at state X_t. Notice that, even if the virtual game VG_t obtained from Q_{θ_t} can not be stored in memory due to the fact that $\mathcal{X}$ is infinite, player k can easily determine it from Q_{θ_t} as needed. Once all individual actions A_t^k are chosen, yielding the joint action A_t, the game moves to a new state X_{t+1} according to the probabilities in P and all players receive the corresponding reward $r(X_t, A_t, X_{t+1})$. All players now use the observed transition $(X_t, A_t, r(X_t, A_t, X_{t+1}), X_{t+1})$ to update each parameter vector θ_t according to (8.2).

Figure 8.1 summarizes the coordinated approximate Q-learning (CAQL) algorithm for one player.

Before addressing the convergence of CAQL, we recall that in Chapter 7 we used the condition

$$\lim_{t\to\infty} \frac{\sqrt{\frac{\log\log(N_t)}{N_t}}}{B(N_t)} = 0 \tag{8.4}$$

to guarantee that ε_t does not decrease too fast towards zero, thus ensuring that no optimal action is ruled out too soon. This condition arose from the known rates of convergence of Q-learning and model-based learning that can be found in [137, 298, 330].

In order to establish convergence of CAQL, we need to derive a similar condition for approximate Q-learning. To that purpose, we resort to a known result on the rates of convergence of stochastic approximation algorithms, described in Appendix D (Theorem D.2.1 in page 292). This theorem, known as the *law of iterated logarithm* for general stochastic approximation processes, states that

$$\limsup_{t\to\infty} \frac{\|\theta_t - \theta^*\|}{\sqrt{\alpha_t \log\left(\sum_{\tau=1}^{t} \alpha_\tau\right)}} \leq K_0, \tag{8.5}$$

where $\{\alpha_t\}$ is the sequence of step-sizes. The bound in (8.5) simply states that the maximum error between the estimated parameter θ_t and the limit parameter θ^* approximately decays with $\sqrt{\alpha_t \log\left(\sum \alpha_\tau\right)}$.

We can readily show that the conditions of Theorem D.2.1 hold for the sequence $\{\theta_t\}$ obtained using the CAQL algorithm: the first condition arises as a consequence of the global asymptotic stability of θ^* as an equilibrium point of the ODE

$$\dot{\theta}_t = h(\theta_t);$$

The second condition arises from the considered step-sizes; the third condition is trivially verified. Therefore, the rate of convergence of θ_t to θ^* verifies the bound in (8.5) and condition (8.4) should be adapted to yield

$$\lim_{t\to\infty} \frac{\sqrt{\frac{\log\left(\sum_{\tau=1}^{t} \frac{1}{n_\tau}\right)}{n_t}}}{B(n_t)} = 0. \tag{8.6}$$

Initialization:

1: **Set** $t = 0$, $\varepsilon_t = \varepsilon_0$ and $\theta_t^k(i) = 0$;

Learning coordination: Given current state X_t

2: **If** $t \leq m$, randomly select an action

3: **else** with GLIE exploitation probability $p_t(a^*)$ **do**

a. **Determine** VG_t as

$$VG_t(X_t, a) = \begin{cases} 1 & \text{if } a \in \mathbf{opt}^{\varepsilon_t}(X_t); \\ 0 & \text{otherwise;} \end{cases}$$

b. **Set** $D = \{a \mid VG_t(X_t, a) = 1\}$;

c. **Set** $H_t = S_m(X_t, \mathcal{H}_t)$;

d. **Set** $h = K\text{-sample}(K, H_t)$;

e. **For all** $a^k \in \mathcal{A}^k$, **set**

$$EP_t(a^k) = \sum_{a^{-k} \in \mathcal{A}^{-k}} VG_t\big(X_t, (a^{-k}, a^k)\big) \frac{n_h(a^{-k})}{K};$$

f. **Set** $BR_t(X_t) = \big\{a^k \mid a^k = \arg\max_{b^k \in \mathcal{A}^k} EP_t(b^k)\big\}$;

g. **If** conditions 1 and 2 above are met, choose the most recent joint action in $h \cap D$;

h. **else** randomly choose an action in $BR_t(X_t)$;

4: **And** with exploration probability $p_t(a)$ select action a;

Learning the game: Given current transition triplet (X_t, A_t, X_{t+1})

5: **Update** θ_t according to (8.2), with $\alpha_t = \frac{1}{t+1}$;

6: **Set** $t = t + 1$;

7: **If** $\varepsilon_t \geq \varepsilon_0 B(t)$, **set** $\varepsilon_t = \varepsilon_0 B(t)$;

Figure 8.1: The CAQL algorithm for one player.

Notice that the expression above is obtained by considering in (8.4) the actual convergence rate for approximate Q-learning obtained from (8.5). However, noticing that

$$\sum_{i=1}^{n} \frac{1}{i} \approx \log(n),$$

we can conclude that

$$\lim_{t\to\infty} \frac{\sqrt{\frac{\log(\log(n_t))}{n_t}}}{B(n_t)} = 0$$

implies (8.6).[4] With this stated, the following theorem immediately follows.

Theorem 8.4.2. *Let $\left(N, \mathcal{X}, (\mathcal{A}^k), \mathsf{P}, r, \gamma\right)$ be a TMG with compact state-space $\mathcal{X} \subset \mathbb{R}^p$ and finite action-space $\mathcal{A}$. Let $(\sigma_\theta)_t$ be a θ-dependent learning strategy obtained from CAQL with the GLIE property w.r.t. the estimate $Q_\theta(x,a) = \xi^\top(x,a)\theta$ and such that $(\sigma_\theta)_t(x,a) > 0$ for all (x,a). Further assume that such strategy verifies, for all pairs (x,a),*

$$|\sigma_\theta(x,a) - \sigma_{\theta'}(x,a)| \leq C \left\|\theta - \theta'\right\| \tag{8.7}$$

for some constant $C > 0$ independent of x and a. Assume that the Markov chain obtained by following the strategy $(\sigma_\theta)_t$, denoted as $(\mathcal{X}, \mathsf{P}_\theta)$, is geometrically ergodic for each θ, with invariant probability measure μ_X^θ. Further assume that

1. *For each θ, the reward function r is continuous μ_X^θ-a.e.;*
2. *The function $B(t)$ decreases monotonically to zero and verifies*

$$\lim_{t\to\infty} \frac{\sqrt{\frac{\log\log(t+1)}{t}}}{B(t+1)} = 0. \tag{8.8}$$

3. *The cardinality m of the sets $S_m(x, \mathcal{H}_t)$ and the length K of the K-sample verify $m \geq K(N+2)$.*

[4] The sum of the first n terms of the harmonic series is given analytically by the n^{th} harmonic number,

$$\sum_{i=1}^{n} \frac{1}{i} = c + \Psi_0(n+1),$$

where c is the Euler-Mascheroni constant and Ψ_0 is the digamma function. On the other hand,

$$\lim_{x\to\infty} \frac{\Psi_0(x)}{\log(x)} = 1$$

implies that

$$\lim_{n\to\infty} \frac{\sum_{i=1}^{n} \frac{1}{i}}{\log(n)} = 1.$$

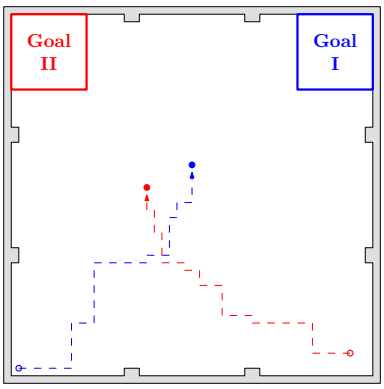


Figure 8.2: Example of a continuous indoor environment.

Then, the sequence $\{\theta_t\}$ generated by CAQL converges w.p.1 to the parameter vector θ^* described in Theorem 8.3.1. Furthermore, all players in N converge in behavior w.p.1 to a common coordinated Nash equilibrium w.r.t. Q_{θ^*}.

Proof See Appendix F. □

One final remark to emphasize that condition (8.7) restricts the way the GLIE strategy is implemented. In particular, notice that in Algorithm 8.1 we have allowed the exploration/exploitation probabilities p_t to depend on the action to be played. For example in line 3, $p_t(a^*)$ represents the probability associated with an optimal joint action. This dependence is made explicit to emphasize the fact that this probability cannot change abruptly, as required by (8.7)

8.5 An illustrative example

We now analyze a continuous version of Example 6.2 explored in the previous chapters. Consider the indoor environment depicted in Figure 8.2. Two mobile robots (I and II) are intended to navigate to the corresponding goal regions, signaled with the bold, colored lines. Each robots starts in the corner diagonally opposite to its goal. The environment is a 1×1 square, and the state of each robot at each time instant is a pair $(\mathbf{x}, \mathbf{y})$ of coordinates.[5] The coordinates of the corners in the goal regions are $(1, 1)$ and $(0, 1)$, respectively, and the corresponding goal regions are 0.1×0.1 squares, as depicted in Figure 8.2. We denote the goal region for robot k by G^k and by G the cartesian product of G^{I} and G^{II}, *i.e.*, $G = G^{\mathrm{I}} \times G^{\mathrm{II}}$. In their trajectories, the robots must avoid crashing into each other, by not being in the same 0.1×0.1-area simultaneously (see Figure 8.3 for an illustration).

We denote the state of robot k at time t by X_t^k. The state of the robot group is a pair $X_t = (X_t^A, X_t^B)$ and can take any value in $([0; 1] \times [0; 1]) \times ([0; 1] \times [0; 1])$.

[5]Once again, we use boldface symbols $\mathbf{x}$ and $\mathbf{y}$ to denote the physical coordinates of one robot to distinguish from the symbols x and y used to denote generic elements of the state-space $\mathcal{X}$.

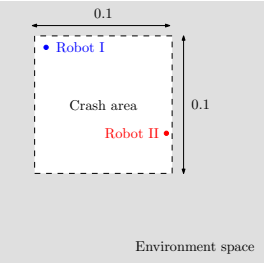


Figure 8.3: Situation of possible crash.

Each robot has 4 actions available, namely N, S, E and W. Each individual action moves the robot in the corresponding direction of a uniform random amount between 0 and 0.3. We consider the movements of the robots to be independent of each other.

This navigation problem can easily be modeled using a TMG $\left(N, \mathcal{X}, (\mathcal{A}^k), \mathsf{P}, r, \gamma\right)$ where

- $N = \{\mathrm{I}, \mathrm{II}\}$ is the set of players;
- $\mathcal{X} = ([0;1] \times [0;1]) \times ([0;1] \times [0;1])$;
- $\mathcal{A}^k = \{N, S, E, W\}$ for $k = \mathrm{I}, \mathrm{II}$;
- The transition probabilities are defined by a kernel P given by

$$\mathsf{P}_a(x, U) = \mathsf{P}^{(\mathrm{I})}_{a^{\mathrm{I}}}(x^{\mathrm{I}}, U^{\mathrm{I}})\mathsf{P}^{(\mathrm{II})}_{a^{\mathrm{II}}}(x^{\mathrm{II}}, U^{\mathrm{II}})$$

  and the kernels $\mathsf{P}^{(k)}$ define the single-robot transition probabilities according to the description above;

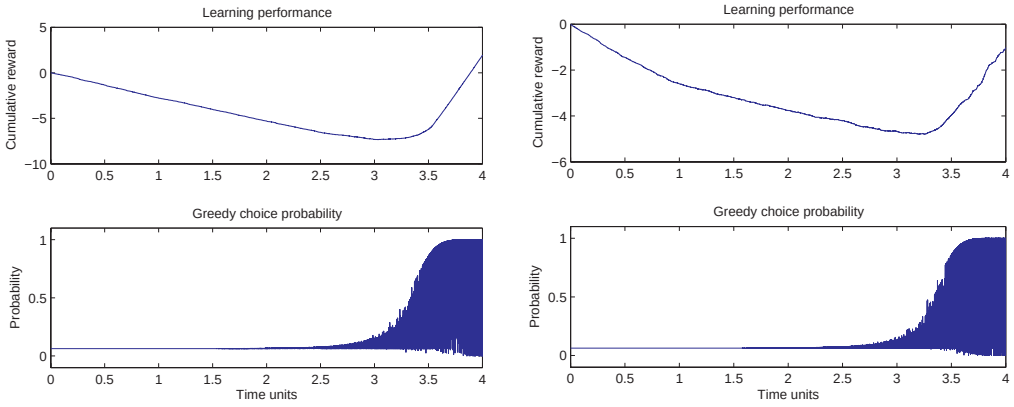
- The reward function r is defined as

$$r(x, a, y) = \begin{cases} 20 & \text{if } y \in G; \\ -10 & \text{if } \left\|y^{\mathrm{I}} - y^{\mathrm{II}}\right\|_\infty < 0.1; \\ 0 & \text{otherwise;} \end{cases}$$

- We consider $\gamma = 0.95$.

We applied CAQL to this Markov game. The agents were allowed to explore and learn during 4×10^5 time steps, and the obtained policy was then evaluated for 100 time units. During learning, *each robot* uses a Boltzmann distributed exploration strategy over the joint state-space, given by

$$p_t(a) = \frac{e^{Q_t(X_t, a)/\tau}}{\sum_{b \in \mathcal{A}} e^{Q_t(X_t, b)/\tau}},$$



a) Grid-based; **b)** Kernel-based.

Figure 8.4: Cumulative reward and greedy choice probability during the 4×10^5-time-units learning period.

where τ is the temperature parameter. The robot will play the corresponding individual action. Exploitation occurs if the chosen joint action corresponds to a coordinated equilibrium.

We ran two experiments, each using a different set of basis functions, to be found in page 370. In Figure 8.4 we depict the total reward obtained during learning in each of the experiments. In Table 8.1 we present the total discounted reward obtained during the 100 time-steps test period.

Figure 8.4 depicts the learning curves for both experiments (corresponding to different sets of basis functions), corresponding to the total reward obtained during learning, and the probability of choosing the "greedy" action (notice that this greedy action is the one prescribed by ABAP). As exploration probability decreases, the robots will expectedly converge to the optimal strategy w.r.t. the learnt Q-function. Notice then that, as the exploration goes to zero, the performance of the robots does improve, as easily seen from the plots in Figure 8.4 by observing the slope of the learning curve.

On the other hand, the slope of the learning curve provides a rough indicator of the performance of the robots as a team and we can conclude from the plots in Figure 8.4 that the performance of CAQL in both methods is quite different: the slope in the grid-based approximation is approximately twice as large as the one in the kernel-based approach. This is due to the representational power of the chosen sets of basis functions. As argued in Chapter 4, the guarantees of convergence for approximate Q-learning imply nothing on the quality of the approximation. Therefore, a set of basis functions that allows a more accurate representation of the optimal Q-function will in general exhibit better final performance. In this case, the grid based approximation must allow a more accurate representation of the optimal Q-function than the kernel-based approximation.

To further understand this difference between the two used approximations, we tested each of the learnt strategies in the environment. We ran the learnt strategies for each approximation during 100 time units and determined the total discounted reward obtained in each case. Table 8.1 represents the results obtained. We ran

Table 8.1: Comparative results for CAQL and approximate Q-learning with no coordination using two different sets of basis functions. The reported results were obtained after the learning period was complete. We present the average total discounted reward and standard deviation obtained over $2,000$ independent Monte-Carlo trials.

Approximation	**Method**	**Total Disc. Reward**
Grid-based	No coordination	14.981 ± 3.246
	CAQL	20.163 ± 4.012
Kernel-based	No coordination	4.538 ± 8.474
	CAQL	5.657 ± 6.142

$2,000$ independent Monte-Carlo trials and present the average and standard deviation obtained using each of the methods. For the purpose of comparison, we also present the results obtained with no coordination mechanism, for each approximation.

The first thing to remark is that, in both cases, CAQL clearly outperforms the uncoordinated team. This indicates that the ABAP coordination mechanism effectively addresses the problem of simultaneous learning and coordination.

The second remark is concerned with the performance of the method for each set of basis functions. Even though using both sets of functions the method converges, the total discounted reward received using a grid-based approach is much larger than that obtained when considering a kernel-based approach. The observed difference confirms our previous analysis on the learning performance of the algorithm.

Another important aspect is the fact that the variance observed when using a kernel-based approximation is much larger than that observed when using a grid-based approximation. A significant variance in the received total reward means that the performance of the method using a kernel-based approximation is more sensitive to the particular trajectory followed. This further supports the conclusion that the observed difference in performance is due to an inferior representational power of the kernel-based approximation.

We emphasize, however, that in both cases the team is still able to coordinate (as expected), since in both situations the performance of the team using the coordination mechanism clearly surpasses that of the team with no coordination.

8.6 Partial observability

In this section, we aim at developing a RL algorithm for TMGs with partial observability. With that purpose in mind, we follow a similar course of action to that used in Section 4.7 and describe a partially observable TMG as a fully observable TMG with infinite state-space. However, as will soon become apparent, the existence of multiple agents significantly hardens the task. We consider several simplifying assumptions, its implications and possible approaches for more general problems.

8.6.1 Partial observability in Markov games

The purpose of this chapter is to discuss possible extensions of the results from the previous chapters. Namely, we discuss how can the reinforcement learning framework be adapted so as to cope with each of the following situations:

- The Markov game model has an infinite state-space $\mathcal{X}$;
- There is some uncertainty regarding the current state of the game (*e.g.*, due to sensor errors);
- Each player is unaware of the plays made by other agents.

So far we have discussed the first of the situations above. We considered infinite state-space TMGs and proposed a suitable learning algorithm and coordination mechanism. We now consider the second and third situations. In particular, we admit that the state of the TMGs cannot be perceived unambiguously and can only be inferred through the use of sensor measurements (with the inherent measurement errors). In this analysis, we once again admit the state-space $\mathcal{X}$ to be finite.

* * *

As in Chapter 7, we once again consider TMGs with a finite state-space $\mathcal{X}$. However, unlike the games in Chapter 7, we now admit that each player $k \in N$ can no longer observe the state of the game at each time instant $t \in \mathcal{T}$. Instead, the players have access to noisy measurements from which they can estimate the underlying state of the game.

Consider a TMG $\left(N, \mathcal{X}, (\mathcal{A}^k), \mathsf{P}, r, \gamma\right)$ and suppose that, at each time instant $t \in \mathcal{T}$, an observation $Z_t = (z_t^1, \ldots, z_t^n)$ taking values in a set $\mathcal{Z} = \times_{k=1}^N \mathcal{Z}^k$ is issued. This observation depends on the state of the chain and on the action at the previous time instant according to the probabilities

$$\mathbb{P}\left[Z_t = z \mid X_t = i, A_{t-1} = a\right] = \mathsf{O}_a(i, z),$$

where z is the tuple $z = (z^1, \ldots, z^N)$. The actual state of the game is unknown to all players; each player $k \in N$ only has access to the individual observation z^k. We refer to an element in $\mathcal{Z}^k$ as an *individual observation* and an element in $\mathcal{Z}$ as a *joint observation* or *observation profile*.

In the following subsections we discuss the two following situations:

- We say that the team has *centralized observations* if $\mathcal{Z}^1 = \ldots = \mathcal{Z}^N$ and for each $z \in \mathcal{Z}$, $z^1 = \ldots = z^N$. This means that the team has centralized sensing capabilities and hence *all players make the same observations*;[6]
- We say that the players have *cognitive autonomy* if they lack centralized observations and are also unable to unambiguously observe the other players' actions.

[6]This occurs, for example, in the RoboCup small-size league, where a global vision system can be used. As another example, consider a team of surveillance robots that has access to the video feeds from the surveillance cameras in the area.

Each of these situations fits in the framework of *stochastic games with incomplete information*: each player acts upon its own *beliefs* on the private information available to itself and other players. The second case also exhibits *imperfect information*: each player has no (direct) access to the plays by the other players.

In the continuation, we address learning in a team of players with centralized observations. We draw a close parallel between this situation and that described in Section 4.7, and derive a similar approach to the problem of partial observability.

Situations with cognitive autonomy are discussed in Subsection 8.6.3. We describe the framework of *decentralized POMDPs*, or Dec-POMDPs, and *interactive POMDPs*, or I-POMDPs as well as several results on the existence of equilibria, computational complexity and approximate solution methods. Finally, we conclude this section by summarizing several bibliographical references.

8.6.2 Centralized observations and state estimation

Let $\left(N, \mathcal{X}, (\mathcal{A}^k), \mathsf{P}, r, \gamma\right)$ be a TMG with finite state and action-spaces. Suppose that, at time instant $t \in \mathcal{T}$, the game is at some state $X_t \in \mathcal{X}_t$ and each player $k \in N$ chooses an individual action $A_t^k \in \mathcal{A}^k$. The game then moves to state X_{t+1} determined according to the transition probabilities defined by the joint action $A_t = (A_t^1, \ldots, A_t^k)$,

$$\mathbb{P}\left[X_{t+1} = j \mid X_t = i, A_t = a\right] = \mathsf{P}_a(i, j).$$

When the game reaches state X_{t+1}, all players receive a numerical reward given by $r(X_t, A_t, X_{t+1})$. Each player also obtains an indirect measurement of the new state of the game, as described in the previous subsection. We denote by Z_{t+1}^k the r.v. corresponding to the individual observation of player k. It depends on X_{t+1} and A_t according to the observation probability function O, *i.e.*,

$$\mathbb{P}\left[Z_{t+1}^k = z^k \mid X_{t+1} = j, A_t = a\right] = \mathsf{O}_a^k(z^k, j),$$

where O^k denotes the k^{th} component of O.

The tuple $(N, \mathcal{X}, (\mathcal{A}^k), (\mathcal{Z}^k), \mathsf{P}, (\mathsf{O}^k), r, \gamma)$ is known as a *partially observable team Markov game* (POTMG), where $\mathcal{X}$ is the state-space, $\mathcal{A} = \times_{k=1}^N \mathcal{A}^k$ is the cartesian product of the individual action-spaces, $\mathcal{Z} = \times_{k=1}^N \mathcal{Z}^k$ is the cartesian product of the individual observation spaces. P is the transition probability function and $\mathsf{O} = \times_{k=1}^N \mathsf{O}^k$ is the (combined) observation probability function. As usual, r and γ are the reward function and discount factor, respectively.

If the team has centralized observations, then $Z_t^1 = \ldots = Z_t^N$ for each t. Therefore, we can simplify the notation and refer to the observation Z_t and the observation probability function O, in both cases omitting the k superscript. In this case, we use the symbol Z_t to denote both the joint observation and the individual observation, remarking that it should be clear from the context which one we refer to.[7]

The assumption of centralized observations allows us to make use of *belief-vectors* as in Chapter 4. Recall that a belief-vector is a probability vector π_t conveying the

[7]Notice that, even in the presence of centralized observations, the decision process is distributed. This means that we cannot reduce a POTMG to a single-agent POMDP.

probability distribution of the state X_t (over the set $\mathcal{X}$) at time instant t. The i^{th} component of π_t is

$$\pi_t(i) = \mathbb{P}\left[X_t = i \mid \mathcal{F}_t\right],$$

where $\mathcal{F}_t$ is the history of the process up to time t. Suppose that the joint action a is played at time instant t and a transition occurs. Then, all players will make some observation $Z_{t+1} = z$ (all players make the same observation) and, as seen in Chapter 4,

$$\mathbb{P}\left[X_{t+1} = j \mid Z_{t+1} = z, A_t = a, X_t \sim \pi_t\right] = \Pi_a(\pi, z) = \\ = \frac{\sum_{i \in \mathcal{X}} \pi_t(i) \mathsf{P}_a(i,j) \mathsf{O}_a(j,z)}{\sum_{i,k \in \mathcal{X}} \pi_t(i) \mathsf{P}_a(i,k) \mathsf{O}_a(k,z)}. \tag{8.9}$$

Recall that the belief vectors are Markovian and require only the knowledge of P and O to be updated.

In a *general* POTMG, the players can have independent observations, and each player $k \in N$ will maintain an individual belief-vector with respect to the current state of the game. And, at any time instant, the individual belief-vectors maintained by each player will generally differ from player to player. This fact motivates the following definition. Let π_t^k denote the individual belief-vector maintained by player k.

Consistent Beliefs

The players in a POTMG $\Gamma = (N, \mathcal{X}, (\mathcal{A}^k), (\mathcal{Z}^k), \mathsf{P}, (\mathsf{O}^k), r, \gamma)$ have *consistent beliefs* at time t if $\pi_t^1 = \ldots = \pi_t^N$. If the players have consistent beliefs at $t = 0$, we say that the game has *consistent initial beliefs*.

In the case of centralized observations, all players make the same observation at each time instant. Therefore, if they have consistent beliefs at some time instant T, they will have consistent beliefs for all $t > T$. In particular, if the game has initial consistent initial beliefs, the players will have consistent beliefs for all t and we can simply consider one sequence of belief-vectors $\{\pi_t\}$: the belief-vector maintained by any player k at time t will be π_t.

This means that in a POTMG $(N, \mathcal{X}, (\mathcal{A}^k), (\mathcal{Z}^k), \mathsf{P}, (\mathsf{O}^k), r, \gamma)$ with consistent initial beliefs the belief sequence $\{\pi_t\}$ is a controlled Markov chain taking values in the $n-1$-dimensional probability simplex $\mathbb{S}^n$ ($n = |\mathcal{X}|$). But this means that we can define an infinite TMG $\left(N, \mathbb{S}^n, (\mathcal{A}^k), \bar{\mathsf{P}}, \bar{r}, \gamma\right)$, where the kernel $\bar{\mathsf{P}}$ is given, for any $\pi \in \mathbb{S}^n$, $a \in \mathcal{A}$ and $U \in \mathcal{B}(\mathbb{S}^n)$, by

$$\bar{\mathsf{P}}_a(\pi, U) = \sum_{z \in \mathcal{Z}} \sum_{i,j \in \mathcal{X}} \pi(i) \mathsf{P}_a(i,j) \mathsf{O}_a(j,z) \mathbb{I}_U(\Pi_a(\pi, z)),$$

$\Pi_a(\pi, z)$ is the updated probability vector obtained from π given the observation z and the joint action a and $\mathbb{I}_U(\pi)$ is the indicator function for the set U.

Notice that, because of the consistent beliefs, *all players have access to the sequence* $\{\pi_t\}$ and the TMG $\big(N, \mathbb{S}^n, (\mathcal{A}^k), \bar{\mathsf{P}}, \bar{r}, \gamma\big)$ is fully observable! Therefore, we have derived a fully observable TMG from a POTMG and the solution of the latter can be computed by computing the solution of the former. In particular, we can apply the CAQL algorithm from Section 8.4 to the fully observable TMG $\big(N, \mathbb{S}^n, (\mathcal{A}^k), \bar{\mathsf{P}}, \bar{r}, \gamma\big)$.

This new equivalent game has a compact state-space and, because of the assumed centralized observations, also verifies the conditions of Theorem 4.7.4 in Section 4.7. In other words, we can combine Theorems 4.7.4 and 8.4.2 and assess the convergence of CAQL when applied to POTMGs with centralized observations and consistent initial beliefs, as long as there is one distinguishable state. This result is summarized in the following theorem, the final convergence result provided in the thesis that subsumes all main contributions in the thesis. We use a similar notation to that used in Theorem 8.4.2.

Theorem 8.6.1. *Let* $(N, \mathcal{X}, (\mathcal{A}^k), \mathcal{Z}, \mathsf{P}, \mathsf{O}, r, \gamma)$ *be a POTMG with centralized observations, consistent initial beliefs and finite state, action and observation spaces* $\mathcal{X}$, $\mathcal{A}$ *and* $\mathcal{Z}$. *Let* $(\sigma_\theta)_t$ *be a* θ*-dependent learning strategy obtained from CAQL and with the GLIE property w.r.t. the estimate* $Q_\theta(\pi, a) = \xi^\top(\pi, a)\theta$ *and such that* $(\sigma_\theta)_t(\pi, a) > 0$ *for all* (π, a). *Further assume that such strategy verifies, for all pairs* (π, a)

$$|\sigma_\theta(\pi, a) - \sigma_{\theta'}(\pi, a)| \leq C \, \|\theta - \theta'\|$$

for some constant $C > 0$ *independent of* π *and* a. *Assume that the Markov chain* $(\mathcal{X}, \mathsf{P}_\theta)$ *obtained by following the strategy* $(\sigma_\theta)_t$ *is irreducible and aperiodic for each fixed* θ *and that there is an observation* $z \in \mathcal{Z}$ *and a state* $i^* \in \mathcal{X}$ *such that, for all* $i \in \mathcal{X}$,

$$\mathsf{O}(i, z) = \begin{cases} 1 & \text{if } i = i^*; \\ 0 & \text{otherwise.} \end{cases}$$

Further assume that

1. *CAQL uses a set* $\Xi = \{\xi_i, i = 1, \ldots, M\}$ *of* M *bounded, linearly independent functions defined on* $\mathbb{S}^n \times \mathcal{A}$ *such that* $\sum_{i=1}^M |\xi_i(x, a)| \leq 1$ *for all* $(\pi, a) \in \mathbb{S}^n \times \mathcal{A}$.

2. *The function* $B(t)$ *decreases monotonically to zero and verifies*

$$\lim_{t \to \infty} \frac{\sqrt{\frac{\log \log(t)}{t}}}{B(t)} = 0.$$

3. *The cardinality* m *of the sets* $S_m(\pi, \mathcal{H}_t)$ *and the length* K *of the* K*-sample verify* $m \geq K(N + 2)$.

Then, the sequence $\{\theta_t\}$ *generated by CAQL converges w.p.1 to the parameter vector* θ^* *satisfying the following recursive relation*

$$Q(\theta^*) = \mathcal{P}_{\mathcal{Q}} \mathbf{H} Q_(\theta^*).$$

Furthermore, all players in N converge in behavior w.p.1 to a common coordinated Nash equilibrium w.r.t. Q_{θ^*}.

We conclude by noting that the state of the new game at time instant t, given by the belief-vector π_t, can be interpreted as an *internal state* for each player, tracking the actual state X_t of the original game.

8.6.3 Cognitive autonomy

We now consider a more general situation, where the players no longer have centralized observations and are unable to explicitly know the actions of the other players.

Let $\Gamma = (N, \mathcal{X}, (\mathcal{A}^k), (\mathcal{Z}^k), \mathsf{P}, (\mathsf{O}^k), r, \gamma)$ be a POTMG and let π_t^k be a probability vector in $\mathbb{R}^n$, each component $\pi_t^k(i)$ indicating player k's belief that the current state of the game is $X_t = i$. The update equation for π_t^k is similar to that described in the previous subsection: if the team plays the joint action a and player k observes z^k and is aware of the played joint action a, the new belief for player k is

$$\pi_{t+1}^k(j) = \frac{\sum_{i \in \mathcal{X}} \pi_t^k(i) \mathsf{P}_a(i,j) \mathsf{O}_a^k(j,z)}{\sum_{i,k \in \mathcal{X}} \pi_t^k(i) \mathsf{P}_a(i,k) \mathsf{O}_a^k(k,z)}.$$

As in the previous subsection we say that the players have *consistent beliefs* at time t if $\pi_t^1 = \ldots = \pi_t^N$.

However, since we no longer have centralized observations, the relation $Z_t^1 = \ldots = Z_t^N$ no longer holds. Even if the players have consistent beliefs at some time instant t, nothing can be said about the consistency of the beliefs at time $t+1$ even if the players are able to observe the played joint action. Therefore, it is no longer possible to use the beliefs π_t^k to define a Markov chain over $\mathbb{S}^n$. Even if all observation probability functions O^k are equivalent, *i.e.*, they assign the same probabilities to the same events, the individual observations made by each player, being independent, will generally lead to inconsistent beliefs.

To deal with this difficulty, several models and methods have been proposed in the literature. We now review two such models, underlying the main advantages and differences between them. We then conclude with a brief overview of related bibliography.

Decentralized POMDPs

A decentralized POMDP (Dec-POMDP), as introduced by Bernstein et al. [23, 24], is a tuple $(N, \mathcal{X}, (\mathcal{A}^k), (\mathcal{Z}^k), \mathsf{P}, (\mathsf{O}^k), r, \gamma)$, where N is a set of agents, $\mathcal{X}$ is a set of states, $\mathcal{A} = \times_{k=1}^N \mathcal{A}^k$ is the cartesian product of the individual action-spaces for the agents and $\mathcal{Z}^k$ are the individual observation spaces. P is the action-dependent transition probability function and O^k is the action-dependent observation probability function for agent k. As usual, r is the common reward function and γ is a discount factor. Dec-POMDPs are, therefore, POTMGs with no centralized observations and in which other agents actions are not observable.

As usual, the purpose of the agents in a Dec-POMDP is to choose the sequence of joint actions $\{A_t\}$ that maximizes the functional

$$V(\{A_t\}, i) = \mathbb{E}\left[\sum_{t=0}^{\infty} \gamma^t R(X_t, A_t) \mid X_0 = i\right]$$

for each initial state $i \in \mathcal{X}$.

Bernstein et al. [24] showed that finding an optimal strategy in a Dec-POMDP is NEXP-hard even if only a finite-horizon is considered, leaving room for little hope on finding efficient computational methods for solving this class of problems. However, in spite of the computational complexity of this class of problems, several researchers have proposed different approximate methods yielding encouraging results in real problems (see Subsection 8.6.4 for further bibliographical references).

In general Dec-POMDPs the agents have cognitive autonomy and will generally have inconsistent beliefs about the state of the game. As argued above, it is not possible to use such beliefs to define some associated Markovian process, as in the case of centralized observations in Subection 8.6.2. Therefore, the agents must resort to their knowledge of the problem and past history of observations to decide upon the action to take.

If each agent $k \in N$ considers the history of observations up to time t as a *signal function* and a suitable payoff function is defined, the decision process at each time instant reduces to a *state Bayesian game*, and the optimal policy for the Dec-POMDP can be approximated by determining the optimal policies for each state Bayesian game.

This approach is successfully pursued in [79], where the BaGA algorithm is proposed to efficiently build the state Bayesian games. As argued in [79, 80], building the exact payoff functions for each state Bayesian game would require a similar effort as that required to solve the original Dec-POMDP. As such, the authors propose the use of a heuristically estimated payoff function that considers the uncertainty to vanish after one step (the so-called Q_{MDP} values).

In order to improve the efficiency of the overall method, in [80] and [81] the authors propose the use of *low probability pruning* and *clustering* in the space of all possible types of the Bayesian game. The method is then applied to several problems, including real robot navigation problems.

Interactive POMDPs

Interactive-POMDPs (I-POMDPs) were introduced by Gmytrasiewicz and Doshi [98] as models extending the POMDP framework to multi-agent scenarios. An I-POMDP describes a *single-agent decision process* but considers the existence of other decision-makers in the environment. As such, it includes in its definition not only a dynamic description of the agent and its task but also a model of other decision-makers that may exist in the environment.

To formalize these ideas we start by introducing the concepts of *frame* and *type* of an agent. The frame of an agent is, in its essence, a POMDP model describing the agent and its task. It includes information on the transition dynamics of the agent,

on the observation process of the agent, on the rewards and on the functional to be maximized. For example, all agents considered in this thesis considered the total discounted reward, represented as a function V, as the functional to be maximized. The type of an agent includes the frame of that agent but also includes information on the current state of the agent (for example, a belief).

We formalize these concepts in the following definition.

Type and Frame

A *type* of an agent k is a tuple $\tau^k = (\pi^k, \mathcal{A}^k, \mathcal{Z}^k, \mathsf{P}, \mathsf{O}^k, r^k, V^k)$, where

- π^k is a belief-state for agent k (more details ahead);
- $\mathcal{A}^k$ is agent k's individual action set;
- $\mathcal{Z}^k$ is agent k's individual observation set;
- P is the transition probability function;
- O^k is agent k's observation probability function;
- r^k is agent k's individual reward function;
- V^k is agent k's *optimality criterion*, represented as a value function V.

A type can be simply written as $\tau^k = (\pi^k, \phi^k)$, where ϕ^k is a *frame* for agent k and is given by $\phi^k = (\mathcal{A}^k, \mathcal{Z}^k, \mathsf{P}, \mathsf{O}^k, r^k, V^k)$. The set of all possible frames for agent k is denoted Φ^k.

As stated before, if we disregard the existence of other agents in the environment, the type of an agent k is similar to a POMDP and can be solved for an optimal action for each belief state π^k. In other words, if the type of a agent is known at some time instant t, the action of that agent *can be predicted*. Therefore, the type of an agent can be seen as a behavioral model for this agent.

I-POMDPs consider multi-agent partially observable interactions from a single agent's perspective. The I-POMDP for an agent k is a tuple $(IS^k, \mathcal{A}, \mathcal{Z}^k, \mathsf{P}, \mathsf{O}^k, r^k)$, where

- IS^k is the set of all *interactive states* for agent k;
- $\mathcal{A} = \times_{k=1}^{N} \mathcal{A}^k$ is the cartesian product of all agents' individual action sets;
- P defines the transition probabilities between states of the world. Therefore, if $\mathcal{X}$ denotes the set of possible states of the world, P is a mapping $\mathsf{P} : \mathcal{X} \times \mathcal{A} \times \mathcal{X} \longrightarrow [0; 1]$;
- $\mathcal{Z}^k, \mathsf{O}^k$ and r^k are as in the definition above.

The set of interactive states (I-states) is described in [69] as the set

$$IS^k = \mathcal{X} \times \Phi^1 \times \ldots \times \Phi^{k-1} \times \Phi^{k+1} \times \ldots \times \Phi^N$$

but this definition is generalized in other works [70, 98] to

$$IS^k = \mathcal{X} \times \mathcal{M}^1 \times \ldots \times \mathcal{M}^{k-1} \times \mathcal{M}^{k+1} \times \ldots \times \mathcal{M}^N,$$

where $\mathcal{M}^k$ is a set of *models* of agent k. In [98] a model $m^k \in \mathcal{M}^k$ is defined as a pair $m^k = (h^k, \sigma^k)$, where h^k is a possible history of observations and σ^k is a strategy mapping each history to an individual action in $\mathcal{A}^k$. If a model is determined from an agent's type, it is said to be an *intentional model*.

Doshi and Gmytrasiewicz [70] further assume the I-POMDP model to be *non-manipulative* and *non-observable*. The first assumption simply states that an agent cannot affect other agents' models directly; the second assumption states that an agent cannot observe other agents' beliefs directly. These two assumptions guarantee the agents to be autonomous, as argued in [98].

In an I-POMDP, each agent maintains a belief over the set of possible I-states and decides upon such beliefs. This raises to two important questions:

- How to propagate the beliefs over time;
- How to compute the policy mapping beliefs into actions.

The solution for the first problem (belief propagation) is achieved by replicating the belief update in (8.9). However, since agent k is unable to observe the action of the other agents, it takes into account its prior belief on the other agents' models, as each model prescribes an action for the corresponding agent. The expression for belief propagation is thus a natural extension of that in (8.9) and can be found in any of the cited works. In [72] a particle-filter-based method is proposed to perform belief propagation more efficiently.

There are several advantages and limitations in using beliefs in the I-POMDP framework.

First of all, considering models of the other agents that include information about their beliefs leads to a problem known as *infinitely nested beliefs*. This means that each agent maintains a belief on the other agents beliefs on its own beliefs and so on. To solve this problem, one may consider only simplified agent models that guarantee finite nesting of the agents' beliefs.

On the other hand, it is possible to show [70, 71] that if the initial beliefs for all agents verify a truth compatibility condition, then the beliefs of all agents converge w.p.1 as the history length t goes to infinity. Furthermore, it is shown that, as $t \to \infty$, the beliefs approach the belief obtained if all agents could share their observations (*i.e.*, the agents had centralized observations). In this case, if each agent adheres to its own optimal strategy w.r.t. its individual belief, the team will converge w.p.1 to a *subjective equilibrium*.[8]

[8] A subjective equilibrium is a strategy from which no individual agent profitably deviates, given its information. Clearly, all Nash equilibria are subjective equilibria; the converse, however, does not generally hold.

However interesting this result may be, it is hard to ensure the initial beliefs to verify such compatibility condition [71] and approximations must be used with much less reassuring convergence properties.

8.6.4 Related work

Markov games with partial observability are commonly referred to as *partially observable stochastic games* (POSG). If complexity results on POMDPs leave room for little hope on its tractability, results on the complexity of POSG are even less encouraging [23, 24]. Nevertheless, several works in the literature propose interesting approaches to address this complex class of games.

Nair et al. [211] propose the use of policy search methods to address POSGs. They introduce JESP,[9] a class of policy search algorithms that converges to a locally optimal Nash equilibrium. They also propose a dynamic programming approach to JESP, considering a generalized belief space and reducing the overall multi-agent problem to a set of single-agent problems. In this approach, Nair et al. establish piecewise-linearity and convexity of the obtained value-function.

Guo and Lesser [114] analyze the general complexity of finding Nash equilibria in POSGs. They also propose an algorithm to perform iterated elimination of dominated strategies. In this line of work, Hansen [118] combines the dynamic programming approach to POMDPs with iterated elimination of dominated strategies to yield an exact dynamic programming algorithm for POSGs considering a finite horizon. Bernstein et al. [25] introduce a multi-agent version of *bounded policy iteration*.[10]

Szer et al. [303] introduce a multi-agent version of A*, MAA*. As described in [303], MAA* is an optimal heuristic search algorithm for POSGs considering a finite horizon.

As described in Subsection 8.6.3, Emery-Montemerlo [79] provides an extensive discussion on the applicability of the POSG model to multi-robot tasks. It describes in detail the POSG framework and the main difficulties arising from considering such a model of robot interaction. It then proceeds with the description of the BaGA algorithm, that considers a POSG as a sequence of smaller Bayesian games. The consideration of these simpler game models allows for approximate solutions that can be used in real-time robotic tasks. The thesis [79] and companion papers [80, 81] also describe a mechanism to consistently build the private information to be used in each Bayesian game for the resulting joint strategy to be coordinated.

Finally, Doshi [69] describes the I-POMDP model. This framework, first introduced in [98], extends the POMDP framework to multi-agent settings. In [70, 71] the authors analyze the existence of equilibria in this class of problems and derive important properties of such equilibria. In [72] a particle-filter-based method is proposed to perform belief propagation more efficiently and in [251] an exact solution method for I-POMDPs is proposed. This method groups player beliefs into

[9] JESP stands for *joint equilibrium-based search for policies*.

[10] Bounded policy iteration is a finite-state controller search algorithm that combines gradient ascent and policy iteration. It was introduced by Poupart and Boutilier [241] to address single agent, partially observable decision problems.

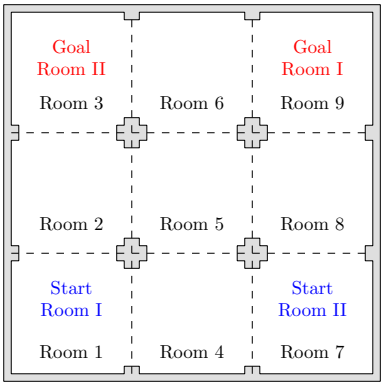


Figure 8.5: Example of an indoor environment.

equivalence classes and discretizes the infinite interactive state-space.

In this thesis, we adopted a simplified model, where the agents have centralized observations and are able to know *a posteriori* each other's actions. With this simplified approach, we were able to address POTMGs using reinforcement learning methods with guaranteed convergence. The methods proposed in this thesis also allow us to successfully address the problem of simultaneous learning and coordination. The extension of the methods studied in the thesis to more elaborate multi-agent models such as Dec-POMDPs or I-POMDPs may constitute an interesting direction for future research. Even if the conditions required by the methods studied here hardly apply to any of the mentioned models, the methods in the thesis may provide useful insights to develop approximate solutions for such complex models.

8.7 An illustrative example

We once again consider the indoor environment from Example 6.2 and repeated in Figure 8.5 for commodity. The corresponding multi-robot navigation problem can be described by an TMG $\big(N, \mathcal{X}, (\mathcal{A}^k), \mathsf{P}, r, \gamma\big)$ where

- $N = \{\mathrm{I}, \mathrm{II}\}$ is the set of players;
- $\mathcal{X} = \mathcal{X}^{\mathrm{I}} \times \mathcal{X}^{\mathrm{II}}$, where $\mathcal{X}^k = \{1, \ldots, 9\}$ for $k = \mathrm{I}, \mathrm{II}$;
- $\mathcal{A}^k = \{N, S, E, W\}$ for $k = \mathrm{I}, \mathrm{II}$;
- The transition probabilities are defined as in Example 6.2;
- The reward function r assigns a reward of $+20$ for every transition triplet (i, a, j) such that $j = (9, 3)$, -10 for every transition triplet (i, a, j) such that $j^{\mathrm{I}} = j^{\mathrm{II}}$ and 0 otherwise;
- We considered $\gamma = 0.95$.

However, unlike Example 6.2, we now admit that the robots are not able to perceive their position in the environment. Only when *both* robots reach their goal

rooms, they both receive a signal indicating that information. Also, whenever the robots collide (with each other), they have sensors that provide them with that information. They have no other sensors available.

This new problem can be modeled using a partially observable team Markov game with centralized observations $(N, \mathcal{X}, (\mathcal{A}^k), \mathcal{Z}, \mathsf{P}, \mathsf{O}, r, \gamma)$, where

- N, $\mathcal{X}$, $\mathcal{A}$, P, r and γ are as defined above;
- $\mathcal{Z} = \{\emptyset, Goal, Crash\}$;
- O represents the observation probability function. For $i = (9, 3)$, $\mathsf{O}(i, Goal) = 1$. For i such that $i^{\mathrm{I}} = i^{\mathrm{II}}$, $\mathsf{O}(i, Crash) = 1$. Otherwise, $\mathsf{O}(i, \emptyset) = 1$.

Whenever both robots reach their goal states, their position is randomly reset to any of the other 80 states, independently of the robots' actions.

Notice that, since the robots are only able to observe the joint goal location and the crash situations, the set of possible observations has only three elements: $\emptyset$, corresponding to the null observation, $Crash$, corresponding to a crash situation, and $Goal$, the observation corresponding to the goal state.

We applied CAQL to the associated team Markov game obtained from the original POTMG $(N, \mathcal{X}, (\mathcal{A}^k), \mathcal{Z}, \mathsf{P}, \mathsf{O}, r, \gamma)$. We used the natural set of functions obtained from the belief state π_t,

$$\Xi = \left\{\xi_{1,(N,N)}, \ldots, \xi_{81,(N,N)}, \xi_{1,(N,S)}, \ldots, \xi_{81,(W,W)}\right\}$$

each $\xi_{i,a}$ given by

$$\xi_{i,a}(\pi, b) = \pi(i)\mathbb{I}_a(b).$$

This corresponds to a total of 81×16 scalar parameters $\theta(i, a)$, and, as in Chapter 4, the overall dimension of the parameter vector θ is similar to the one in the fully observable case. The robots were allowed to explore/learn during 3×10^5 time units, and the obtained strategy was then evaluated for 100 time units. During learning we used Boltzmann exploration in all experiments.

Figure 8.6 represents the total reward obtained by the robots during learning. The point where coordination is attained is clearly noticeable by observing the slope of the learning curve.

We also tested the learnt strategy in the environment. We ran the learnt strategy for 100 time units and determined the total discounted reward obtained by the team. Table 8.2 represents the results obtained. We have run $2,000$ independent Monte-Carlo trials and present the average and standard deviation obtained using the policy learnt with CAQL. We also present the results obtained with a team of robots with no coordination mechanism as well as those of a team with full observability.

We conclude this section with two final comments.

First of all, notice that the difference in the performance between the two methods is not very significant. This can be easily explained. In this scenario, few observations are available and, most of the time, the agent must decide upon beliefs traducing great uncertainty on the underlying state of the game. It is not surprising that, in this situation, "almost any" joint action (coordinated or not) is equally good.

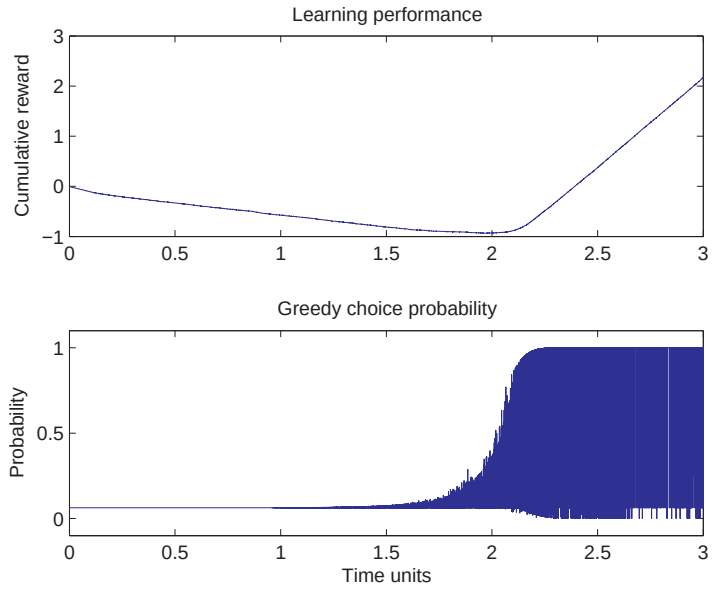


Figure 8.6: Cumulative reward obtained during the 3×10^5-time-units learning period.

Table 8.2: Comparative results of CAQL and uncoordinated approximate Q-learning against those of an optimal (fully-observable) team. The reported results were obtained after the learning period was complete. We present the average total discounted reward and standard deviation obtained over $2,000$ independent Monte-Carlo trials.

Method	**Total Disc. Reward**
No coordination	64.579 ± 9.832
CAQL	67.528 ± 8.924
Optimal	90.236 ± 9.761

On the other hand, this uncertainty is lifted essentially in states where coordination is not essential (after a crash or when the goal is reached).

On the other hand, the goal and "crash" states clearly verify the conditions of Theorem 8.6.1, and we expect CAQL to converge, as established in Theorem 8.6.1 and confirmed by in Figure 8.6 and Table 8.2.

8.8 Concluding remarks

To conclude this chapter, we summarize the main ideas presented so far. We discuss the applicability of the RL framework introduced in this chapter to multi-robot navigation problems. We conclude with an overview of the theoretical approach described throughout the thesis.

8.8.1 Summary

In the previous chapters, we described TMGs as suitable models to address multi-agent topological navigation problems. We described and introduced methods that ensure that a team of robots is able to *learn* and *coordinate* in the implementation of the optimal decision rule in a given environment. However, in the approach pursued in the previous chapters, two essential assumptions were made:

- The TMGs had finite state-space;
- At each time instant, each player (robot) is able to perceive the state of the game with no uncertainty, as well as the actions taken by the other players (robots).

In this chapter we discussed how the previous assumptions can be alleviated.

With respect to the first assumption, we introduced a generalization of the CQL method, specifically designed to address TMGs with infinite state-spaces. We combined approximate Q-learning with an extended version of BAP (approximate BAP) and showed that the combination of these two methods, named *coordinated approximate Q-learning* (CAQL), converges w.p.1 to a suitable approximation of the optimal Q-function for the game, while ensuring coordination in an optimal Nash equilibrium (with respect to the learnt Q-function).

With respect to the second assumption, we started by discussing a simplified problem where the team has *centralized observations*, *i.e.*, all players share the same observations. We provided a method to reformulate this class of games as fully observable team stochastic games with infinite state-spaces, thus suitable for the application of CAQL. We then discussed more general models where the players have *cognitive autonomy*. In this situation it is no longer possible to define an easily solvable equivalent game and more elaborate solution techniques are required. We briefly described two frameworks to address this class of problems, Dec-POMDPs and I-POMDPs, and reviewed the main properties of each framework and corresponding solution methods.

We concluded the chapter by illustrating the use of CAQL in a familiar multi-robot navigation example.

8.8.2 Discussion

Along the thesis, we described/introduced several RL algorithms for various purposes. In this chapter we brought that study to a conclusion by describing a RL algorithm for TMGs with infinite state-spaces, CAQL. Under suitable conditions, the algorithm converges asymptotically to an approximation of the optimal Q-function and the players converge to an optimal Nash equilibrium w.r.t. this approximation.

We then proposed the application of this algorithm to POTMGs under the simplifying assumption of centralized observations. We now further comment on the applicability of CAQL in partially observable scenarios. We also comment on the implications of Theorem 8.4.2, where convergence of CAQL is established, as well as several issues that can be addressed in future research.

CAQL and partially observable scenarios: applicability and limitations

Coordinated approximate Q-learning, described in Figure 8.1, is a multi-agent RL algorithm that uses linear function approximation and ABAP to ensure coordination.

The fundamental elements contributing for the convergence of CAQL are

- An adequate learning strategy;
- An adequate set of basis functions to use in the approximation;
- An underlying "well-behaved" Markov chain.

The adequate learning strategy prevents that the players get "stuck" too soon in a suboptimal greedy strategy, which would prevent them from eventually converging to an optimal joint strategy. The adequate set of basis functions prevents divergence problems in the algorithm.[11] Finally, the underlying Markov chain models the time evolution of the game. It allows the analysis of the algorithm to be performed in terms of an equivalent, stationary process and permits a more intuitive understanding of the conditions required for convergence.

When we consider TMGs as models for mobile robot navigation, we implicitly consider the movement of the group of robots to verify the Markov property. This means that the movement of the whole team of robots can be described as a *single* Markov chain.

An alternative to this approach would be to consider separate Markov chains to model the movement of each robot, for example by disregarding the existence of other decision-makers in the environment. But then, one of the following necessarily holds:

- Each robot has a specific mission, independent of the other robots, and therefore no interaction/coordination is required from the robots to complete their individual missions. In this case, each robot can be modeled individually, and we get a set of N single-agent systems that can be solved independently;
- There is the need for interaction/coordination between the robots (this occurs, for example, if the robots must share some resource in the environment).

In the latter situation, the other robots can be modeled as part of the environment. However, and unless if they follow some stationary action course, the "environment" is not *stationary* and the Markov chain model will not even be ergodic (much less geometrically ergodic).

When addressing POTMGs, we assumed the simplifying assumption that the players had *centralized observations*. This assumption guarantees that, even if the players keep individual belief on the state of the game, these beliefs are *coincident*. We can thus look at these beliefs as defining a Markov chain to which all players have access, and reduce the POTMG to a fully observable TMG.

[11] Divergence/non-convergence has been reported in several works in the literature [11, 44, 102, 321].

Of course that, implicitly, we also assume that the agents are able to know *a posteriori* the actions chosen by each member of the team. If no communication at all is available, the assumption of action observability is somewhat unrealistic, and we would like to alleviate it. To achieve this, we could consider an extension of our learning algorithm that can handle *infinite action-spaces*. It is our belief that, with due modifications, approximate Q-learning and ABAP can be adapted so as to cope with infinite action-spaces.[12] Then, by still considering centralized observations, it should be possible to maintain consistent beliefs both on the state of the game and on the joint actions played.

If the assumption of centralized observations is dropped, it is no longer possible to define a fully observable Markov chain to which all players have access. This requires a whole different approach to the problem of partial observability, such as Dec-POMDPs or I-POMDPs. In particular, I-POMDPs make use of beliefs (over the I-states) and should be amenable to the use of single-agent approximate Q-learning, as described in Chapter 4.

A unified reinforcement learning framework

To conclude this chapter, we would like to further comment on Theorem 8.4.2.

First of all, Theorem 8.4.2 is a multi-agent counterpart to Theorem 4.5.2 provided in Chapter 4. Both these theorems convey a rather reassuring result on the convergence of Q-learning when linear function approximation is used. This result has long been an open question in the RL community and, even if the restrictions on the basis functions may seem too restrictive, they nevertheless include as special cases several results on the convergence of Q-learning with linear function approximation.

On the other hand, the generalization of BAP also provides a sound coordination method for multi-agent systems, even if function approximation is used. In fact, Theorem 8.4.1 guarantees that, given the approximation obtained with the learning algorithm, the team of robots will perform no worse than a centralized decision-maker would (except on a negligible set of points).

The interesting aspect that we would like to emphasize is that the finite versions of these algorithms (in particular, tabular Q-learning and BAP) can easily be seen as a particular case of the infinite ones. Therefore, convergence of Q-learning and BAP results as an immediate corollary of Theorem 8.4.2. This means that CAQL and Theorem 8.4.2 provide a pleasant unified approach to RL problems, both with finite/infinite state-spaces and with single/multiple agents. And, under suitable conditions, this same framework can even be used to address problems with partial observability.

On the other hand, it is important to remark that, unlike the approach in [321], the algorithms described herein make no use of eligibility traces. It is our belief that, with no significant trouble, the algorithms described in the thesis can be adapted so as to accommodate eligibility traces. This may improve the overall performance of

[12] As a reference, Borkar [32] proposes a functional version of Q-learning that considers infinite state and action-spaces.

the algorithms, as the use of eligibility traces generally leads to tighter error bounds and faster convergence.

We also find that further work is required to perceive the conditions under which non-linear function approximation can be used with Q-learning and other RL algorithms. It is our belief that, in spite of the counter-examples in the literature [27, 321], positive convergence results are possible if an adequate family $\{Q_\theta\}$ of functions is considered (probably lying in a manifold of some sort). This would further generalize the applicability of RL methods as well as providing a whole new branch of approximation architectures to be used with RL algorithms.

* * *

With the methods presented in this chapter, we conclude the theoretical development of the thesis. In the next chapter we apply the methodology developed in this chapter to several different problems of increasing complexity. These experimental results will bring the thesis to a conclusion, and in Chapter 10 we summarize the main contributions and discuss some issues to be addressed in future research.

Chapter 9

Results in Multi-Robot Navigation

In this chapter we bring this second part of the thesis to a satisfying conclusion by applying the CAQL algorithm developed in the previous chapter to several multi-robot navigation problems.

We describe the model for each robot in the team and the POTMG framework arising from this model. We then describe several navigation tasks, making use of the large scenarios already used in the experiments in Chapter 5, and apply the CAQL algorithm to each of these problems. We ensure the applicability of the method by verifying that the convergence conditions stated in Theorem 8.6.1 are verified in the problems under consideration. We conclude by discussing the performance of the method in the proposed problems as well as its general applicability in robotic navigation problems.

9.1 Introductory remarks

This chapter brings to a conclusion the work in this thesis by experimentally assessing the validity of the methods developed along the last chapters to the class of problems we initially proposed to address.

Recall that, in Section 1.2, we proposed to address situations in which

- A robot or group of robots must complete some navigation task in an environment described by a *topological map*;
- This navigation task is *initially unknown* to the robots;
- The robots receive *evaluative feedback*, indicating how well they are performing;
- The robots control their movements by choosing among a set of possible *actions* with uncertain outcome;
- The robots have access to *noisy measurements* from which they infer their current location/configuration;
- The robots do not know beforehand the actions taken by the other robots, *i.e.*, they do not *explicitly communicate*. Coordination must *emerge* as a consequence of the interaction between the robots.

By considering such tasks as being modeled by a *finite* POTMGs with centralized observations, it is immediate that

- The players in the game represent the group of robots;
- The states in the game and the transition probability model account for the topological representation of the environment: each state in the game corresponds to a configuration of the robots in the environment and the transition probabilities model the uncertainty in the outcome of the control primitives of the robots;
- The common reward function "encodes" the joint task of the team and is initially unknown to the robots; it works as evaluative feedback from which the team can learn how to optimally execute the desired task;
- Partial observability and centralized observations work as the noisy measurements from which the team infers the underlying state of the game, this corresponding to the configuration of the team;
- Finally, the absence of explicit communication in the aforementioned sense is naturally considered in the POTMG framework.[1]

[1]Notice, though, that we have *a posteriori* action observability.

In the previous chapter we described the CAQL algorithm, assessed its convergence properties and applied it to a toy multi-robot navigation problem. Although encompassing all the features above described, its main purpose was to allow a simple understanding of the applicability of CAQL as well as of its properties.

We now apply CAQL to a more extensive set of problems, aiming at analyzing its performance in more realistic problems. These application problems are, essentially, multi-agent variations of the single-agent examples considered in Chapter 5. With these problems, we illustrate the use of POTMG models with centralized observations in robotic navigation problems. We also assess experimentally the usefulness of CAQL in relatively large-sized problems. For example, the CMU environment described in Chapter 5 with only two robots corresponds to a POTMG with around 3×10^5 states!

In each of the problems addressed in this chapter, we consider a teams of autonomous mobile robots moving in an environment described as a topological map. The robots are allowed to explore the environments for a learning period using CAQL and each robot then uses the learnt strategy to control its movement. The main goal of this chapter is to illustrate the successful application of CAQL in multi-robot navigation problems with partial observability.

As in Chapter 5, we have no access to the optimal solution of the problems studied and are therefore unable to establish a comparison between our results and those obtained with the optimal strategy. Therefore, we present the results obtained with an omniscient team of robots—capable of observing the actual state of the game at all times. These results provide an upper bound to the performance of CAQL.

The chapter is organized as follows. We start by describing the set of problems considered: we describe the environments and the robots, from which we derive the structure of the POTMG model used. We then describe the conducted experiments and present the obtained results. We conclude the chapter by discussing these results.

9.2 The experimental setup

In this section we describe the scenarios, the mobile robots and the obtained POSTG model for the different experiments.

9.2.1 The scenarios

To better understand the applicability of CAQL to navigation tasks, we conducted several tests that illustrate the different properties of the CAQL algorithm.

The first set of tests assesses the coordination ability of CAQL. To that purpose we consider scenarios in which coordination is essential for the mission to be successfully accomplished. In the examples considered in the previous chapters, coordination was already important for the robots to avoid negative rewards (penalties). In our first set of tests we move one step further and analyze situations in which coordination is *essential* to achieve the goal: in these scenarios, it is impossible to reach the final configuration if the robots mis-coordinate.

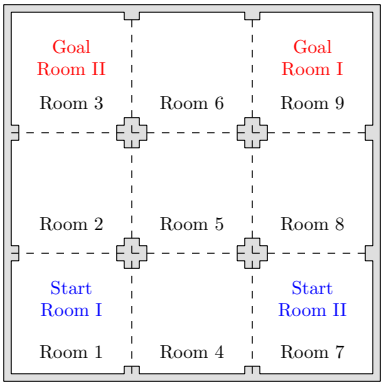


Figure 9.1: Indoor environment from Example 6.2.

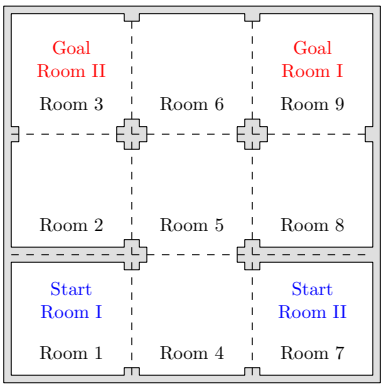


Figure 9.2: Alternative indoor environment to assess coordination.

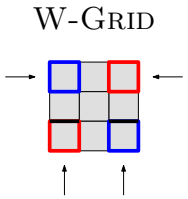


Figure 9.3: Compact representation of the W-GRID test scenario with 9 nodes.

Consider once again the scenario from Example 6.2, used extensively in this second part of the thesis and reproduced in Figure 9.1.

For each robot to avoid an accident in this scenario, it suffices to avoid approaching the other robot, as seen in the different strategies depicted in Figure 6.3 of Chapter 6. We now consider a variation of this problem that requires one of the robots to *wait* for the other for the mission to be successful. This alternative scenario is represented in Figure 9.2. Notice that, if both robots try to simultaneously move to the central room in the bottom row, they will crash and remain in the same state. In order for the robots to leave the initial configuration, one of the robots must *wait*, allowing the other robot to move through the only free passage.

A more compact representation of this environment is depicted in Figure 9.3. We henceforth refer to this scenario as W-GRID, standing for *grid-world with walls*.

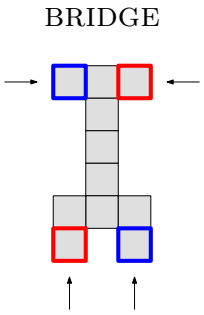


Figure 9.4: BRIDGE environment with 11 nodes.

Table 9.1: Number of states in the POTMGs for each of the test scenarios.

Environment	# States
W-GRID (2 robots)	1 296
BRIDGE (2 robots)	1 936
ISR (2 robots)	29 584
MIT (2 robots)	38 416
PENTAGON (2 robots)	43 264
CIT (2 robots)	78 400
SUNY (2 robots)	87 616
CMU (2 robots)	283 024

The second test scenario, BRIDGE, is depicted in Figure 9.4. It represents a "bridge" in which one of the robots must also wait for the other for the mission to be successful. Unlike the examples considered in the previous chapter, when both robots move into one same "cell" in the map, they both receive a penalty. In this situation, unlike the previous chapters, *the movement does not succeed*. In other words, when the robots "crash" they receive a penalty and remain in the same position they were before "moving".

In the second set of tests, we consider the same scenarios used in Chapter 5. In particular, we apply CAQL to a team of robots moving in the ISR, MIT, PENTAGON, CIT, SUNY and CMU scenarios. We repeat the representation of these environments in Figures 9.5 through 9.10 to help specify the initial and final states for the robots in each experiment.

In Figures 9.3 through 9.10, each pair of color-matching cells represents a starting/goal pair of cells for a particular robot in the team. As in Chapter 5, we consider that in each cell the robots can be in one of four possible orientations, according to the four compass directions: N, S, E and W. We summarize in Table 9.1 the total number of states in the POTMG corresponding to each of the test scenarios.

9.2.2 The robots

In our experiments we consider teams of two robots moving in the environments depicted in Figures 9.5 through 9.10. As in Chapter 5, each robot has three available

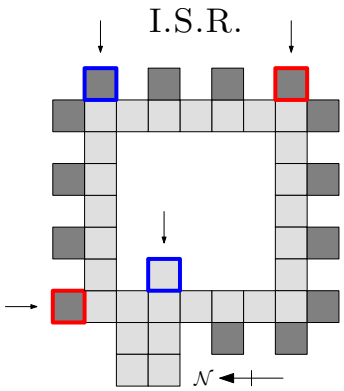


Figure 9.5: Topological representation of the ISR environment with 43 nodes.

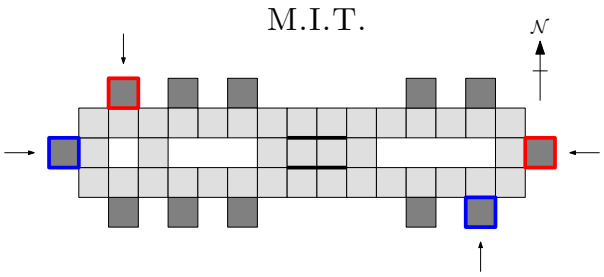


Figure 9.6: Topological representation of the MIT environment with 49 nodes.

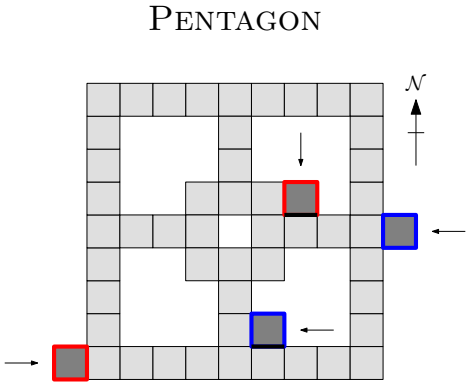


Figure 9.7: Topological representation of the PENTAGON environment with 52 nodes.

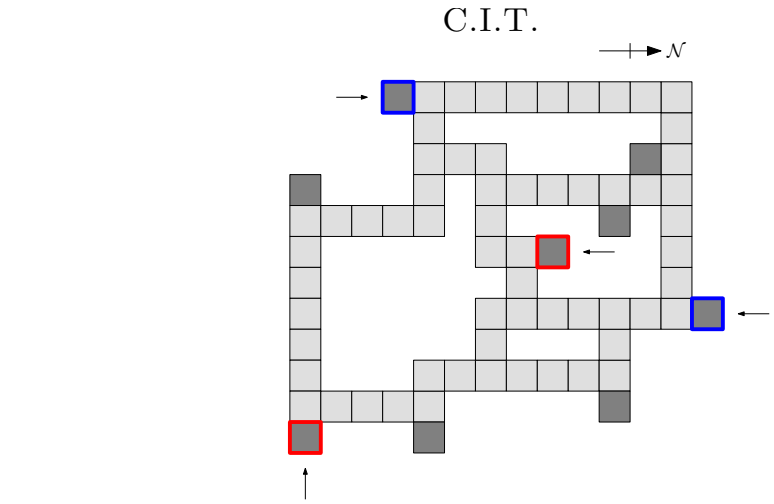


Figure 9.8: Topological representation of the CIT environment with 70 nodes.

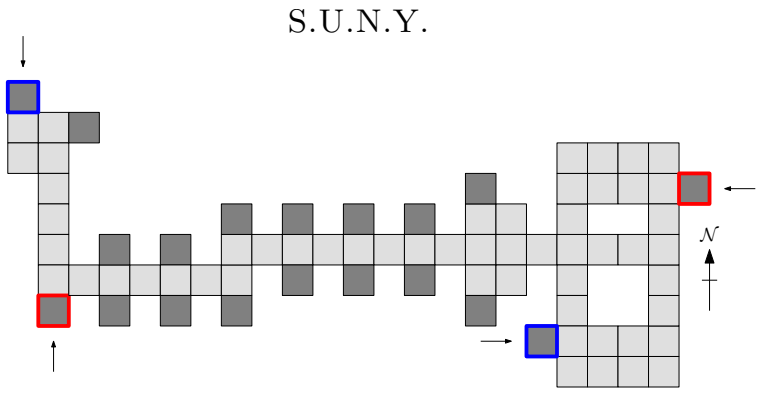


Figure 9.9: Topological representation of the SUNY environment with 74 nodes.

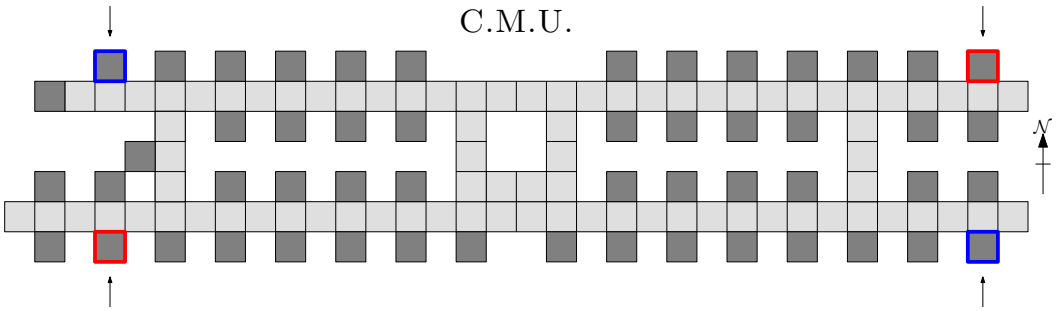


Figure 9.10: Topological representation of the CMU environment with 133 nodes.

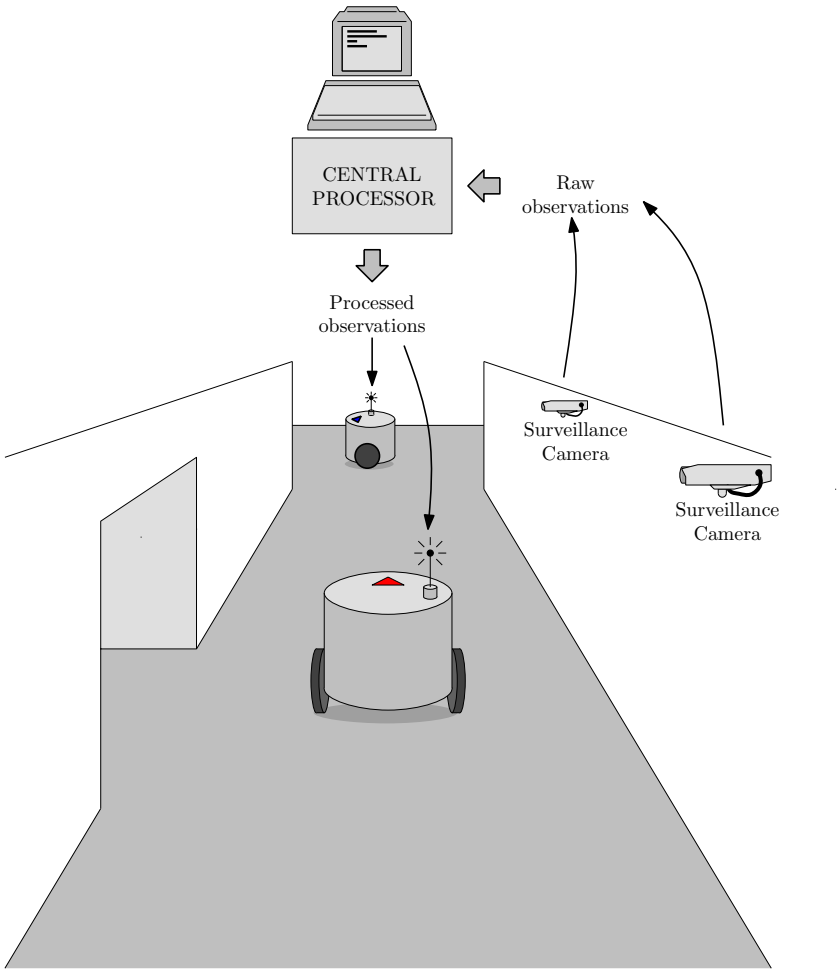


Figure 9.11: Environment layout and sensor architecture.

actions at each time: move forward, turn left and turn right. Each of these actions has an uncertain outcome described by the probabilities in Table 5.2. The probabilities for the POTMG can be obtained by considering the movement of the different robots to be independent except in the crash situations. If $\mathsf{P}_{a^k}^{(k)}(i^k, j^k)$ denotes a particular transition probability for robot k obtained from Table 5.2, the transition probabilities for the POTMG are obtained as

$$\mathsf{P}_a(i,j) = \prod_{k=1}^{N} \mathsf{P}_{a^k}^{(k)}(i^k, j^k).$$

For the experiments in this chapter, we consider $N = 2$.

We consider in all scenarios that surveillance cameras keep the environment monitored. The images from the cameras are processed in a central processor and the processed data is then sent to all the robots. This processed data contains the state of each robot as perceived by the cameras. Figure 9.11 presents a general overview of the architecture.

We consider that each robot has a distinctive feature that allows the cameras to

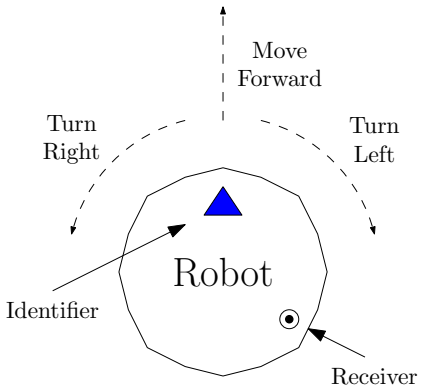


Figure 9.12: The robot, its sensors and actions.

distinguish the different robots while perceiving the state (position and orientation) of each robot. We represented such identifier as a colored mark indicating the orientation of the robot, as depicted in Figure 9.12. The color is used to identify the robots and the orientation of the mark provides the orientation of the robot.

Similarly to the transitions between states, the observations are also prone to errors, and there is a non-zero probability of error in the observed configuration. To illustrate the observation probabilities, suppose that robot A is in position i and robot B in position j, disregarding for the moment their orientation. Then, the surveillance system will observe configuration (i^A, j^B) with probability 0.7 and configuration (j^A, i^B) with probability 0.3. In other words, the camera system misidentifies two robots with a 30% probability. For each robot, the camera system provides the right orientation with probability 0.60 and the two adjacent orientations each with a 0.20 probability. For example, suppose a robot is in state i oriented towards the North. Then, the camera system will identify the robot as being oriented towards the North with a 0.6 probability and, with a 0.4 probability the orientation of the robot is mis-identified either as being East or West. These orientation observation probabilities are summarized in Table 9.2.

The described uncertainty in the observation process occurs in all states *except when the robots reach the goal configuration.* In the state corresponding to that configuration, all robots are able to unambiguously perceive their location and that of all other robots.

Finally, the robots receive a reward of +20 whenever they reach their joint goal (the final configuration), of −10 whenever they crash and of 0 every other time step.

9.2.3 The experiments

We used a simulator to generate state transitions, observations and immediate rewards in the various environments described above. For each environment, the colored cells in Figures 9.3 through 9.10 denote a pair of starting/goal states; each different color concerns a different robot. For example, considering the environment in Figure 9.3, a robot (I) starts in the leftmost "blue state" and must reach the rightmost blue state; a second robot (II) starts in the rightmost "red state" and must reach the leftmost red state. In Table 9.3 we summarize the starting states in

Table 9.2: Orientation observation probabilities for a single robot.

True orientation	**Observed orientation**		
North	North	East	West
	0.60	0.20	0.20
South	South	East	West
	0.60	0.20	0.20
East	East	North	South
	0.60	0.20	0.20
West	West	North	South
	0.60	0.20	0.20

Table 9.3: Starting states for the robots in the several experiments.

Experiment	**Starting states**	**Figure**
W-GRID	Leftmost blue; rightmost red	9.3
BRIDGE	Leftmost blue; rightmost red	9.4
ISR	Leftmost blue; rightmost red	9.5
MIT	Rightmost blue; leftmost red	9.6
PENTAGON	Rightmost blue; leftmost red	9.7
CIT	Rightmost blue; leftmost red	9.8
SUNY	Leftmost blue; rightmost red	9.9
CMU	Leftmost blue; rightmost red	9.10

each environment.

The initial belief state for all robots in the team corresponds to a uniform distribution over all non-goal states. Every time the team reaches the goal configuration, it is reset to the initial configuration.[2] As in the examples in the previous chapters, the team reaches its joint goal at time t if the state X_t of the process corresponds to a configuration in which all robots are in the corresponding goal states.

For the smaller test scenarios (W-GRID and BRIDGE), we allowed CAQL to explore the environment for 5×10^5 time steps. For the remaining environments, we allowed CAQL to explore the environment for 1×10^6 time steps.

We then conducted a series of trials on each learnt policy to evaluate its performance. In all experiments, a single trial consisted of a truncated trajectory of

[2]Recall that, in the experiments in Chapter 5, the position of the robot was randomly reset upon arrival at the goal. However, to prevent situations in which the robots may end-up "stuck" in a crash situation, we always reset the position of the robot to the initial configuration.

250 simulated steps starting from the initial state. The immediate rewards were appropriately discounted and then added to yield a sample of the total discounted reward. This was repeated for 2,000 independent Monte-Carlo trials and the reported results are the averages over all trials. The discount factor was 0.95 for all experiments.

Also, as in Chapter 5, we recorded for each trial whether the team was able to successfully reach the goal configuration within the 250 time steps. We determined the percentage of successful trials and used this percentage as a second performance measure.

We applied CAQL to each TMG $\left(N, \mathbb{S}^n, (\mathcal{A}^k), \bar{\mathsf{P}}, \bar{r}, \gamma\right)$ obtained from the original POTMG with centralized observations $(N, \mathcal{X}, (\mathcal{A}^k), (\mathcal{Z}^k), \mathsf{P}, (\mathsf{O}^k), r, \gamma)$. As in Chapter 5, we used the natural basis functions arising from the beliefs π_t. In particular, we used the set of basis functions

$$\Xi = \{\xi_{i,a}, i = 1, \ldots, |\mathcal{X}|, a = 1, \ldots, |\mathcal{A}|\},$$

with each $\xi_{i,a}$ given by

$$\xi_{i,a}(\pi, u) = \pi(i)\mathbb{I}_a(u),$$

for all $u \in \mathcal{A}$ and $\pi \in \mathbb{S}^n$, where $\mathbb{I}_a$ is the indicator function for the set $\{a\}$. Using this approximation, the learnt parameter vector θ has the same dimension as the Q-functions for the underlying TMG and we used the Q_{MDP} values to initialize the parameter vector in all tests, so as to speed the learning process. We used Boltzmann exploration to ensure a suitable exploration/exploitation tradeoff.

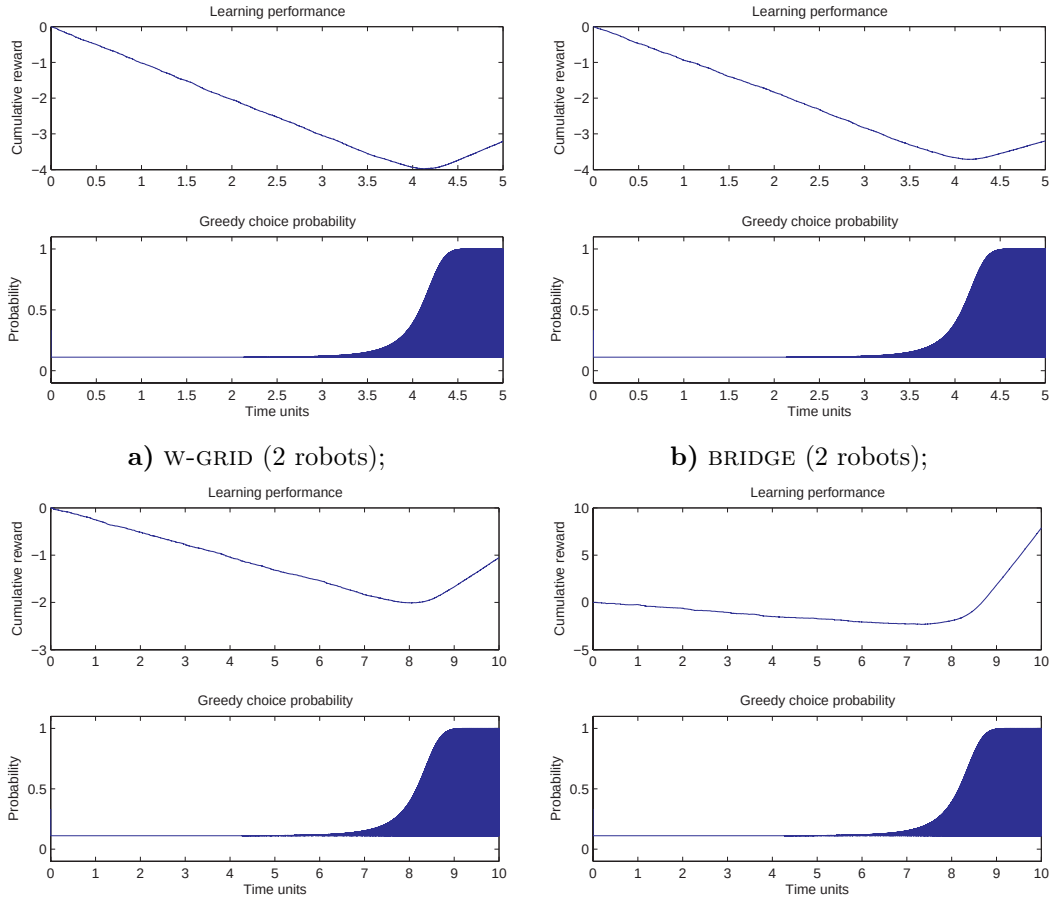
9.3 Experimental results

We present in Figures 9.13 and 9.14 the cumulative reward obtained during the learning period. Notice the difference in the time scale between the environments W-GRID and BRIDGE and the remaining environments, resulting from the shorter learning period. The total discounted reward obtained using the learnt policy and the percentage of successful missions are summarized in Table 9.4. For the sake of comparison, we provide in Table 9.5 the performance of a team of robots that uses no coordination mechanism and in Table 9.6 the results of a team with full-state observability.

9.3.1 Discussion

We now discuss the results of the experiments summarized in Figures 9.13 and 9.14 and in Tables 9.4 through 9.6.

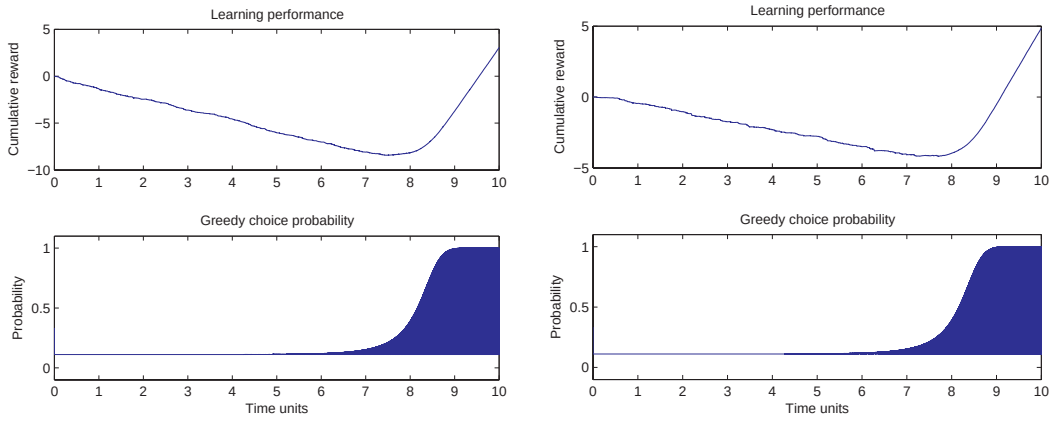
As in Chapter 5, exact methods are of little use in environments with the dimensions of those considered herein. That is the reason why we reported the results of the optimal policy in the underlying fully observable team Markov game in Table 9.6. As expected, the optimal fully-observable policy is always able to reach the goal configuration: since the state is fully observable there is no ambiguity on the configuration of the team in the environment, and the robots can easily coordinate in the optimal strategy.



a) W-GRID (2 robots); **b)** BRIDGE (2 robots);

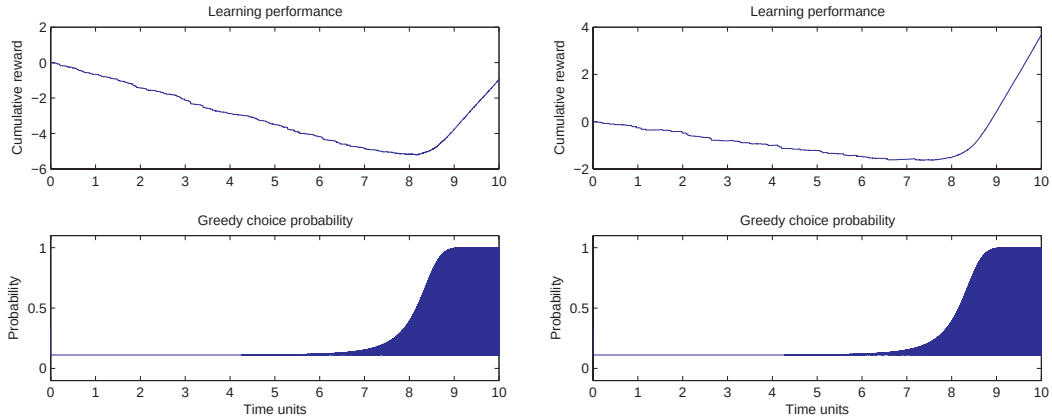
c) ISR (2 robots); **d)** MIT (2 robots);

Figure 9.13: Cumulative reward and greedy choice probability during the learning period for the W-GRID, BRIDGE, ISR and MIT environments.



e) PENTAGON (2 robots);

f) CIT (2 robots);



g) SUNY (2 robots);

h) CMU (2 robots).

Figure 9.14: Cumulative reward and greedy choice probability during the learning period for the PENTAGON, CIT, SUNY and CMU environments.

Table 9.4: Total discounted reward and percentage of successful missions in the nine experiments using CAQL after the learning period is complete. We present the average total discounted reward and standard deviation obtained over 2,000 independent Monte-Carlo trials.

Experiment	Total Disc. Reward	Success %
w-grid (2 robots)	18.304 ± 7.964	100.00 %
bridge (2 robots)	10.916 ± 6.661	100.00 %
isr (2 robots)	9.916 ± 2.250	100.00 %
mit (2 robots)	6.963 ± 1.355	100.00 %
pentagon (2 robots)	9.275 ± 1.840	100.00 %
cit (2 robots)	5.900 ± 1.258	100.00 %
suny (2 robots)	1.800 ± 0.961	100.00 %
cmu (2 robots)	1.628 ± 0.373	100.00 %

Table 9.5: Total discounted reward and percentage of successful missions in the 9 experiments using approximate Q-learning without coordination. We present the average total discounted reward and standard deviation obtained over 2,000 independent Monte-Carlo trials.

Experiment	Total Disc. Reward	Success %
w-grid (2 robots)	7.701 ± 9.504	100.00 %
bridge (2 robots)	1.791 ± 8.620	100.00 %
isr (2 robots)	5.976 ± 4.058	100.00 %
mit (2 robots)	3.992 ± 2.283	100.00 %
pentagon (2 robots)	5.036 ± 2.730	100.00 %
cit (2 robots)	3.666 ± 2.073	99.90 %
suny (2 robots)	0.090 ± 0.664	94.85 %
cmu (2 robots)	0.438 ± 0.475	99.10 %

Table 9.6: Total discounted reward and percentage of successful missions in the nine experiments using the optimal, fully observable strategy. We present the average total discounted reward and standard deviation obtained over 2,000 independent Monte-Carlo trials.

Experiment	Total Disc. Reward	Success %
w-grid (2 robots)	25.692 ± 4.350	100.00 %
bridge (2 robots)	15.531 ± 3.669	100.00 %
isr (2 robots)	12.622 ± 1.815	100.00 %
mit (2 robots)	8.883 ± 0.741	100.00 %
pentagon (2 robots)	12.550 ± 0.945	100.00 %
cit (2 robots)	8.028 ± 0.679	100.00 %
suny (2 robots)	2.435 ± 0.624	100.00 %
cmu (2 robots)	2.122 ± 0.221	100.00 %

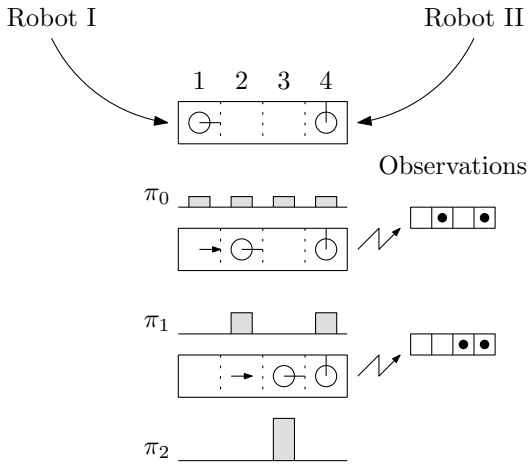


Figure 9.15: Uncertainty elimination.

Remarkably enough, the same thing happens in the experiments with partial observability. In these experiments, described in Table 9.4, the robots can only infer its position from the data provided by the surveillance feed and the past history. However, unlike the examples in Chapter 5, the perceptual aliasing is not too severe: each observation corresponds (at most) to 18 possible states. This means that, even if there is significant uncertainty in the initial belief, the dynamics of the robots (translated in the transition probabilities) and the observation process effectively eliminate most such uncertainty after few steps.

The following very simple example clarifies the process by which uncertainty is eliminated.

Example 9.1. Consider the situation represented in Figure 9.15. We suppose that the state of each robot consists only on its position in the environment (1, 2, 3 or 4). We also suppose that there is only one action (Move forward). This action causes robot I to move to the right (unless if the robot is in the rightmost state) and robot II to stand still. We emphasize that the effects of the only action are *part of the specification of the problem*: they correspond to the information in the transition matrix P. We will only track the leftmost robot (Robot I).

Initially, we have no idea of where robot I is, so the belief consists on a uniform probability vector that we denoted by π_0. Both robots choose the action "move forward" and hence, at $t = 1$, robot I moves to Room 2, as described by the arrow in Figure 9.15, and Robot II remains in Room 4. Both robots get the observation "Robots in Rooms 2 and 4". Given this observation, Robot I must be in state 2 or 4 and the belief is updated as depicted in Figure 9.15 under π_1.

Now if both robots choose the action "move forward" again, at time $t = 3$ robot I will move to Room 3 and robot II will remain in Room 4. Clearly, both robots will receive the observation "Robots in Rooms 3 and 4". But, because of the transition model described above, this means that the robot in Room 2 at $t = 2$ moved to Room 3 at time $t = 3$ and the robot in *Room* 4 remained

Table 9.7: Average number of mis-coordinations, corresponding to the average number of negative rewards received by the team. We present the average obtained over 2,000 independent Monte-Carlo trials.

Experiment	Optimal	CAQL
W-GRID (2 robots)	3.200	4.092
BRIDGE (2 robots)	1.885	2.700
ISR (2 robots)	1.135	1.523
MIT (2 robots)	0.000	0.000
PENTAGON (2 robots)	0.000	0.021
CIT (2 robots)	0.000	0.000
SUNY (2 robots)	0.213	0.479
CMU (2 robots)	0.000	0.001

still. Since the robot that can actually move is robot I, it is easy to conclude that robot I was the one moving and is now in state 3 with 100% probability.

◇

Back to our results, if we take into consideration the fact that the dynamic model and observation model gradually eliminate the uncertainty in the belief, it is not surprising that the CAQL team is able to attain the impressive performance in Table 9.4, much superior than that observed in the single-robot case (see Chapter 5).

On the other hand, the success rate was measured upon 250-time-step runs, and all test scenarios have more than 1,000 states. Therefore, the success rate obtained cannot arise from simple exhaustive exploration.[3] By comparing the performance of CAQL with that of the fully observable team in terms of reward (Table 9.6), we conclude that even in the presence of partial observability the team is able to coordinate. This becomes clearer if we observe in Table 9.7 the average number of mis-coordinations (crashes) in each of the test environments. For example, in the the W-GRID scenario the average number of mis-coordinations for CAQL is 4.092. This means that, during the 250-time-units test period, the robots crash about 4 times in average. We remark that, as expected, the number of mis-coordinations is larger in the smaller environments, since the team has less "room" to move about.

When comparing the results of the "uncoordinated" team (Table 9.5) with those of the coordinated team (Table 9.4), it is evident that the use of the coordination mechanism greatly improves the performance of the team. Notice that both teams learn the same Q-values, as they both use the same algorithm to learn the game. This makes even more striking the difference in performance observed in the two tests.

Notice that, even if the smaller environments exhibit a larger number of mis-coordinations, mostly arising from the uncertainty in the transitions, the effect of these mis-coordinations in terms of the *success rate* of the team is much more relevant

[3]Exhaustive exploration would not guarantee that the goal state is ever visited within the 250 time-frame, and would lead to severe penalization arising from mis-coordinations.

in the larger environments. This fact can easily be interpreted. Notice, first of all, that the success rate measures the number of trials that the team was able to reach the final configuration. In the smaller environments, the final configuration can be reached very rapidly, as long as the robots are able to minimally coordinate. Therefore, mis-coordinations do affect the total discounted reward received, but will hardly prevent the team from reaching the goal. In the larger scenarios, because of the size of the environments, reaching the goal takes a significant amount of time and even one mis-coordination may translate in a delay that the team cannot afford. This indicates that coordination mechanisms may have a decisive influence in the team's ability to complete complex missions.

Finally, it is worth mentioning that all navigation problems have been formulated so as to meet the conditions of Theorem 8.6.1. Therefore, Theorem 8.4.2 guarantees that CAQL converges w.p.1 and coordinates in all but a null-measured set of states. On the other hand, Theorems 8.3.1 and 4.7.4 also guarantee that approximate Q-learning converges to the same approximate Q-function (even without coordination).

9.4 Concluding remarks

To conclude this chapter, we summarize the main ideas presented so far. We describe the general reinforcement learning framework introduced in the two previous chapters and discuss its applicability in multi-robot navigation problems. We also hint on how this framework can be extended to multi-agent settings, a topic to be developed in the second part of this thesis.

9.4.1 Summary

In this chapter we tested the CAQL algorithm in the large benchmark problems already considered in Chapter 5. In these large test scenarios, a team of mobile robots with centralized sensing capabilities must navigate from an initial configuration to a target configuration. The robots do not know beforehand how to coordinate in this task and must learn it as they interact in the common environment.

We applied CAQL to this set of problems and verified that, in all situations, the team is able to coordinate and reach the target configuration, exhibiting a nearly-perfect performance.

9.4.2 Discussion

This chapter brings the thesis to a conclusion. Although we leave for Chapter 10 the fundamental conclusions, we herein stress some important issues that arise from the results presented in this chapter. Together with those in Chapter 5, the results we presented herein assess experimentally the applicability of reinforcement learning in robotic navigation tasks. The classes of tasks considered fit into the description provided in Section 1.2, and we bring to a satisfying conclusion the study developed along the thesis.

In Chapters 6 through 8 we introduced and developed a general learning methodology to address multi-robot navigation tasks with sound theoretical properties. This

parallelled the development in Part I and culminates in this Chapter with the experimental validation of those methods. We must emphasize that the methods developed in the thesis (both in Part I and II) have a broader applicability than robotic navigation tasks, even if in the thesis these methods were approached envisioning this specific application. In fact, CAQL can be used to address any general multi-agent decision problem where coordination among the decision-makers is fundamental.

Finally, we notice that several of the concluding remarks in Chapter 5 are also applicable here. In particular, those regarding the fact concerning the structure of robot navigation tasks. Recall that, as argued in Chapter 5, robotic tasks exhibit some locality in the transitions – a robot cannot "jump" between arbitrary states. This locality helps to decrease uncertainty (as illustrated in Example 9.1) and makes robotic tasks particularly amenable to a RL approach relying in belief-states. When considering multiple robots, some of this locality is lost, precisely due to the existence and interaction of multiple robots. On the other hand, a team of robots with centralized observations will have access to more sensorial information than a single robot; this makes sense if we think that several people observing the same scene will be able to more accurately describe it as the observations of each one complement those of the others. This attenuates the effect of partial observability which, in turn, will often lead to better performance.

Chapter 10

General Conclusions

In this chapter we conclude the thesis, providing a general overview of all material presented. We summarize the main contributions and discuss several issues that have been briefly addressed along the thesis but whose discussion was postponed to this final chapter. We also point out several directions to be considered in future research.

10.1 Overview of the thesis

In this thesis we addressed mobile robot navigation from a learning perspective. We considered navigation tasks in which a robot/group of robots has to move from an initial position/configuration to a target position/configuration. The robots have access to a topological representation of the environment and should coordinate in the execution of the joint navigation task. We assumed that the task is initially unknown to the robots; they should be able to successfully explore the environment and use the evaluative feedback collected during this exploration period to learn the optimal way of completing the task at hand.

The thesis was divided in two parts. In Part I, we focused on single-robot navigation problems, where a single robot must navigate from an initial position to a final position in an environment described topologically. The use of *topological maps* provides environment representations with high-level of abstraction and inherent scalability properties. Topological environment representations are also closely related with *discrete-event models* describing the robot [188, 189].

The relation between the topological model of the environment and the discrete-event model of the robot led to the use of *Markov decision processes* (MDPs) as suitable models to address this class of navigation problems. We described several well-known methods (VI, ARTDP methods, Q-learning and SARSA) for computing the optimal decision-rule (or policy), given a finite MDP. However, all these methods require *full state observability*: the robot should be able to unambiguously determine its location in the environment in terms of the topological map.

Since perfect state perception is an unrealistic assumption in most situations, we adopted a more general model that encompasses partial observability. We showed

that these *partially observable Markov decision processes* (POMDPs) have an equivalent representation as fully observable MDPs with an infinite state-space. Therefore, we developed two new methods that make use of linear function approximation and tackle MDPs with infinite state-spaces. These methods, dubbed *approximate Q-learning* and *approximate SARSA*, and corresponding analysis, constituted the main contribution in the first part of the thesis. We concluded this part by validating the use of these methods in several large benchmark navigation problems.

In Part II, we addressed multi-robot problems, where a group of robots must jointly move from an initial configuration to a final configuration, while avoiding accidents and other undesirable situations. In this multi-agent setting, each robot moves autonomously and is, therefore, an independent decision-maker. However, the robots work as a team, in that they have a joint common goal (reach the target configuration). The robots should cooperate in attaining this joint goal without resorting to any explicit communication mechanism. Also, as in the problems considered in Part I, the robots do not know their task beforehand.

We adopted *team Markov games* (TMGs) as suitable models to address multi-robot navigation problems. We discussed three important issues arising in multi-agent cooperative settings: task representation, team performance evaluation and cooperation mechanism. The use of TMGs automatically settles the former two issues by means of a reward function that not only defines the task to be completed but also provides a natural way to evaluate the performance of the team. On the other hand, in the particular class of problems considered, cooperation arises in the form of *coordination*. We argue that coordination must be explicitly addressed and propose the use of *biased adaptive play* (BAP) as a coordination mechanism.

We then proceeded by extending several methods from Part I to multi-agent settings (namely ARTQI and Q-learning). These methods can be combined with BAP and used to determine an optimal coordinated decision-rule for finite TMGs. However, all these methods require *full state and action observability*. As in single-robot problems, full state observability means that each robot is able to unambiguously perceive the state of the whole group. Full action observability means that each robot is fully aware of the actions taken by all robots as soon as such actions are taken.

To tackle partial state observability we adopted the more general model of *partially observable team Markov games* (POTMGs) with *centralized observations* and showed this model to have an equivalent representation as a fully observable TMG with an infinite state-space. We developed a new coordination mechanism for infinite TMGs, *approximate biased adaptive play* (ABAP), and analyzed its main properties. We then combined ABAP with approximate Q-learning, leading to the *coordinated approximate Q-learning* algorithm, the final contribution of the thesis. This algorithm allows for a unified treatment of several different reinforcement learning problems and, in particular, of all navigation problems considered in the thesis. Finally, we discussed several approaches from the literature that address situations in which full action observability is not required and concluded the thesis by validating the use of CAQL in several large benchmark navigation problems.

10.1.1 Summary of contributions

We now present a very concise summary of all contributions of the thesis.

1. We proposed *approximate Q-learning*, an algorithm that combines Q-learning with linear function approximation, and established its convergence w.p.1 when using a fixed learning policy (Theorem 4.5.2);

2. We proposed *approximate* SARSA, an algorithm that combines SARSA with linear function approximation, and established its convergence w.p.1 when using a GLIE learning policy (Theorem 4.5.3);

3. As a corollary, we established the convergence of approximate Q-learning when using a GLIE learning policy (Corollary 4.5.4);

4. We established a partially observable Markov chain $(\mathcal{X}, \mathcal{Z}, \mathsf{P}, \mathsf{O})$ to be equivalent to a geometrically ergodic, fully observable Markov chain $(\mathbb{S}^n, \bar{\mathsf{P}})$ as long as the chain $(\mathcal{X}, \mathsf{P})$ is irreducible, aperiodic and there is an observable state (Theorem 4.7.4);

5. We proposed *coordinated Q-learning* (CQL), an algorithm that combines Q-learning with biased adaptive play, and established its convergence w.p.1 to an optimal Nash equilibrium (Theorem 7.3.3);

6. We proposed *approximate biased adaptive play* (ABAP), a coordination mechanism for infinite team Markov games, and established its convergence w.p.1 in all but a null-measured set of states (Theorem 8.4.1);

7. We then combined ABAP with approximate Q-learning, leading to the *coordinated approximate Q-learning* (CAQL) algorithm. We established convergence of CAQL w.p.1, unifying in a single, general result most previous results along the thesis (Theorem 8.4.2).

Contribution 1 was previously published in [193]. Contribution 4 can be found in [190, 192] along with the algorithm in Appendix E. Contribution 5 can be found in [189, 193]. A journal paper is in preparation that includes Contributions 6 and 7.

10.2 General discussion

Before concluding our presentation, it is important to discuss several important issues that arose along the thesis and whose discussion was postponed to these concluding remarks.

10.2.1 Reinforcement learning related issues

We start by discussing several general issues related with the reinforcement learning framework that do not restrict to the navigation problems considered in the thesis.

General conditions for convergence

We now analyze the conditions stated in the several convergence theorems. We also refer to the discussion in Chapters 4 and 8.

Most algorithms in the thesis are stochastic approximation algorithms, taking the general form

$$\theta_{t+1} = \theta_t + \alpha_t H(\theta_t, X_{t+1})$$

where H is a perturbed version of a function h whose zero is to be determined. The sequence $\{X_t\}$ is a Markov chain and the sequence $\{\alpha_t\}$ is a step-size sequence, defining the rate of convergence of the algorithm to the desired limit point.[1]

In most convergence theorems along the thesis (in particular, in Theorems 3.4.1, 3.4.2, 3.4.3, 4.5.2, 4.5.3, 7.3.3, 8.3.1, 8.4.2 and 8.6.1) we require the step size sequence to have infinite sum, while converging to zero. More exactly, we require that $\sum_t \alpha_t^2 < \infty$. The infinite sum guarantees that the algorithm converges to the desired limit, independently of how far the initial estimate is; the convergence to zero guarantees that the algorithm actually converges, preventing it from infinitely oscillating around the desired limit point. However, the convergence guarantee thus obtained is only asymptotic.

If only a finite number of iterations is possible, then the sequence of step-sizes should be *non-vanishing*. Convergence results for non-vanishing step-sizes are thoroughly available in the literature [21] and we refer to the detailed discussion in [51] on the practical implementation of such stochastic approximation methods.

On a more practical notice, we refer that most methods introduced in the thesis are *on-line* methods: the policy is learnt as the agent/agents interact with the environment. This means that sufficient time must be provided for the agent/agents to explore the environment and the possible inter-agent interactions. Even if the computational complexity involved in performing the updates in the methods is not significant, the actual implementation of these learning methods will always require a learning period for the agents to acquire the optimal policy. As already stated, this period permits sufficient exploration and its duration is usually adjusted experimentally.

With respect to the remaining conditions of convergence, two observations are in order. First of all, we have provided in Theorem 4.7.4 a set of conditions that guarantee that a POMDP can be transformed into an equivalent geometrically ergodic, fully observable MDP. The conditions are related with the ergodicity of the underlying MDP and with the existence of one *distinguishable state*. We have claimed that the latter assumption is not unreasonable in that we expect that the robots are at least able to distinctly recognize the *goal* of their task.

Secondly, simple ergodicity of the underlying unobserved MDP is, in general, not sufficient to guarantee the geometric ergodicity of the corresponding belief process [135]. For general partially observable models, geometric ergodicity of the belief process has been established using several approaches in different works [159, 304]. As in Theorem 4.7.4, all such results require the underlying unobserved process to

[1] To clarify the relation between the step-size sequence $\{\alpha_t\}$ and the rate of convergence of the algorithm, see Appendix D.

be ergodic. The advantage of our approach lies on the fact that the added condition is extremely simple to verify and trivially holds in many actual problems.

The choice of basis functions

Our approximate methods from Chapters 4 and 8 (approximate Q-learning, approximate SARSA and CAQL) make use of a finite-dimensional function space $\mathcal{Q}$ and approximate the best representation of the optimal Q-function in this space. This space is the linear span of a pre-defined set of basis functions $\Xi = \{\xi_1, \ldots, \xi_M\}$ and, as discussed in Chapter 4, the quality of the obtained approximation greatly depends on the family of basis functions chosen.

Two important issues immediately arise:

- *How to choose the basis functions.* The choice of basis functions may require some labor to provide a satisfying approximation. The importance of this problem has made it a topic of intense current research.

  Several authors have experimentally inferred the usefulness of different classes of basis functions, such as Boyan and Moore [44], Sutton [290] or Kretchmar and Anderson [146]. One particularly popular approximation relies on *tile coding*. In its original formulation, tile coding was known as CMAC (cerebellar model articulatory controller).[2] Tile coding produces a feature representation of the state-space which is similar to a network of fixed radial basis functions (RBF). One argued advantage of tile coding over standard RBF approximations lies on the computational efficiency with which tile coding can be implemented [290]. RBFs and tile coding are further compared in [146]. Examples of successful applications of tile coding within RL can be found in several works [40, 286, 337].

  In partially observable scenarios, the use of beliefs appears quite naturally and, as seen in Chapters 5 and 9, with quite satisfactory results in robotic applications. We also refer to Appendix E and references therein, where the use of point-sample projections provide a natural set of basis functions that yield an easily interpretable approximation.

  Finally, some researchers have focused on the problem of determining representative families of basis functions for which some theoretical performance guarantees can be provided. We refer the works by Glaubius and Smart [96], Menache et al. [197], Parr et al. [232] and Keller et al. [139].

- *How to choose the number M of basis functions.* Clearly, richer sets of basis functions lead to greater representative power and allow for more accurate representations of a broader class of functions. However, this affects the memory requirements and learning time of the algorithms. Therefore, there is an evident tradeoff between representational power of the set of basis functions and efficiency of the learning algorithm. In [6, 208, 301], the authors explore

[2] We refer to http://www.cs.ualberta.ca/%7Esutton/RL-FAQ.html#CMACs for the origin of the "tile coding" designation.

this tradeoff, providing explicit performance bounds for approximated value iteration.[3]

The use of GLIE policies/strategies and the problem of exploration

Along the thesis and in the practical implementation of all algorithms, we made generous use of GLIE policies/strategies. The policies/strategies with this property, first introduced by Singh et al. [280], have the evident advantage of converging to the greedy (and hopefully optimal) policy/strategy while ensuring sufficient exploration. However, this does not settle the exploration vs. exploitation tradeoff. It may so be the case that a limited time is available for learning and the agent should learn as much as possible in that time. In that case, a GLIE policy may not be adequate. On the other hand, it may occur that some actions are particularly undesirable and, so, exploration should be approached carefully. All these issues should be taken into account when considering the practical implementation of the algorithm.

Even so, we should remark that in the multi-agent setting, emerging coordination necessarily requires the several decision-makers to interact. GLIE strategies manage to provide this required interaction while the decision-makers are in the process of learning the game.

On-policy and off-policy algorithms

As established in Theorems 4.5.2 and 4.5.3, given a linear function space $\mathcal{Q}$, approximate Q-learning and approximate SARSA converge w.p.1 to the same limit point, verifying the fixed-point recursion

$$Q(\theta^*) = \mathcal{P}_{\mathcal{Q}}\mathbf{H}Q(\theta^*).$$

However, such convergence guarantee is asymptotic: convergence of the sequence $\{Q_t\}$ of estimates occurs only as $t \to \infty$. As seen in Chapter 4, the update rules for approximate Q-learning and approximate SARSA are distinct in the temporal-differences used—see (4.13) and (4.14).

The distinction between Q-learning and SARSA (and between approximate Q-learning and approximate SARSA) can be more clearly understood by sketching the trajectories of both algorithms. Such trajectories are pictorially represented in Figure 10.1. Suppose that both methods depart from the same initial condition Q_0, follow a similar GLIE policy with initial learning policy δ_0. The fact that the used policy has the GLIE property means that it "changes" slower than the sequence of estimates $\{Q_t\}$. From the sampled trajectory, SARSA will approach the Q-function corresponding to the policy δ_0, while Q-learning will approach the optimal Q-function. When the learning policy is "updated", Q-learning will exhibit a policy which is closer to optimal then SARSA's. This means that, in general, SARSA will be slightly slower due to its on-policy updates. This phenomenon was often observed in the results in Chapters 3, 4 and 5. Therefore, in finite time, there

[3] Although value iteration methods are quite distinct from stochastic approximation methods such as Q-learning, the referred bounds derived also hold for the latter class of algorithms.

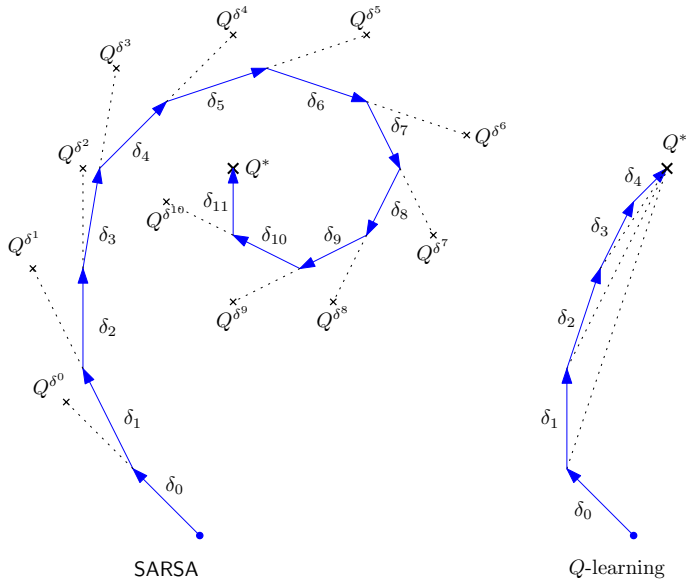


Figure 10.1: Pictorial representation of the difference between Q-learning and SARSAwith a GLIE learning policy.

will generally be a difference between the policies and functions learnt by Q-learning and SARSA. As seen in Chapter 4, this difference is more significant in environments which are very informative (*i.e.*, rewards are available in a large portion of the state-space).

Observability and communication

In all methods described and implemented in Part II, there was a fundamental assumption of full-action observability. In fact, even if we discussed in Section 8.6 several approaches from the literature that do not assume action observability, the methods considered in the thesis do rely on this assumption. Therefore, it is important to fully realize the exact implications of this assumption.

In a POTMG, the observations received by each player may provide information about the actions by the other players, *i.e.*, the actions may be perceived from *their effects* on the environment. When this is the case, each player will be able to *infer* the action taken by the other players from the sensorial information received. The player could thus maintain a *belief* on the actions taken by the other players and use this belief to build its individual strategy. This approach is simplified version of the I-POMDP approach described in Chapter 8.

Another possibility is to consider that the players *communicate a posteriori* the actions taken at each time instant. Using communication to overcome the problem of action observability is a common methodology, implicitly adopted in several works in the literature (see, for example, the discussion in [24, 212]). Notice, also, that the approach followed in Chapters 8 and 9 necessarily relies on some form of communication to ensure *centralized observations*. We could argue that, since some form of communication must always exist, then we could take advantage of this to ensure coordination. However, using communication to maintain "synchronized" joint information is fundamentally different from using it to achieve coordinated

decision-making: in the former case, the process of decision depends on the communication only indirectly. Failure in the communications will, in this case, delay the learning process and the coordination process but no guarantees of convergence are lost. On the other hand, if communication is used to ensure coordination, failure in the communications *may impair the ability of the robots to coordinate*.

In any case, using communication to address multi-agent problems with cognitive autonomy can significantly reduce the computational complexity of this class of problems [212]. Dec-POMDP models that explicitly consider communication, dubbed as COMM-MTDP (for communicating multiagent team decision process), exhibit a computational complexity similar to that of single agent POMDPs [246, 247]. And even if communication is not free, it may be used advantageously to decrease the overall complexity of the multi-agent decision problem [212].

To implement efficient communication Roth et al. [256, 257] propose the Dec-Comm algorithm that determines in which situations communication may bring the agents actual benefits, thus avoiding communication in situations where it brings no significant advantage. This idea is further pursued in [258], where further communication constraints are analyzed. In the latter paper, the authors address the problem of *what to communicate*, in the presence of communication costs or bandwidth limitations. Finally, in [259], the authors take advantage of factored representations for Dec-MDPs and are able to reduce communication to a peer-to-peer protocol.

This idea can be further explored, by considering separately situations in which an agent in a group can act independently and situations in which cooperation is mandatory. This line of work was already addressed in the aforementioned works [256, 257], that rely on communication to decentralize the execution of a joint strategy computed off-line. It would be interesting to develop a similar simplification but requiring no joint strategy to be determined. In those situations where the agents can act independently, the agents could simply follow a single-agent strategy, whose determination is computationally much less expensive than multi-agent strategies. In those situations where coordination is required, communication could be used to simplify the complexity of multi-agent decision-making, as suggested in [212].

Finally, the recent advances in actor-critic methods [145, 239, 296], their convergence guarantees and impressive robustness when combined with function approximation suggests that this class of algorithms may be successfully applied to multi-agent settings, even considering completely independent learners. Furthermore, such approach could be combined with ideas from [40] and from the recent developments in the theory of generalized fictitious play processes [161, 162].

10.2.2 Reinforcement learning in robot navigation

We now discuss several issues related with the application of the methods and framework described in the thesis to robotic navigation problems. We also review several works from the literature more or less related to the approach in the thesis.

RL approaches to robot control and navigation

The powerful methods and impressive results of RL [62, 307, 316] have rendered this framework quite popular among the computer science and robotic communi-

ties. Therefore, it is not surprising that several works address classical control and navigation problems using the RL formalism and methods. For example, Bradtke [47] and ten Hagen [306] apply reinforcement learning methods to perform linear quadratic regulation. Forbes [86] combines an instance-based function approximation architecture with Q-learning to the control of autonomous vehicles.

Closer to the approach in this thesis, Nourbakhsh et al. [222] describes the general architecture of the DERVISH robot, winner of the 1994 Robot Competition and Exhibition held as part of the 13th National Conference on Artificial Intelligence (AAAI'94). The paper describes the state-space representation and navigation algorithms used by the robot to navigate in an office environment. Its navigation algorithm relies on a *topological map* and maintains a belief-like representation of the position of the robot. Although not using the exact same approach, its working principle is very close to that used in this thesis.

In another work, Simmons and Koenig [142, 277] use a POMDP model to address autonomous navigation in an office environment. In this work, the authors address the particular problems of translating prior topological information and sensorial data into this partially observable MDP model. Shatkay and Kaelbling [274] further explore this approach by incorporating odometric information to replace the prior topological information, using an extension of the Baum-Welch algorithm to deal with relational and observation data.[4] It also discusses several interesting advantages of using topological maps in robot navigation. Finally, Cassandra et al. [53] follow on the previous works, addressing the general problem of belief-based robot navigation using a POMDP model. The authors also discuss several practical issues such as how to build the actual POMDP transition and observation matrices.

Roy and Thrun [261] also use a POMDP model for robot navigation. In this work, the authors uses a grid partition of the state-space (consisting of position-orientation tuples) and increase this state-space with entropy information.[5] The authors implement localization using belief tracking and propose a new method relying on *state entropy* to handle the poor POMDP scalability. The paper concludes with the application of this methodology in coastal robot navigation, using the MINERVA robot.

Thrun [311] addresses POMDPs with continuous state and action spaces combining importance sampling with Monte-Carlo methods for belief propagation. This method is then applied to a robotic locate-and-retrieve task.

In a different line of work, some researchers focused on continuous state-space problems, while disregarding partial observability. For example, Smart and Kaelbling [282] feature an application of Q-learning to mobile robot navigation in infinite spaces. In their approach, the authors consider two distinct learning phases: in a first phase, the robot follows a given pre-specified policy that samples the "interesting" parts of the state-space (those providing non-zero rewards). In the second phase, the robot combines Q-learning with a powerful function approximation architecture and uses the experience gathered during the first phase to speed the learning. Yen

[4] Baum-Welch is an applied expectation-maximization algorithm [248] usually used to learn HMMs.

[5] Belief entropy provides a useful measure of uncertainty and has been used in several approximate POMDP solution methods, such as [169, 191, 261].

and Hickey [344] make use of a local, feature-based representation of the environment and implement an episodic variation of Q-learning with a tunable forgetting factor. For a large value of the forgetting factor, the robot will re-explore in every learning episode. For a small forgetting factor, the robot will essentially exploit, using the information from the previous episodes. The lack of information due to the local representation is overcome by implementing a hierarchical architecture, where a lower-level layer handles simpler tasks such as local navigation and the higher-level layer handles global planning.

To conclude, we remark that both belief-based POMDP approaches and those disregarding partial observability but considering infinite state-spaces fall into a common category, in light of our results from Chapter 4. In this thesis we provide a unified approach to both classes of problems and further extend them to multi-robot settings.

Learning and policy implementation

In Chapter 2 we have motivated the use of a topological representation of the environment in addressing mobile robot navigation, due to its scalability and high level of abstraction. Also, as argued in Chapter 6, such topological representation brings added advantages in heterogeneous multi-robot situations, by providing a high-level environment representation than can be internally be translated into a suitable representation for each robot.

We also motivated the use of the reinforcement learning formalism and methodology to this class of problems: RL provides a class of powerful methods that allow a user to program a robot/group of robots by means of evaluative feedback, and constitutes an appealing alternative to exhaustive programming. As seen throughout the thesis, the reward function plays in RL a fundamental role, by providing the learner with the required feedback for the task at hand. From a formal point of view, *the reward function defines the task to be learnt.* Therefore, it is of the greatest importance that the reward function be defined in accordance with the task intended.

In the particular case of mobile robot navigation, as considered in this thesis, the reward functions used were often very simple: the robots were rewarded a positive reinforcement upon reaching their goal and punished with a negative reinforcement upon reaching undesirable situations. This, as intended, "pushes" the robots towards the goal and away from the undesirable situations. More complex tasks will often require more complex reward functions and the crafting of reward functions is an important topic of research, closely related with *utility theory.* This research led to several important works, regarding *policy invariance under reward shaping* [182, 221] and *inverse reinforcement learning* [1, 64, 220, 249].

Specifically concerning the learning process, two observations are in order. First of all, given a particular task for a robot to learn, and due to the exploration requirements during learning, the initial performance of the robot will be very modest. Furthermore, the more complex the task, the longer it will generally take to learn. For this reason, it is customary to consider an extensive learning period, during which the robot is able to explore at will. During this learning period, as argued

above, a non-vanishing learning rate should be used, to:

- speed up the convergence of the learning algorithm;
- minimize the effects of the finite learning time.

Once the learning period is over, the learnt policy can be directly implemented in the robot as a simple map from states to actions. Such simple policies are straightforward to implement and require little resources from the robot. Furthermore, they provide the explicit "state-wise" programming of the actions of the robots. On the other hand, if desired, it is also possible to simply eliminate all exploration but allowing the learning process to proceed. This will allow the robot to track slow changes in the environment and to eventually further improve its policy.

The second remark is concerned with the implementation of the learning period. Allowing the learning period to be conducted with the actual robot is often time-consuming and in some situations even lead to damage of the robot. Therefore, it is customary to develop simplified simulation models to run the learning process. Such simulation models often capture the fundamental situations where decision-making is required from the robot and allow for a much faster and hazard-safe learning period. However, simulation models can only provide approximate representations of the actual situations and it is convenient that, once the learnt policy is implemented in the real robot, the learning process is allowed to continue (with no exploration) as the robot interacts with the actual environment.

The use of robot and sensor models

In Chapters 4 and 8 we made use of belief-states to overcome the problem of partial observability. A belief-state is nothing but a probability distribution over the state-space updated using a simple Bayes rule. Belief-tracking in navigation tasks is equivalent to Markov localization, as seen in [87, 89]. Markov localization has been successfully applied in real robot navigation tasks with the RHINO and MINERVA robots [88].

We remark, however, that belief-tracking as described in Chapter 4 requires knowledge of the probabilities in P and in O. Recall that the probabilities $\mathsf{P}_a(i,j)$ define the dynamic model of the robot:

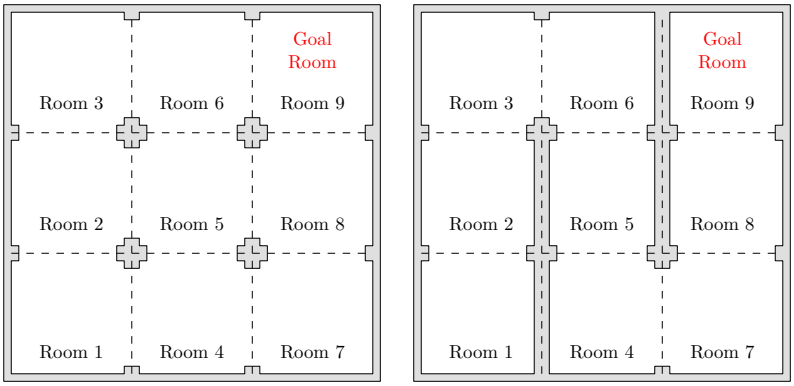
$$\mathsf{P}_a(i,j) = \mathbb{P}\left[X_{t+1} = j \mid X_t = i, A_t = a\right]$$

and the probabilities $\mathsf{O}_a(i,z)$ define the model for the sensors:

$$\mathsf{O}_a(i,z) = \mathbb{P}\left[Z_{t+1} = z \mid X_{t+1} = i, A_t = a\right].$$

Knowledge of P and O is equivalent to the knowledge of the dynamic model of the robot and the sensors. In robotic applications such as those envisioned in this thesis, knowledge of the dynamics of the robot and the model of the sensors is a fairly reasonable assumption.

On the other hand, it may often be the case that the dynamic/sensor model available are *inaccurate*. Although we did not address this problem in the thesis, in



a) Environment of Example 2.1 **b)** Modified environment

Figure 10.2: Original environment of Example 2.1 and modified environment that includes extra obstacles.

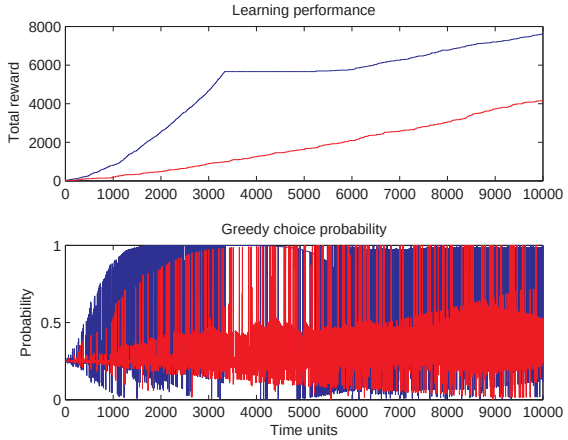


Figure 10.3: Learning performance of the robot when the environment dynamics are altered at time $t = 3330$. We also plot (in red) the learning performance of a second robot that only trained in the modified environment.

many situations the effect of inaccuracies in terms of belief-states and policy learning is easily accommodated by the learning algorithm.

To see this, consider the following example.

Example 10.1. We return to Example 2.1 and consider once again a mobile robot moving in the environment of Figure 10.2.a). The robot can be modeled by the MDP $(\mathcal{X}, \mathcal{A}, \mathsf{P}, r, \gamma)$ described Chapter 2 and the navigation problem can be solved by using any of the methods in the first part of the thesis.

Suppose now that, during learning, the environment is modified to accommodate two large obstacles that did not exist originally, as depicted in Figure 10.2.b). We tested the effect of this on the learning of the robot and plotted the obtained learning performance in Figure 10.3.

For the purpose of comparison, we allowed a second robot to learn in the modified environment from the start. We refer to the first robot as the *test*

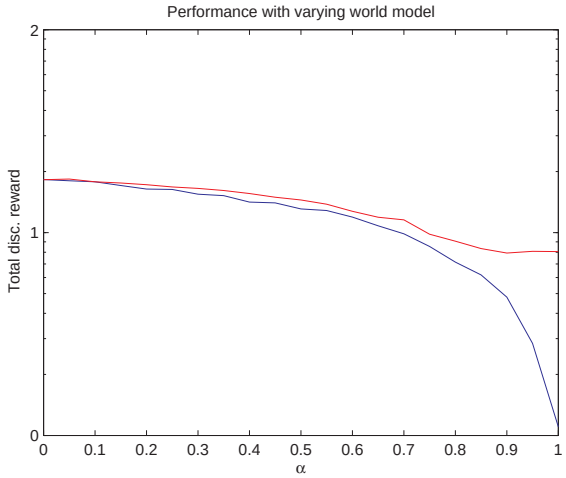


Figure 10.4: Performance of the robot following policy δ^0 as α goes from 0 to 1. For each α we also plot (in red) the performance of the optimal policy δ^α.

robot (plotted in blue) and the second robot as the *control robot* (plotted in red).

Notice two important phenomena. First of all, the abrupt change in the slope of the learning curve for the test robot clearly indicates that the sudden change in the scenario severely affected the performance of the robot. This is also visible in the plot indicating the probability of choosing the optimal action, that abruptly decreased.

The second important aspect to observe is that, after some time, the slope of the learning curve for the test robot again changes, indicating convergence to the optimal policy in the new environment. This can be verified by observing that the final slope of the learning curve for the test robot matches that of the control robot.

Denote by $\mathsf{P}^{(\mathrm{I})}$ the transition probability matrix for the environment in Figure 10.2.a) and $\mathsf{P}^{(\mathrm{II})}$ the transition probability matrix for the environment in Figure 10.2.b). Given a nonnegative $\alpha \leq 1$ the matrix P^α given by

$$\mathsf{P}_a^\alpha(x,y) = (1-\alpha)\mathsf{P}^{(\mathrm{I})} + \alpha\mathsf{P}^{(\mathrm{II})}$$

is clearly a transition probability matrix, a "mixture" of the two matrices $\mathsf{P}^{(\mathrm{I})}$ and $\mathsf{P}^{(\mathrm{II})}$. Denote by δ^α the optimal policy for the MDP $(\mathcal{X}, \mathcal{A}, \mathsf{P}^\alpha, r, \gamma)$, where $\mathcal{X}$, $\mathcal{A}$ and r are as defined in Example 2.1.

To further analyze the robustness of the learnt policy with respect to changes in the environment, we tested the performance of δ^0 as we changed α from 0 to 1. For each value of α, we tested δ^0 for 10 time units and plot in Figure 10.4 the average total discounted reward obtained over $2,000$ independent Monte-Carlo trials. We also plot, for each α, the average total discounted reward obtained using policy δ^α.

Notice that, for α approximately between 0 and 0.8 the performance of δ^0 is not significantly worse than that of the optimal policy δ^α. Only when the environment changes significantly (in this case, some of the paths are blocked) does the performance significantly worsen, as expected. ◇

This example illustrates that, for small changes in the model/sensor behavior, the learnt policy should not suffer a too severe degradation in performance. However, in the presence of severe model/sensor inaccuracies, the use of belief-tracking may lead to undesirable situations. For example, it may lead to beliefs that are inconsistent with the actual state of the process and induce meaningless action choice. In the presence of severe model inaccuracies, alternative solution methods must be considered. The study of *partially observable Markov decision processes with imprecise parameters* has been very recently introduced in [130] following on previous work on bounded-parameter MDPs [94].

Convergence conditions in practice

The several convergence theorems provided along the text identified multiple conditions for the corresponding conclusions to hold. These conditions can be grouped into two main classes: the conditions *on the algorithms* (that depend only on a careful implementation of the learning algorithm) and the conditions *on the process*. We now provide a brief overview of the latter class of conditions in terms of robotic navigation, in order to convey a clearer understanding of the theoretical requirements of our results in practical terms.

ψ-irreducibility The concept of irreducibility of Markov chains is deeply related with the ability of the chain to visit its state-space. Roughly speaking, the irreducibility of a Markov chain is, in a sense, similar to the concept of controllability in dynamic systems. The Markov processes considered in the thesis correspond to navigation problems and describe the movement of a robot/group of robots in an environment described by a topological map. The state-space of the process corresponds to the possible locations in this map, and irreducibility corresponds to the ability of a robot/group of robots to reach every part of the state-space. Therefore, this condition is trivially verified in robotic navigation tasks, since non-visitable states can be eliminated from the map.

Aperiodicity The concept of aperiodicity in Markov chains is related with the existence of periodic orbits in the trajectories of the Markov chain. Requiring the Markov chain to be aperiodic is the same as requiring the trajectories of the robot not to be forcefully periodic. In most scenarios, the need for infinite exploration automatically renders the chain aperiodic.

Recurrence The concept of recurrence is related with the ability of a chain to *re-visit* its state-space. This is also a reasonable condition to require, since it simply ensures that the robot does not get stuck in a part of the state-space. This can easily be prevented by resorting to straightforward analysis of the discrete-event model of the robot [194, 196].

Geometric ergodicity In chains with finite state-space, geometric ergodicity is an immediate consequence of irreducibility and aperiodicity. It corresponds to one of the strongest concepts of *stability* in Markov chains. A chain is geometrically ergodic if its "average" state-distribution converges exponentially

to a stationary measure. The verification of this condition for the problems of interest is conducted in Theorem 4.7.4, where an additional condition for geometric ergodicity of a POMDPs(besides irreducibility and aperiodicity) is identified that relies on the existence of a *distinguishable state*.

Similarity function Finally, the use of a similarity function to ensure coordination in infinite settings is, in fact, simple to justify. If a group of robots is able to coordinate in a given configuration X_t, it is only reasonable to assume that they should be able to coordinate (in a similar fashion) in nearby configurations. The use of a similarity function (which is user defined and can be chosen "almost" freely) simply formalizes this concept of "nearby" configuration.

10.3 Future work

In concluding this thesis, we now discuss several open issues that can yield interesting topics for future research.

First of all, as stated in Chapter 1, the thesis does not address the interface between the decision-maker, working at a high level of abstraction, and the low level interaction with the world (sensor data processing, actuator control, etc.). This means that the most immediate direction for future work is to pursue the development of such interface, aiming at validating the approach described herein in real robots. Some of the works referred in Subsection 10.2.2 and in Chapter 2 provide useful departing points, but a significant amount of work must be developed in the implementation of reliable low-level interface to allow the implementation of decision-making at the level discussed in this thesis.

A second direction for future work, already mentioned in Chapter 8, we make no use of eligibility traces in our algorithms. However, it is just expectable that the methods described herein can easily be adapted to accommodate eligibility traces, this eventually yielding better approximations (with tighter error bounds) [27, 321, 326].

On the other hand, although several negative counter-examples have been reported in the literature, it is our belief that convergence of RL methods with non-linear approximation architectures should be possible. This may be achieved by restricting the non-linear approximations to lie in some "well-behaved" manifold (*e.g.*, convex, smooth...). We belief that it is likely that the theory of projected dynamic systems [60] can be used to define a non-linear iterative method that relies on some sort of projection into a convex manifold.

Yet in the subject of non-linear approximation, it would be interesting if, while using essentially a linear approximation architecture, some parameters of the basis functions could also be learnt. This would perhaps lie as an "intermediate step" between linear and non-linear architectures.

With respect to the application of our approximate algorithms in partially observable scenarios, we implicitly assume the decision-maker to be able to maintain accurate beliefs on the state of the underlying Markov chain, this being a central point in the application of the algorithms. However, in large-dimensional spaces, this may not always be possible, and the decision-makers may only be able to maintain

approximate beliefs on the Markov chain. It is important to understand the impact of approximate beliefs in the convergence of our methods. This may be especially troublesome in multi-agent scenarios, if the use of approximated beliefs allows for different decision-makers having non-consistent beliefs. In this case, we once again face the problems occurring in problems with no centralized observations.

Also, as discussed in the end of Chapter 8, it would also be important to extend our framework and algorithms to situations where the agents have cognitive autonomy. As suggested in the aforementioned chapter, the use of I-POMDPs (with the individual beliefs over I-states) may be adequately combined with approximate Q-learning, yielding a learning algorithm applicable to each individual agent and that may converge to a subjective equilibrium [70, 98]. This encompasses multi-agent problems where the different agents may have distinct and even conflicting missions.

On a different perspective, the intractability of the several models proposed to address multi-agent problems with partial observability has led several researchers to consider several simplifying assumptions that may, potentially, decrease the complexity of solution methods. For example, the use of explicit communication may be used advantageously to decrease the overall complexity of the multi-agent decision problem [212]. How to make advantageous use of communication is, therefore, an interesting problem that may foster important advances in the theory of multi-agent decision-making.

Finally, it would be interesting to further extend the applicability of this RL approach to even more complex systems. A very challenging topic for future research is the combination of optimization strategies arising both from classical control and RL to hybrid or continuous-time problems.

Part III

Appendices

Appendix A

Some Mathematical Background

In this appendix, we present some mathematical tools that are used in the thesis.

We present a brief review on martingale sequences and its convergence. We then present two important inequalities that are used in different contexts: Hölder's inequality and Gronwall's inequality. We proceed with a result from probability theory describing the speed of convergence of particular sequences of random variables, such as those featured in the laws of large numbers or in the central limit theorem and conclude with some basic notions on measure theory.

A.1 Martingale sequences

A family of σ-fields $\{\mathcal{F}_n\}$ is said to be *increasing* if $\mathcal{F}_n \subset \mathcal{F}_{n+1}$. Given a σ-field $\mathcal{F}$, a r.v. X is $\mathcal{F}$-*measurable* if

$$\mathbb{E}[X \mid \mathcal{F}] = X.$$

Given an increasing family of σ-fields $\{\mathcal{F}_n\}$, a sequence of r.v.s $\{X_n\}$ is $\mathcal{F}_n$-*adapted* if X_n is $\mathcal{F}_n$-measurable, for all n. A sequence of r.v.s $\{X_n\}$ is said to be *previsible* if X_n is $\mathcal{F}_{n-1}$-measurable for all n.

Martingale Sequence

A *martingale* is a sequence of r.v.s $\{M_n\}$ such that

- For some increasing family of σ-fields $\{\mathcal{F}_n\}$, M_n is $\mathcal{F}_n$-adapted;
- $\mathbb{E}\left[|M_n|\right] < \infty$ for all n; and
- $\mathbb{E}\left[M_{n+1} \mid \mathcal{F}_n\right] = M_n$ w.p.1.

The same definition holds if $\{M_n\}$ is a sequence of random vectors, replacing the absolute value $|\cdot|$ by some norm $\|\cdot\|$ in the previous definition.

We now present some results on the convergence of martingale sequences. Proofs can be found in any standard book on probability theory or stochastic processes [58, 105].

Theorem A.1.1. *Let $\{M_n\}$ be a martingale and suppose that there is a constant $K > 0$ such that*

$$\mathbb{E}\left[|M_n|\right] < K, \tag{A.1}$$

for all n. Then, there is a r.v. M such that, w.p.1,

$$\lim_{n\to\infty} M_n = M$$

and $\mathbb{E}\left[M\right] < \infty$.

Notice that the condition (A.1) basically states that the sequence $\{M_n\}$ is uniformly bounded. Before introducing the next result, the following definition is needed.

Martingale Increment

If M_n is a martingale, the sequence $\{S_n\}$ given for each n by $S_n = M_n - M_{n-1}$ is called a *martingale increment*.

The next theorem relates the convergence of a martingale with the boundedness of the corresponding martingale increment.

Theorem A.1.2. *Let $\{M_n\}$ be a martingale and $\{S_n\}$ the corresponding martingale increment. If there is $1 \leq p \leq 2$ such that, w.p.1,*

$$\sum_{n=1}^{\infty} \mathbb{E}\left[|S_n|^p \mid \mathcal{F}_{n-1}\right] < \infty, \tag{A.2}$$

then the martingale $\{M_n\}$ *converges w.p.1.*

A.2 Several useful inequalities

In this section, we provide several inequalities that will prove useful throughout the thesis.

Lemma A.2.1 (Hölder Inequality)**.** *Let* X, Y *be r.v.s. For any* $p : 1 < p < \infty$ *and* $1/p + 1/q = 1$*,*

$$|\mathbb{E}[XY]| \leq \mathbb{E}[|XY|] \leq \mathbb{E}[|X|^p]^{\frac{1}{p}} \mathbb{E}[|Y|^q]^{\frac{1}{q}}. \tag{A.3}$$

By taking $Y \equiv 1$ in (A.3), we get a new, very useful, inequality

$$\mathbb{E}[|X|] \leq \mathbb{E}[|X|^p]^{\frac{1}{p}}.$$

We next present the Gronwall inequality, commonly used to bound the difference between the solutions of two differential equations $\dot{y}(t) = f(t, y(t))$ and $\dot{z}(t) = g(t, z(t))$ in terms of the difference between the initial conditions and/or the difference between f and g. The original inequality seems to have first appeared in [106]; a proof can be found in [126].

Theorem A.2.2 (Gronwall Inequality)**.** *Let* $\mathcal{X}$ *be a Banach space and* $U \subset \mathcal{X}$ *an open set in* $\mathcal{X}$*. Let* $f, g : [a, b] \times U \to \mathcal{X}$ *be continuous functions and let* $y, z : [a, b] \to U$ *satisfy the initial value problems*

$$\begin{aligned} \dot{y}(t) &= f(t, y(t)); & y(a) &= y_0; \\ \dot{z}(t) &= g(t, z(t)); & z(a) &= z_0. \end{aligned}$$

Also assume there is a constant $C > 0$ *so that*

$$\|g(t, x_2) - g(t, x_1)\| \leq C \|x_2 - x_1\|$$

and a continuous function $\varphi : [a, b] \to [0\infty)$ *so that*

$$\|f(t, y(t)) - g(t, y(t))\| \leq \varphi(t).$$

Then for $t \in [a, b]$*,*

$$\|y(t) - z(t)\| \leq e^{C|t-a|} \|y_0 - z_0\| + e^{C|t-a|} \int_a^t e^{-C|s-a|} \varphi(s) ds.$$

A.3 The law of the iterated logarithm

Let $\{X_n\}$ be a sequence of independent, identically distributed random variables, and let $S_n = \sum_{i=1}^{n} X_i$. Assume that $\mathbb{E}[X_n] = \mu < \infty$ and $\mathbb{E}\left[|X|^2\right] = \sigma^2$, where $0 < \sigma^2 < \infty$.

By the law of large numbers, we know that

$$\frac{S_n}{n} \to \mu \tag{A.4}$$

w.p.1 and, by the central limit theorem,

$$\frac{S_n - n\mu}{\sigma\sqrt{n}} \to \mathsf{N}(0,1) \tag{A.5}$$

in distribution. We denoted by $\mathsf{N}(0,1)$ the normal (Gaussian) distribution.

However, none of the two referred convergence results describes the amplitude of the fluctuations of each of the terms in the sequences S_n/n and $S_n/\sigma\sqrt{n}$ around the limits described in (A.4) and (A.5). The *law of the iterated logarithm* provides a bound on these fluctuations, conveying an estimate on the "speed of convergence" of these sequence to the corresponding limits.

Theorem A.3.1 (Law of the Iterated Logarithm)**.** *Let $\{X_n\}$ be a sequence of independent, identically distributed random variables verifying $\mathbb{E}[X_n] = \mu$ and $\mathbb{E}[X^2] = \sigma^2$, where $\mu < \infty$ and $0 < \sigma^2 < \infty$. Then,*

$$\limsup_{n\to\infty} \frac{|S_n - n\mu|}{\sigma\sqrt{2n\log\log n}} = 1 \tag{A.6}$$

w.p.1.

Notice that the law of iterated logarithm implies that given any constant $K > 1$,

$$\left|\frac{S_n}{n} - \mu\right| \leq K\sigma\sqrt{2n^{-1/2}\log\log n}$$

w.p.1, for sufficiently large n.

A.4 Some notes on measure spaces and norms

In this appendix, we provide some mathematical details on measure spaces and norms that play an important role in the interpretation of some of the results in the thesis.

Let X be a set and $\mathcal{B}(X)$ a σ-field on X. This means that $\mathcal{B}(X)$ is a family of subsets of X with the following properties:

- $X \in \mathcal{B}(X)$;

- If $U \in \mathcal{B}(X)$ then $X - U \in \mathcal{B}(X)$;
- For any index set I, if $U_i \subset X, i \in I$, then $\bigcup_{i \in I} U_i \in \mathcal{B}(X)$.

and the sets in $\mathcal{B}(X)$ are *measurable sets*. If X is endowed with a topology, we define the Borel σ-field on X as the σ-field generated by the open sets in X (*i.e.*, the "smallest" σ-field that includes all open sets). A *measure space* is a tuple $(X, \mathcal{B}(X), \mu)$, where μ is a *measure* on $\mathcal{B}(X)$. Without going into too many details, a measure on $\mathcal{B}(X)$ is a function $\mu : \mathcal{B}(X) \longrightarrow \mathbb{R}$ that assigns a "volume" to each set in $\mathcal{B}(X)$ and verifies

$$\mu\left(\bigcup_{i \in I} U_i\right) = \sum_{i \in I} \mu(U_i),$$

for any collection of sets $U_i \in \mathcal{B}(X)$ such that $U_i \cap U_j = \emptyset$ if $i \neq j$. It is a *probability measure* if $\mu(U) \geq 0$ for all $U \in \mathcal{B}(X)$ and $\mu(X) = 1$.

Let X be a set and $\mathcal{B}(X)$ a σ-field on X. Given two measures ν and μ on $\mathcal{B}(X)$, ν is *absolutely continuous* w.r.t. μ if $\nu(U) = 0$ for every measurable set U such that $\mu(U) = 0$ and we denote this as $\nu \ll \mu$. When $\nu \ll \mu$ and $\mu \ll \nu$, the two measures are said *equivalent*. When $\nu \ll \mu$, there is a μ-measurable function $f : X \longrightarrow \mathbb{R}$ such that, for any $U \in \mathcal{B}(X)$,

$$\nu(U) = \int_U f(x)\mu(dx).$$

The function f is known as the Radon-Nikodym derivative of ν w.r.t. μ and is usually denoted as

$$f = \frac{d\nu}{d\mu}.$$

A function $f : X \longrightarrow \mathbb{R}$ is *measurable* if $f^{-1}(U) \in \mathcal{B}(X)$ for any measurable set $U \subset \mathbb{R}$. A function $f : X \longrightarrow \mathbb{R}$ is a *simple function* if there is a collection of measurable sets $U_i, i = 1, \ldots, n$ and a set of real numbers $b_i, i = 1, \ldots, n$ such that $f(x) = b_i$ if $x \in U_i$. In this case, we write

$$\int_X f d\mu = \int_X f(x)\mu(dx) = \sum_{i=1}^{n} b_i \mu(U_i).$$

Simple functions are always measurable and for any non-negative measurable function there is a sequence of simple functions $\{f_k\}$ such that, for every $x \in X$, $f(x) = \lim_{k \to \infty} f_k(x)$ and we can take

$$\int_X f(x)\mu(dx) = \lim_{k \to \infty} \int_X f_k(x)\mu(dx).$$

For a general measurable function, we can always write $f(x) = f^+(x) - f^-(x)$, with $f^+(x) = \max(f(x), 0)$ and $f^-(x) = \max(-f(x), 0)$ and

$$\int_X f(x)\mu(dx) = \int_X f^+(x)\mu(dx) - \int_X f^-(x)\mu(dx).$$

Generally, when integrating over the whole space X, we represent the above expression succinctly as (μf) or, if μ is a probability measure, as $\mathbb{E}[f]$.

The total variation norm of a measure μ on $\mathcal{B}(X)$ is defined as

$$\|\mu\| = \sup_{|f|\leq 1} |\mu f| = \sup_{U\in\mathcal{B}(X)} \mu(U) - \inf_{U\in\mathcal{B}(X)}$$

where f is any real valued function defined on X.

Two functions $f, g : \mathcal{X} \longrightarrow \mathbb{R}$ are *equal μ-a.e.* if

$$\int_X |f(x) - g(x)|\, \mu(dx) = 0.$$

Notice that this defines an equivalence relation on all real functions defined on X. The space of such equivalent classes is *linear* and can be endowed with an inner product or a norm. We refer to such space as the *space of measurable functions on X*. When this is the case, we usually abusively refer to the *norm of a function f* while meaning the *norm of the equivalence class of f*.

Given a measure space $(X, \mathcal{B}(X), \mu)$, we define the following norms on the space of measurable functions on X:

$$\|f\|_1 = \int_X |f(x)|\, \mu(dx);$$
$$\|f\|_2 = \left(\int_X f^2(x)\mu(dx)\right)^{\frac{1}{2}};$$

and, more generally

$$\|f\|_p = \left(\int_X f^p(x)\mu(dx)\right)^{\frac{1}{p}}.$$

We consider one further norm

$$\|f\|_\infty = \sup_{x\in X} |f(x)|,$$

where the sup above corresponds to the *essential supremum*, defined for the equivalence class of function f as

$$\sup_{x\in X} f(x) = \inf_{g\sim f} \sup_{x\in X} g(x),$$

where the sup in the right-hand side of the expression above now corresponds to the standard supremum. When clear from the context, we use the sup operator without specifying whether referring to the standard supremum or the essential supremum.

Finally, given a measure space $(X, \mathcal{B}(X), \mu)$, the space of all measurable functions f such that $\|f\|_p < \infty$ is a *Banach space* and is denoted as $\mathcal{L}^p(X)$. In particular,

- $\mathcal{L}^1(X)$ is known as the *space of absolutely integrable functions*;

- $\mathcal{L}^2(X)$ is a *Hilbert space* with the inner product

$$\langle f, g \rangle = \int_X f(x)g(x)\mu(dx)$$

and is known as the *space of square-integrable functions*;

- The space of all measurable functions f such that $\|f\|_\infty < \infty$ is a Banach space denoted as $\mathcal{L}^\infty(X)$ and known as the space of *essentially bounded functions*.

Appendix B

Markov Chains and Stochastic Stability

The purpose of this appendix is to provide a set of analytical tools regarding the stability of Markov chains in general state-spaces, namely, geometric ergodicity.

We present fundamental concepts such as ψ-irreducibility, aperiodicity and recurrence. We describe how these properties relate with the stable behavior of the trajectories of the chain. We also provide conditions under which the law of large numbers, the central limit theorem and the law of iterated logarithm hold along the trajectories of the chain, and introduce the Poisson equation.

Throughout the appendix, we closely follow the terminology and notation of [201] and we refer to this book for formal proofs of the statements presented here.

B.1 Markov chains and transition probabilities

Before actually getting into Markov chains, we introduce some general concepts on stochastic processes, to establish the notation.

A *stochastic process* $\{X_t\}$ is a sequence of r.v.s, each $X_t \in \mathcal{X}$ and where t is a parameter taking values in some index set $\mathcal{T}$. Parameter t is usually referred as *time* although it may bear no relation with any actual time.[1] A stochastic process can be classified as *discrete-time* or as *continuous-time*, depending on the set $\mathcal{T}$ being discrete or continuous.

Given a probability measure μ defined on $\mathcal{B}(\mathcal{X})$, suppose that

$$\mathbb{P}[X_0 \in U] = \mu(U).$$

Then, μ is called the *initial measure* for the process $\{X_t\}$. We use $\mathbb{P}_\mu[\cdot]$ to denote the probability induced by the initial measure μ and $\mathbb{E}_\mu[\cdot]$ the corresponding expected value. In particular, if $\mu = \delta_x$ for some $x \in \mathcal{X}$, we write $\mathbb{P}_x[\cdot]$ instead of $\mathbb{P}_\mu[\cdot]$ and $\mathbb{E}_x[\cdot]$ instead of $\mathbb{E}_\mu[\cdot]$.[2]

Given any initial measure μ, define the probability kernel P as

$$\mathsf{P}(x, U) = \mathbb{P}_x[X_1 \in U],$$

with $x \in \mathcal{X}$ and $U \in \mathcal{B}(\mathcal{X})$.

Time-homogeneous Markov Chain

A discrete-time stochastic process $\{X_t\}$ is a *time-homogeneous Markov chain* if, given any family of sets $\{U_0, \ldots, U_t\}$ in $\mathcal{B}(\mathcal{X})$,

$$\mathbb{P}_\mu[X_0 \in U_0, \ldots, X_t \in U_t] = \int_{U_0} \ldots \int_{U_{t-1}} \mathsf{P}(dx_{t-1}, U_t) \ldots \mathsf{P}(x_0, dx_1) d\mu(x_0), \tag{B.1}$$

for all $t \in \mathcal{T}$.

Notice that the property expressed in (B.1) is equivalent to the more known property

$$\mathbb{P}[X_{t+1} \in U \mid \mathcal{F}_t] = \mathbb{P}[X_t \in U \mid X_{t-1} = x] = \mathsf{P}(x, U),$$

in that any chain verifying any of the two necessarily verifies the other. We denoted by $\mathcal{F}_t$ the σ-field generated by the history of the process up to time t. The kernel P is called a *transition probability kernel*. A Markov chain is thus defined by its state-space $\mathcal{X}$ and the corresponding transition probabilities, determined by P.

It can also be shown that, given an arbitrary set $\mathcal{X}$ and a probability kernel P there is a Markov chain with state-space $\mathcal{X}$ and corresponding transition probabilities defined by P. Then, it is possible to refer to a Markov chain as a pair $(\mathcal{X}, \mathsf{P})$ where $\mathcal{X}$ is a set and P is a properly defined probability kernel. We refer interchangeably to a Markov chain either as a pair $(\mathcal{X}, \mathsf{P})$ or a sequence $\{X_t\}$.

[1] This nomenclature goes back to the early works on stochastic processes in which such processes were used to study time series of random events.

[2] We denoted by δ_x the point-mass function at x.

Notice that, given an initial measure μ, it is possible to determine the probability $\mathbb{P}_\mu[X_t \in U]$ for any $t \in \mathcal{T}$. From (B.1) it is immediate that

$$\mathbb{P}_\mu[X_t \in U] = \int_{\mathcal{X}} \dots \int_{\mathcal{X}} \mathsf{P}(dx_{t-1}, U) \dots \mathsf{P}(x_0, dx_1) d\mu(x_0).$$

In order to simplify the notation, define the kernel P^k recursively as

$$\mathsf{P}^1(x, U) = \mathsf{P}(x, U)$$

and

$$\mathsf{P}^k(x, U) = \int \mathsf{P}(y, U) \mathsf{P}^{k-1}(x, dy).$$

The kernel P^k is known as the *k-step transition probability kernel*, since it defines the a k-step transition probability:

$$\mathsf{P}^k(x, U) = \mathbb{P}[X_{t+k} \in U \mid X_t = x].$$

The kernels introduced next play an important role in the theory of Markov chains.

m-Skeleton Chain

Given a Markov chain $(\mathcal{X}, \mathsf{P})$, the *m-skeleton chain* of $(\mathcal{X}, \mathsf{P})$ is the Markov chain $(\mathcal{X}, \mathsf{P}^m)$.

Resolvent Chain

Given a Markov chain $(\mathcal{X}, \mathsf{P})$, the *resolvent chain* for $(\mathcal{X}, \mathsf{P})$ is the Markov chain defined for $0 < \varepsilon < 1$ as $(\mathcal{X}, \mathsf{K}_{a_\varepsilon})$, with the kernel $\mathsf{K}_{a_\varepsilon}$ given by

$$\mathsf{K}_{a_\varepsilon}(x, U) = (1 - \varepsilon) \sum_{i=0}^{\infty} \varepsilon^i \mathsf{P}^i(x, U),$$

for every $x \in \mathcal{X}$ and every $A \in \mathcal{B}(\mathcal{X})$.

Sampled Chain

Let a be a probability measure defined on $\mathcal{B}(\mathbb{N})$. Given a Markov chain $(\mathcal{X}, \mathsf{P})$, the *a-sampled chain* for $(\mathcal{X}, \mathsf{P})$ is a Markov chain $(\mathcal{X}, \mathsf{K}_a)$, with the kernel K_a defined as

$$\mathsf{K}_a(x, U) = \sum_{i=0}^{\infty} a(i)\mathsf{P}^i(x, U),$$

for every $x \in \mathbb{S}$ and every $U \in \mathcal{B}(\mathcal{X})$. The probability measure a is referred to as a *sampling distribution*.

Each of the chains $(\mathcal{X}, \mathsf{P}^m)$, $(\mathcal{X}, \mathsf{K}_{a_\varepsilon})$ and $(\mathcal{X}, \mathsf{K}_a)$ can be interpreted as a Markov chain obtained from $(\mathcal{X}, \mathsf{P})$ by sampling it at random times such that the time-difference between two consecutive samples, t_s, verifies respectively

$$\begin{aligned} \mathbb{P}[t_s = m] &= 1; \\ \mathbb{P}[t_s = i] &= (1-\varepsilon)\varepsilon^i; \\ \mathbb{P}[t_s = i] &= a(i). \end{aligned}$$

B.2 Irreducibility

The concept of irreducibility of a Markov chain was first introduced in the context of countable state-space Markov chains. In this setting, an irreducible Markov chain visits every state in the state-space with non-zero probability. This is a desirable property for a Markov chain, since it ensures that the state-space $\mathcal{X}$ is not split into several regions that do not communicate. It will soon become apparent that irreducibility is one of the most basic requirements for a Markov chain to exhibit "stable" behavior.

When dealing with infinite state-space Markov chains, the concept of irreducibility becomes slightly more elaborate, giving rise to the alternate notions of irreducibility measure and ψ-irreducibility.

Given a Markov chain $(\mathcal{X}, \mathsf{P})$ and a set $U \in \mathcal{B}(\mathcal{X})$, let τ_U denote the first time that the Markov chain reaches the set U, *i.e.*,

$$\tau_U = \min_{t \in \mathcal{T}} \{X_t \in U, t \geq 1\}.$$

The value τ_U is a r.v. representing the first time instant in which the chain enters the set U. As such, it is usually referred to as the *first return time* to set U.

Irreducibility Measure

Let φ be some measure on $\mathcal{B}(\mathcal{X})$. A Markov chain $(\mathcal{X}, \mathsf{P})$ is *φ-irreducible* if,

given any set $U \in \mathcal{B}(\mathcal{X})$, whenever $\varphi(U) > 0$ we have, for all $x \in \mathcal{X}$,

$$\mathbb{P}_x \left[\tau_U < \infty\right] > 0.$$

In a φ-irreducible chain, every φ-large set is reached in finite time with non-zero probability. However, the converse may not hold: there may be sets reached in finite time with non-zero probability that are φ-small. This leads to

Proposition B.2.1. *Let $(\mathcal{X}, \mathsf{P})$ be a φ-irreducible chain, for some measure φ. Then, there is a* maximal irreducibility measure *ψ such that*

- *$(\mathcal{X}, \mathsf{P})$ is ψ-irreducible;*
- *Given any measure φ', if $(\mathcal{X}, \mathsf{P})$ is φ'-irreducible, then $\psi \gg \varphi'$;*
- *If $\psi(U) = 0$, then $\psi\big(\{x \in \mathcal{X} \mid \mathsf{P}_x[\tau_U < \infty] > 0\}\big) = 0$;*

Proposition B.2.1 motivates the following definition.

ψ-Irreducible Chain

A Markov chain $(\mathcal{X}, \mathsf{P})$ is *ψ-irreducible* if it is φ-irreducible for some measure φ and ψ is a maximal irreducible measure verifying the conditions in Proposition B.2.1.

A set $U \in \mathcal{B}(\mathcal{X})$ is *full* if its complementary set $U^c = \mathcal{X} - U$ has zero ψ-measure, *i.e.*, if $\psi(U^c) = 0$. A set $U \in \mathcal{B}(\mathcal{X})$ is *absorbing* if $\mathsf{P}(x, U) = 1$, for all $x \in U$.

It can be shown that all irreducibility measures are equivalent [201]. Therefore, given a ψ-irreducible Markov chain, we denote by $\mathcal{B}^+(\mathcal{X})$ the family of all sets with positive ψ-measure without specifically identifying the the maximal irreducibility measure ψ.

B.3 Minorization properties

Markov chains have been around since 1906, when they were first formalized by the Russian mathematician Andrei Markov (1856-1922) [180, 181]. Afterwards, Markov chains were generalized to countable spaces by Andrei Kolmogorov (1903-1987) [144]. Markov chains in countable state-spaces have been the central topic in numerous publications and have been applied in many different fields of research,

from engineering to economic modeling. A good reference on countable state-space Markov chains is the book by Kijima [141].

The concepts introduced in this section establish the conditions that allow the state-space of a Markov chain to be interpreted as an countable agglomerate of "small" sets, each behaving as a single state in terms of transition probabilities.

Small Set

Given a Markov chain $(\mathcal{X}, \mathsf{P})$, a set $C \in \mathcal{B}(\mathcal{X})$ is a *small set* if there exists some $m > 0$ and a non-trivial measure ν_m defined on $\mathcal{B}(\mathcal{X})$ such that

$$\mathsf{P}^m(x, U) \geq \nu_m(U), \tag{B.2}$$

for all $x \in C$ and all $U \in \mathcal{B}(\mathcal{X})$. Whenever (B.2) holds, the set C is said to be ν_m-small.

A small set may be roughly interpreted as a set of states that behave as a whole, single "super-state" in the m-skeleton chain $(\mathcal{X}, \mathsf{P}^m)$ (in terms of transition probabilities). A generalization of this concept for a generically sampled Markov chain is given in the following definition.

Petite Sets

Given a Markov chain $(\mathcal{X}, \mathsf{P})$ and a sampling distribution a, a set $C \in \mathcal{B}(\mathcal{X})$ is a *ν_a-petite set* if there exists a non-trivial measure ν_a on $\mathcal{B}(\mathcal{X})$ such that

$$\mathsf{K}_a(x, U) \geq \nu_a(U), \tag{B.3}$$

for all $x \in C$ and all $U \in \mathcal{B}(\mathcal{X})$.

It should be clear from the very definition of petite set that every small set is petite. In the next section we see that the converse also holds for ψ-irreducible, aperiodic chains.

Proposition B.3.1. *Given a ψ-irreducible Markov chain $(\mathcal{X}, \mathsf{P})$, the following statements hold:*

1. *If a set $U \in \mathcal{B}(\mathcal{X})$ is ν_a-petite, there is a sampling distribution b and a maximal irreducibility measure ψ_b such that U is also ψ_b-petite;*

2. *The union of two petite sets is petite;*

3. *There is a sampling distribution c and an increasing sequence $\{C_i\}$ of ψ_c-petite sets, all corresponding to the same sampling distribution c, such that $\bigcup_i C_i = \mathcal{X}$, where ψ_c is a maximal irreducibility measure equivalent to ψ.*

The importance of the previous proposition lies in the fact that it establishes a close relation between the classical concept of irreducibility for Markov chains with countable state-space and the concept of ψ-irreducibility for Markov chains with a general state-space.

Since each set C_i, being petite, behaves as a "whole" in terms of transition probabilities for the chain $(\mathcal{X}, \mathsf{K}_c)$, we can interpret each C_i as a "generalized state" of the sampled chain. Assertion 3 of Proposition B.3.1 now states that the state-space $\mathcal{X}$ is simply the union of a countable collection of such generalized states. Therefore, we expect the chain $(\mathcal{X}, \mathsf{K}_c)$ to be similar to a chain with countable state-space, in terms of irreducibility behavior.

B.4 Periodicity

In the theory of Markov chains, periodicity is a property deeply related to the behavior of the trajectories of a particular chain. Therefore, such concept plays a central role in establishing the concept of stability for Markov chains.

Theorem B.4.1. *Consider a ψ-irreducible Markov chain $(\mathcal{X}, \mathsf{P})$ and let $C \in \mathcal{B}^+(\mathcal{X})$ be a ν_m-small set. Let*

$$E_C = \{n \geq 1 \mid C \text{ is } \nu_n\text{-small, with } \nu_n = k_n \nu_m\},$$

where the k_n are positive constants for each $n \in E_C$. If d is the g.c.d. of E_C, there exists a family $\mathcal{D}$ of disjoint sets $D_1, \ldots, D_d \in \mathcal{B}(\mathcal{X})$ such that

- *For every $x \in D_i$, $\mathsf{P}(x, D_{i+1}) = 1$, $i = 1, \ldots, d \pmod d$;*
- *$\psi\big(\mathcal{X} - \bigcup_{i=1}^{d} D_i\big) = 0$.*

The family $\mathcal{D}$ is called a d-cycle and is maximal.

We notice that maximal in the context of the theorem means that if there is a d'-cycle, then either d' divides d or $d' = d$. In the latter case, and upon an index reordering, $D_i = D_{i'}$ almost everywhere.

As stated in [201], the d-cycle does not depend on the particular small set C chosen, except, perhaps, on a ψ-null set of points.

Periodic and Aperiodic Markov Chains

Let $(\mathcal{X}, \mathsf{P})$ be a ψ-irreducible Markov chain. The largest d for which there is a d-cycle for the Markov chain is called the *period* of $(\mathcal{X}, \mathsf{P})$. If $d = 1$, the chain is said to be *aperiodic* and *periodic* otherwise.

A Markov chain with period d returns to a set $D_i \in \mathcal{D}$ at time t only if $t = kd+i$, for some $k \in \mathbb{N}$.

The following theorem establishes an important relation between aperiodicity and minorization properties in a Markov chain.

Theorem B.4.2. *If $(\mathcal{X}, \mathsf{P})$ is an aperiodic ψ-irreducible Markov chain, then every petite set is small.*

B.5 Topology in Markov chains

So far, Markov chains have been considered as having general state-spaces with no particular properties. However, in many practical situations, the state-space is actually endowed with a suitable topological structure (for example, $\mathcal{X}$ is a subset of $\mathbb{R}^p$, for some $p \in \mathbb{N}$). If this is the case, it is important to perceive the relation between concepts such as petiteness and common topological concepts, such as compactness or openness.

Whenever a Markov chain $(\mathcal{X}, \mathsf{P})$ is defined in a topological space $\mathcal{X}$, we implicitly assume that $\mathcal{X}$ is a locally compact, separable and metrizable topological space, with $\mathcal{B}(\mathcal{X})$ the corresponding Borel σ-field.

Feller Chain

Consider a Markov chain $(\mathcal{X}, \mathsf{P})$ defined in a topological space $\mathcal{X}$. If, given any open set $O \in \mathcal{B}(\mathcal{X})$, the function

$$P_O(x) = \mathsf{P}(x, O)$$

is a lower semi-continuous function, then $(\mathcal{X}, \mathsf{P})$ is called a *Feller chain.*

A function f is lower semi-continuous if

$$\liminf_{y \to x} f(y) \geq f(x).$$

Therefore, the Feller property basically ensures that, given a point $x \in \mathcal{X}$ and an open set O, the transition probabilities from x to O do not "drastically" decrease in points around x. The following theorem provides some tools that may help to determine whether a given chain verifies the Feller property.

Theorem B.5.1. *A Markov chain $(\mathcal{X}, \mathsf{P})$ verifies the Feller property (i.e., is a Feller chain) if and only if P maps the set $\mathcal{C}(\mathcal{X})$ of all bounded, continuous functions defined on $\mathcal{X}$ into itself.*

The following results, however simple they may appear, provide powerful tools in assessing the stability of Markov chains.

Theorem B.5.2. *Let $(\mathcal{X}, \mathsf{P})$ be a ψ-irreducible Feller chain. If there is an open petite set $O \in \mathcal{B}^+(\mathcal{X})$, then all compact subsets of $\mathcal{X}$ are petite.*

Theorem B.5.3. *Let $(\mathcal{X}, \mathsf{P})$ be a ψ-irreducible Feller chain. If ψ has non-empty support, then all compact subsets of $\mathcal{X}$ are petite.*

B.6 Invariant measures

Invariant measures play an important role in the analysis of convergence of Markov chains. An invariant measure μ^* is a measure on $\mathcal{B}(\mathcal{X})$ that remains unchanged under P. Therefore, invariant measures in Markov chains play a similar role to that of equilibrium points in dynamical systems.

Invariant Measure

Given a Markov chain $(\mathcal{X}, \mathsf{P})$, a probability measure μ^* is called invariant if

$$\int_{\mathcal{X}} \mu^*(dx)\mathsf{P}(x, U) = \mu^*(U). \tag{B.4}$$

It is now convenient to introduce the following notation. Given a measurable

function f defined on $\mathcal{X}$ and a probability measure μ on $\mathcal{B}(\mathcal{X})$, we define

$$(\mathsf{P}f)(x) = \mathbb{E}_x\left[f(X_1)\right] = \int_{\mathcal{X}} f(y)\mathsf{P}(x,dy);$$
$$(\mu\mathsf{P})(U) = \mathbb{P}_\mu\left[X_1 \in U\right] = \int_{\mathcal{X}} \mathsf{P}(x,U)d\mu(x);$$
$$(\mu f) = \mathbb{E}_\mu\left[f(X_0)\right] = \int_{\mathcal{X}} f(x)d\mu(x).$$

In terms of these operators, (B.4) can be rewritten as

$$\mu^*(U) = (\mu^*\mathsf{P})(U).$$

It arises as a consequence of the definition of invariant measure that, given an initial probability measure μ on $\mathcal{B}(\mathcal{X})$, if the limit distribution

$$\bar{\mu} = \lim_{n\to\infty} \mu\mathsf{P}^n$$

exists, then it must be an invariant distribution. This can be seen by performing the following computation:

$$\bar{\mu} = \lim_{n\to\infty} \mu\mathsf{P}^n = \lim_{n\to\infty} \mu\mathsf{P}^{n+1} = \lim_{n\to\infty} \mu\mathsf{P}^n\mathsf{P} = \bar{\mu}\mathsf{P}. \tag{B.5}$$

Positive Chain

If $(\mathcal{X},\mathsf{P})$ is a ψ-irreducible Markov chain admitting an invariant probability measure μ^*, then $(\mathcal{X},\mathsf{P})$ is called a *positive chain*.

The existence of an invariant probability measure greatly depends on the long-term behavior of the Markov chain and on the frequency with which the chain returns to ψ-large sets. This recurrent behavior is described and explored in the following section. For now, we conclude this section with the following result.

Theorem B.6.1. *If μ^* is the invariant probability measure for a positive ψ-irreducible Markov chain, then μ^* is equivalent to ψ.*

B.7 Recurrence and drift

As referred in the previous section, the existence of an invariant probability measure μ^* is related to the recurrence of the trajectories described by a Markov chain. Define

η_U as the number of visits to a set $U \in \mathcal{B}(\mathcal{X})$, *i.e.*,

$$\eta_U = \sum_{t=0}^{\infty} \mathbb{I}_U(X_t),$$

where $\mathbb{I}_U$ is the indicator function of the set U.

Recurrent and Transient Sets

Given a Markov chain $(\mathcal{X}, \mathsf{P})$, a set $U \in \mathcal{B}(\mathcal{X})$ is *recurrent* if

$$\mathbb{E}_x [\eta_U] = \infty,$$

for any $x \in \mathcal{X}$. Otherwise, U is *transient*. In particular, if a constant $M > 0$ exists such that, for all $x \in \mathcal{X}$,

$$\mathbb{E}_x [\eta_U] < M,$$

the set U is *uniformly transient*.

The following theorem establishes an essential relation between ψ-irreducibility and recurrence/transience.

Theorem B.7.1. *If $(\mathcal{X}, \mathsf{P})$ is a ψ-irreducible Markov chain, one of the two following statements holds:*

- *Every set in $\mathcal{B}^+(\mathcal{X})$ is recurrent, and $(\mathcal{X}, \mathsf{P})$ is called a* recurrent chain*;*
- *There is a countable covering of $\mathcal{X}$ by uniformly transient sets, and $(\mathcal{X}, \mathsf{P})$ is called a* transient chain*. In this case, every petite set is uniformly transient.*

Another important concept in the analysis of the long-term behavior of a Markov chain is the concept of *drift*. The drift measures the "expected change" in the chain during one time step. It is possible to define a Lyapunov-like criterion to establish the stability/unstability of a Markov chain in terms of this drift.

Drift

Given a non-negative measurable function $V : \mathcal{X} \longrightarrow \mathbb{R}$, the *drift operator* Δ is

defined as

$$(\Delta V)(x) = (\mathsf{P}V)(x) - V(x) = \int_{\mathcal{X}} V(y)\mathsf{P}(x, dy) - V(x),$$

for all $x \in \mathcal{X}$.

The referred introduction of the Lyapunov-like criterion to establish stability of a Markov chain in terms of convergence to an invariant measure is addressed in detail in Section B.8. Such criterion is sustained by the recurrence properties of Markov chains. A first and very simplified version of such result is provided in the following theorem.

Theorem B.7.2. *Consider a ψ-irreducible Markov chain $(\mathcal{X}, \mathsf{P})$. Then,*

1. *$(\mathcal{X}, \mathsf{P})$ is transient if and only if there exists a bounded non-negative function V and a set $C \in \mathcal{B}^+(\mathcal{X})$ such that, for all $x \in C^c$,*

$$(\Delta V)(x) \geq 0$$

and there is a set B defined as

$$B = \left\{ x \in \mathcal{X} \mid V(x) > \sup_{y \in C} V(y) \right\}$$

such that $B \in \mathcal{B}^+(\mathcal{X})$.

2. *$(\mathcal{X}, \mathsf{P})$ is recurrent if there exists a petite set $C \in \mathcal{B}(\mathcal{X})$ and a non-negative measurable function V such that, for all $x \in C^c$,*

$$(\Delta V)(x) \leq 0$$

and the family of sets $\{B_t\}$ defined as

$$B_t = \{x \in \mathcal{X} \mid V(x) \leq t\}$$

is such that each B_t is petite.

The principle behind this theorem is simple to explain. Interpret the function V as a Lyapunov-like function and the function (ΔV) as its temporal derivative. Statement 1 states that once the chain is off the set C, it is not likely to return to C. Then C is transient and, since $C \in \mathcal{B}^+(\mathcal{X})$, Theorem B.7.1 implies that $(\mathcal{X}, \mathsf{P})$ is transient. On the other hand, statement 2 establishes C as a petite recurrent set which, by Theorem B.7.1, implies $(\mathcal{X}, \mathsf{P})$ to be recurrent.

The notion recurrence can be further strengthened to that of Harris recurrence.

Harris Recurrence

A set $U \in \mathcal{B}(\mathcal{X})$ is said to be *Harris recurrent* if, for any $x \in \mathcal{X}$,

$$\mathbb{P}_x [\eta_U = \infty] = 1.$$

A ψ-irreducible Markov chain in which all sets $U \in \mathcal{B}^+(\mathcal{X})$ are Harris recurrent is a *Harris (recurrent) chain.*

B.8 Ergodicity

In this section, we address the problem of existence of the invariant probability measure μ^* and the convergence behavior of the chain with respect to this measure.

An invariant measure μ^* remains unchanged under the operator P. As such, a Markov chain with initial distribution $\mu_0 = \mu^*$ is described, at all times, by the invariant measure μ^* and is therefore *stationary.*

In dynamic systems theory, the study of stability is related with the convergence of the system towards stationary behavior and the same occurs in Markov chains. In particular, the most powerful stability result presented here bears interesting resemblances with the known Lyapunov stability theorem for dynamic systems.

Theorem B.8.1. *Every recurrent Markov chain $(\mathcal{X}, \mathsf{P})$ admits a unique (up to constant multiples) invariant measure μ^*. In particular, if there is a petite set $C \in \mathcal{B}(\mathcal{X})$ such that*

$$\sup_{x \in C} \mathbb{E}_x [\tau_C] < \infty,$$

the invariant measure μ^* is finite and the chain $(\mathcal{X}, \mathsf{P})$ is positive.

The concept of stability for Markov chains is related with the convergence of the trajectories of the chain towards stationarity. Therefore, we need the following concepts.

Ergodic Markov Chain

A Markov chain $(\mathcal{X}, \mathsf{P})$ is ergodic if

$$\left| \mathsf{P}(x, U)^t - \mu^*(U) \right| \to 0$$

for any $x \in \mathcal{X}$ and any $U \in \mathcal{B}(\mathcal{X})$.

Geometrically Ergodic Markov Chain

A Markov chain $(\mathcal{X}, \mathsf{P})$ is geometrically ergodic if, given any initial measure μ_0 on $\mathcal{B}(\mathcal{X})$,

$$\sum_{t=0}^{\infty} r^t \left\| \mu_0 \mathsf{P}^t - \mu^* \right\| < \infty$$

where r is some constant such that $r > 1$ and $\|\cdot\|$ is the total variation norm.

Uniformly Ergodic Markov Chain

A Markov chain $(\mathcal{X}, \mathsf{P})$ is uniformly ergodic if

$$\sup_{x \in \mathcal{X}} \left\| \mathsf{P}^t(x, \cdot) - \mu^* \right\| \to 0$$

as $t \to \infty$.

The two following theorems establish sufficient conditions to guarantee ergodicity and geometric ergodicity.

Theorem B.8.2. *Let $(\mathcal{X}, \mathsf{P})$ be a positive aperiodic Harris chain with invariant probability measure μ^* on $\mathcal{B}(\mathcal{X})$. Then, the following statements hold:*

1. *There is a petite set $C \in \mathcal{B}(\mathcal{X})$ such that*

$$\sup_{x \in C} \mathbb{E}_x \left[\tau_C \right] < \infty;$$

2. *There is a petite set $C \in \mathcal{B}(\mathcal{X})$ and a non-negative function V finite at some point $x_0 \in \mathcal{X}$ such that*

$$(\Delta V)(x) \leq -1 + b \mathbb{I}_C(x),$$

for some constant $b > 0$;

3. *For any* $x \in \mathcal{X}$,
$$\sup_{U \in \mathcal{B}(\mathcal{X})} |\mathsf{P}(x, U) - \mu^*(U)| \to 0. \tag{B.6}$$

Theorem B.8.3. *Let* $(\mathcal{X}, \mathsf{P})$ *be a* ψ*-irreducible and aperiodic Markov chain. If the chain is positive recurrent with invariant probability measure* μ^* *on* $\mathcal{B}(\mathcal{X})$*, the following statements are equivalent:*

1. *There is a petite set* $C \in \mathcal{B}(\mathcal{X})$ *such that*
$$\sup_{x \in C} \mathbb{E}_x \left[k^{\tau_C}\right] < \infty,$$
for some constant $k > 1$*;*

2. *There is a petite set* $C \in \mathcal{B}(\mathcal{X})$ *and a non-negative function* $V \geq 1$ *finite in some point* $x_0 \in \mathcal{X}$ *such that*
$$(\Delta V)(x) \leq -\beta V(x) + b\mathbb{I}_C(x),$$
where β *and* b *are constants verifying* $b < \infty$ *and* $\beta > 0$*;*

3. *For any* $x \in \mathcal{X}$,
$$\sum_{t=0}^{\infty} r^t \left\|\mu_0 \mathsf{P}^t - \mu^*\right\| < \infty, \tag{B.7}$$
for some constant $r > 1$.

We conclude this section with the following convergence theorem, binding all concepts of ergodicity introduced.

Theorem B.8.4. *For any Markov chain* $(\mathcal{X}, \mathsf{P})$ *the following are equivalent:*

1. *The chain is uniformly ergodic;*

2. *There are constants* $r > 1$ *and* $R < \infty$ *such that, for all* $x \in \mathcal{X}$,
$$\left\|\mathsf{P}^t(x, \cdot) - \mu^*\right\| \leq R r^{-t};$$

3. *There is* $t \in \mathcal{T}$ *such that*
$$\sup_{x \in \mathcal{X}} \left\|\mathsf{P}^t(x, \cdot) - \mu^*\right\| < 1;$$

4. *The state-space $\mathcal{X}$ is ν_m-small for some m;*

5. *The chain is aperiodic and there is a petite set $C \in \mathcal{B}(\mathcal{X})$ with*

$$\sup_{x \in \mathcal{X}} \mathbb{E}_x [\tau_C] < \infty,$$

in which case for every $U \in \mathcal{B}^+(\mathcal{X})$,

$$\sup_{x \in \mathcal{X}} \mathbb{E}_x [\tau_U] < \infty;$$

6. *The chain is aperiodic and there is a petite set $C \in \mathcal{B}(\mathcal{X})$ and a constant $k > 1$ with*

$$\sup_{x \in \mathcal{X}} \mathbb{E}_x [k^{\tau_C}] < \infty,$$

in which case for every $U \in \mathcal{B}^+(\mathcal{X})$,

$$\sup_{x \in \mathcal{X}} \mathbb{E}_x [k^{\tau_U}] < \infty;$$

7. *The chain is aperiodic and there is a petite set $C \in \mathcal{B}(\mathcal{X})$ and a everywhere bounded function $V \geq 1$ such that*

$$(\Delta V)(x) \leq -\beta V(x) + b\mathbb{I}_C(x),$$

where β and b are constants verifying $b < \infty$ and $\beta > 0$.

B.9 Limit theorems and the Poisson equation

Given a Markov chain $(\mathcal{X}, \mathsf{P})$ and a measurable function $f : \mathcal{X} \longrightarrow \mathbb{R}$, the Poisson equation for the chain w.r.t. f is

$$(\mathbf{I} - \mathsf{P})\nu(x) = f(x) - \mu f, \tag{B.8}$$

where $\mathbf{I}$ is the identity operator. If the chain $(\mathcal{X}, \mathsf{P})$ is geometrically ergodic, the solution ν to (B.8) is well defined for all $x \in \mathcal{X}$ and verifies

$$\nu(x) = \sum_{t=0}^{\infty} \left(\mathsf{P}^t(f - \mu f)\right)(x).$$

We can interpret the solution $\nu(x)$ of the Poisson equation as being the total "error" obtained by replacing each term $f(x_t)$ by its mean value μf along a trajectory $\{x_t\}$ of the chain starting at a point $x_0 = x$.

The existence of a solution for the Poisson equation (B.8) is important in deriving limit theorems for Markov chains such as the law of large numbers or the law of the

iterated logarithm. Let $f : \mathcal{X} \longrightarrow \mathbb{R}$ be a measurable function and $(\mathcal{X}, \mathsf{P})$ a positive Harris chain with invariant probability measure μ^*. Define the sequence $\{S_t(f)\}$ as

$$S_t(f) = \sum_{i=0}^{t} f(X_i)$$

for each $t \in \mathcal{T}$. We have the following result.

Theorem B.9.1. *If $(\mu^* |f|) < \infty$ then the law of large numbers holds for the sequence $\{S_t\}$. In other words,*

$$\lim_{t\to\infty} \frac{1}{t} S_t(f) = (\mu^* f)$$

w.p.1.

By further strengthening the conditions on the Markov chain, we obtain the following results.

Theorem B.9.2. *Let $(\mathcal{X}, \mathsf{P})$ be a uniformly ergodic Markov chain and f a measurable function such that $f^2(x) \leq 1$. Denote by F the centered function*

$$F(x) = f(x) - (\mu^* f).$$

Then, the constant

$$\sigma^2 = \mathbb{E}_{\mu^*}\left[F^2(X_0)\right] + 2\sum_{i=1}^{\infty} \mathbb{E}_{\mu^*}\left[F(X_0)F(X_i)\right]$$

is well defined, non-negative and finite. Furthermore,

$$\lim_{t\to\infty} \frac{1}{t} S_t^2(F) = \sigma^2.$$

Theorem B.9.3. *Under the conditions of Theorem B.9.2, if $\sigma^2 > 0$ the central limit theorem and the law of the iterated logarithm hold for the sequence $\{S_t(F)\}$, i.e.,*

$$\lim_{t\to\infty} \frac{S_t(F)}{\sigma\sqrt{t}} \stackrel{d}{=} \mathsf{N}(0,1)$$

in distribution and

$$\limsup_{t\to\infty} \frac{S_t(F)}{\sigma\sqrt{2t\log\log t}} = 1.$$

* * *

Consider now a controlled Markov chain $(\mathcal{X}, \mathsf{P}_\theta)$, where the transition kernel P_θ is allowed to depend on a parameter θ. Let μ_θ be the invariant probability measure for the chain $(\mathcal{X}, \mathsf{P}_\theta)$. A simpler version of the following theorems can be found in the notes on stochastic approximation by Delyon [67] and in a more general form in the book by Benveniste et al. [21]. In the following theorems, $\|\cdot\|$ denotes a generic norm.

Theorem B.9.4. *Let $(\mathcal{X}, \mathsf{P}_\theta)$ be a controlled Markov chain on a compact space $\mathcal{X}$ with control parameter θ. If, for θ in some compact $\mathcal{C}$, the Markov chain $(\mathcal{X}, \mathsf{P}_\theta)$ is geometrically ergodic and there is $q \geq 1$ such that*

$$\int_{\mathcal{X}} \|\mathsf{P}_\theta^q(x, dy) - \mathsf{P}_{\theta'}^q(x, dy)\| \leq K_1 \|\theta - \theta'\| \tag{B.9}$$

for some $K_1 > 0$, then there is a constant $\rho : 0 < \rho < 1$ independent of θ such that

$$\|\mathsf{P}_\theta^n - \mu_\theta\| < K_2 \rho^n,$$

for some $K_2 > 0$ (which may depend on θ).

Theorem B.9.5. *Let $(\mathcal{X}, \mathsf{P}_\theta)$ be a controlled Markov chain on a compact space $\mathcal{X}$ with control parameter θ. Suppose that the conditions of Theorem B.9.4 are verified. Let $H : \mathcal{C} \times \mathcal{X} \longrightarrow \mathbb{R}^M$ be a measurable function such that, for all θ in some compact set $\mathcal{C}$,*

$$\|H(\theta, x)\| \leq K_3(1 + \|\theta\|) \tag{B.10a}$$

$$\|\mathsf{P}_\theta^n H(\theta, \cdot) - \mathsf{P}_{\theta'}^n H(\theta', \cdot)\| \leq K_3 \|\theta - \theta'\|, \tag{B.10b}$$

for $0 \leq n \leq q$ and some positive constant K_3. Then, for each $\theta \in \mathcal{C}$ there is a solution ν_θ for the Poisson equation

$$(\mathbf{I} - \mathsf{P}_\theta)\nu_\theta(x) = H(\theta, x) - (\mu_\theta H)(\theta),$$

and the solutions ν_θ verify

$$\|\nu_\theta(x)\| \leq K_4$$

and

$$\|\nu_\theta(x) - \nu_{\theta'}(x)\| \leq K_4 \|\theta - \theta'\|,$$

for all $x \in \mathcal{X}$, $\theta, \theta' \in \mathcal{C}$ and some positive constant K_4.

B.10 Discrete state-spaces

All the results presented so far in this appendix concern Markov chains with a general (infinite) state-space. However, it is very common in applications that the actual state-space is countable or even finite. The following results are simple restatements of the central theorems presented so far when considering the particular case of countable/finite Markov chains.

Consider a Markov chain $(\mathcal{X}, \mathsf{P})$ on a countable state-space $\mathcal{X}$. Since $\mathcal{X}$ is countable, we assume with no lack of generality that $\mathcal{X} = \{1, 2, \ldots, n, \ldots\}$. In this particular case, the kernel P can be represented as a matrix.

Given any two elements $i, j \in \mathcal{X}$, j *is accessible from* i (denoted $i \to j$) if there is $k > 0$ such that

$$\mathsf{P}^k(i, j) = \mathsf{P}[X_{t+k} = j \mid X_t = i] > 0.$$

If i and j are mutually accessible, then we say that they *communicate* and denote it as $i \leftrightarrow j$. Notice that the relation $\leftrightarrow$ is an equivalence relation. Denote by $\mathcal{E}(i)$ the equivalence class of element i.

Irreducible Markov Chain

A Markov chain $(\mathcal{X}, \mathsf{P})$ defined in a countable set $\mathcal{X}$ is *irreducible* if for some $i \in \mathcal{X}$, $\mathcal{E}(i) = \mathcal{X}$.

Positive Markov Chain

A Markov chain $(\mathcal{X}, \mathsf{P})$ defined in a countable state-space $\mathcal{X}$ is *positive* if there is an invariant probability vector, *i.e.*, if there is a probability vector μ^* such that

$$\mu^{*\top} \mathsf{P} = \mu^{*\top}$$

and $\mu^*(i) > 0$ for all $i \in \mathcal{X}$.

We conclude this appendix on Markov chains with the following propositions.

Proposition B.10.1. *Every irreducible, aperiodic, positive Markov chain* $(\mathcal{X}, \mathsf{P})$ *defined in a countable set* $\mathcal{X}$ *is ergodic.*

Proposition B.10.2. *Every irreducible and aperiodic Markov chain* $(\mathcal{X}, \mathsf{P})$ *defined in a finite set* $\mathcal{X}$ *is ergodic.*

Proposition B.10.3. *Every ergodic Markov chain* $(\mathcal{X}, \mathsf{P})$ *defined in a countable set* $\mathcal{X}$ *is geometrically ergodic.*

Appendix C

Game Theory and Markov Games

In this appendix we briefly review several important concepts on game theory.

We present several fundamental concepts such as pure and mixed strategy, best response and Nash equilibrium. We describe the principles behind fictitious play and adaptive play and provide important theorems regarding the convergence of these strategies. For the particular case of zero-sum games we provide a brief description of the minimax principle and a result on the existence of a max-minimization strategy.

Throughout the appendix, we closely follow the terminology and notation of [226, 227] and we refer to these books for formal proofs of the statements presented herein.

C.1 Strategic games

Game theory provides a mathematical framework to model situations in which several decision-makers interact. These interactions can take place at different levels,

going from "games" in the common everyday sense to more complex interactions, such as those taking place in economical or biological systems.

Consider a set of N players, which we abusively denote by N. We write $\mathcal{A}^k$ to represent the set of all actions available to player k. The global set of possible actions is denoted by $\mathcal{A} = \times_{k=1}^{N} \mathcal{A}^k$, where $\times_{k=1}^{N} \mathcal{A}^k$ represents the cartesian product of the family of sets $\left\{\mathcal{A}^k, k = 1, \ldots, N\right\}$. An element $a \in \mathcal{A}$ is a N-tuple $a = (a^1, \ldots, a^N)$ and is called a *joint action* or *action profile*. Each element $a^k \in \mathcal{A}^k$ is an individual action. The tuple

$$a^{-k} = (a^1, \ldots, a^{k-1}, a^{k+1}, \ldots, a^N)$$

is a *reduced joint action*, and we write

$$a = (a^{-k}, a^k)$$

to indicate that the individual action of player k in the joint action a is a^k.

We need the following definition.

Preference relation

A binary relation $\succsim$ on a set A is called a *preference relation* if it is complete, reflexive and transitive.[1]

A strategic game is a possible model of interaction between decision-makers. This model consists of a set of players, a set of possible actions for each player and a set of preferences over the set of possible actions. Formally,

Strategic game

A *strategic game* is a tuple $\left(N, (\mathcal{A}^k), (\succsim^k)\right)$, where

- N is the set of players;
- For each player $k \in N$, $\mathcal{A}^k$ represents the set of all possible actions available to player k;
- For each player $k \in N$, $\succsim^k$ is a preference relation on the set $\mathcal{A} = \times_{k=1}^{N} \mathcal{A}^k$— the preference relation of player k.

The preference relation $\succsim^k$ of player k can often be represented using a *reward function* $r^k : \mathcal{A} \longrightarrow \mathbb{R}$. The reward function is also known as *payoff function* or

[1] A binary relation $\sim$ on a set X is *complete* if, given any two elements $a, b \in X$, either $a \sim b$ or $b \sim a$; it is *reflexive* if, given any $a \in X$, $a \sim a$; it is *transitive* if, given any $a, b, c \in X$, $a \sim b$ and $b \sim c$ implies that $a \sim c$.

Table C.1: The game of matching pennies. The numbers represent the win/loss in Euros.

	Heads	*Tails*
Heads	$+1, -1$	$-1, +1$
Tails	$-1, +1$	$+1, -1$

utility function, and is built so as to verify, for any $a, b \in \mathcal{A}$, $r^k(a) \geq r^k(b)$ whenever $a \succsim^k b$. Therefore, whenever the preferences for each player are established through the rewards r^k, we represent a strategic game as a tuple $\big(N, (\mathcal{A}^k), (r^k)\big)$.[2]

Example C.1 (Matching Pennies). Consider the game of matching pennies, in which two players simultaneously choose whether to show the head or the tail of a coin. If the sides of the two coins match, player 2 pays player 1 the amount of 1 €. Otherwise player 1 pays player 2 the amount of 1 €. This game is summarized in table C.1. The choice of player 1 being *Heads* or *Tails* corresponds to the first and second rows in Table C.1. Similarly, the choice of player 2 being *Heads* or *Tails* corresponds to the first and second columns in Table C.1. The numbers in the table correspond to the rewards received by players 1 and 2, respectively.

In this game, the interest of the players are exact opposites: the win of one implies the loss of the other. Games where the interests of the different players are exact opposites are known as being *strictly competitive* or *zero-sum games* on account of the fact that the sum of the rewards of the players is always zero. These games are further analyzed in Section C.3. ◇

Example C.2 (Prisoner's Dilemma). Suppose that two suspects of a major crime are held under custody for interrogation. The interrogators keep the two suspects in separate cells. There is enough evidence on each of the suspects to keep them on custody for minor accusations, but no evidence that relates either of the suspects with the major crime, unless one (or both) of the suspects acts as an informer. If either of the suspects acts as an informer, it will provide evidence to convict the other of the major crime and will be released as a witness. However, if *both* suspects decide to confess, they will both be convicted of the major crime. This game is summarized in table C.2.

[2]Defining a preference relation in terms of a pay-off function can be somewhat misleading. Consider a situation where $\mathcal{A} = \{a, b, c\}$. Suppose that the preferences of player k are given by the reward function r^k, where $r^k(a) = 0$, $r^k(b) = 1$ and $r^k(c) = 100$. This payoff function should be interpreted as simply stating that $c \succsim^k b \succsim^k a$. A second payoff function r_1^k such that $r_1^k(a) = 0$, $r_1^k(b) = 100$ and $r_1^k(c) = 101$ represents just as well the preferences of player k, and so does any payoff function r^k verifying $r^k(c) > r^k(b) > r^k(a)$.

Table C.2: The prisoner's dilemma. The numbers represent the number of years in liberty.

	Quiet	*Inform*
Quiet	−1, −1	−4, 0
Inform	0, −4	−3, −3

Unlike in the game of matching pennies, in this game the interests of the players are not exact opposites.

Notice that both players have to win from cooperating, since they both prefer the action profile (*Quiet, Quiet*) to the action profile (*Inform, Inform*). On the other hand, since no player can be sure of what the other player's action choice, it will do best by choosing the individual action *Inform*.

Take, for example, player 1. If player 2 chooses to remain quiet, then player 1 does better by informing. However, if player 2 chooses to inform, player 1 will also do better by informing. The same reasoning can be replicated for player 2. Therefore, the outcome of the game will be the (unexpected) action profile (*Inform, Inform*). ◇

The payoffs in games such as the prisoner's dilemma or matching pennies can be represented by matrices, as evident from Tables C.1 and C.2. This is usually the case in strategic games defined by reward functions, and such games are also known as *matrix games*.

We now address the process of decision-making in strategic games.

C.1.1 Nash equilibria

In strategic games (and other classes of games) there is an implicit assumption of rationality on the players. This means that each player i chooses from all its individual actions available the best action according to its preference relation $\succsim^i$.

When choosing its action, each player is expected to compare the different actions and opt for the "best" action as provided by its preference relation. However, the best action for each player in a game often depends on the other player's choice of actions. This leads to the following definition.

Nash Equilibrium

A *Nash equilibrium* of a strategic game $\big(N, (\mathcal{A}^k), (\succsim^k)\big)$ is an action profile $a^* \in \mathcal{A}$ such that, for every player $k \in N$,

$$a^* \succsim^k \big((a^*)^{-k}, a^k\big),$$

for all $a^k \in \mathcal{A}^k$.

Table C.3: Equivalent reward function for the prisoner's dilemma.

	Quiet	*Inform*
Quiet	2, 2	0, 3
Inform	3, 0	1, 1

A Nash equilibrium can be interpreted as an action profile capturing a *steady-state play* in the strategic game $\big(N, (\mathcal{A}^k), (\succsim^k)\big)$. In fact, if a^* is a Nash equilibrium, no player benefits from individually deviating its play from a^*.

Example C.2. (cont.) Consider once again the game of the prisoner's dilemma. The payoff function summarized in Table C.2 is equivalent to Table C.3 (see footnote 2 in page 267).

As already seen, since there is no communication between the players, the outcome of the game will be (*Inform, Inform*). In fact, assuming both players to be rational, given any of the other player's choice, each player can always do better by choosing *Inform*. The action profile (*Inform, Inform*) is, therefore, a Nash equilibrium for the prisoner's dilemma. ◇

Not every strategic game has a Nash equilibrium. For example, the matching pennies game does not have a Nash equilibrium. The problem of existence of equilibria is addressed next.

C.1.2 Best response

In finding the Nash equilibria for a strategic game it is often useful to consider each player's *best response functions*. The best response function for player $k \in N$ is a set-valued function $B^k : \mathcal{A}^{-k} \longrightarrow 2^{\mathcal{A}^k}$ such that, for every $a^k \in B^k(a^{-k})$,

$$r^k\big(a^{-k}, a^k\big) \geq r^k(a^{-k}, b^k),$$

for any $b^k \in \mathcal{A}^k$. In other words, the individual actions in $B^k(a^{-k})$ are the best responses for player k when the other players play the reduced action a^{-k}. A Nash equilibrium is an action profile a^* such that

$$(a^*)^k \in B^k\big((a^*)^{-k}\big), \qquad \text{for all } k \in N.$$

Define the set-valued function $B : \mathcal{A} \longrightarrow 2^{\mathcal{A}}$ by

$$B(a) = \times_{k=1}^{N} B^k(a^{-k}).$$

Clearly, a Nash equilibrium is a point $a^* \in \mathcal{A}$ such that $a^* \in B(a^*)$. Using Kakutani's fixed point theorem, it is possible to show the following

Proposition C.1.1. *A strategic game* $\left(N, (\mathcal{A}^k), (\succsim^k)\right)$ *has a Nash equilibrium if, for all* $k \in N$*, the set* $\mathcal{A}^k$ *is a non-empty, compact and convex subset of* $\mathbb{R}^p$ *and the preference relation* $\succsim^k$ *is*

- *Continuous;*
- *Quasi-concave on* $\mathcal{A}^k$.[2]

The conditions in Proposition C.1.1 are sufficient but not necessary. For example, any game in which $\mathcal{A}^k$ has finitely many elements is non-convex and therefore violates the conditions of the theorem. However, there are strategic games (such as the prisoner's dilemma) that have finitely many actions and for which Nash equilibria do exist.

C.1.3 Bayesian games

We now describe a class of games where the players lack some information on the other players. This can occur in many different situations: the players may be unaware of the other players' preference relations or of any other aspect relevant to the decision making process. As such, Bayesian games are also known as *strategic games with imperfect information.*

In a Bayesian game, the *state* of the game is the complete set of relevant features of the game. We denote such state by X and admit X to take values in a finite set $\mathcal{X}$ of possible states. Obviously, the state of a Bayesian game is determined upon the start of the game and each player $k \in N$ holds a prior belief π^k. The vector π^k is a probability distribution over $\mathcal{X}$ and each $\pi^k(x)$ translates the belief of player k on the state of the game taking the particular value $x \in \mathcal{X}$.

Before choosing its action, each player $k \in N$ receives a *signal* $\tau^k(x)$ conveying the player k some information on the actual state of the game. Each *signal function* τ^k depends deterministically on the state X of the game and takes values in some set T^k of possible signals. The elements t^k of T^k are known as the *possible types* of player k. When a player k receives a particular signal $t^k \in T^k$, it knows the state of the game to lie in the set $(\tau^k)^{-1}(t^k)$ and adequately updates its belief on the state of the game using its prior beliefs π^k. A player $k \in N$ is of *type* t^k if it receives the signal t^k.

All this leads to the following definition.

Bayesian game

A *Bayesian game* is a tuple $\left(N, \mathcal{X}, (\mathcal{A}^k), (\tau^k), (r^k)\right)$, where

[2] A function $f : X \longrightarrow \mathbb{R}$ is *quasi-concave* if, for every $a \in \mathbb{R}$, the set $L_f(a) = \{x \in X \mid f(x) \geq a\}$ is convex. X is some subset of an Euclidean space.

- N is the set of players;
- $\mathcal{X}$ is the set of game states;
- For each player $k \in N$, $\mathcal{A}^k$ represents the set of all possible actions available to player k;
- For each player $k \in N$, τ^k represents the corresponding signal function;
- For each player $k \in N$, r^k is a payoff function over the set $\mathcal{A} \times \mathcal{X}$, translating the preference relation of player k.

It is possible to associate with each Bayesian game an equivalent strategic game with perfect information as follows. The new set of players $\hat{N}$ is defined as the set of all pairs (k, t^k), where $t^k \in T^k$. The set of possible actions for each player (k, t^k) is $\mathcal{A}^k$. Finally, denote by $a(k, t^k)$ the action of player (k, t^k). When the state of the game is x, player k receives the signal $\tau^k(x)$ and its type is, thus, $\tau^k(x)$. We denote by $a^k(x)$ the action of player $(k, \tau^k(x))$. The expected payoff of player k when its type is t^k and it chooses action a^k is thus

$$\hat{r}^k(a^{-k}, a^k) = \sum_{x \in \mathcal{X}} \mathbb{P}\left[x \mid t^k\right] r^k((a^{-k}(x), a^k), x).$$

With this new payoff function, we obtain a strategic game $(\hat{N}, (\mathcal{A}^k), (\hat{r}^k))$, where $\hat{N} = \left\{(k, t^k) \mid k \in N, t^k \in T^k\right\}$.

Making use of the strategic game associated with a Bayesian game, we have the following definition.

Bayesian game Nash Equilibrium

A *Nash equilibrium* for a Bayesian game $\left(N, \mathcal{X}, (\mathcal{A}^k), (\tau^k), (r^k)\right)$ is a Nash equilibrium for the associated strategic game.

C.2 Mixed equilibria

So far, we have considered strategic games as triples $\left(N, (\mathcal{A}^k), (\succsim^k)\right)$, where the preference relation $\succsim^k$ of each player $k \in N$ is defined over the set $\mathcal{A} = \times_{k=1}^{N} \mathcal{A}^k$. However, it is often the case that the players choose their actions in a non-deterministic way, and it is necessary to include in the model some mechanism traducing the player's preferences over possible *lotteries* over the actions in $\mathcal{A}$. In the remainder of this appendix, we consider only strategic games in the form $\left(N, (\mathcal{A}^k), (r^k)\right)$, *i.e.*, strategic games in which the preferences are given by a payoff function.

Strategy

A *strategy* σ^k for player k in a strategic game $\left(N, (\mathcal{A}^k), (r^k)\right)$ is a probability distribution over the set $\mathcal{A}^k$.

A strategy σ^k assigns a probability $\sigma^k(a^k)$ to each action $a^k \in \mathcal{A}^k$. We say that player k follows strategy σ^k when playing the game $\left(N, (\mathcal{A}^k), (r^k)\right)$ if it chooses each action $a^k \in \mathcal{A}^k$ with probability $\sigma^k(a^k)$. If a strategy σ^k assigns probability 1 to some action $a^k \in \mathcal{A}^k$, then σ^k is a *pure strategy*. Otherwise, it is called a *mixed strategy*.

The tuple $\sigma = (\sigma^1, \ldots, \sigma^N)$ is a *joint strategy* or *strategy profile*. A *reduced strategy profile* or *reduced joint strategy* is a tuple

$$\sigma^{-k} = (\sigma^1, \ldots, \sigma^{k-1}, \sigma^{k+1}, \ldots, \sigma^N),$$

and write write

$$\sigma = (\sigma^{-k}, \sigma^k)$$

to indicate that the individual strategy of player k in the joint strategy σ is σ^k. The *support* of a strategy σ^k is the set of all actions $a^k \in \mathcal{A}^k$ such that $\sigma^k(a^k) > 0$.

Consider a strategic game $\left(N, (\mathcal{A}^k), (r^k)\right)$ and suppose that the agents are allowed to choose their actions randomly. Denote by R^k the random payoff received by player k in a play of the game.

Mixed Strategy Nash Equilibrium

A *mixed strategy Nash equilibrium* of a strategic game $\left(N, (\mathcal{A}^k), (r^k)\right)$ is a strategy profile σ^* such that, for every player $k \in N$,

$$\mathbb{E}_{\sigma^*}\left[R^k\right] \geq \mathbb{E}_{((\sigma^*)^{-k},\, \sigma^k)}\left[R^k\right] \tag{C.1}$$

for all strategies σ^k.

The next theorem was established by John F. Nash in 1949 [216].

Theorem C.2.1. *Every strategic game $\left(N, (\mathcal{A}^k), (r^k)\right)$ with finite $\mathcal{A}$ has a mixed strategy Nash equilibrium.*

A strategic game verifying the conditions of the previous theorem is known as a *finite* strategic game. The next result further characterizes mixed strategy Nash equilibria.

Theorem C.2.2. *Given a finite strategic game* $\big(N, (\mathcal{A}^k), (r^k)\big)$*, a strategy* σ^* *is a mixed strategy Nash equilibrium if and only if, for every* $k \in N$*,*

- *Given the reduced strategy* $(\sigma^*)^{-k}$*, the expected payoff of every action* a^k *in the support of* $(\sigma^*)^k$ *is the same;*
- *Given the reduced strategy* $(\sigma^*)^{-k}$*, the expected payoff of every action* a^k *not in the support of* $(\sigma^*)^k$ *is, at most, the expected payoff of any action in the support of* $(\sigma^*)^k$*.*

C.3 Strictly competitive games

As illustrated in the example of matching pennies, there are games in which two players have opposing goals. Such games are known as *strictly competitive* or *zero-sum*, since the payoffs of the players in such games usually add up to 0. This leads to the following general definition.

Strictly Competitive Game

A strategic game $\big(N, (\mathcal{A}^k), (r^k)\big)$ is *strictly competitive* if $N = 2$ and

$$r^1(a^1, a^2) \geq r^1(b^1, b^2) \quad \Rightarrow \quad r^2(a^1, a^2) \leq r^2(b^1, b^2).$$

A player k is said to *maxminimize* if it chooses its actions assuming that the other player will always choose its own action so as to hurt player k as much as possible. This leads to the following definition.

Maxminimizer

Let $\big(\{1,2\}, (\mathcal{A}^k), (r^k)\big)$ be a strictly competitive strategic game. An individual strategy $(\sigma^1)^*$ is a *maxminimizer for player 1* if

$$\min_{\sigma^2} \mathbb{E}_{((\sigma^1)^*,\, \sigma^2)} \left[R^1\right] \geq \min_{\sigma^2} \mathbb{E}_{(\sigma^1,\, \sigma^2)} \left[R^1\right]$$

for any individual strategy σ^1 of player 1. Similarly, an individual strategy $(\sigma^2)^*$ is a *maxminimizer for player 2* if

$$\min_{\sigma^1} \mathbb{E}_{(\sigma^1,\, (\sigma^2)^*)} \left[R^2\right] \geq \min_{\sigma^1} \mathbb{E}_{(\sigma^1, \sigma^2)} \left[R^2\right]$$

for any individual strategy σ^2 of player 2.

It is now possible to introduce the following result.

Theorem C.3.1. *Let* $\left(\{1,2\},(\mathcal{A}^k),(r^k)\right)$ *be a strictly competitive strategic game. Then,*

- *If* $((\sigma^1)^*,(\sigma^1)^*)$ *is a mixed strategy Nash equilibrium,* $(\sigma^1)^*$ *is a maxminimizer for player 1,* $(\sigma^2)^*$ *is a maxminimizer for player 2, and*

$$\max_{\sigma^1}\min_{\sigma^2}\mathbb{E}_{(\sigma^1,\,\sigma^2)}\left[R^1\right]=\min_{\sigma^2}\max_{\sigma^1}\mathbb{E}_{(\sigma^1,\,\sigma^2)}\left[R^1\right]=\mathbb{E}_{((\sigma^1)^*,\,(\sigma^2)^*)}\left[R^1\right];$$

- *If*

$$\max_{\sigma^1}\min_{\sigma^2}\mathbb{E}_{(\sigma^1,\,\sigma^2)}\left[R^1\right]=\min_{\sigma^2}\max_{\sigma^1}\mathbb{E}_{(\sigma^1,\,\sigma^2)}\left[R^1\right],$$

  $(\sigma^1)^*$ is a maxminimizer for player 1 and $(\sigma^2)^*$ is a maxminimizer for player 2, then $((\sigma^1)^*,(\sigma^2)^*)$ is a mixed strategy Nash equilibrium.

This result yields the following corollary.

Corollary C.3.2. *Let* $\left(\{1,2\},(\mathcal{A}^k),(r^k)\right)$ *be a strictly competitive strategic game. If* (σ^1,σ^2) *and* $(\bar{\sigma}^1,\bar{\sigma}^2)$ *are two mixed strategy Nash equilibria, then so are* $(\bar{\sigma}^1,\sigma^2)$ *and* $(\sigma^1,\bar{\sigma}^2)$*. Furthermore, all the previous equilibria yield the same expected payoff for the two players.*

We conclude this section with the observation that, in a strictly competitive game, if player k plays a Nash equilibrium then its expected payoff is at least the equilibrium payoff.

C.4 Fully cooperative games

In the previous section we addressed strictly competitive games, in which two players with opposing interests interact in a common scenario. The converse to this situation would be if the two players had the exact same interests. Such games are known as *fully cooperative*, and we address them in this section.

We start with a formal definition.

Fully Cooperative Game

A strategic game $\big(N, (\mathcal{A}^k), (r^k)\big)$ is *fully cooperative* if

$$r^k(a) \geq r^k(b) \quad \Rightarrow \quad r^l(a) \geq r^l(b) \tag{C.2}$$

for all $k, l \in N$ and all $a, b \in \mathcal{A}$.

Fully cooperative games are also known as *team games*. Notice that (C.2) implies that all players share the same preference relation $\succsim^k$ on $\mathcal{A}$ and, therefore, we can assume with no loss of generality that

$$r^1 = r^2 = \ldots = r^N = r.$$

Therefore, a fully cooperative game can be represented by a tuple $\big(N, (\mathcal{A}^k), r\big)$ with a unique payoff function r for all players.

In a general strategic game $\big(N, (\mathcal{A}^k), (r^k)\big)$, an action profile $a^* \in \mathcal{A}$ is a *coordinated* or *Pareto-optimal* (Nash) equilibrium if it maximizes the payoff received for every player. This leads to the following definition.

Coordinated Equilibrium

Let $\big(N, (\mathcal{A}^k), (r^k)\big)$ be a strategic game. A strategy profile σ^* is a *coordinated equilibrium* for the game if

$$\mathbb{E}_{\sigma^*}\left[R^k\right] = \max_{a \in \mathcal{A}} r^k(a),$$

for all $k \in N$.

Clearly, not all strategic games have coordinated equilibria. For those that do have strategic equilibria, it is always possible to find at least one pure strategy equilibrium. This is the case of fully cooperative games, as stated in the following result.

Theorem C.4.1. *Every fully cooperative game $\big(N, (\mathcal{A}^k), r\big)$ has at least one coordinated (Nash) equilibrium σ^*.*

This result yields the following corollary.

Corollary C.4.2. *Let $\big(N, (\mathcal{A}^k), r\big)$ be a fully cooperative strategic game. Then, all coordinated equilibria for the game yield the same expected payoff for all the players.*

C.5 Stochastic games

Stochastic games can be seen as multi-state generalizations of strategic games or, on the other hand, as a multi-agent generalizations of Markov decision processes. They were first introduced by Shapley [273] in 1953.

Consider a finite MDP $(\mathcal{X}, \mathcal{A}, \mathsf{P}, r, \gamma)$. At each time instant, the state of the process is a r.v. X_t taking values in $\mathcal{X}$. The transition probabilities depend on the control parameter at time t, represented by A_t, and are given by the transition probability kernel P according to

$$\mathbb{P}\left[X_{t+1} = j \mid X_t = i, A_t = a\right] = \mathsf{P}_a(i, j).$$

In a standard MDP, the control parameter A_t is chosen by a single decision-maker so as to control the time-evolution of the chain $\{X_t\}$. Each transition (i, a, j) grants the decision maker with a reward $r(i, a, j)$.

Suppose now that $\mathcal{A}$ is the cartesian product of N smaller sets $\mathcal{A}^k$, *i.e.*, $\mathcal{A} = \times_{k=1}^{N} \mathcal{A}^k$. The control parameter at each time t is now a N-tuple $A_t = (A_t^1, \ldots, A_t^N)$, with each $A_t^k \in \mathcal{A}^k$. Suppose also that r is a function defined in $\mathcal{X} \times \mathcal{A} \times \mathcal{X}$ but taking values in $\mathbb{R}^N$. It is therefore possible to write

$$r(i, a, j) = \big(r^1(i, a, j), \ldots, r^N(i, a, j)\big),$$

each r^k defined on $\mathcal{X} \times \mathcal{A} \times \mathcal{X}$ and taking values in $\mathbb{R}$.

A stochastic game is a generalized MDP $(\mathcal{X}, \mathcal{A}, \mathsf{P}, r, \gamma)$ in which the set $\mathcal{A}$ and the reward function r are as described above. There are N independent decision-makers (the players), each choosing the value of one individual control parameter A_t^k. Each transition (i, a, j) grants player k with a reward $r^k(i, a, j)$. This leads to the following definition.

Stochastic Game

A *stochastic game* or *Markov game* is a tuple $\big(N, \mathcal{X}, (\mathcal{A}^k), \mathsf{P}, (r^k), \gamma\big)$, where

- N is the set of players in the game;
- $\mathcal{X}$ is the set of game-states;
- $\mathcal{A} = \times_{k=1}^{N} \mathcal{A}^k$ is the cartesian product of the individual action sets $\mathcal{A}^k$;

- P represents the transition probability kernels. In the simplest case where $\mathcal{X}$ is finite, each P_a is a matrix with xy^{th} entry given by

$$\mathsf{P}_a(i,j) = \mathbb{P}\left[X_{t+1} = j \mid X_t = i, A_t = a\right];$$

- $r = (r^1, \ldots, r^N)$ is the reward function, assigning a reward $r^k(x,a,y)$ to player k each time a transition from x to y occurs "under" the joint action a.

In a stochastic game, each player seeks to maximize its total expected payoff, defined as in Markov decision processes by

$$V^k(\{A_t\}, i) = \mathbb{E}\left[\sum_{t=0}^{\infty} \gamma^t R^k(X_t, A_t) \mid X_0 = i\right]$$

with $i \in \mathcal{X}$ and $R^k(i,a)$ the random reward received by player k for taking action a in state i.

An individual strategy for player k can now be seen as a sequence $\left\{\sigma_t^k\right\}$. Each σ_t^k is a mapping $\sigma_t^k : \mathcal{X} \times \mathcal{A} \longrightarrow [0,1]$ and the control sequence $\left\{A_t^k\right\}$ generated by $\left\{\sigma_t^k\right\}$ verifies

$$\mathbb{P}\left[A_t^k = a^k \mid X_t = i\right] = \sigma_t^k(i,a), \tag{C.3}$$

for all t. A strategy profile $\{\sigma_t\} = \times_{k=1}^N \left\{\sigma_t^k\right\}$ generates a control sequence $\{A_t\}$, where each element A_t is a N-tuple $(A_t^1, \ldots, A_t^N)$, each A_t^k verifying (C.3) for all $t \in \mathcal{T}$ and all $k \in N$. To simplify the notation, we denote a strategy profile $\{\sigma_t\}$ simply by σ_t. We write $(V^{\sigma_t})^k(i)$ instead of $V^k(\{A_t\}, i)$ whenever the control sequence $\{A_t\}$ is generated by the strategy profile σ_t and refer to V^{σ_t} as being the *value function* associated with strategy σ_t. A strategy σ_t is *stationary* if

$$\sigma_1 = \ldots = \sigma_t = \ldots,$$

in which case we represent it simply by $\sigma = \sigma_1$.

Stochastic Game Nash Equilibrium

A Nash equilibrium for a stochastic game $\left(N, \mathcal{X}, (\mathcal{A}^k), \mathsf{P}, (r^k), \gamma\right)$ is a strategy profile σ_t^* such that, for every player $k \in N$,

$$(V^{\sigma_t^*})^k(i) \geq (V^{((\sigma_t^*)^{-k}, \sigma_t^k)})^k(i),$$

for all strategies σ_t^k and all states $i \in \mathcal{X}$.

The following result was established by Fink [85].

Theorem C.5.1. *Every finite stochastic game* $\big(N, \mathcal{X}, (\mathcal{A}^k), \mathsf{P}, (r^k), \gamma\big)$ *has a stationary Nash equilibrium.*

We conclude this section by defining a Q-function in the context of stochastic games. Given a stochastic game $\big(N, \mathcal{X}, (\mathcal{A}^k), \mathsf{P}, (r^k), \gamma\big)$ the Q-function associated with a strategy σ_t is defined for player k as

$$(Q^{\sigma_t})^k(i,a) = \sum_{j \in \mathcal{X}} \mathsf{P}_a(i,j)\big[r^k(i,a,j) + \gamma (V^{\sigma_t})^k(j)\big],$$

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

C.6 Fictitious play

Fictitious play is an iterative procedure originally proposed by Brown [49] to determine the solution for a strictly competitive game. This procedure was shown to converge in this class of games by Robinson [254] and later extended to other particular classes of games by several authors (see, for example, [22, 125, 147, 202]).

In its original formulation, two players repeatedly engaged in a strictly competitive game $\big(\{1,2\}, (\mathcal{A}^k), (r^k)\big)$. Each player maintains an estimate of the other player's strategy as follows: let $n_t(a)$ denote the number of times that the individual action a was played up to (and including) the t^{th} play. At play t, each player k estimates the other player's strategy to be

$$\hat{\sigma}_t^{-k}(a) = \frac{n_t(a)}{t}, \tag{C.4}$$

for each $a \in \mathcal{A}^{-k}$. Player k then estimates the expected payoff of each of its individual actions according to

$$EP(a^k) = \sum_{a^{-k} \in \mathcal{A}^{-k}} r^k(a^{-k}, a^k) \hat{\sigma}_t^{-k}(a^{-k})$$

and randomly chooses its action from the set of best responses,

$$BR_t = \left\{ a^k \in \mathcal{A}^k \mid a^k = \arg\max_{b^k \in \mathcal{A}^k} EP(b^k) \right\}.$$

Robinson [254] showed that this methodology yields two sequences $\{\hat{\sigma}_t^1\}$ and $\{\hat{\sigma}_t^2\}$ converging respectively to σ^1 and σ^2 such that $\sigma = (\sigma^1, \sigma^2)$ is a Nash equilibrium for the game $\big(\{1,2\}, (\mathcal{A}^k), (r^k)\big)$. The sequences $\{\hat{\sigma}_t^1\}$ and $\{\hat{\sigma}_t^2\}$ define a *fictitious play processes.*

In a general strategic game $\big(N, (\mathcal{A}^k), (r^k)\big)$, the players repeatedly engage in the game and each player $k \in N$ maintains at each play an estimate $\hat{\sigma}_t^{-k}$ of the reduced strategy σ^{-k} used by the remaining players. This estimate is determined similarly to

the one in (C.4) and is used to determine the strategy to follow: the agent uses the current estimate to determine the expected payoff of each of its individual actions and randomly chooses its action from the corresponding best response set.

It is not possible to ensure that fictitious play converges in all games. However, for particular classes of games (see the references at the beginning of this section), it is possible to establish the convergence of fictitious play, and this methodology can be used by a set of agents to learn and converge in behavior to a Nash equilibrium.

Fictitious Play Property

We say that a game has the *fictitious play property* if every fictitious play process converges in beliefs to an equilibrium.

The following two results are particularly useful for the work in the thesis [59, 323].

Theorem C.6.1. *Every strictly competitive game has the fictitious play property.*

Theorem C.6.2. *Every fully cooperative game has the fictitious play property.*

C.7 Adaptive play

Adaptive play was first proposed by Young [345] as an alternative method to fictitious play. The basic underlying idea is similar to fictitious play, but the actual method works differently from fictitious play.

Young [345] was able to establish that for games that are *weakly acyclic*, adaptive play converges w.p.1 to a pure strategy Nash equilibrium. Since the two methods described in this section (adaptive play and biased adaptive play) are central in the algorithms presented in Chapters 7 and 8, we provide a detailed description of both methods and their convergence properties.

C.7.1 Adaptive play for repeated games

Before describing the process of adaptive play, we need the concept of K-sample.

Let K and m be two integers such that $1 \leq K \leq m$ and let h be a vector of length m. We refer to any set of K samples randomly drawn from h without replacement as a *K-sample* and denote it generically by $K(h)$.

Let $\Gamma = \big(N, (\mathcal{A}^k), (r^k)\big)$ be a strategic game. Let t be some time index taking values in an index set $\mathcal{T}$ and suppose that, for every $t \in \mathcal{T}$, the players repeatedly engage in the game Γ. Each player $k \in N$ chooses an action $a_t^k \in \mathcal{A}^i$ as described below, and the action profile $a_t = (a_t^1, \ldots, a_t^N)$ is referred to as the *play at time* t. The history of plays up to time t is a set $\mathcal{H}_t = \{a_1, \ldots, a_t\}$.

At each time instant $t \in \mathcal{T}$, each player $k \in N$ chooses its actions a_t^k as follows. Let K and m be two integers such that $1 \leq K \leq m$. For $t \leq m$, each player chooses its actions randomly; for $t \geq m+1$, each player k inspects K plays drawn without replacement from the most recent m plays. We denote by H_t the m most recent plays at time t. Let $n_K(a^{-k})$ be the number of times that the reduced action a^{-k} appears in the K-sample $K(H_t)$. Player k then uses the K-sample $K(H_t)$ to choose its best response strategy by calculating its expected payoff w.r.t. the estimated strategy played by its opponents: for each individual action $a^k \in \mathcal{A}^k$, player k determines the corresponding expected payoff $EP(a^k)$ as

$$EP(a^k) = \sum_{a^{-k} \in \mathcal{A}^{-k}} r^k(a^{-k}, a^k) \frac{n_K(a^{-k})}{K}$$

It then randomly chooses its action from the set of best responses,

$$BR_t = \left\{ a^k \in \mathcal{A}^k \mid a^k = \arg\max_{b^k \in \mathcal{A}^k} EP(b^k) \right\}.$$

Notice that this procedure is similar to fictitious play in that it chooses the best response action w.r.t. the estimated reduced strategy $\hat{\sigma}^{-k}$. The only difference lies in the fact that adaptive play uses *incomplete history sampling*, while fictitious play uses the complete history to estimate σ^{-k}.

Notice that the sample history of m plays is a Markov chain $\{H_t\}$. Given a history vector h with length m and an action $a^k \in \mathcal{A}^k$, we define $p^k(i) = \mathbb{P}\left[a^k \mid h\right]$ as the probability of player k choosing action a^k given h. This probability is non-zero if and only if there is a K-sample $K(h)$ such that a^k is the best response to the strategy $\hat{\sigma}^{-k}$ estimated from $K(h)$. Given any two history vectors h and h', the transition probability for the chain $\{H_t\}$ is given by

$$\mathbb{P}\left[H_{t+1} = h' \mid H_t = h\right] = \prod_{k=1}^{N} p^k(a^k \mid h),$$

where a^k is k^{th} component of the last action in h'.

As shown in [345], a history h is an absorbing state of the chain $\{H_t\}$ if and only if it consists of m successive plays of a strict Nash equilibrium a^*. Young [345] also provides the conditions under which adaptive play converges to one such absorbing state.

To formally present such result, we need the following definitions.

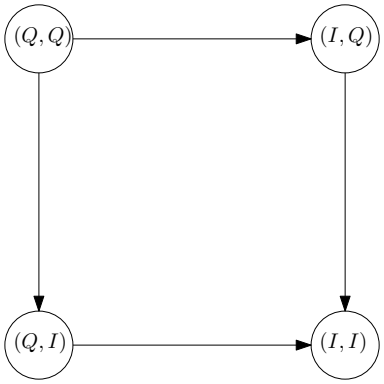


Figure C.1: Best response graph for the prisoner's dilemma.

Best Response Graph

Let $\Gamma = \big(N, (\mathcal{A}^k), (r^k)\big)$ be a strategic game with finite action-space $\mathcal{A} = \times_{k=1}^{N} \mathcal{A}^k$. The *best response graph* for game Γ is a directed graph $\mathcal{G} = (V, E)$, where $V = \mathcal{A}$ and, given any two vertices $a, b \in V$, $(a, b) \in E$ if and only if $a \neq b$ and there is exactly one $k \in N$ such that $b^k \in B(a^{-k})$ and $a^{-k} = b^{-k}$.

A best response graph is build by considering all joint actions in $\mathcal{A}$ as vertices and setting a directed edge from a joint action a to a joint action b if the two actions are composed of the same individual actions for all players except one. For that single player k, b^k is a best response to the reduced action a^{-k}. The following example illustrates this concept for the prisoner's dilemma.

Example C.2. (cont.) Consider once again the prisoner's dilemma, whose payoff function is represented in Table C.3. The best response graph for this example is depicted in Figure C.1

There are four vertices, (Q, Q), (Q, I), (I, Q) and (I, I), each corresponding to one of the actions in $\mathcal{A}$: (*Quiet*, *Quiet*), (*Quiet*, *Inform*), (*Inform*, *Quiet*) and (*Inform*, *Inform*).

The edges are set as follows. If player 1 chooses action *Quiet*, then the best response for player 2 is to play *Inform*. This means that there should be a directed edge from (Q, Q) to (Q, I). A similar reasoning leads to the conclusion that there should also be a directed edge from (Q, Q) to (I, Q). Finally, if any of the players chooses action *Inform*, the other player's best response is to also play *Inform*. This justifies the two remaining edges in the graph. ◇

Table C.4: Fully cooperative game with multiple equilibria.

	a	b
a	**10**	-20
b	-20	5

Weakly Acyclic Game

A strategic game $\Gamma = \big(N, (\mathcal{A}^k), (r^k)\big)$ is *weakly acyclic* if, given any vertex a in its best response graph, there is a directed path to a vertex a^* from which there is no exiting edge (a sink).

It should be clear that a sink as described in the previous definition corresponds necessarily to a strict Nash equilibrium. Consider, for example, the best response graph in Figure C.1. The graph is weakly acyclic, as can easily be checked from the definition. On the other hand, the only sink is the vertex (I, I). This vertex corresponds to the action (*Inform*, *Inform*) which, as seen before, is the only Nash equilibrium for the game of prisoner's dilemma.

Given a weakly acyclic strategic game $\Gamma = \big(N, (\mathcal{A}^k), (r^k)\big)$, we denote by $L(a)$ the shortest path from the vertex a to a strict Nash equilibrium in the best response graph of Γ. Let $L(\Gamma) = \max_{a \in \mathcal{A}} L(a)$.

Theorem C.7.1. *Let $\Gamma = \big(N, (\mathcal{A}^k), (r^k)\big)$ be a weakly acyclic strategic game. If*

$$K \leq \frac{m}{L(\Gamma) + 2},$$

then adaptive play converges w.p.1 to a strict Nash equilibrium.

C.7.2 Biased adaptive play

Biased adaptive play is a variant of adaptive play proposed by Wang and Sandholm [330]. This method is designed to address problems of equilibrium selection in fully cooperative games. In this class of games, fictitious play is known to converge to a Nash equilibrium. However, if there are multiple equilibria with different values, there is no guarantees that the fictitious play process converges to the coordinated equilibrium.

Consider, for example, the fully cooperative game in Table C.4. The boldface entry represents the only coordinated equilibrium, (a, a). However, the action (b, b)

is also a Nash equilibrium, and there are no guarantees that either fictitious play or adaptive play converges to the coordinated equilibrium.

To describe how biased adaptive play works, we start with the following definition. Let $\Gamma = \big(N, (\mathcal{A}^k), r\big)$ be a fully cooperative strategic game and let D be a set containing some of the Nash equilibria in Γ (and no other joint actions).

Weakly Acyclic Game w.r.t a Bias Set

Γ is a weakly acyclic game w.r.t. the bias set D if, given any vertex a in the best response graph of Γ, there is a directed path to either a Nash equilibrium in D or a strict Nash equilibrium.

Given a fully cooperative strategic game $\Gamma = \big(N, (\mathcal{A}^k), r\big)$, we construct a virtual game $VG = \big(N, (\mathcal{A}^k), r_{VG}\big)$, where $r_{VG}(a) = 1$ if a is a coordinated equilibrium for Γ and $r_{VG}(a) = 0$ otherwise. By setting $D = \{a \in \mathcal{A} \mid r_{VG}(a) = 1\}$, the game VG is weakly acyclic w.r.t. the set D. Furthermore, as argued in [330], by learning how to play a Nash equilibrium in the virtual game VG, the players learn how to play a coordinated equilibrium in the original game Γ.

Given the virtual game VG, the players proceed as in adaptive play: given a K-sample $K(H_t)$ obtained from the history of the m most recent plays, each player k checks if

1. There is a joint action $a^* \in D$ such that, for all the actions $a \in K(H_t)$, $a^{-k} = (a^*)^{-k}$;
2. There is at least one action $a^* \in D$ such that $a^* \in K(H_t)$.

If these two conditions are verified, player k is "lead to believe" that the remaining players have coordinated in an action a^* in D. Therefore, if conditions 1 and 2 are met, player k chooses its best response $(a^*)^k$ such that

$$a^* = \max_{t \in \mathcal{T}} \{a_t \mid a_t \in K(H_t) \text{ and } a_t \in D\}.$$

If either 1 or 2 or both do not hold, then player k chooses its action as in adaptive play.

We now have the following theorem, found in [330].

Theorem C.7.2. *Let $\Gamma = \big(N, (\mathcal{A}^k), r\big)$ be a fully cooperative, weakly acyclic game w.r.t. some bias set D. If*

$$K \leq \frac{m}{L(\Gamma) + 2},$$

then biased adaptive play converges w.p.1 to either a strict Nash equilibrium or a Nash equilibrium in D.

Once again, the constant $L(\Gamma)$ is defined as $L(\Gamma) = \max_{a \in \mathcal{A}} L(a)$, where $L(a)$ is now the shortest path in the best response graph of Γ going from vertex a to either a strict Nash equilibrium or a Nash equilibrium in D.

We conclude this appendix with two observations. The first observation concerns the use of a virtual game to ensure that the biased adaptive play converges to a coordinated equilibrium. Notice that this technique can easily be used with fictitious play, guaranteeing convergence of this method to a coordinated equilibrium. However, computationally speaking, biased adaptive play presents similar convergence guarantees while being computationally more efficient, because of the incomplete history sampling.

The second observation is related with the application of fictitious play, adaptive play and biased adaptive play to fully cooperative stochastic games. Although these methods were presented for a repeated game framework, it is possible to apply them *mutatis mutandis* to the stochastic game framework. In fact, the Q-values for a coordinated equilibrium define for each state $x \in \mathcal{X}$ a fully cooperative strategic game $(N, (\mathcal{A}^k), Q)$ to which the three referred methods can be applied, as long as every state x is visited infinitely often [330].

Appendix D

Stochastic Approximation

In this appendix, we present some background on stochastic approximation.

We present a convergence result for general stochastic approximation algorithms and discriminate the conditions necessary to ensure convergence w.p.1. We provide an interpretation of stochastic approximation algorithms in terms of solutions of ordinary differential equations, and describe the corresponding limit points in terms of equilibria of such ODEs. We also provide several simpler convergence results of great use on several proofs of convergence along the thesis. We conclude with a law of iterated logarithm describing the rate of convergence of such methods.

D.1 Convergence of stochastic approximation algorithms

Stochastic approximation algorithms are known since the 19^{th} century, but a formal treatment of convergence was not completed until the pioneer works of Robbins and Monro [253] and Kiefer and Wolfowitz [140]. Since then, these algorithms have been fundamental in numerous applications such as stochastic optimization, system identification, signal modeling, adaptive filtering or machine learning, and have inspired numerous works, such as the books by Ljung and Söderström [172], Benveniste et al. [21] or Kushner and Yin [153].

The original stochastic approximation algorithms address the problem of determining the zero of a real-valued function h when the function is not known but noise-corrupted samples of h are available at any desired point. The algorithm takes

the form

$$\theta_{t+1} = \theta_t + \alpha_t H(\theta_t, X), \tag{D.1}$$

where θ_t is the current estimate of the point of interest, $\{\alpha_t\}$ is a sequence of positive step-sizes and $H(\theta_t, X)$ is a noisy sample of $h(\theta_t)$ that depends on the r.v. X.

In the 1970s, Ljung introduced the ODE method for the analysis of stochastic approximation algorithms [171]. This method studies the convergence of the sequence $\{\theta_t\}$ generated by algorithm (D.1) by comparing it with the trajectory of an associated ODE

$$\dot{\theta} = h(\theta).$$

This method allows the application of results from dynamical systems theory (such as the Lyapunov theorem) to the study of convergence of stochastic approximation algorithms.

The ODE method proved to be a powerful tool in the analysis of stochastic approximation algorithms. Some illustrative applications of this method can be found in Abounadi et al. [5], Borkar [31], Borkar and Meyn [34], Kushner and Lakshmivarahan [152]. Kushner and Clark [151] provided a rather general set of conditions for convergence, known as the *Kushner-Clark conditions*, afterwards explored in several other works [148, 153, 198].

D.1.1 A general convergence result

In this subsection, we present a general convergence theorem used in establishing the convergence of several algorithms introduced in the thesis.

Consider the generic stochastic approximation algorithm

$$\theta_{t+1} = \theta_t + \alpha_{t+1} H(\theta_t, X_{t+1}) + \alpha_{t+1}^2 C_{t+1}(\theta_t, X_{t+1}), \tag{D.2}$$

where $\{X_t\}$ is a controlled Markov chain taking values in a compact subset $\mathcal{X} \subset \mathbb{R}^p$. The corresponding transition probabilities are given by

$$\mathbb{P}\left[X_{t+1} \in U \mid \mathcal{F}_n\right] = \mathsf{P}_{\theta_t}(X_n, U),$$

for all $U \in \mathcal{B}(\mathcal{X})$. The set $\{\alpha_t\}$ is a decreasing sequence of positive step-sizes verifying

$$\sum_t \alpha_t = \infty \qquad \text{and} \qquad \sum_t \alpha_t^{1+\lambda} < \infty,$$

for some $\lambda \in [0.5; 1]$. Let $h : \mathbb{R}^M \longrightarrow \mathbb{R}^M$ be defined from H as

$$h(\theta) = \mathbb{E}_X\left[H(\theta, X_t)\right], \tag{D.3}$$

where $\mathbb{E}_X[\cdot]$ represents the expectations with respect to the steady-state probability measure of the chain $\{X_t\}$ (see Appendix B).

For the purposes of the convergence theorem below, we admit the following

assumptions to hold for some norm $\|\cdot\|$:

$$\begin{aligned}
\|H(\theta,x)\| &\leq K_1(1+\|\theta\|)(1+\|x\|^{q_1}); && \text{(Ass. 1.a)}\\
\mathbb{E}\left[1+\|X_{t+1}\| \mid X_0=x,\theta_0\right] &\leq K_2(1+\|x\|^{q_2}); && \text{(Ass. 1.b)}\\
\|\nu_\theta(x)\| &\leq K_3(1+\|\theta\|)(1+\|x\|^{q_3}); && \text{(Ass. 1.c)}\\
\|C_t(\theta,x)\| &\leq K_4(1+\|\theta\|)(1+\|x\|^{q_4}), && \text{(Ass. 1.d)}
\end{aligned}$$

where K_1, K_2, K_3 and K_4 are positive constants and q_1, q_2, q_3 and q_4 are positive integers. In (Ass. 1.c), ν_θ is the solution (for fixed θ) of the Poisson equation

$$(\mathbf{I}-\mathsf{P}_\theta)\nu_\theta(x) = H(\theta,x) - (\mu_\theta H)(\theta),$$

and μ_θ is the invariant probability measure for the chain $(\mathcal{X},\mathsf{P}_\theta)$.

We further admit that

$$\|\mathsf{P}_\theta\nu_\theta(x) - \mathsf{P}_{\theta'}\nu_{\theta'}(x)\| \leq K_5(1+R^{1-\lambda})\,\|\theta-\theta'\|^\lambda\,(1+\|x\|^{q_5}), \qquad \text{(Ass. 2)}$$

where $\|\theta\| \leq R$, $\|\theta'\| \leq R$, K_5 is a positive constant and q_5 is a positive integer.

The following generic convergence theorem can be found in the book by Benveniste et al. [21], Theorem 17.

Theorem D.1.1. *Suppose that* (Ass. 1) *and* (Ass. 2) *hold. Then,*

1. *if there exists a positive function $\mathcal{U} : \mathbb{R}^M \longrightarrow \mathbb{R}$ with bounded continuous second derivative such that, for all $\theta : \|\theta\| \geq K_0$,*
   (a) *$d\mathcal{U}(\theta)/dt = \langle d\mathcal{U}(\theta)/d\theta, h(\theta)\rangle \leq 0$;*
   (b) *there is $\beta > 0$ such that $\mathcal{U}(\theta) \geq \beta\,\|\theta\|^2$;*

   then the sequence $\{\theta_t\}$ is bounded w.p.1;
2. *if, furthermore, there exists a vector $\theta^* \in \mathbb{R}^M$ such that*
   (a) *$d\mathcal{U}(\theta)/dt = \langle d\mathcal{U}(\theta)/d\theta, h(\theta)\rangle < 0$ for all $\theta \neq \theta^*$;*
   (b) *$\mathcal{U}(\theta) = 0$ iff $\theta = \theta^*$;*

   then the sequence $\{\theta_t\}$ converges to θ^* w.p.1.

Since Theorem D.1.1 plays a fundamental role in establishing the convergence of the several algorithms in this thesis, we briefly analyze its assumptions and assertions so as to gain some insight on the ideas behind this result.

We start by rewriting (D.2) as

$$\theta_{t+T} = \theta_t + \sum_{i=t}^{t+T} \alpha_{i+1} H(\theta_i, X_{i+1}). \qquad \text{(D.3)}$$

The summation on the right-hand side can be seen as a sample average of $H(\theta, x)$ over T samples of the process $\{X_t\}$. As such, we could write

$$\theta_{t+T} \approx \theta_t + \sum_{i=t}^{t+T} \alpha_{i+1} \mathbb{E}\left[H(\theta_i, X_{i+1})\right] + \sum_{i=t}^{t+T} \alpha_{i+1} e_i, \tag{D.4}$$

where the e_is are error terms. Suppose, in particular, that the process $\{X_t\}$ quickly becomes stationary. Then, (D.4) further simplifies to

$$\theta_{t+T} \approx \theta_t + \sum_{i=t}^{t+T} \alpha_{i+1} \mathbb{E}\left[H(\theta_i, X)\right]. \tag{D.5}$$

where X is the "stationary version" of $\{X_t\}$. On the other hand, if H is slowly varying in θ, we can write

$$\theta_{t+T} \approx \theta_t + \sum_{i=t}^{t+T} \alpha_{i+1} \mathbb{E}\left[H(\theta_t, X)\right]. \tag{D.6}$$

Let $\tau = \sum_{i=t}^{t+T} \alpha_{i+1}$ and $h(\theta) = \mathbb{E}\left[H(\theta_t, X)\right]$. Then, (D.6) becomes

$$\theta_{t+T} \approx \theta_t + \tau h(\theta_t). \tag{D.7}$$

Equation (D.7) is the Euler forward approximation of the ODE

$$\dot{\theta}_t = h(\theta_t)$$

and we would expect that, under suitable conditions, the trajectories of this ODE are close to those of (D.2).

The several assumptions in Theorem D.1.1 ensure that the several approximations in (D.4), (D.5), (D.6) and (D.7) are "valid". If such assumptions are fulfilled, the algorithm closely follows the associated ODE. Therefore, if this ODE asymptotically converges to some equilibrium point, so does the algorithm. The function $\mathcal{U}$ referred in the Theorem is simply a Lyapunov function ensuring the existence of an asymptotically stable equilibrium point for the ODE.

From all stated so far, we can coarsely interpret the several assumptions of Theorem D.1.1 as follows.

- The assumptions on the step-sizes ensure that the sum in (D.3) has an averaging effect, in part allowing the approximation in (D.4);
- The bounds on $\|H\|$ and $\mathbb{E}\left[\|X_t\|\right]$, the existence of ν_θ and corresponding bounde on $\|\nu_\theta\|$ ensure that the error resulting from replacing $H(\theta, X_t)$ by its mean field $h(\theta)$ (approximation (D.5)) are not too significant;[1]
- Finally, the bounds on $\|H\|$ and assumption (Ass. 2) ensure that both H and

[1]The solution $\nu_\theta(x)$ of the Poisson equation can be interpreted as the total error obtained by replacing $H(\theta, X_t)$ by $h(\theta)$ along a trajectory $\{x_t\}$ of the chain verifying $x_0 = x$.

h vary slowly with θ, ensuring that the error in the approximation (D.6) is also not significant.

A more detailed argument on the approximations involved in the ODE method can be found in the fundamental work by Ljung [171] and on the more general survey by Bharath and Borkar [29].

D.1.2 Simpler convergence results

We now present several simpler convergence results that can be established using Theorem D.1.1. These results and corresponding proofs can can be found in [131, 280, 300].

Lemma D.1.2. *Let X be a finite set and define the iterative process*

$$\Delta_{t+1}(x) = (1 - g_t(x))\Delta_t(x) + f_t(x)F_t(x)$$

for $x \in X$. Then $\Delta_t \to 0$ as long as

- $\sum_t f_t(x) = \infty$ *and* $\sum_t f_t^2(x) < \infty$*;*
- $\sum_t g_t(x) = \infty$ *and* $\sum_t g_t^2(x) < \infty$*;*
- $\mathbb{E}\left[f_t(x) \mid \mathcal{F}_t\right] \leq \mathbb{E}\left[g_t(x) \mid \mathcal{F}_t\right]$ *uniformly w.p.1;*
- $\left\|\mathbb{E}\left[F_t(x) \mid \mathcal{F}_t\right]\right\|_W \leq \gamma \left\|\Delta_t\right\|_W$*, with* $0 < \gamma < 1$*;*
- $\mathbf{var}\left[F_t(x) \mid \mathcal{F}_t\right] \leq C(1 + \left\|\Delta_t\right\|_W)^2$*, for some constant* $C > 0$*.*

We denoted by $\|\cdot\|_W$ the weighted maximum norm w.r.t. some vector W. The following lemma is very similar to Lemma D.1.2, allowing however the analysis of a somewhat different class of stochastic algorithms.

Lemma D.1.3. *Let X be a finite set and define the iterative process*

$$\Delta_{t+1}(x) = (1 - f_t(x))\Delta_t(x) + f_t(x)F_t(x)$$

for $x \in X$. Then $\Delta_t \to 0$ as long as

- $0 \leq f_t(x) \leq 1$*,* $\sum_t f_t(x) = \infty$ *and* $\sum_t f_t^2(x) < \infty$*;*
- $\left\|\mathbb{E}\left[F_t(x) \mid \mathcal{F}_t\right]\right\|_W \leq \gamma \left\|\Delta_t\right\|_W + c_t$*, with* $0 \leq \gamma < 1$ *and* $c_t \to 0$ *w.p.1;*
- $\mathbf{var}\left[F_t(x) \mid \mathcal{F}_t\right] \leq C(1 + \left\|\Delta_t\right\|_W)^2$*, for some constant* $C > 0$*.*

These simple results allow for clear and straightforward proofs of convergence for several of the algorithms described in this thesis.

The following result is more elaborate and therefore more broadly applicable. It was first introduced by Szepesvári and Littman in the context of generalized Markov decision processes and used to establish the convergence of several classes of reinforcement learning methods [166, 168, 252, 299, 300, 302].

Let B be a normed vector space and T an operator on B. Let $\mathbf{T} = \{T_0, T_1, \ldots\}$ be a sequence of random operators, $T_t : B \times B \longrightarrow B$. Given a subset $U \subset B$, let $\mathbf{F}_0 : U \longrightarrow 2^B$ be a mapping assigning to each element $u \in U$ a set $\mathbf{F}_0(u)$ of possible *initial conditions*.

We are interested in analyzing the convergence of the iteration $u_{t+1} = T_t(u_t, u_t)$ to a fixed point f^* of T. The following definition formalizes the idea that the sequence $\mathbf{T}$ approximates the operator T.

The sequence $\mathbf{T}$ *approximates* T *on the set* U *and with initial values from* $\mathbf{F}_0(u)$ if, for all $u \in U$ and all initial conditions $u_0 \in \mathbf{F}_0(u)$, the iteration $v_{t+1} = T_t(v_t, u)$ converges to $T(u)$ w.p.1. Furthermore, if $U = \{u\}$, we say that $\mathbf{T}$ *approximates* T *on* u*, with initial values from* $\mathbf{F}_0(u)$.

The following definition will also be useful in the theorem below.

A set $U \subset B$ is *invariant under an operator* $T_t : B \times B \longrightarrow B$ if, for all $u, v \in U$, $T_t(u, v) \in U$. If $\mathbf{T}$ is a sequence $\{T_0, T_1, \ldots\}$ of operators, we say that U is *invariant under* $\mathbf{T}$ if it is invariant under each $T_t \in \mathbf{T}$.

We are now in position to present the following theorem.

Theorem D.1.4. *Let X be an arbitrary set and let B be the space of all bounded functions on X. Let $T : B \longrightarrow B$ be an operator with fixed-point u^* and let $\mathbf{T} = \{T_0, T_1, \ldots\}$ be a sequence of operators approximating T at u^* for initial values from $\mathbf{F}_0(u^*)$. For some $u_0 \in \mathbf{F}_0(u^*)$, define the sequence $\{u_t\}$ recursively as $u_{t+1} = T_t(u_t, u_t)$.*

Then, the sequence $\{u_t\}$ converges to u^* w.p.1 as long as

- *There exist random functions $0 \leq f_t(x) \leq 1$ and $0 \leq g_t(x) \leq 1$ such that*

$$f_t(x) \leq \gamma(1 - g_t(x)),$$

  for some $\gamma < 1$;

- *For all $v_1, v_2 \in B$,*

$$|T_t(v_1, u^*)(x) - T_t(v_2, u^*)(x)| \leq g_t(x)\,|v_1(x) - v_2(x)|\,;$$

- *For all* $v, v_1 \in B$,

$$|T_t(v, u^*)(x) - T_t(v, v_1)(x)| \leq f_t(x)(\|u^*(x) - v_1(x)\| + c_t),$$

where $c_t \to 0$ *w.p.1 as* $t \to 0$*;*

- *For all* $k > 0$,

$$\prod_{t=k}^{n} g_t(x) \to 0$$

as $n \to \infty$ *uniformly in* x.

D.2 Asymptotic behavior

The following result, from [234], describes a law of iterated logarithm for general stochastic algorithms. Several works in the literature address the rate of convergence of specific reinforcement learning methods, and can be seen as particular applications of the more general result presented here. We refer, in particular, the works by Szepesvári [298], Kearns and Singh [137] and Even-Dar and Mansour [82].

Rewrite equation D.2 as

$$\theta_{t+1} = \theta_t + \alpha_{t+1}\big[h(\theta_t) + H(\theta_t, X_{t+1}) - h(\theta_t)\big] + \alpha_{t+1}^2 C_{t+1}(\theta_t, X_{t+1}), \tag{D.8}$$

and write

$$\eta_{t+1} = H(\theta_t, X_{t+1}) - h(\theta_t)$$

and

$$c_{t+1} = \alpha_{t+1} C_{t+1}(\theta_t, X_{t+1}).$$

We denoted by h the $\mathbb{R}^M$-valued function defined in (D.3). Using this new notation, (D.8) becomes

$$\theta_{t+1} = \theta_t + \alpha_{t+1}\big[h(\theta_t) + \eta_{t+1}\big] + \alpha_{t+1} c_{t+1}.$$

Let θ^* be the equilibrium point of the ODE

$$\dot{\theta}_t = h(\theta_t).$$

We suppose the following assumptions to hold.

(Ass. 1) There is a neighborhood $\mathcal{V}$ of θ^* such that, for all $\theta \in \mathcal{V}$,

$$h(\theta) = \mathbf{M}(\theta - \theta^*) + \mathrm{O}(\|\theta - \theta^*\|^a),$$

for some stable matrix $\mathbf{M}$ and some constant $a > 1$. We denote by $-\lambda$ the largest real part of the eigenvalues of $\mathbf{M}$.

(Ass. 2) The gain sequence verifies $\alpha_t = \alpha_0/t$ for some $\alpha_0 > 1/2\lambda$.[2]

(Ass. 3) There is a positive constant K such that, for $\|\theta_t - \theta^*\| \leq K$,

$$\mathbb{E}\left[c_{t+1} \mid \mathcal{F}_t\right] = 0$$

and

$$\sup_{t \geq 0} \mathbb{E}\left[\|c_{t+1}\|^b \mid \mathcal{F}_t\right] < \infty,$$

for some $b > 2$.

The following result follows.

Theorem D.2.1. *Suppose that (Ass. 1) through (Ass. 3) hold. Then, if $\theta_t \to \theta^*$ w.p.1,*

$$\limsup_{t \to \infty} \frac{\|\theta_t - \theta^*\|}{\sqrt{\alpha_t \log\left(\sum_{\tau=1}^{t} \alpha_\tau\right)}} \leq K_0, \tag{D.9}$$

w.p.1, where K_0 is some constant.

[2] In the paper [234], Pelletier actually considers more general step-size sequences. We present only this simplified version of the result in [234], since it is sufficient for our purposes.

Appendix E

Q-learning using sample-based approximation

In this appendix, we describe a linear approximation mechanism to use with Q-learning that is closely related with sample-based strategies such as interpolation-based Q-learning [302].

We describe a variation of Q-learning with linear function approximation different from the one in Chapter 4 and derive a set of conditions that imply the convergence of this method w.p.1, when a fixed learning policy is used. We show that, under these conditions, the algorithm exhibits approximation errors that verify similar bounds to those in [321]. We discuss the relation between the results herein and those obtained in several related works in the literature, such as interpolated Q-learning or Borkar's functional Q-learning. The approximation mechanism described is passible of interpretation according to two different perspectives: as a sample-based approximation mechanism using convex interpolation or as a linear approximation mechanism relying on a set of basis functions.

E.1 Sample-based approximation

In this appendix we address the problem of function approximation in Q-learning from a different perspective: instead of deriving conditions that ensure the orthogonal projection to be non-expansive in the sup-norm, we propose a *sample-based projection* that is naturally non-expansive in the sup-norm.

The use of sample-based projection operators was thoroughly used in the algorithm proposed by Szepesvári and Smart [302] and dubbed as interpolation-based Q-learning. This method makes use of a set of sample points and defines a projection operator that simply considers the value of the projected function at those sample points. The set of sample points can actually be augmented so as to improve the accuracy of the approximation: the authors establish a mechanism that ensures that, as the number of samples grows to infinity, the obtained approximation converges to the optimal function.

In kernel-based reinforcement learning [225], the authors also pursue the idea of using sample points to approximate a desired functional. In this method, the transitions in the history of the process are used as samples from which the operator **H** introduced in (4.3) is approximated.

Finally, in a somewhat different approach, Spaan and Vlassis [283, 284, 327] introduce approximate VI algorithms for POMDPs based on point samples obtained according to the dynamics of the POMDP. They provide error bounds for the obtained approximation.

The approach described in this appendix is similar to that described in Chapter 4 in that we use linear function approximation with Q-learning. However, the conditions imposed on the basis functions used allow for the choice of a set of "sample"-points that are then used in an interpolation-like fashion to attain the desired approximations.

In this appendix we consider a parameterized family $\mathcal{Q}$ of functions $Q_\theta : \mathcal{X} \times \mathcal{A} \longrightarrow \mathbb{R}$, where θ is a parameter in $\mathbb{R}^M$. We want to determine the point θ^* in parameter space such that Q_{θ^*} is the best approximation of Q^* in $\mathcal{Q}$, in a sense yet to be made clear. By defining a suitable recursion for θ, we reduce the determination of the infinite-dimensional function Q^* to the determination of a finite-dimensional vector θ^*.

> REMARK: Throughout the appendix, we use the symbol π to denote a policy, instead of the customary symbol δ. We reserve the symbol δ to represent the Dyrac delta. $\diamond$

E.2 Main result

In this section, we establish the convergence properties of Q-learning when using linear function approximation. We identify the conditions ensuring convergence w.p.1 and derive error bounds for the obtained approximation. As will soon become apparent, the results derived herein are deeply related with other approaches described in the literature, *e.g.*, [32, 103, 279, 302, 320].

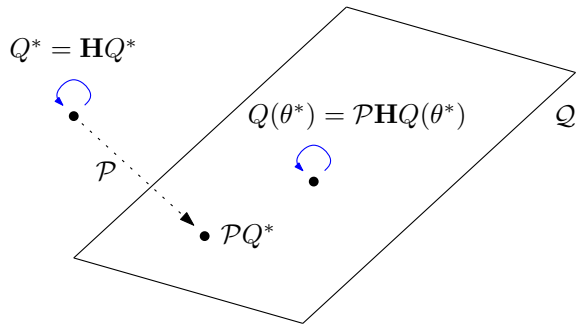


Figure E.1: Optimal function $Q^* = \mathbf{H}Q^*$ and the fixed-point of $Q(\theta^*)$ of the combined operator $\mathcal{P}_\mathcal{Q}\mathbf{H}$. Notice that, in general $Q(\theta^*) \neq \mathcal{P}_\mathcal{Q}Q^*$.

E.2.1 Combining Q-learning with linear function approximation

In our pursuit to approximate Q^*, we start by considering a family of functions $\mathcal{Q} = \{Q_\theta\}$ parameterized by a finite-dimensional parameter vector $\theta \in \mathbb{R}^M$. If we replace the iterative procedure to find Q^* by a suitable "equivalent" procedure to find a parameter θ^* so as to best approximate Q^* by a function in $\mathcal{Q}$, we move from a search in an infinite dimensional function space to a search in a finite dimensional space ($\mathbb{R}^M$). This has an immediate implication: unless if $Q^* \in \mathcal{Q}$, we will not be able to determine Q^* exactly. Instead, we will determine the fixed point of a combined operator $\mathcal{P}_\mathcal{Q}\mathbf{H}$, where $\mathcal{P}_\mathcal{Q}$ is some mapping that "projects" a function q defined in $\mathcal{X} \times \mathcal{A}$ to a point in $\mathcal{Q}$ (see Fig. E.1).

We admit the family $\mathcal{Q}$ to be linear in that if $q_1, q_2 \in \mathcal{Q}$, then so does $\alpha q_1 + q_2$ for any $\alpha \in \mathbb{R}$. $\mathcal{Q}$ is therefore the linear span of some set of linearly independent functions $\xi_i : \mathcal{X} \times \mathcal{A} \longrightarrow \mathbb{R}$, and each $q \in \mathcal{Q}$ can be written as

$$q(x, a) = \sum_{i=1}^{M} \xi_i(x, a)\theta(i),$$

where $\theta(i)$ is the ith component of the vector $\theta \in \mathbb{R}^M$. If $\Xi = \{\xi_1, \ldots, \xi_M\}$ is a set of linearly independent functions, we interchangeably use Q_θ and $Q(\theta)$ to denote the function

$$Q_\theta(x, a) = \sum_{i=1}^{M} \xi_i(x, a)\theta(i) = \xi^\top(x, a)\theta, \tag{E.1}$$

where $\xi(x, a)$ is a vector in $\mathbb{R}^M$ with ith component given by $\xi_i(x, a)$.

We throughout let $\Xi = \{\xi_i, i = 1, \ldots, M\}$ be a set of M bounded, linearly independent functions verifying

$$\sum_i |\xi_i(x, a)| \leq 1 \tag{E.2}$$

for all $(x, a) \in \mathcal{X} \times \mathcal{A}$ and eventually introduce further restrictions on the set Ξ as needed.

E.2.2 Linear approximation using sample-based projection

We now consider a sample-based approximation model that, while imposing somewhat strict conditions on the set of functions Ξ, will allow us to derive useful error bounds for the obtained approximation Q_{θ^*}. For that we assume that the functions in Ξ verify

$$\|\xi_i\|_\infty = 1. \tag{E.3}$$

We remark that if (E.2) and (E.3) simultaneously hold, linear independence of the functions in Ξ arises as an immediate consequence. To see this, notice that for each function $\xi_i \in \Xi$ there is a point (x, a) such that $|\xi_i(x, a)| = 1$, as $\|\xi_i\|_\infty = 1$. Then, since $\sum_i |\xi_i(x, a)| \leq 1$, $\xi_j(x, a) = 0$ for all $j \neq i$. This, in turn, implies the functions in Ξ are linearly independent. As in the previous subsection, we take the family $\mathcal{Q}$ as the linear span of Ξ.

For each function $\xi_i \in \Xi$ take a point (x_i, a_i) in $\mathcal{X} \times \mathcal{A}$ such that $|\xi_i(x_i, a_i)| = 1$, and denote by I the set obtained by gathering M of such points, one for each $\xi_i \in \Xi$. If $\mathscr{B}$ is the set of all (essentially) bounded functions defined on $\mathcal{X} \times \mathcal{A}$ and taking values on $\mathbb{R}$, we define a mapping $\mathcal{P} : \mathscr{B} \longrightarrow \mathbb{R}^M$ as

$$(\mathcal{P}f)(i) = f(x_i, a_i), \tag{E.4}$$

where f is an arbitrary function in $\mathscr{B}$, $(\mathcal{P}f)(i)$ is the ith component of the vector $\mathcal{P}f$ and (x_i, a_i) is the point in I corresponding to ξ_i. Notice that $\mathcal{P}f$ is properly defined for every $f \in \mathscr{B}$ and verifies

$$\|\mathcal{P}f\|_\infty \leq \|f\|_\infty$$

and

$$\mathcal{P}_{\alpha f_1 + f_2} = \alpha \mathcal{P} f_1 + \mathcal{P} f_2.$$

Our variant of Q-learning iteratively determines the point $\theta^* \in \mathbb{R}^M$ verifying the fixed-point recursion

$$\theta^* = \mathcal{P}\mathbf{H}Q(\theta^*), \tag{E.5}$$

where $\mathbf{H}$ is the operator defined in (3.2). Since $\mathbf{H}$ is a contraction in the maximum norm and $\sum_i |\xi_i(x, a)| \leq 1$, the fixed point in (E.5) is properly and uniquely defined.

To derive the expression of the algorithm, we remark that (E.5) can be explicitly written as

$$\theta^*(i) = \int_{\mathcal{X}} \delta_{(x_i,a_i)}(x, a) \int_{\mathcal{X}} \left[r(x, a, y) + \gamma \max_u \xi^\top(y, u)\theta^* \right] \mathsf{P}_a(x, dy) d\mu(x, a),$$

where μ is some probability measure on $\mathcal{X} \times \mathcal{A}$ and $\delta_{(x_i,a_i)}$ is the Dirac delta centered around (x_i, a_i).

We are now in position to describe the algorithm. Let g_ε be a smooth Dirac

approximation,[1] such that

$$\int g_\varepsilon(x,a;y,u)d\mu(y,u) = 1$$
$$\lim_{\varepsilon\to 0}\int g_\varepsilon(x,a;y,u)f(y,u)d\mu(y,u) = f(x,a).$$

Let π be a stochastic stationary policy and suppose that $\{x_t\}$, $\{a_t\}$ and $\{r_t\}$ are sampled trajectories from the MDP $(\mathcal{X},\mathcal{A},\mathsf{P},r,\gamma)$ using policy π. Then, given any initial estimate θ_0, we generate a sequence $\{\theta_t\}$ according to the update rule

$$\theta_{t+1}(i) = \theta_t(i) + \alpha_t g_{\varepsilon_t}(x_i,a_i;x_t,a_t)\big[r_t + \gamma\max_{u\in\mathcal{A}}\xi^\top(x_{t+1},u)\theta_t - \xi^\top(x_t,a_t)\theta_t\big],$$

where $\{\varepsilon_t\}$ is a sequence verifying

$$\varepsilon_{t+1} = (1-\beta_t)\varepsilon_t.$$

More generally, we can have

$$\varepsilon_{t+1} = \varepsilon_t + \beta_t h(\varepsilon_t),$$

where h is chosen so that the ODE $\dot{x}_t = h(x_t)$ has a globally asymptotically stable equilibrium in the origin.

Under some regularity assumptions on the Markov chain $(\mathcal{X},\mathsf{P}_\pi)$ obtained using the policy π and on the step-sizes α_t and β_t, the trajectories of the algorithm closely follow those of an associated ODE with a globally asymptotically stable equilibrium point θ^*. Therefore, the sequence $\{\theta_t\}$ will converge w.p.1 to the equilibrium point θ^* of the ODE.

We now state our main convergence result. Given an MDP $(\mathcal{X},\mathcal{A},\mathsf{P},r,\gamma)$, let π be a stationary stochastic policy and $(\mathcal{X},\mathsf{P}_\pi)$ the corresponding Markov chain with invariant probability measure μ_X. Assume μ_X to be absolutely continuous w.r.t. the Lebesgue measure on $\mathcal{X}$ and bounded away from zero. Denote by $\mathbb{E}_\pi[\cdot]$ the expectation w.r.t. the probability measure μ_π defined for every set $Z\times U\subset\mathcal{X}\times\mathcal{A}$ as

$$\mu_\pi(Z\times U) = \int_Z\sum_{a\in U}\pi(x,a)\mu_X(dx).$$

Also, define $\hat{\alpha}_t(i)$ as

$$\hat{\alpha}_t(i) = \alpha_t g_{\varepsilon_t}(x_i,a_i;x_t,a_t).$$

Theorem E.2.1. *Let $(\mathcal{X},\mathcal{A},\mathsf{P},r,\gamma)$ be a Markov decision process and assume the Markov chain $(\mathcal{X},\mathsf{P}_\pi)$ to be geometrically ergodic with invariant probability measure*

[1] There are several common smooth Dirac approximations, *e.g.*,

$$g_\varepsilon(x;y) = \frac{1}{\varepsilon\sqrt{\pi}}e^{-\|x-y\|^2/\varepsilon^2}.$$

μ_X absolutely continuous w.r.t. the Lebesgue measure on $\mathcal{X}$ and bounded away from 0. Further suppose that $\pi(x,a) > 0$ for all $a \in \mathcal{A}$ and μ_X-almost all $x \in \mathcal{X}$.

Let $\Xi = \{\xi_i, i = 1, \dots, M\}$ be a set of M functions defined on $\mathcal{X} \times \mathcal{A}$ and taking values in $\mathbb{R}$. In particular, admit the functions in Ξ to verify $\|\xi_i\|_\infty = 1$ and $\sum_i |\xi_i(x,a)| \leq 1$.

Then, the following hold:

1. ***Convergence:*** *For any initial condition $\theta_0 \in \mathbb{R}^M$, the algorithm*

$$\theta_{t+1}(i) = \theta_t(i) + \alpha_t g_{\varepsilon_t}(x_i, a_i; x_t, a_t)\big[r_t + \gamma \max_{u \in \mathcal{A}} \xi^\top(x_{t+1}, u)\theta_t - \xi^\top(x_t, a_t)\theta_t\big], \tag{E.6a}$$

$$\varepsilon_{t+1} = (1 - \beta_t)\varepsilon_t. \tag{E.6b}$$

converges w.p.1 as long as the step-size sequences $\{\alpha_t\}, \{\beta_t\}$ are such that

$$\sum_t \alpha_t = \infty; \qquad \sum_t \alpha_t^2 < \infty; \tag{E.7a}$$

$$\sum_t \beta_t = \infty; \qquad \sum_t \beta_t^2 < \infty, \tag{E.7b}$$

$\beta_t = \mathrm{o}(\alpha_t)$ and α_t is built so that $\min_i \sum_t \hat{\alpha}_t(i) = \infty$.

2. ***Limit of convergence:*** *Under these conditions, the limit function $Q(\theta^*)$ of* (E.6) *verifies*

$$Q_{\theta^*}(x,a) = (\mathcal{P}_{\mathcal{Q}} \mathbf{H} Q_{\theta^*})(x,a), \tag{E.8}$$

where $\mathcal{P}_{\mathcal{Q}} : \mathscr{B} \to \mathcal{Q}$ denotes the operator given by

$$(\mathcal{P}_{\mathcal{Q}} Q)(x,a) = \xi^\top(x,a)\mathcal{P}Q.$$

3. ***Error bounds:*** *Under these conditions, the limit function Q_{θ^*} verifies the bound*

$$\|Q(\theta^*) - Q^*\|_\infty \leq \frac{1}{1-\gamma} \|\mathcal{P}_{\mathcal{Q}} Q^* - Q^*\|_\infty. \tag{E.9}$$

E.3 Proof of Theorem E.2.1

We separately establish each of the three assertions of Theorem E.2.1. To prove the first assertion, we establish the trajectories $\{\theta_t\}$ generated by algorithm (E.6) to closely follow those of an associated ODE with a globally asymptotically stable equilibrium point. As long as the iterates of the algorithm remain bounded, this will imply the convergence to the equilibrium point of the associated ODE.

To prove the second assertion of Theorem E.2.1, we provide an interpretation of the equilibrium point of the associated ODE as the fixed point of a composite operator. This interpretation will then lead to the third assertion of Theorem E.2.1.

E.3.1 Convergence of the iterates

To prove the convergence of the sequence $\{\theta_t\}$ generated by (E.6), we follow a similar argument to that in [32, 33].

Consider a general ODE in $\mathbb{R}^M$,

$$\frac{d}{dt}Z(t) = h(Z(t)), \tag{E.10}$$

for a Lipschitz map $h : \mathbb{R}^p \longrightarrow \mathbb{R}^p$. Further suppose that the ODE (E.10) has a globally asymptotically stable equilibrium Z^*.

Given any $T > 0$ and $\sigma > 0$, a bounded measurable function $z : \mathbb{R}^+ \longrightarrow \mathbb{R}^M$ is a (T, σ)-*perturbation of* (E.10) if there is a sequence T_n of positive real numbers such that $T_0 = 0$, $T_n \to \infty$, with $T_{n+1} - T_n > T$ and the following holds

$$\sup_{t\in[T_n,T_{n+1}]} \|Z^n(t) - z(t)\| \leq \sigma,$$

where $Z^n(t)$ is a solution of (E.10) defined in the interval $[T_n, T_{n+1}]$.

We now introduce the Hirsch lemma [124], whose proof can be found, for example, in [33].

Lemma E.3.1 (Hirsch Lemma)**.** *Given any $\rho > 0$ and $T > 0$, there is a σ_0 such that, for all $\sigma < \sigma_0$, every (T, σ)-perturbation of* (E.10) *converges to an ρ-neighborhood of Z^*.*

Consider the ODE

$$\frac{d}{dt}\theta_t(i) = (\mathcal{P}\mathbf{H}Q(\theta_t))_i - \theta_t(i). \tag{E.11}$$

It is not hard to see that this ODE has a globally asymptotically stable equilibrium θ^* verifying

$$\theta^*(i) = (\mathcal{P}\mathbf{H}Q(\theta^*))_i$$

since, as remarked in Subsection E.2.2, $\mathcal{P}$ is a non-expansion in the sup-norm, $\mathbf{H}$ is a contraction in the sup-norm and $\|\xi_i\|_\infty = 1$ for $i = 1, \ldots, M$.

Consider, on the other hand, the ODE

$$\begin{aligned}\frac{d}{dt}\theta_t^\varepsilon(i) = h^\varepsilon(\theta_t) = \\ = \int g_\varepsilon(x_i, a_i; x, a)\big[(\mathbf{H}Q_{\theta_t^\varepsilon})(x, a) - Q_{\theta_t^\varepsilon}(x, a)\big]d\mu_\pi(x, a).\end{aligned} \tag{E.12}$$

By hypothesis, $g_\varepsilon(x_i, a_i; \cdot) \to \delta_{(x_i,a_i)}$ as $\varepsilon \to 0$. By taking $\theta_0^\varepsilon = \theta_0$, a standard argument using the Gronwall inequality leads to the conclusion that the solutions θ_t^ε of (E.12) verify $\theta_t^\varepsilon \to \theta_t$ as $\varepsilon \to 0$, and this convergence holds uniformly in

compact time intervals.[2] This implies that, given any $T > 0$ and $\sigma > 0$, θ_t^ε is a $(T, \frac{\sigma}{2})$-perturbation of (E.11) for sufficiently small ε.

Our purpose is to establish that, for fixed ε, the trajectories of the algorithm (E.6) closely follow those of (E.12). Thus, for fixed ε, rewrite (E.6) in the form

$$\theta_{t+1} = \theta_t + \alpha_t H^\varepsilon(\theta_t, Y_{t+1}), \tag{E.13}$$

where $Y_{t+1} = (X_t, A_t, X_{t+1})$. Since the chain $\{X_t\}$ is geometrically ergodic and $\pi(x,a) > 0$ for μ_π-almost all $x \in \mathcal{X}$, it follows that so is the chain $\{Y_t\}$.

Using the geometric ergodicity of the chain $\{Y_t\}$ and the Poisson equation involving $H^\varepsilon(\theta, Y)$, we can rewrite (E.13) as

$$\theta_{t+1} = \theta_t + \alpha_t h^\varepsilon(\theta_t) + \alpha_t(e_{t+1} + \eta_{t+1}), \tag{E.14}$$

where, for each $i = 1, \ldots, M$, $\sum_{t=0}^\infty \alpha_t e_t(i) < \infty$ and $\|\eta_t\| \to 0$. To see that this is so, recall that the Poisson equation involving $H^\varepsilon(\theta, Y)$ is given by

$$\upsilon_\theta(y) - (\mathsf{P}_\pi \upsilon_\theta)(y) = H^\varepsilon(\theta, y) - h^\varepsilon(\theta),$$

where υ_θ is the solution for a given θ. Under the geometrical ergodicity of $\{Y_t\}$ the solution for this equation always exists [201] and we can rewrite

$$\begin{aligned} H^\varepsilon(\theta_t, X_{t+1})_i &= h^\varepsilon(\theta_t)_i + \upsilon_{\theta_t}(X_{t+1})_i - (\mathsf{P}_\pi \upsilon_{\theta_t})(X_{t+1})_i = \\ &= h^\varepsilon(\theta_t)_i + \upsilon_{\theta_t}(X_{t+1})_i - (\mathsf{P}_\pi \upsilon_{\theta_t})(X_t)_i + \\ &\quad + (\mathsf{P}_\pi \upsilon_{\theta_t})(X_t)_i - (\mathsf{P}_\pi \upsilon_{\theta_{t+1}})(X_{t+1})_i + \\ &\quad + (\mathsf{P}_\pi \upsilon_{\theta_{t+1}})(X_{t+1})_i - (\mathsf{P}_\pi \upsilon_{\theta_t})(X_{t+1})_i = \\ &= h^\varepsilon(\theta_t)_i + \zeta_{t+1}(i) + (u_t(i) - u_{t+1}(i)) + \eta_{t+1}(i), \end{aligned}$$

with

$$\begin{aligned} \zeta_{t+1}(i) &= \upsilon_{\theta_t}(X_{t+1})_i - (\mathsf{P}_\pi \upsilon_{\theta_t})(X_t)_i; \\ u_t(i) &= (\mathsf{P}_\pi \upsilon_{\theta_t})(X_t)_i; \\ u_{t+1}(i) &= (\mathsf{P}_\pi \upsilon_{\theta_{t+1}})(X_{t+1})_i; \\ \eta_{t+1}(i) &= (\mathsf{P}_\pi \upsilon_{\theta_{t+1}})(X_{t+1})_i - (\mathsf{P}_\pi \upsilon_{\theta_t})(X_{t+1})_i. \end{aligned}$$

Finally, setting $e_{t+1}(i) = \zeta_{t+1}(i) + (u_t(i) - u_{t+1}(i))$ leads to (E.14). The two aforementioned properties of e_t and η_t, namely $\sum_{t=0}^\infty \alpha_t e_t(i) < \infty$ and $\|\eta_t\| \to 0$, arise as consequences of the geometric ergodicity assumption on the underlying Markov chain and of the properties of the solution υ_θ of the Poisson equation (see Appendix B and the references [66, 201]).

[2] In particular, in an interval $[t, t+\tau]$ we have

$$\|\theta_t^\varepsilon - \theta_t\| \le K_\varepsilon(e^{C\tau} - 1),$$

for some positive constant C and some positive, ε-dependent constant K_ε that goes to 0 with ε.

We now proceed as in [33]. Define the sequences

$$\tau_0 = 0; \qquad \tau_k = \sum_{t=0}^{k-1} \alpha_t;$$
$$T_0 = 0; \qquad T_{n+1} = \min\left\{\tau_k \mid \tau_k > T_n + T\right\},$$

and let $\{\theta_t\}$ be a sample trajectory obtained using (E.13) with constant ε. From $\{\theta_t\}$ build the continuous-time process $\bar{\theta}^0(t)$ by taking $\bar{\theta}^0(\tau_k) = \theta_k$ and using linear interpolation in the interval $[\tau_k, \tau_{k+1}]$. In particular (E.14) yields

$$\bar{\theta}^0(\tau_{t+1}) = \bar{\theta}^0(\tau_t) + (\tau_{t+1} - \tau_t)h^\varepsilon(\theta^0(\tau_t)) + \rho(\tau_t), \tag{E.15}$$

with

$$\rho(\tau_t) = \alpha_t(e_{t+1} + \eta_{t+1}).$$

By interpreting (E.15) as a discretized version of the ODE (E.12) with noise $\rho(\tau_t)$, we can once again resort to the Gronwall inequality to bound the distance between $\bar{\theta}^0(t)$ and θ_t^ε in each interval $[T_k, T_{k+1}]$ by a constant that can be made arbitrarily small for large enough k. But this means that, for any $\sigma > 0$, the process $\bar{\theta}^{t_0}(t) = \bar{\theta}^0(t + t_0)$ is a $(T, \frac{\sigma}{2})$-perturbation of (E.12) for t_0 large enough.

By combining this conclusion with the previous conclusion on θ_t^ε, we see that, for any $T > 0$ and $\sigma > 0$, $\bar{\theta}^{t_0}(t)$ is a (T, σ)-perturbation of (E.11) (for ε sufficiently small).

Consider now the ODE

$$\frac{d}{dt}\varepsilon_t = 0. \tag{E.16}$$

and the approximate process

$$\varepsilon_{t+1} = \varepsilon_t - \alpha_t \frac{\beta_t}{\alpha_t}\varepsilon_t. \tag{E.17}$$

Repeating the exact same procedure used for $\bar{\theta}_t^0$, and by noticing that $\beta_t = \mathrm{o}(\alpha_t)$, we build a process $\bar{\varepsilon}_t^{t_0}$ such that, for any $\sigma > 0$, $\bar{\varepsilon}_t^{t_0}$ is a (T, σ)-perturbation of (E.16) for t_0 large enough. Finally, since $\sum_t \hat{\alpha}_t(i) = \infty$, this leads to the conclusion that, given any $T > 0$ and $\sigma > 0$, $(\bar{\theta}_t^{t_0}, \bar{\varepsilon}_t^{t_0})$ is a (T, σ)-perturbation of the system of ODEs

$$\frac{d}{dt}\theta_t(i) = (\mathcal{P}\mathbf{H}Q(\theta_t))_i - \theta_t(i); \tag{E.18a}$$
$$\frac{d}{dt}\varepsilon_t = 0, \tag{E.18b}$$

for t_0 large enough.

We now notice that, in general, a smaller ε will require a larger t_0 to ensure that $\bar{\theta}^{t_0}(t)$ is a (T, σ)-perturbation of (E.12). On the other hand, ε_t as generated by (E.6b) converges to 0. Therefore, to guarantee that there is t_0 such that $(\bar{\theta}_t^{t_0}, \bar{\varepsilon}_t^{t_0})$ is a (T, σ)-perturbation of (E.18), it is necessary to ensure that ε_t approaches zero sufficiently slowly. By choosing the sequence α_t so that $\sum_t \hat{\alpha}_t(i) = \infty$, we ensure

that the trajectories $\{\theta_t\}$ approach the trajectories of the ODE (E.12) faster than g_{ε_t} approaches the Dirac delta and are in position to apply the Hirsh lemma to conclude that, for any $\rho > 0$, the process $(\bar{\theta}_t^{t_0}, \bar{\varepsilon}_t^{t_0})$ converges to a ρ-neighborhood of $(\theta^*, 0)$. This, in turn, implies that $\theta_t \to \theta^*$ as long as the sequence $\{\theta_t\}$ remains bounded, which we establish in the continuation.

E.3.2 Boundedness of the iterates

To establish the boundedness of the iterates, we replicate the procedure in [34].

Let $\{\theta_t\}$ be a trajectory generated by (E.6). We build a scaled sequence $\{\hat{\theta}_t\}$ by setting

$$\hat{\theta}_t = \frac{\theta_t}{\lambda_n}$$

where $\lambda_n = \max\{\|\theta_{T_n}\|, 1\}$, for every t in $[T_n, T_{n+1})$ and T_i are as defined in Subsection E.3.1. If the sequence θ_t is unbounded, this means that $\limsup \lambda_n = \infty$, so we analyze the behavior of $\hat{\theta}_t$ as $\lambda_n \to \infty$.

In Subsection E.3.1, we established the trajectories $\{\theta_t\}$ to closely follow those of the ODE

$$\frac{d}{dt}\theta_t = h(\theta_t),$$

where $h(\theta) = \mathcal{P}\mathbf{H}Q(\theta) - \theta$. For the scaled sequence $\hat{\theta}_t$, we now consider the function $h_\lambda(\theta)$, given by

$$h_\lambda(\theta) = \frac{h(\lambda\theta)}{\lambda},$$

with $\lambda > 0$. Notice that, as $\lambda \to \infty$, h_λ approaches the function h_∞ given by

$$h_\infty(\theta)_i = \gamma \max_{b\in\mathcal{A}} \int \xi^\top(y, b)\theta \mathsf{P}_{a_i}(x_i, dy) - \theta_i.$$

Define the operator $\mathbf{F} : \mathbb{R}^M \longrightarrow \mathbb{R}^M$ with ith component given by

$$\mathbf{F}(\theta)_i = \gamma \max_{b\in\mathcal{A}} \int \xi^\top(y, b)\theta \mathsf{P}_{a_i}(x_i, dy).$$

This operator is a contraction in the sup-norm (due to the fact that $\sum_i |\xi_i(x, a)| \leq 1$) and, hence has a single fixed point. Since the origin is a fixed point of $\mathbf{F}$, the ODE associated with h_∞ has a single equilibrium point at the origin and this equilibrium point is globally asymptotically (exponentially) stable.

By repeating the procedure used in Subsection E.3.1, we can build by interpolation a continuous time process from the scaled sequence $\hat{\theta}_t$. In each interval $[T_n, T_{n+1}]$, this continuous-time process is a (T, σ)-perturbation of the ODE,

$$\frac{d}{dt}\theta_t = h_{\lambda_n}(\theta_t), \tag{E.19}$$

for any $T > 0$ and any $\sigma > 0$ (by eventually considering a time-shifted version of the continuous time process, as in Subsection E.3.1). This implies the boundedness of θ_t as a consequence.

In fact, suppose that $\{\theta_t\}$ is not bounded. This implies that $\lambda_n \to \infty$ eventually along a subsequence. Since the solutions of (E.19) converge exponentially fast to an arbitrarily small neighborhood of the origin (depending on λ_n), by taking n large enough we can ensure that $\left\|\hat{\theta}_{T_{n+1}}\right\| \leq C$ for any $C < 1$. But this implies that

$$\frac{\left\|\theta_{T_{n+1}}\right\|}{\|\theta_{T_n}\|} \leq C$$

or, equivalently, $\left\|\theta_{T_{n+1}}\right\| \leq C \|\theta_{T_n}\|$. Therefore, whenever θ_t leaves, say, the unit ball in $\mathbb{R}^M$, it returns exponentially fast toward it, and θ_t remains bounded.

We refer to [32, 34], where a similar process is apllied to establish boundedness of an iterative process.

E.3.3 Limit of convergence and error bounds

We have established that the sequence $\{\theta_t\}$ generated by (E.6) converges w.p.1 to a point θ^*. The limit point θ^* is the globally asymptotically stable equilibrium of the ODE (E.11), verifying the following recursive relation:

$$\theta^* = \mathcal{P}\mathbf{H}Q(\theta^*).$$

This provides an interpretation for the limit point of $\{\theta_t\}$ as the fixed point of the combined operator $\mathcal{P}\mathbf{H}Q(\cdot)$, where Q is now understood as a mapping from $\mathbb{R}^M$ to $\mathscr{B}$.

To conclude the proof of Theorem E.2.1, it remains to establish statement 3, thus providing the error bounds for the approximation. To this, we perform some explicit computations, yielding

$$\begin{aligned}
\|Q(\theta^*) - Q^*\|_\infty &= \|Q(\theta^*) - Q(\mathcal{P}Q^*) + Q(\mathcal{P}Q^*) - Q^*\|_\infty \leq \\
&\leq \|Q(\theta^*) - Q(\mathcal{P}Q^*)\|_\infty + \|Q(\mathcal{P}Q^*) - Q^*\|_\infty = \\
&= \|Q(\theta^* - \mathcal{P}Q^*)\|_\infty + \|Q(\mathcal{P}Q^*) - Q^*\|_\infty = \\
&= \|Q(\mathcal{P}\mathbf{H}Q(\theta^*) - \mathcal{P}\mathbf{H}Q^*)\|_\infty + \|Q(\mathcal{P}Q^*) - Q^*\|_\infty
\end{aligned}$$

Using the fact that $\|\xi_i\| = 1$, we get

$$\begin{aligned}
&\|Q(\theta^*) - Q^*\|_\infty \leq \\
&\quad = \|\mathcal{P}\mathbf{H}Q(\theta^*) - \mathcal{P}\mathbf{H}Q^*\|_\infty + \|Q(\mathcal{P}Q^*) - Q^*\|_\infty \leq \\
&\quad = \gamma \|Q(\theta^*) - Q^*\|_\infty + \|Q(\mathcal{P}Q^*) - Q^*\|_\infty ,
\end{aligned}$$

and this finally leads to

$$\|Q(\theta^*) - Q^*\|_\infty \leq \frac{1}{1-\gamma} \|Q(\mathcal{P}Q^*) - Q^*\|_\infty .$$

This concludes the proof of Theorem E.2.1.

E.4 Discussion

We emphasize the similarity between interpolation-based Q-learning (IBQL) by Szepesvári and Smart [302] and the algorithm in this appendix. The fundamental difference between these two methods lies on the fact that IBQL only makes use of the estimated Q-function to predict the value of the next state. The updates of IBQL rely on a vector $\hat{d}_t$ of modified temporal differences with ith component given by

$$\begin{aligned}\hat{d}_t(i) &= r_t + \gamma \max_{u \in \mathcal{A}} Q_{\theta_t}(x_{t+1}, u) - \theta_t(i) = \\ &= r_t + \gamma \max_{u \in \mathcal{A}} Q_{\theta_t}(x_{t+1}, u) - Q_{\theta_t}(x_i, a_i).\end{aligned}$$

Notice that each $\hat{d}_t(i)$ is not a temporal-difference in the strict sense, since it does not provide a one-step estimation "error". This means that the information provided by $\hat{d}_t(i)$ may lead to "misleading" updates. Although not affecting the convergence of IBQL in the long-run, IBQL may exhibit slower convergence because of this. On the other hand, if IBQL is used with a vanishing ε, the effect of these misleading updates will vanish as $t \to \infty$. In the experimental results portrayed in [302], a vanishing ε was used. Nevertheless, IBQL exhibited initially slower convergence than of other methods, probably because of this reported effect.

We also remark that, in [302], the convergence result requires the underlying Markov chain to be positive Harris and aperiodic. These conditions are actually weaker than the geometric ergodicity required by our result. However, in many practical situations, the former conditions will actually imply the latter.[3] This means that the conditions on the problem required in Theorem E.2.1 are essentially similar to those in [302] placing the results of both papers in a common line of work and, basically, leading to concordant conclusions.

Finally, we also refer the close relation between our method and the algorithm by Borkar [32]. In the aforementioned work, Borkar provides a convergence analysis of what we may refer to as *functional Q-learning*. This functional Q-learning can be seen as an extension of classical Q-learning to functional spaces, and arises from the approach proposed by Baker [13] to stochastic approximation in function spaces. The update equation for this method is fundamentally similar to (E.6). The main difference is that, while we consider only a fixed, finite set of points $I = \{(x_1, a_1), \ldots, (x_M, a_M)\}$, the algorithm in [32] maintains a *complete representation of Q^**, each component of which is updated at each iteration. Clearly, maintaining such a representation of Q^* is computationally impossible. Therefore, the algorithm in [32] boils down to maintaining a complete record of the history of past events $\mathcal{H} = \{(x_0, a_0), \ldots, (x_t, a_t), \ldots\}$ and of the estimates Q_t at each of these points. Then, the value of Q^* at a generic point $(x, a) \in \mathcal{X} \times \mathcal{A}$ is estimated as

$$Q_{t+1}(x, a) = Q_0(x, a) + \sum_{k=0}^{t} \alpha_k g_{\varepsilon_k}(x_k, a_k; x, a)\big[r_k + \gamma \max_{u \in \mathcal{A}} Q_t(x_{k+1}, u) - Q_t(x_k, a_k)\big].$$

[3] An aperiodic, positive Harris chain is geometrically ergodic as long as the support of μ_X has non-empty interior.

Appendix F

Proofs

In this appendix, we provide the formal proofs of the main results presented along the thesis.

F.1 Proofs for Chapter 3

The proofs in this section belong to the theorems found in Chapter 3 of the main text.

F.1.1 Proof of Theorem 3.3.1

The proof closely follows [300]. We use Theorem D.1.4 of Appendix D.

Let B be the space of bounded functions defined on $\mathcal{X}$ and consider the randomized operator $T_t : B \times B \longrightarrow B$ defined by

$$T_t(U,V)(i) = \begin{cases} \sum_{j\in\mathcal{X}} \hat{\mathsf{P}}_t(i_t,j)\big(\hat{r}_t(i_t,j) + \gamma V(j)\big) & \text{if } i = i_t; \\ U(i) & \text{otherwise.} \end{cases}$$

We want to show that the operator T_t approximates the dynamic programming operator $\mathbf{T}^\delta$ defined in (3.3).

Notice that, by assumption, all states $i \in \mathcal{X}$ are visited infinitely often; furthermore, when a state i is visited, $U_{t+1}(i)$ does not depend on U_t. Therefore, to show that T_t approximates $\mathbf{T}^\delta$ we need only to show that

$$\left| \sum_{j\in\mathcal{X}} \hat{\mathsf{P}}_t(i,j)\big(\hat{r}_t(i,j) + \gamma V(j)\big) - \sum_{j\in\mathcal{X}} \mathsf{P}_\delta(i,j)\big(r_\delta(i,j) + \gamma V(j)\big) \right| \to 0$$

for all $i \in \mathcal{X}$, where r_δ is defined in Subsection 3.3.1.

Some explicit computations now yield

$$\begin{aligned} &\left| \sum_{j\in\mathcal{X}} \hat{\mathsf{P}}_t(i,j)\big(\hat{r}_t(i,j) + \gamma V(j)\big) - \sum_{j\in\mathcal{X}} \mathsf{P}_\delta(i,j)\big(r_\delta(i,j) + \gamma V(j)\big) \right| \leq \\ &\quad \leq \max_{i\in\mathcal{X}} \left| \sum_{j\in\mathcal{X}} \hat{\mathsf{P}}_t(i,j)\big(\hat{r}_t(i,j) + \gamma V(j)\big) - \sum_{j\in\mathcal{X}} \mathsf{P}_\delta(i,j)\big(r_\delta(i,j) + \gamma V(j)\big) \right| \leq \\ &\quad \leq \max_{i\in\mathcal{X}} \left| \sum_{j\in\mathcal{X}} \hat{\mathsf{P}}_t(i,j)\big(\hat{r}_t(i,j) + \gamma V(j)\big) - \sum_{j\in\mathcal{X}} \hat{\mathsf{P}}_t(i,j)\big(r_\delta(i,j) + \gamma V(j)\big) \right| + \\ &\qquad + \max_{i\in\mathcal{X}} \left| \sum_{j\in\mathcal{X}} \hat{\mathsf{P}}_t(i,j)\big(r_\delta(i,j) + \gamma V(j)\big) - \mathsf{P}_\delta(i,j)\big(r_\delta(i,j) + \gamma V(j)\big) \right| \leq \\ &\quad \leq \max_{i,j\in\mathcal{X}} |\hat{r}_t(i,j) - r_\delta(i,j)| + \max_{i\in\mathcal{X}} \left| \sum_{j\in\mathcal{X}} \big(\hat{\mathsf{P}}_t(i,j) - \mathsf{P}_\delta(i,j)\big)\big(r_\delta(i,j) + \gamma V(j)\big) \right|. \end{aligned}$$

Since, by the law of large numbers, $\hat{\mathsf{P}}_t \to \mathsf{P}_\delta$ and $\hat{r}_t \to r_\delta$, the last term goes to zero as $t \to \infty$ and, T_t approximates $\mathbf{T}^\delta$.

Now, by defining the functions $f_t(i)$ and $g_t(i)$ as

$$f_t(i) = \begin{cases} \gamma & \text{if } i = i_t; \\ 0 & \text{otherwise} \end{cases}$$

and

$$g_t(i) = \begin{cases} 0 & \text{if } i = i_t; \\ 1 & \text{otherwise,} \end{cases}$$

all conditions of Theorem D.1.4 are verified, and the conclusion immediately follows. □

F.1.2 Proof of Theorem 3.3.2

The proof is basically similar to that of Theorem 3.3.1. We use Theorem D.1.4 of Appendix D.

Let B be the space of bounded functions defined on $\mathcal{X} \times \mathcal{A}$ and consider the randomized operator $T_t : B \times B \longrightarrow B$ defined by

$$T_t(Q, Q')(i, a) = \begin{cases} \sum_{j \in \mathcal{X}} \hat{\mathsf{P}}_t(i_t, a_t, j)\big(\hat{r}_t(i_t, a_t, j) + \gamma \max_{b \in \mathcal{A}} Q'(j, b)\big) & \text{if } (i, a) = (i_t, a_t); \\ Q(i, a) & \text{otherwise.} \end{cases}$$

We want to show that the operator T_t approximates the dynamic programming operator $\mathbf{H}$ defined in (3.2).

Repeating the reasoning in the proof of Theorem 3.3.1, we have that

$$\left| \sum_{j \in \mathcal{X}} \hat{\mathsf{P}}_t(i, a, j)\big(\hat{r}_t(i, a, j) + \gamma \max_{b \in \mathcal{A}} Q'(j, b)\big) - \sum_{j \in \mathcal{X}} \mathsf{P}_a(i, j)\big(r(i, a, j) + \gamma \max_{b \in \mathcal{A}} Q'(j, b)\big) \right| \to 0$$

for all $(i, a) \in \mathcal{X} \times \mathcal{A}$ and this implies that T_t approximates $\mathbf{H}$. Defining

$$f_t(i, a) = \begin{cases} \gamma & \text{if } (i, a) = (i_t, a_t); \\ 0 & \text{otherwise} \end{cases}$$

and

$$g_t(i, a) = \begin{cases} 0 & \text{if } (i, a) = (i_t, a_t); \\ 1 & \text{otherwise,} \end{cases}$$

we can apply Theorem D.1.4 and the assertion of the theorem follows. □

F.1.3 Proof of Theorem 3.4.1

The proof closely follows [131] We use Lemma D.1.3 of Appendix D.

For that purpose, we rewrite the update rule for TD(0) as

$$V_{t+1}(i_t) = (1 - \alpha_t(i_t))V_t(i_t) + \alpha_t(i_t)\big(r_t + \gamma V_t(i_{t+1})\big).$$

Subtracting $V^\delta(i_t)$ in both sides and writing $\Delta_t(i_t) = V_t(i_t) - V^\delta(i_t)$, we get

$$\Delta_{t+1}(i_t) = (1 - \alpha_t(i_t))\Delta_t(i_t) + \alpha_t(i_t)\big(r_t + \gamma V_t(i_{t+1}) - V^\delta(i_t)\big).$$

By identifying the set X in Lemma D.1.3 of Appendix D with $\mathcal{X}$ and $F_t(x)$ with

$r_t + \gamma V_t(i_{t+1}) - V^\delta(i_t)$ we have, for $i = i_t$,

$$\mathbb{E}\left[F_t(i) \mid \mathcal{F}_t\right] = \sum_{j \in \mathcal{X}} \mathsf{P}_\delta(i_t, j)\big[r_\delta(i_t, j) + \gamma V_t(j) - V^\delta(i_t)\big]$$

and 0 otherwise. We denoted by $r_\delta(i_t, j)$ the reward $r(i_t, \delta(i_t), j)$, defined in Section 3.3.

Now replacing the definition of $V^\delta(i_t, a_t)$ in the expression for $\mathbb{E}\left[F_t(i) \mid \mathcal{F}_t\right]$ yields

$$\begin{aligned}
&\mathbb{E}\left[F_t(i_t) \mid \mathcal{F}_t\right] = \\
&\quad = \sum_{j \in \mathcal{X}} \mathsf{P}_\delta(i_t, j)\big[r_\delta(i_t, j) + \gamma V_t(j, b) - V^\delta(i_t)\big] = \\
&\quad = \gamma \sum_{j \in \mathcal{X}} \mathsf{P}_\delta(i_t, j)\big[V_t(j) - V^\delta(j)\big].
\end{aligned}$$

By computing the norm we obtain

$$\begin{aligned}
&\left\|\mathbb{E}\left[F_t(i) \mid \mathcal{F}_t\right]\right\|_\infty = \\
&\quad = \max_i \left|\mathbb{E}\left[F_t(i) \mid \mathcal{F}_t\right]\right| \leq \\
&\quad \leq \gamma \max_i \sum_{j \in \mathcal{X}} \mathsf{P}_\delta(i, j)\left|V_t(j) - V^\delta(j)\right| \leq \\
&\quad \leq \gamma \left\|\Delta_t\right\|_\infty .
\end{aligned}$$

This ensures the second condition of Lemma D.1.3. The third condition arises immediately as a consequence of the fact that the rewards r are bounded. Therefore, by Lemma D.1.3, $\Delta_t \to 0$ w.p.1 or, equivalently, $V_t \to V^\delta$ w.p.1.

F.1.4 Proof of Theorem 3.4.2

The proof of this theorem is essentially similar to that of Theorem 3.4.1 and closely follows [131]. We start by rewriting the update rule for Q-learning as

$$Q_{t+1}(i_t, a_t) = (1 - \alpha_t(i_t, a_t))Q_t(i_t, a_t) + \alpha_t(i_t, a_t)\big(r_t + \gamma \max_{b \in \mathcal{A}} Q_t(i_{t+1}, b)\big)$$

and subtract $Q^*(i_t, a_t)$ in both sides. We let $\Delta_t(i_t, a_t) = Q_t(i_t, a_t) - Q^*(i_t, a_t)$, yielding

$$\Delta_{t+1}(i_t, a_t) = (1 - \alpha_t(i_t, a_t))\Delta_t(i_t, a_t) + \alpha_t(i_t, a_t)\big(r_t + \gamma \max_{b \in \mathcal{A}} Q_t(i_{t+1}, b) - Q^*(i_t, a_t)\big).$$

By identifying the set X in Lemma D.1.3 of Appendix D with the set $\mathcal{X} \times \mathcal{A}$ and $F_t(x)$ with $r_t + \gamma \max_{b \in \mathcal{A}} Q_t(i_{t+1}, b) - Q^*(i_t, a_t)$ we have, when $(i, a) = (i_t, a_t)$,

$$\mathbb{E}\left[F_t(i, a) \mid \mathcal{F}_t\right] = \sum_{j \in \mathcal{X}} \mathsf{P}_{a_t}(i_t, j)\big[r(i_t, a_t, j) + \gamma \max_{b \in \mathcal{A}} Q_t(j, b) - Q^*(i_t, a_t)\big]$$

and 0 otherwise. Using the definition of $Q^*(i_t, a_t)$ in the expression of $\mathbb{E}\left[F_t(i, a) \mid \mathcal{F}_t\right]$,

we have

$$\begin{aligned}
&\mathbb{E}\left[F_t(i_t,a_t)\mid\mathcal{F}_t\right]=\\
&\quad=\sum_{j\in\mathcal{X}}\mathsf{P}_{a_t}(i_t,j)\big[r(i_t,a_t,j)+\gamma\max_{b\in\mathcal{A}}Q_t(j,b)-Q^*(i_t,a_t)\big]=\\
&\quad=\gamma\sum_{j\in\mathcal{X}}\mathsf{P}_{a_t}(i_t,j)\big[\max_{b\in\mathcal{A}}Q_t(j,b)-\max_{b\in\mathcal{A}}Q^*(j,b)\big].
\end{aligned}$$

By computing the norm we obtain, easily,

$$\begin{aligned}
&\left\|\mathbb{E}\left[F_t(i,a)\mid\mathcal{F}_t\right]\right\|_\infty=\\
&\quad=\max_{i,a}\left|\mathbb{E}\left[F_t(i,a)\mid\mathcal{F}_t\right]\right|\leq\\
&\quad\leq\gamma\max_{i,a}\sum_{j\in\mathcal{X}}\mathsf{P}_a(i,j)\max_{j,b}\left|Q_t(j,b)-\max_{b\in\mathcal{A}}Q^*(j,b)\right|=\\
&\quad=\gamma\left\|\Delta_t\right\|_\infty.
\end{aligned}$$

This ensures the second condition of Lemma D.1.3.

Like in the proof of Theorem 3.4.1, the third condition arises from the fact that the rewards r are bounded. Lemma D.1.3 ensures that $\Delta_t\to 0$ w.p.1 or, equivalently, $Q_t\to Q^*$ w.p.1.

On the other hand, the fact that the learning policy is GLIE ensures that, as $t\to\infty$, $\delta_t(i,a)\to 1$ if $a=\arg\max_{a\in\mathcal{A}}Q_t(i,a)$ and 0 otherwise. Since, as seen, $Q_t\to Q^*$ w.p.1, it is immediate that $\delta_t\to\delta^*$ w.p.1. □

F.1.5 Proof of Theorem 3.4.3

This proof follows from the proof of Theorem 3.4.2 and closely follows [280]. We use Lemma D.1.3 of Appendix D.

As in the proof of Theorem 3.4.2, we start by rewriting the update rule for SARSA as

$$\begin{aligned}
Q_{t+1}(i_t,a_t)=(1-\alpha_t(i_t,a_t))Q_t(i_t,a_t)+\alpha_t(i_t,a_t)\big(r_t+\gamma\max_{b\in\mathcal{A}}Q_t(i_{t+1},b)\big)+\\
+\alpha_t(i_t,a_t)\gamma\big(Q_t(i_{t+1},a_{t+1})-\max_{b\in\mathcal{A}}Q_t(i_{t+1},b)\big).
\end{aligned}$$

Subtracting $Q^*(i_t,a_t)$ in both sides and writing $\Delta_t(i_t,a_t)=Q_t(i_t,a_t)-Q^*(i_t,a_t)$, we get

$$\begin{aligned}
\Delta_{t+1}(i_t,a_t)=(1-\alpha_t(i_t,a_t))\Delta_t(i_t,a_t)+\alpha_t(i_t,a_t)\big(r_t+\gamma\max_{b\in\mathcal{A}}Q_t(i_{t+1},b)-Q^*(i_t,a_t)\big)+\\
+\alpha_t(i_t,a_t)\gamma\big(Q_t(i_{t+1},a_{t+1})-\max_{b\in\mathcal{A}}Q_t(i_{t+1},b)\big)=\\
=(1-\alpha_t(i_t,a_t))\Delta_t(i_t,a_t)+\alpha_t(i_t,a_t)\big(F_t^Q(i_t,a_t)+C_t(i_t,a_t)\big),
\end{aligned}$$

where $F^Q(i,a)$ is

$$F_t^Q(i,a) = \begin{cases} r_t + \gamma \max_{b\in\mathcal{A}} Q_t(i_{t+1},b) - Q^*(i_t,a_t) & \text{if } (i,a) = (i_t,a_t) \\ 0 & \text{otherwise} \end{cases}$$

and

$$C_t(i,a) = \begin{cases} \gamma\big(Q_t(i_{t+1},a_{t+1}) - \max_{b\in\mathcal{A}} Q_t(i_{t+1},b)\big) & \text{if } (i,a) = (i_t,a_t) \\ 0 & \text{otherwise.} \end{cases}$$

From the proof of Theorem 3.4.2, we have that

$$\left\| \mathbb{E}\left[F_t^Q(i,a) \mid \mathcal{F}_t \right] \right\| \leq \gamma \left\| \Delta_t \right\|.$$

On the other hand, the learning policy δ is admittedly GLIE and, therefore,

$$c_t = \left\| \mathbb{E}\left[C_t(i,a) \mid \mathcal{F}_t \right] \right\| \to 0,$$

as $t \to \infty$.[1]

The bound on the variance of $F_t^Q(i,a) + C_t(i,a)$ arises as a consequence of the corresponding bound on the variance of $F_t^Q(i,a)$. □

F.2 Proofs for Chapter 4

The proofs in this section belong to several results found in Chapter 4 of the main text.

F.2.1 Proof of Theorem 4.5.2

We separately establish each of the statements in Theorem 4.5.2. To prove Statement 1, we write (4.15) in the form

$$\theta_{t+1} = \theta_t + \alpha_{t+1} H(\theta_t, Y_{t+1}),$$

and use Theorem D.1.1 of Appendix D: we show that the associated ODE has a globally asymptotically stable equilibrium point, which implies the existence of a Lyapunov function $\mathcal{U}$ verifying the requirements of the referred theorem. That establishes convergence of (4.15) w.p.1.

To prove Statement 2 of Theorem 4.5.2, we provide an interpretation of the equilibrium point of the ODE associated with algorithm (4.15) as the fixed point of a composite operator.

[1] This holds since the estimates Q_t remain bounded and the set $\mathcal{X}\times\mathcal{A}$ is finite. The boundedness of Q_t arises as a consequence Theorem 3.4.2: the trajectories $\{Q_t\}$ of the SARSA algorithm are upper-bounded by those of Q-learning and lower-bounded by those of an alternate version of Q-learning that minimizes Q_t instead of maximizing it.

Regularity assumptions on Markov chains

In this subsection we prove that, under the assumptions of Theorem 4.5.2, Ass. 1 and Ass. 2 of Theorem D.1.1 actually hold.

Let $Y_t = (X_{t-1}, A_{t-1}, X_t)$ for each $t \in \mathcal{T}$. The sequence $\{Y_t\}$ is a stochastic process, since X_{t-1}, A_{t-1} and X_t are r.v.s for all t. Furthermore, denoting by $\mathcal{F}_t$ the σ-field generated by the history

$$\mathcal{H}_t = \{\theta_0, X_0, \ldots, X_t, A_0, \ldots, A_t\},$$

we have

$$\mathbb{P}\left[Y_{t+1} \in U \mid \mathcal{F}_t\right] = \mathbb{P}\left[X_{t+1} \in U_2, A_t \in U_A, X_t \in U_1 \mid \mathcal{F}_t\right],$$

for any set $U = U_1 \times U_A \times U_2$, with $U_1, U_2 \in \mathcal{B}(\mathcal{X})$ and $U_A \subset \mathcal{A}$. This, in turn, leads to

$$\mathbb{P}\left[Y_{t+1} \in U \mid \mathcal{F}_t\right] = \mathbb{P}\left[X_{t+1} \in U_2, A_t \in U_A, X_t \in U_1 \mid X_t = x\right] = \mathbb{P}\left[Y_{t+1} \in U \mid Y_t\right]$$

and this means that $\{Y_t\}$ is a Markov chain with state-space $\mathcal{X}\times\mathcal{A}\times\mathcal{X}$ and transition probabilities given by

$$\mathbb{P}\left[Y_{t+1} \in U \mid Y_t = (x, a, y)\right] = \begin{cases} \sum_{b \in U_A} \delta(y, b) \mathsf{P}_b(y, U_2), & \text{if } y \in U_1; \\ 0 & \text{otherwise,} \end{cases} \tag{F.1}$$

where $U = U_1 \times U_A \times U_2$.

Let $\mathcal{Y} = \mathcal{X} \times \mathcal{A} \times \mathcal{X}$ denote the state-space and P_Y denote the transition probability kernel for the Markov chain $\{Y_t\}$. Since $\{X_t\}$ is assumed geometrically ergodic with invariant probability measure μ_X, $\delta(x, a) > 0$ for every a and μ_X-almost every x and $\mathcal{A}$ is a finite set, $\{Y_t\}$ is also geometrically ergodic, with invariant probability measure μ_Y verifying

$$\mu_Y(U) = \int_{U_1} \sum_{a \in U_A} \delta(x, a) \mathsf{P}_a(x, U_2) d\mu_X(x),$$

with $U = U_1 \times U_A \times U_2$, $U_1, U_2 \in \mathcal{B}(\mathcal{X})$ and $U_A \subset \mathcal{A}$. That μ_Y above is indeed invariant with respect to the transition probabilities P_Y in (F.1) can easily be checked by applying directly the definition of invariant measure (see Appendix B).

We now write algorithm (4.15) as

$$\theta_{t+1} = \theta_t + \alpha_{t+1} H(\theta_t, Y_{t+1}),$$

where $\{Y_t\}$ is as defined above and the i^{th} component of H is given, for a generic θ, by

$$H(\theta, y)_i = \xi_i(x, a)\Big(r(x, a, z) + \gamma \max_{b \in \mathcal{A}} \xi^\top(z, b)\theta - \xi^\top(x, a)\theta\Big), \tag{F.2}$$

with $y = (x, a, z)$. This leads to the following lemma.

Lemma F.2.1. *Under the conditions of Theorem 4.5.2 there is a constant $K > 0$ such that*

$$\|H(\theta, y)\| \leq K(1 + \|\theta\|),$$

where $\|\cdot\|$ denotes the sup-norm.

Proof We establish the assertion of the lemma by successively bounding the several terms of H. We have, for $y = (x, a, z)$,

$$\begin{aligned}\|H(\theta, y)\| &= \max_{i=1,\dots,N} |H_i(\theta, y)| \leq \\ &\leq \max_{i=1,\dots,N} \left| \xi_i(x,a)\big(r(x,a,z) + \gamma \max_{b\in\mathcal{A}} \xi^\top(z,b)\theta\big) - \xi^\top(x,a)\theta \right| \leq \\ &\leq \max_{i=1,\dots,N} |\xi_i(x,a) r(x,a,z)| + \gamma \max_{b\in\mathcal{A}} \left|\xi_i(x,a)\xi^\top(z,b)\theta\right| + \left|\xi^\top(x,a)\theta_i\right|.\end{aligned}$$

Since, by assumption, $|r(x, a, z)| \leq \mathcal{R}$ and $|\xi_i(x, a)| \leq 1$, we have

$$\|H(\theta, y)\| \leq \mathcal{R} + (1 + \gamma)\,\|\theta\|$$

and, finally, by setting $K = \max\{\mathcal{R}; (1 + \gamma)\}$,

$$\|H(\theta, y)\| \leq K(1 + \|\theta\|).$$

□

This leads to the following proposition.

Proposition F.2.2. *Under the conditions of Theorem 4.5.2, Ass. 1 and Ass. 2 of Theorem D.1.1 hold.*

Proof Ass. 1.a follows trivially from Lemma F.2.1. Ass. 1.b follows from the fact that $\mathcal{Y}$ is compact. Ass. 1.c follows from Theorem B.9.5 (see Appendix B) and since P_Y does not depend on θ, Ass. 2 trivially holds. □

Convergence of the iterates

Given that the requirements of Theorem D.1.1 of Appendix D are satisfied, that same result guarantees the convergence of $\{\theta_t\}$ w.p.1 as long as the ODE

$$\dot{\theta}_t = h(\theta_t), \tag{F.3}$$

with

$$h(\theta) = \mathbb{E}_{\mu_Y}\Big[\xi(x,a)\big(r(x,a,z) + \gamma \max_{b\in\mathcal{A}} \xi^\top(z,b)\theta - \xi^\top(x,a)\theta\big)\Big], \tag{F.4}$$

has a globally asymptotically stable equilibrium θ^*. This is established in the next proposition.

Proposition F.2.3. *The ODE*

$$\dot{\theta}_t = h(\theta_t), \tag{F.5}$$

where h is defined in (F.4)*, is globally asymptotically stable.*

Proof We start by rewriting h as

$$h(\theta) = h_1(\theta) - h_2(\theta),$$

with

$$h_1(\theta) = \mathbb{E}_{\mu_Y}\left[\xi(x,a)\big(r(x,a,z) + \gamma \max_{b\in\mathcal{A}} \xi^\top(z,b)\theta\big)\right]$$

and

$$h_2(\theta) = \mathbb{E}_{\mu_Y}\left[\xi(x,a)\xi^\top(x,a)\theta\right].$$

Some simple calculations lead to the conclusion that

$$\|h_1(\theta_1) - h_1(\theta_2)\|_\infty \leq \gamma \|\theta_1 - \theta_2\|_\infty \tag{F.6}$$

and

$$\|h_2(\theta_1) - h_2(\theta_2)\|_\infty \leq \|\theta_1 - \theta_2\|_\infty . \tag{F.7}$$

On the other hand, suppose that $\{\theta_t^1\}$ and $\{\theta_t^2\}$ are two trajectories of the ODE starting in different initial conditions. Explicit calculations now yield

$$\begin{aligned}\frac{d}{dt}\left\|\theta_t^1 - \theta_t^2\right\|_p = \left\|\theta_t^1 - \theta_t^2\right\|_p^{1-p} \sum_i (\theta_t^1(i) - \theta_t^2(i))^{p-1} \cdot \\ \cdot (h_1(\theta_t^1)_i - h_1(\theta_t^2)_i) - \\ - \left\|\theta_t^1 - \theta_t^2\right\|_p^{1-p} \sum_i (\theta_t^1(i) - \theta_t^2(i))^{p-1} \cdot \\ \cdot (h_2(\theta_t^1)_i - h_2(\theta_t^2)_i),\end{aligned}$$

where we denoted by $h_1(\theta)_i$ the i^{th} component of $h_1(\theta)$ and similarly for h_2. Applying Hölder's inequality to the summations leads to

$$\frac{d}{dt}\left\|\theta_t^1 - \theta_t^2\right\|_p \leq \left\|h_1(\theta_t^1) - h_1(\theta_t^2)\right\|_p - \left\|h_2(\theta_t^1) - h_2(\theta_t^2)\right\|_p .$$

Taking the limit as $p \to \infty$ and using (F.6) and (F.7) leads to

$$\frac{d}{dt}\left\|\theta_t^1 - \theta_t^2\right\|_\infty \leq \gamma \left\|\theta_t^1 - \theta_t^2\right\|_\infty - \left\|\theta_t^1 - \theta_t^2\right\|_\infty$$

which, in turn, leads to

$$\frac{d}{dt}\left\|\theta_t^1 - \theta_t^2\right\|_\infty \leq (\gamma - 1)\left\|\theta_t^1 - \theta_t^2\right\|_\infty . \tag{F.8}$$

Let $\lambda = 1 - \gamma > 0$. Upon integration, (F.8) becomes

$$\left\|\theta_t^1 - \theta_t^2\right\|_\infty \leq e^{-\lambda t}\left\|\theta_0^1 - \theta_0^2\right\|_\infty .$$

This means that all trajectories of the algorithm approach exponentially fast and the ODE is globally exponentially stable. □

By further noticing that the ODE (F.5) is autonomous (does not depend explicitly on t), we can conclude (by taking $\theta_t^2 = \theta_{t+T}^1$ in the reasoning above) that no periodic orbits can exist, which establishes the existence of a globally asymptotically stable equilibrium point for (F.5). The fact that the ODE has a globally asymptotically stable equilibrium is sufficient to ensure the existence of a function $\mathcal{U}$ verifying the conditions of Theorem D.1.1,[2] which in turn establishes the convergence of the sequence $\{\theta_t\}$ w.p.1.

Limit of convergence

An equilibrium point θ^* for the ODE above verifies $h(\theta^*) = 0$. By explicitly writing h, it is clear that $h(\theta^*) = 0$ is equivalent to

$$\mathbb{E}_{\mu_Y}\left[\xi(x,a)\big(r(x,a,z) + \gamma \max_{b\in\mathcal{A}} \xi^\top(z,b)\theta\big)\right] = \\ = \mathbb{E}_{\mu_Y}\left[\xi(x,a)\xi^\top(x,a)\theta\right]$$

which in turn leads to

$$\theta^* = \Sigma^{-1}\mathbb{E}_{\mu_Y}\left[\xi(x,a)\big(r(x,a,z) + \gamma \max_{b\in\mathcal{A}} \xi^\top(z,b)\theta\big)\right].$$

Therefore, the sequence $\{\theta_t\}$ generated by (4.15) converges w.p.1 to the limit point θ^* verifying the following recursive relation:

$$\theta^* = \mathcal{P}\mathbf{H}Q(\theta^*)$$

or, more explicitly,

$$\theta^* = \Sigma^{-1}\mathbb{E}_Y\left[\xi(x,a)(\mathbf{H}Q_{\theta^*})(x,a)\right], \tag{F.9}$$

where $Q(\theta)$ is the function

$$Q_\theta(x,a) = \xi^\top(x,a)\theta.$$

The purpose of (4.15) is to approximate the optimal Q-function Q^* by an element of the linear space $\mathcal{Q}$ spanned by the basis functions $\xi_i \in \Xi$. We would like to determine the element $q^* \in \mathcal{Q}$ that, in some sense, "best approximates" Q^*.

[2] This guarantee arises from standard converse Lyapunov theorems.

One possibility is to consider the orthogonal projection of Q^* on $\mathcal{Q}$. Denote the orthogonal projection operator on $\mathcal{Q}$ by $\mathcal{P}_{\mathcal{Q}}$.

Finding the projection of Q^* on $\mathcal{Q}$ translates into finding the function $f \in \mathcal{Q}$ verifying

$$f(x,a) = (\mathcal{P}_{\mathcal{Q}} Q^*)(x,a) = (\mathcal{P}_{\mathcal{Q}} \mathbf{H} Q^*)(x,a).$$

However, f thus defined is not a fixed point of any of the involved operators, and there is not an obvious procedure to write a stochastic approximation algorithm to find f. Therefore, we instead consider the function g verifying

$$g(x,a) = (\mathcal{P}_{\mathcal{Q}} \mathbf{H} g)(x,a). \tag{F.10}$$

The function g *is* a fixed point of the operator $\mathcal{P}_{\mathcal{Q}}\mathbf{H}$ and can easily be implemented using stochastic approximation. The projection $\mathcal{P}_{\mathcal{Q}}$ can be expressed as

$$(\mathcal{P}_{\mathcal{Q}} f)(x,a) = \xi^\top(x,a)\Sigma^{-1}\mathbb{E}_Y\left[\xi(z,u)f(z,u)\right]$$

for any function f. But this means that, given the limit point θ^* of algorithm (4.15), the corresponding function Q_{θ^*} verifies

$$\begin{aligned} Q_{\theta^*}(x,a) &= \xi^\top(x,a)(\mathcal{P}\mathbf{H}Q_{\theta^*}) = \\ &= \xi^\top(x,a)\Sigma^{-1}\mathbb{E}_Y\left[\xi(z,u)\mathbf{H}Q_{\theta^*}(z,u)\right] = \\ &= (\mathcal{P}_{\mathcal{Q}}\mathbf{H}Q_\theta)(x,a) \end{aligned}$$

where the second equality comes from (F.9). Finally, this implies that Q_{θ^*} verifies the fixed point equation in (F.10), which establishes statement (4.16) of Theorem 4.5.2.

This concludes the proof of Theorem 4.5.2. □

F.2.2 Proof of Theorem 4.5.3

To establish the convergence of approximate SARSA with function approximation, we once again resort to Theorem D.1.1 of Appendix D. We start by writing the update rule as

$$\begin{aligned} \theta_{t+1} = \theta_t &+ \alpha_{t+1} H^*(\theta_t, Y_{t+1}) + \\ &+ \alpha_{t+1}\big(\mathbb{E}\left[H_t(\theta_t, Y_{t+1}, A_{t+1})\right] - H^*(\theta_t, Y_{t+1})\big) + \\ &+ \alpha_{t+1}\big(H_t(\theta_t, Y_{t+1}, A_{t+1}) - \mathbb{E}\left[H_t(\theta_t, Y_{t+1}, A_{t+1})\right]\big). \end{aligned}$$

We consider the Markov chain $\{Y_t\}$ defined for each t as $Y_t = (X_{t-1}, A_{t-1}, X_t)$. We denote by $\mathcal{Y}$ the corresponding domain of definition, $\mathcal{Y} = \mathcal{X}\times\mathcal{A}\times\mathcal{X}$. The mappings H^* and H_t are given by

$$H^*(\theta, Y_{t+1}) = \xi(X_t, A_t)\big(r(X_t, A_t, X_{t+1}) + \gamma \max_{b\in\mathcal{A}} \xi^\top(X_{t+1}, b)\theta - \xi^\top(X_t, A_t)\theta\big);$$

$$H_t(\theta_t, Y_{t+1}, A_{t+1}) = \xi(X_t, A_t)\big(r(X_t, A_t, X_{t+1}) + \gamma\xi^\top(X_{t+1}, A_{t+1})\theta - \xi^\top(X_t, A_t)\theta\big).$$

We now define two auxiliary mappings H and C_t as

$$H(\theta, Y_{t+1}, A_{t+1}) = H^*(\theta_t, Y_{t+1}) + H_t(\theta_t, Y_{t+1}, A_{t+1}) - \mathbb{E}\left[H_t(\theta_t, Y_{t+1}, A_{t+1})\right]$$
$$C_t(\theta, Y_{t+1}) = \frac{1}{\alpha_t}\big(\mathbb{E}\left[H_t(\theta_t, Y_{t+1}, A_{t+1})\right] - H^*(\theta_t, Y_{t+1})\big),$$

and rewrite the update rule once again as

$$\theta_{t+1} = \theta_t + \alpha_{t+1} H(\theta_t, Y_{t+1}, A_{t+1}) + \alpha_t^2 C_t(\theta_t, Y_{t+1}).$$

We notice that, since the policy $(\delta_\theta)_t$ is admittedly GLIE, there must be a constant K_0 such that, w.p.1,

$$\|C_t(\theta, y)\| \leq K_0(1 + \|\theta\|). \tag{F.11}$$

In fact, since $(\delta_\theta)_t$ is GLIE, the action choice is greedy in the limit and, therefore

$$\mathbb{E}\left[H_t(\theta_t, Y_{t+1}, A_{t+1})\right] - H^*(\theta_t, Y_{t+1}) \to 0.$$

This, in turn, implies that

$$\|C_t(\theta, y)\| \leq K \frac{c_t}{\alpha_t} \|\theta\|$$

for some $K > 0$ and some sequence $c_t \to 0$ and (F.11) follows.[3]

On the other hand, although the policy $(\delta_\theta)_t$ now depends on the parameter vector θ, it is not hard to check that Propositions F.2.2 and F.2.3 still hold and that the ODE associated with approximate SARSA has a globally asymptotically stable equilibrium point θ^* verifying

$$\theta^* = \mathcal{P}\mathbf{H}Q_{\theta^*}.$$

Therefore, by Theorem D.1.1, approximate SARSA with function approximation converges to the same limit point of Q-learning with function approximation and statement (4.20) follows. □

F.2.3 Proof of Lemma 4.7.1

Notice that the fact that $(\mathcal{X}, \mathsf{P})$ is irreducible means that for any state $i \in \mathcal{X}$, there is $m > 0$ such that

$$\mathsf{P}^m(i, i^*) > 0.$$

On the other hand, since i^* is admittedly "observable", this means that if $X_t = i^*$ for some $t \in \mathcal{T}$, then $\pi_t(i) = \mathbb{I}_{i^*}(i)$. We will denote such belief by π^*.

The measure φ defined for any set $U \in \mathcal{B}(\mathbb{S}^n)$ as $\varphi(U) = \mathbb{I}_{\pi^*}(U)$ is an irreducibility measure for the chain $(\mathbb{S}^n, \bar{\mathsf{P}})$ since

$$\varphi(U) > 0 \Rightarrow \mathbb{P}_\pi\left[\tau_U < \infty\right] > 0,$$

for any $\pi \in \mathbb{S}^n$. Therefore, by Proposition B.2.1, there is an irreducibility measure

[3] As remarked in Chapter 3, it holds that $\lim c_t/\alpha_t = M$ for some constant M, since otherwise $(\delta_\theta)_t$ would not ensure sufficient exploration.

ψ on $\mathcal{B}(\mathbb{S}^n)$ and $(\mathbb{S}^n, \bar{\mathsf{P}})$ is ψ-irreducible. □

F.2.4 Proof of Lemma 4.7.2

We proceed by contradiction.

Suppose that under the conditions of the lemma the chain $(\mathbb{S}^n, \bar{\mathsf{P}})$ is periodic, with period $d > 1$. It is clear that if a d-cycle exists with $d > 1$, then there is a set D_k in the d-cycle such that $\pi^* \in D_k$, where π^* is as defined in the proof of Lemma 4.7.1. This means that $\mathsf{P}^m(i^*, i^*) > 0$ only if $m = nd + k$, for some $n \in \mathbb{N}$. This in turn means that i^* has period d and hence (since $(\mathcal{X}, \mathsf{P})$ is irreducible) so do all the states in $\mathcal{X}$. But this implies that the chain $(\mathcal{X}, \mathsf{P})$ is periodic with period d, which is false, by assumption. Then, $(\mathbb{S}^n, \bar{\mathsf{P}})$ must be aperiodic and the proof is complete. □

F.2.5 Proof of Lemma 4.7.3

To prove the desired result, we show that there is a petite set $C \in \mathcal{B}^+(\mathbb{S}^n)$ and that the chain $(\mathbb{S}^n, \bar{\mathsf{P}})$ is weak Feller. We then apply Theorem B.5.2 to establish $\mathbb{S}^n$ to be petite.

Notice, first of all, that since $(\mathcal{X}, \mathsf{P})$ is irreducible and aperiodic and $\mathcal{X}$ is finite, the chain $(\mathcal{X}, \mathsf{P})$ is ergodic and, consequently, geometrically ergodic.

Let π^* be as defined in the proof of Lemma 4.7.1. Take an arbitrary element $z \in \mathcal{Z}$ and define π_0 as

$$\pi_0(j) = \frac{\sum_{i \in \mathcal{X}} \pi^*(i) \mathsf{P}(i,j) \mathsf{O}(j,z)}{\sum_{i,k \in \mathcal{X}} \pi^*(i) \mathsf{P}(i,k) \mathsf{O}(k,z)},$$

i.e., π_0 is the belief state succeeding to π^* when observation z occurs. Since

$$\sum_i \pi_0(i) = 1,$$

this means that there is a state $j \in \mathcal{X}$ such that $\pi_0(j) > 0$. Let

$$J = \min_i \{\pi_0(i) \mid \pi_0(i) > 0\}$$

and

$$I = \arg\min_i \{\pi_0(i) \mid \pi_0(i) > 0\}.$$

Take any $0 < \varepsilon_1 < J$ and consider the set

$$C = \{\pi \in \mathbb{S}^n \mid \pi(I) > \varepsilon_1\}.$$

Clearly, the set C is open in $\mathbb{R}^n$ and, consequently, it is open in the subspace topology on $\mathbb{S}^n$. On the other hand, $\pi_0 \in C$.

We now show that C is ν_M-small, for some measure ν_M and some $M \in \mathbb{N}$.

Since the chain $(\mathcal{X}, \mathsf{P})$ is irreducible by assumption, there is $M > 0$ such that

$$\mathsf{P}^{M-1}(I, i^*) > 0.$$

Let $\varepsilon_2 = \mathsf{P}^{M-1}(I, i^*)$. Using the Chapman-Kolmogorov inequality, for any $\pi \in C$ and any set $U \in \mathcal{B}(\mathbb{S}^n)$,

$$\begin{aligned}\bar{\mathsf{P}}^M(\pi, U) &\geq \bar{\mathsf{P}}^{M-1}(\pi, \{\pi^*\})\bar{\mathsf{P}}(\pi^*, U) = \\ &= \sum_{j \in \mathcal{X}} \pi(j)\mathsf{P}^{M-1}(j, i^*)\bar{\mathsf{P}}(\pi^*, U) > \\ &> \varepsilon_1\varepsilon_2\bar{\mathsf{P}}(\pi^*, U).\end{aligned}$$

Then, the measure ν_M defined for each set $U \in \mathcal{B}(\mathbb{S}^n)$ as $\nu_M(U) = \varepsilon_1\varepsilon_2\bar{\mathsf{P}}(\pi^*, U)$ is a measure on $\mathcal{B}(\mathbb{S}^n)$ and C is ν_M-small and therefore petite. On the other hand, it is immediate to see that, since $\pi_0 \in C$, $\psi(C) > 0$.

We now show that the chain $(\mathbb{S}^n, \bar{\mathsf{P}})$ is a (weak) Feller chain.

Let f be a bounded continuous function defined on $\mathbb{S}^n$. Then, since $\mathbb{S}^n$ is compact and $\bar{\mathsf{P}}(\pi, \cdot)$ is a probability measure on $\mathcal{B}(()\,\mathbb{S}^n)$,

$$(\bar{\mathsf{P}}f)(\pi) = \int_{\mathbb{S}^n} \bar{\mathsf{P}}(\pi, d\omega) f(\omega) \leq \max_{\pi \in \mathbb{S}^n} |f|\,,$$

and $(\bar{\mathsf{P}}f)$ is bounded. On the other hand, notice that, for any $z \in \mathcal{Z}$, the n-dimensional function $\pi \to \Pi(\pi, z)$ given by

$$\Pi(\pi, z)_j = \frac{\sum_{i \in \mathcal{X}} \pi(i)\mathsf{P}(i, j)\mathsf{O}(j, z)}{\sum_{i,k \in \mathcal{X}} \pi(i)\mathsf{P}(i, k)\mathsf{O}(k, z)}, \tag{F.12}$$

is clearly continuous on π. Then, $f_z(\pi)$ defined for each $\pi \in \mathbb{S}^n$ and each $z \in \mathcal{Z}$ as

$$f_z(\pi) = f(\Pi(\pi, z)),$$

where π_z is given by (F.12) is the composition of two continuous functions and, hence, continuous. Finally,

$$(\bar{\mathsf{P}}f)(\pi) = \sum_z \sum_{i,j} \pi(i)\mathsf{P}(i, j)\mathsf{O}(j, z) f_z(\pi)$$

and it is clearly also a continuous function of π. Then, $\bar{\mathsf{P}}$ maps $\mathcal{C}(\mathbb{S}^n)$ into $\mathcal{C}(S^n)$ and Theorem B.5.1 ensures that $(\mathbb{S}^n, \bar{\mathsf{P}})$ is a weak Feller chain.

We have established $(\mathbb{S}^n, \bar{\mathsf{P}})$ to be a ψ-irreducible Feller chain and have found a petite set $C \in \mathcal{B}^+(\mathbb{S}^n)$. Theorem B.5.2 now ensures that all compact subsets of $\mathbb{S}^n$ are petite and, since $\mathbb{S}^n$ is itself compact, it is petite and the proof is complete. □

F.3 Proofs for Chapter 7

The proofs in this section belong to the results found in Chapter 7 of the main text. We also introduce the framework of generalized Markov decision processes by Szepesvári and Littman [299]. The use of this framework will greatly simplify several of the proofs in this section.

F.3.1 Generalized Markov decision processes

Generalized Markov decision processes are, in its essence, similar to MDPs as described in Chapter 2 and the methods featured are also similar to those described in Chapter 3. However, the use of generalized operators will allow us to formally cast team Markov games as a particular of a generalized MDP and thus use all the method available for the latter class of processes. The generalized MDP (GMDP) framework was introduced by Szepesvári and Littman [299] and further explored in [168, 300].

A *generalized Markov decision process* is also a tuple $(\mathcal{X}, \mathcal{A}, \mathsf{P}, r, \gamma)$, where $\mathcal{X}$, $\mathcal{A}$, P, r and γ are as in standard MDPs. For a matter of simplicity, we assume $\mathcal{X}$ to be finite, but all definitions and results hold *mutatis mutandis* if $\mathcal{X}$ is infinite. We will assume that P and r are unknown. A generalized MDP represents a *generalized decision-maker* trying to optimize its expected discounted cumulative reward, with a discount factor $0 < \gamma < 1$. We referred a *generalized decision-maker* since we make no a priori assumptions on the decision-making process. In particular, we do not restrict decision-making to be ruled by a *single* deciding entity, thus including the possibility of multiple decision-makers.

We also referred the *optimization* of the discounted cumulative reward, since we make no prior assumptions on the optimization process. In particular, we do not restrict this optimization to be a maximization, requiring it only to be *non-expansive* and to verify

$$\min_{a\in\mathcal{A}} f(x,a) \leq (\bigotimes f)(x) \leq \max_{a\in\mathcal{A}} f(x,a)$$

for any function $f : \mathcal{X} \times \mathcal{A} \to \mathbb{R}$, where we denoted the generalized optimization operator by $\bigotimes$.

As in standard MDPs, the objective of the generalized decision-maker is to optimize over all possible sequences of actions $\{A_t\}$ the infinite-horizon total discounted reward, defined as in (2.5). The difference is that the *optimal value function* V^* is now defined for each state $i \in \mathcal{X}$ as

$$V^*(i) = \bigotimes_{\{A_t\}} \mathbb{E}\left[\sum_{t=0}^{\infty} \gamma^t R(X_t, A_t) \mid X_0 = i\right]$$

and verifies the *generalized Bellman equation*

$$V^*(i) = \bigotimes_{a\in\mathcal{A}} \sum_j \mathsf{P}_a(i,j)\big[r(i,a,j) + \gamma V^*(j)\big].$$

The function Q^* is naturally defined from V^* as

$$Q^*(i,a) = \sum_j \mathsf{P}_a(i,j)\big[r(i,a,j) + \gamma V^*(j)\big]$$

and the optimal policy δ^* as

$$\delta^*(i) = \arg \bigotimes_{a\in\mathcal{A}} Q^*(i,a),$$

for all $i \in \mathcal{X}$. We once again emphasize that GMDPs and MDPs are, in its essence, equivalent formulations. However, the use of generalized operator $\bigotimes$ in GMDPs will prove very useful: by specifying the operator $\bigotimes$ we can handle MDPs and TMGs using a common formulation and the results in any of the two frameworks carries immediately to the other.

We now introduce the "generalized" versions of the ARTQI and Q-learning algorithms, described in their original versions in Chapter 3.

We start by generalized ARTQI. In generalized MDPs, the maximization is replaced by the generalized optimization operator $\bigotimes$. However, the expressions to estimate P and r remain the same as those introduced in Chapter 3. As in ARTQI, we apply the fixed point iteration for Q^* using $\hat{\mathsf{P}}_t$ and $\hat{r}_t$ instead of P and r:

$$Q_{t+1}(i_t,a_t) = \hat{\mathsf{P}}_t(i_t,a_t,j)\big(\hat{r}_t(i_t,a_t,j) + \gamma \bigotimes_{b\in\mathcal{A}} Q_t(j,b)\big),$$

The generalized version of Q-learning starts with any initial estimate Q_0, and uses the update

$$Q_{t+1}(i_t,a_t) = Q_t(i_t,a_t) + \alpha_t(i_t,a_t)\left(r_t + \gamma \bigotimes_{b\in\mathcal{A}} Q_t(i_{t+1},b) - Q_t(i_t,a_t)\right) \tag{F.13}$$

to generate a sequence $\{Q_t\}$ of estimates for the optimal Q-function.

Convergence of the sequences $\{Q_t\}$ obtained with both generalized ARTQI and generalized Q-learning can be established using Theorem D.1.4 in Appendix D. This is summarized in the following results.

Theorem F.3.1. *Given a generalized MDP $(\mathcal{X},\mathcal{A},\mathsf{P},r,\gamma)$, the sequence of estimates $\{Q_t\}$ generated by generalized ARTQI converges to Q^* w.p.1, as long as every state-action pair $(i,a) \in \mathcal{X}\times\mathcal{A}$ is visited infinitely often.*

PROOF The proof is similar to that of Theorem 3.3.2. We use Theorem D.1.4 of Appendix D.

Let B be the space of bounded functions defined on $\mathcal{X}\times\mathcal{A}$ and consider the

randomized operator $T_t : B \times B \longrightarrow B$ defined by

$$T_t(Q, Q')(i,a) = \begin{cases} \hat{\mathsf{P}}_t(i_t, a_t, j)\big(\hat{r}_t(i_t, a_t, j) + \gamma \bigotimes_{b\in\mathcal{A}} Q'(j,b)\big) & \text{if } (i,a) = (i_t, a_t); \\ Q(i,a) & \text{otherwise.} \end{cases}$$

We want to show that the operator T_t approximates the operator T defined as

$$(TQ)(i,a) = \mathsf{P}_a(i,j)\big[r(i,a,j) + \gamma \bigotimes_{b\in\mathcal{A}} Q^*(j,b)\big].$$

By assumption, all state-action pairs $(i,a) \in \mathcal{X} \times \mathcal{A}$ are visited infinitely often; furthermore, when a pair (i,a) is visited, $Q_{t+1}(i,a)$ does not depend on Q_t. Therefore, to show that T_t approximates T defined above, we need only to show that

$$\left| \hat{\mathsf{P}}_t(i,a,j)\big[\hat{r}_t(i,a,j) + \gamma \bigotimes_{b\in\mathcal{A}} Q'(j,b)\big] - \mathsf{P}_a(i,j)\big[r(i,a,j) + \gamma \bigotimes_{b\in\mathcal{A}} Q'(j,b)\big] \right| \to 0$$

for all pairs (i,a).

Repeating the computations in the proof of Theorem 3.3.1 yields

$$\begin{aligned} &\left| \hat{\mathsf{P}}_t(i,a,j)\big[\hat{r}_t(i,a,j) + \gamma \bigotimes_{b\in\mathcal{A}} Q'(j,b)\big] - \mathsf{P}_a(i,j)\big[r(i,a,j) + \gamma \bigotimes_{b\in\mathcal{A}} Q'(j,b)\big) \right| \leq \\ &\qquad \leq \max_{i,a,j} |\hat{r}_t(i,a,j) - r(i,a,j)| + \max_{i,a} \left| [p_t(i,a,j) - \mathsf{P}_a(i,j)] \left[r(i,a,j) + \gamma \bigotimes_{b\in\mathcal{A}} Q'(j,b) \right] \right|. \end{aligned}$$

The law of large numbers ensures that $\hat{\mathsf{P}}_t \to \mathsf{P}$ and, clearly, $\hat{r}_t \to r$. Thus, the last term in the previous expression goes to zero as $t \to \infty$ and T_t approximates T.

Defining $f_t(i,a)$ and $g_t(i,a)$ as

$$f_t(i,a) = \begin{cases} \gamma & \text{if } (i,a) = (i_t, a_t); \\ 0 & \text{otherwise} \end{cases}$$

and

$$g_t(i,a) = \begin{cases} 0 & \text{if } (i,a) = (i_t, a_t); \\ 1 & \text{otherwise,} \end{cases}$$

all conditions of Theorem D.1.4 are verified and the conclusion of the theorem follows. □

Theorem F.3.2. *Given a generalized MDP $(\mathcal{X}, \mathcal{A}, \mathsf{P}, r, \gamma)$, the sequence of estimates $\{Q_t\}$ generated by generalized Q-learning converges to Q^* w.p.1 for any initial estimate Q_0, as long as*

$$\sum_{t\in\mathcal{T}} \alpha_t(i,a) = \infty; \qquad \sum_{t\in\mathcal{T}} \alpha_t^2(i,a) < \infty,$$

and $\alpha_t(i,a) = 0$ *if* $(i,a) \neq (i_t, a_t)$.

PROOF We use Theorem D.1.4 of Appendix D.

Let B be the space of bounded functions defined on $\mathcal{X} \times \mathcal{A}$ and consider the randomized operator $T_t : B \times B \longrightarrow B$ defined by

$$T_t(Q,Q')(i,a) = \begin{cases} (1-\alpha_t(i_t,a_t))Q(i_t,a_t) + \alpha_t(i_t,a_t)\big(r(i_t,a_t,i_{t+1}) + \gamma \bigotimes_{b\in\mathcal{A}} Q'(i_{t+1},b)\big) & \text{if } (i,a) = (i_t,a_t); \\ Q(i,a) & \text{otherwise.} \end{cases}$$

Further define the functions $f_t(i,a)$ and $g_t(i,a)$ to be

$$f_t(i,a) = \begin{cases} \gamma\alpha_t(i_t,a_t) & \text{if } (i,a) = (i_t,a_t); \\ 0 & \text{otherwise} \end{cases}$$

and

$$g_t(i,a) = \begin{cases} (1-\alpha_t(i_t,a_t)) & \text{if } (i,a) = (i_t,a_t); \\ 1 & \text{otherwise.} \end{cases}$$

The restrictions imposed on the sequence of step-sizes $\{\alpha_t\}$ ensures that these functions verify the conditions of Theorem D.1.4.

It remains only to show that the operator T_t defined above approximates T at Q^*. To see this, notice that, given any initial condition $q_0 \in B$, the sequence $\{q_t\}$ generated by the recursion $q_{t+1}(i,a) = T_t(q_t, Q^*)(i,a)$ verifies

$$q_{t+1}(i_t,a_t) = (1-\alpha_t(i_t,a_t))q(i_t,a_t) + \alpha_t(i_t,a_t)\big(r(i_t,a_t,i_{t+1}) + \gamma \bigotimes_{b\in\mathcal{A}} Q^*(i_{t+1},b)\big).$$

Subtracting $Q^*(i_t,a_t)$ on both sides and defining $\Delta_t(i,a) = q_t(i,a) - Q^*(i,a)$ yields

$$\Delta_{t+1}(i_t,a_t) = (1-\alpha_t(i_t,a_t))\Delta(i_t,a_t) + \alpha_t(i_t,a_t)\big(r(i_t,a_t,i_{t+1}) + \gamma \bigotimes_{b\in\mathcal{A}} Q^*(i_{t+1},b) - Q^*(i_t,a_t)\big).$$

Repeating the steps in the proof of Theorem 3.4.2, we apply Lemma D.1.3 of Appendix D to ensure that $\Delta_t \to 0$ w.p.1 or, equivalently, $q_t \to Q^*$ w.p.1. Notice, in particular, that the inequality

$$\left\| \bigotimes_{b\in\mathcal{A}} Q_t(j,b) - \bigotimes_{b\in\mathcal{A}} Q^*(j,b) \right\| \leq \|\Delta_t\|$$

holds since $\bigotimes$ is admittedly a non-expansion.

Then, by Theorem D.1.4, the conclusion of the Theorem follows. □

In the next proofs we take advantage of the fact that TMGs can be recast as generalized MDPs to trivially extend the proofs from Chapter 3.

F.3.2 Proof of Theorem 7.2.1

To prove Theorem 7.2.1 we make use of Theorem F.3.1 above by reducing a TMG to a GMDP. To this we only need to show that the update in (7.3) can be written in terms of a generalized operator $\bigotimes$ verifying the conditions above. By simple inspection, we conclude that

$$\bigotimes_{a\in\mathcal{A}} f(i,a) = \max_{a\in\mathcal{A}} f(i,a)$$

that clearly verifies

$$\left\| \bigotimes_{a\in\mathcal{A}} f - \bigotimes_{a\in\mathcal{A}} g \right\|_\infty \leq \|f-g\|_\infty$$

and

$$\min_{a\in\mathcal{A}} f(x,a) \leq (\bigotimes f)(x) \leq \max_{a\in\mathcal{A}} f(x,a).$$

This means that TMGs are generalized MDPs with

$$\bigotimes_{a\in\mathcal{A}} f(i,a) = \max_{a\in\mathcal{A}} f(i,a)$$

and the conclusion follows from Theorem F.3.1. □

F.3.3 Proof of Theorem 7.2.2

As seen in the proof of Theorem 7.2.1, TMGs are generalized MDPs with

$$\bigotimes_{a\in\mathcal{A}} f(i,a) = \max_{a\in\mathcal{A}} f(i,a)$$

and the conclusion of the Theorem follows immediately from Theorem F.3.2. □

F.3.4 Proof of Lemma 7.3.1

The proof closely follows the proof of Lemma 6 in [330].

Let $i \in \mathcal{X}$ be some fixed state and let λ_T be the event that, for all $t > T$,

$$\max_{a\in\mathcal{A}} |Q_t(i,a) - Q^*(i,a)| < \frac{K_0}{2} B(N_t).$$

Since, by assumption,

$$\lim_{t\to\infty} \frac{\mathsf{rate}(N_t)}{B(N_t)} = 0,$$

it holds that

$$\lim_{t\to\infty} \frac{K_1 \mathsf{rate}(N_t)}{K_0 B(N_t)} = 0,$$

for any positive constant K_1. Therefore, and since, by hypothesis,

$$\|Q_t - Q^*\| \leq K_0 \mathsf{rate}(N_t)$$

w.p.1, given any $\rho > 0$ there is a time instant $T_0(\rho) > 0$ such that, for all $t > T_0$,

$$\mathbb{P}\left[\lambda_t\right] > 1 - \rho. \tag{F.14}$$

Take now two actions $a, b \in \mathcal{A}$ such that $a \in \mathbf{opt}(i)$ and b verifies

$$b = \arg\max_{u \notin \mathbf{opt}(i)} Q^*(i, u),$$

where we used the compact notation $\mathbf{opt}(i)$ to represent $\mathbf{opt}^0(i)$. Define the quantity $\delta = |Q^*(i, a) - Q^*(i, b)|$. By assumption, $B(N_t) \to 0$ and, therefore, there is a time instant T_1 such that, for all $t > T_1$,

$$K_0 B(N_t) < \frac{\delta}{2}. \tag{F.15}$$

Let $T = \max\{T_0, T_1\}$. For all $t > T$ it holds with probability $p > 1 - \rho$ that, given any action $b \notin \mathbf{opt}(i)$,

$$\begin{aligned} Q_t(i, b) + K_0 B(N_t) &< Q^*(i, b) + K_0 B(N_t) + \frac{K_0}{2} B(N_t) < \\ &< Q^*(i, b) + \frac{\delta}{2} + \frac{\delta}{4} \leq \\ &\leq \max_{u \in \mathcal{A}} Q^*(i, u) - \frac{\delta}{4} < \\ &< \max_{u \in \mathcal{A}} Q_t(i, u). \end{aligned} \tag{F.16}$$

The first inequality arises from (F.14); the second inequality arises from (F.15); the third inequality arises from the definition of δ and the last inequality arises from (F.14) once again. On the other hand, for all $t > T$ it holds with probability $p > 1 - \rho$ that, given any action $a \in \mathbf{opt}(i)$,

$$Q_t(i, a) + K_0 B(N_t) > Q^*(i, a) + \frac{K_0}{2} B(N_t) > \max_{u \in \mathcal{A}} Q_t(i, u). \tag{F.17}$$

The first inequality arises from (F.14) and the second inequality from (F.15).

By construction, we have that $\varepsilon_t \leq \varepsilon_0 B(N_t)$, and hence $\varepsilon_t \leq K_0 B(N_t)$ as long as $\varepsilon_0 \leq K_0$. Then, joining (F.16) and (F.17), it holds with probability $p > 1 - \rho$ that, for all $t > T$,

$$\begin{aligned} Q_t(i, b) &< \max_{u \in \mathcal{A}} Q_t(i, u) - \varepsilon_t \\ Q_t(i, a) &> \max_{u \in \mathcal{A}} Q_t(i, u) - \varepsilon_t, \end{aligned}$$

for any actions $a \in \mathbf{opt}^{\varepsilon_t}(i)$ and $b \notin \mathbf{opt}^{\varepsilon_t}(i)$. The first expression implies that,

for any $t > T$, no suboptimal action belongs to $\mathbf{opt}^{\varepsilon_t}(i)$; the second expression implies that all optimal actions do belong to $\mathbf{opt}^{\varepsilon_t}(i)$. This means that, for all $t > T$, $VG_t = VG$ with probability $p > 1 - \rho$ and, therefore, $\mathbb{P}[\Lambda_T] > 1 - \rho$. The conclusion of the theorem follows. □

F.3.5 Proof of Theorem 7.3.2

We repeat the proof of Theorem 7 from [330]. We will need an intermediate result also from [330] that we reproduce here for the sake of completeness. This lemma generalizes the conditions for convergence of BAP presented in Theorem C.7.2 of Appendix C. In the following lemma, K is the length of the K-samples and $L(\Gamma)$ is as defined in Appendix C, page 282.

Lemma F.3.3. *Let $\Gamma = \big(N, (\mathcal{A}^k), r\big)$ be a fully cooperative, weakly acyclic game w.r.t. some bias set D. If each player $k \in N$ follows an individual strategy with the GLIE property and*

$$K \leq \frac{m}{L(\Gamma) + 2},$$

then BAP converges w.p.1 to either a strict Nash equilibrium or a Nash equilibrium in D.

Proof See [330]. □

Convergence of Q_t to Q^* arises as an immediate consequence of Theorem 7.2.1, since the GLIE policy ensures sufficient exploration.

We now establish converge in behavior to a coordinated equilibrium in the TMG $\big(N, \mathcal{X}, (\mathcal{A}^k), \mathsf{P}, r, \gamma\big)$ by establishing convergence in behavior to a coordinated equilibrium in the state-game Γ_i^*, for any state $i \in \mathcal{X}$. This, as seen, is sufficient to ensure convergence of OAL for all $i \in \mathcal{X}$ due to the assumption of infinite exploration under the GLIE policy.

We start by remarking that the rate of convergence for model-based learning has been shown to verify the law of iterated logarithm [137, 330]. This means that, w.p.1,

$$\max_{a \in \mathcal{A}} |Q_t(i,a) - Q^*(i,a)| \leq \|Q_t - Q^*\|_\infty \leq K_0 \sqrt{\frac{\log\log(N_t)}{N_t}},$$

for some positive constant K_0.

Take some state $i \in \mathcal{X}$ and let Λ_T be the event that, for all $t > T$, $VG_t(i,\cdot) = VG(i,\cdot)$ for some (and hence all) agent $k \in N$. Since the conditions of Lemma 7.3.1 are verified by hypothesis, it holds that, given any $\rho_1 > 0$ there is $T_1(\rho_1)$ such that

$$\mathbb{P}[\Lambda_t] > 1 - \rho_1$$

for all $t > T_1$.

Suppose now that Λ_{T_1} occurs for some $T_1 > 0$ and let $\Gamma = \big(N, (\mathcal{A}^k), Q^*(i,\cdot)\big)$. As shown in [330], $L(\Gamma) < N$; by Lemma F.3.3, BAP converges w.p.1 to a strict Nash equilibrium or a Nash equilibrium in D. By construction, there are no strict Nash equilibria outside D and all equilibria in D are coordinated equilibria. Therefore, if Λ_{T_1} occurs, BAP converges w.p.1 to a coordinated Nash equilibrium w.p.1. In other words, there is a time $T_2(\rho_2, T)$ such that, for any $t > T_2$,

$$\mathbb{P}\left[\lambda_t \mid \Lambda_T\right] > 1 - \rho_2,$$

where ρ_2 is an arbitrary positive constant and λ_T is the event that, for $t > T$, all agents play an optimal joint action at state i.

All the reasoning so far implies that there is a time instant $T_3(\rho_1, \rho_2)$ such that, for all $t > T_3$,

$$\mathbb{P}\left[\lambda_t\right] > \mathbb{P}\left[\lambda_t \mid \Lambda_t\right]\mathbb{P}\left[\Lambda_t\right] = (1-\rho_1)(1-\rho_2) > 1 - \rho_1 - \rho_2.$$

Since ρ_1 and ρ_2 are arbitrary, the conclusion of the theorem follows. □

F.3.6 Proof of Theorem 7.3.3

This proof is basically similar to that of Theorem 7.3.2.

Convergence of Q_t to Q^* arises as an immediate consequence of Corollary 7.2.2, since the GLIE policy ensures sufficient exploration and the step-sizes verify the conditions of the corollary.

Converge in behavior is established by by noticing that Q-learning has been shown to verify w.p.1 the following error bound [137, 298]:

$$\max_{a\in\mathcal{A}} |Q_t(i,a) - Q^*(i,a)| \leq \|Q_t - Q^*\|_\infty \leq K_0\sqrt{\frac{\log\log(N_t)}{N_t}},$$

for some positive constant K_0. Convergence in behavior now follows from the exact same steps used in the proof of Theorem 7.3.2. □

F.4 Proofs for Chapter 8

The proofs in this section belong to the theorems found in Chapter 8 of the main text.

F.4.1 Proof of Theorem 8.2.1

Before getting into the proof of Theorem 8.2.1, and for the sake of completeness, we provide several intermediate results that have been assumed as certain in the main text but that have not been formally established. These results will gradually build up to Theorem 8.2.1.

We start by establishing the existence of an optimal *deterministic* policy in infinite MDPs.

Lemma F.4.1. *For every Markov decision process $M = (\mathcal{X}, \mathcal{A}, \mathsf{P}, r, \gamma)$ with finite action-space $\mathcal{A}$ and compact state-space $\mathcal{X} \subset \mathbb{R}^p$ there is an optimal deterministic policy.*

Proof Suppose that given a non-deterministic, non-stationary policy δ_t is optimal. The optimal value function is given, at any state $x \in \mathcal{X}$, by

$$V_0^*(x) = \mathbb{E}_{\delta_t}\left[\sum_{t=0}^{\infty} \gamma^t R(X_t, A_t) \mid X_0 = x\right].$$

We remark that the index 0 in V_0^* refers to the initial time instant. The use of this index will become apparent in what follows. Isolating the first reward $R(X_0, A_0)$ leads to

$$V_0^*(x) = \mathbb{E}_{\delta_t}\left[R(X_0, A_0) \mid X_0 = x\right] + \mathbb{E}_{\delta_t}\left[\gamma \sum_{t=0}^{\infty} \gamma^t R(X_{t+1}, A_{t+1}) \mid X_0 = x\right].$$

If we explicitly write the expectation with respect to the policy δ_t, we get

$$\begin{aligned} V_0^*(x) = &\sum_{a\in\mathcal{A}} \delta_0(x,a) \int_{\mathcal{X}} r(x,a,y) \mathsf{P}_a(x,dy) + \\ &+ \gamma \sum_{a\in\mathcal{A}} \delta_0(x,a) \int_{\mathcal{X}} \mathbb{E}_{\delta_t}\left[\sum_{t=0}^{\infty} \gamma^t R(X_{t+1}, A_{t+1}) \mid X_1 = y\right] \mathsf{P}_a(x,dy) \end{aligned}$$

which, by noticing the term between square brackets to be $V_1^*(y)$ (observe the time index), leads to

$$V_0^*(x) = \sum_{a\in\mathcal{A}} \delta_0(x,a) \int_{\mathcal{X}} \left[r(x,a,y) + \gamma V_1^*(y)\right] \mathsf{P}_a(x,dy).$$

Let

$$b = \arg\max_{a\in\mathcal{A}} \int_{\mathcal{X}} \left[r(x,a,y) + \gamma V_1^*(y)\right] \mathsf{P}_a(x,dy). \tag{F.18}$$

Clearly, we have that

$$\int_{\mathcal{X}} \left[r(x,b,y) + \gamma V_1^*(y)\right] \mathsf{P}_b(x,dy) \geq \sum_{a\in\mathcal{A}} \delta_0(x,a) \int_{\mathcal{X}} \left[r(x,a,y) + \gamma V_1^*(y)\right] \mathsf{P}_a(x,dy),$$

since $\delta_0(x,a) \leq 1$ and $\sum_{a\in\mathcal{A}} \delta_0(x,a) = 1$. But then, the deterministic policy δ_0^* defined for every $x \in \mathcal{X}$ by

$$\delta_0^*(x) = \arg\max_{a\in\mathcal{A}} \int_{\mathcal{X}} \left[r(x,a,y) + \gamma V_1^*(y)\right] \mathsf{P}_a(x,dy)$$

must be optimal. Repeating the same reasoning for every $t \in \mathcal{T}$, the conclusion of

the lemma follows. □

REMARK: Notice that the action b in (F.18) is well defined since $\mathcal{A}$ is considered finite. This finiteness assumption could be relieved by considering a compact set $\mathcal{A}$ and continuous r and P. Also the compactness condition on $\mathcal{X}$ is not necessary. However, for the purposes of the thesis, we have no need for the more general the result and hence its current formulation. ◇

We now establish the existence of a optimal *stationary* policy in infinite MDPs.

Lemma F.4.2. *For every Markov decision process $M = (\mathcal{X}, \mathcal{A}, \mathsf{P}, r, \gamma)$ with finite action-space $\mathcal{A}$ and compact state-space $\mathcal{X} \subset \mathbb{R}^p$ there is an optimal stationary policy.*

PROOF From the proof of Lemma F.4.1, the policy δ_t defined at each time instant as

$$\delta_t(x) = \arg\max_{a \in \mathcal{A}} \int_{\mathcal{X}} \left[r(x, a, y) + \gamma V^*_{t+1}(y) \right] \mathsf{P}_a(x, dy) \tag{F.19}$$

is optimal. We would like to show that this policy is stationary or, which amounts to the same thing, that $V^*_t(x)$ is independent of t. We establish this last fact by contradiction.

Let δ^*_t be some optimal policy. The definition of the optimal value function for two consecutive time instants T and $T+1$ is

$$V^*_T(x) = \mathbb{E}_{\delta^*_t} \left[\sum_{t=0}^{\infty} \gamma^t R(X_{T+t}, A_{T+t}) \mid X_T = x \right]$$
$$V^*_{T+1}(x) = \mathbb{E}_{\delta^*_t} \left[\sum_{t=0}^{\infty} \gamma^t R(X_{T+t+1}, A_{T+t}) \mid X_{T+1} = x \right].$$

Suppose that $V^*_T(y) \neq V^*_{T+1}(y)$ for some $y \in \mathcal{X}$. This can only occur due to the time-differences in the policy δ^*_t, for if δ^*_t were time independent, the two previous definitions would agree in every point and $V^*_T(x) = V^*_{T+1}(x)$.

Define now two policies δ'_t and δ''_t as follows. For every $x \in \mathcal{X}$,

$$\delta'_t(x) = \delta^*_{t+1}(x)$$

and

$$\delta''_t(x) = \delta^*_{t-1}(x).$$

By definition, we have that $V^{\delta'_t}_T(x) = V^*_{T+1}(x)$ and $V^{\delta''_t}_{T+1}(x) = V^*_T(x)$ for every $x \in \mathcal{X}$. Now, one of two things can happen:

1) $V_T^*(y) < V_{T+1}^*(y)$. But then, $V_T^{\delta_t'}(y) > V_T^*(y)$ which is a contradiction since δ_t^* is optimal and therefore maximizes the expected discounted total reward in every point.

2) $V_T^*(y) > V_{T+1}^*(y)$. But then, $V_{T+1}^{\delta_t''}(y) > V_{T+1}^*(y)$ which is also a contradiction for the same reason.

This implies that $V_t^*(x)$ does not depend on t and the policy δ_t^* defined in (F.19) is stationary. □

With the two previous lemmas we established the existence of an optimal, stationary and deterministic policy in infinite MDPs. We now proceed with the proof of Theorem 8.2.1. To that purpose, we start by proving that every team Markov game has an associated MDP. The existence of a pure stationary equilibrium in the game is established by showing every deterministic, stationary policy in the associated MDP to constitute one such equilibrium for the game.

Let $\Gamma = \big(N, \mathcal{X}, (\mathcal{A}^k), \mathsf{P}, r, \gamma\big)$ be a team Markov game with compact state-space $\mathcal{X} \subset \mathbb{R}^p$. Suppose that, at each time instant $t \in \mathcal{T}$, each player $k \in N$ chooses its action a^k according to some criterion provided by a centralized decision-maker. From this perspective, we can look at the game Γ as an MDP $(\mathcal{X}, \mathcal{A}, \mathsf{P}, r, \gamma)$ where $\mathcal{X}$, P and r coincide and $\mathcal{A} = \times_{k=1}^N \mathcal{A}^k$.[4] Since the set $\mathcal{A}$ is the cartesian product of N sets $\mathcal{A}^1, \ldots, A^N$, each element $a \in \mathcal{A}$ is a N-tuple $(a^1, \ldots, a^k)$ each a^k being an element of $\mathcal{A}^k$. Define, for each $k = 1, \ldots, N$ a map $\phi^k : \mathcal{A} \longrightarrow \mathcal{A}^k$ assigning to each action $a \in \mathcal{A}$ its k^{th} component a^k.

Lemmas F.4.1 and F.4.2 ensure that there is a deterministic, stationary optimal policy δ^*. We now show that this policy is a pure, stationary Nash equilibrium in Γ. For each $k \in N$, define the individual strategy $\sigma^k(x)$ as

$$\sigma^k(x) = \phi^k(\delta^*(x)),$$

for all $x \in \mathcal{X}$. Clearly, the joint strategy $\sigma = (\sigma^1, \ldots, \sigma^N)$ is simply a "distributed" version of δ^* and, therefore,

$$V^\sigma(x) = \max_{\{A_t\}} \mathbb{E}\left[\sum_{t=0}^{\infty} \gamma^t R(X_t, A_t) \mid X_0 = x\right].$$

It is immediate that σ must be a Nash equilibrium. In fact, given any individual strategy $(\sigma^k)'$,

$$V^\sigma(x) \geq V^{(\sigma^{-k}, (\sigma^k)')}(x)$$

for all $x \in \mathcal{X}$ and any $k \in N$. In the other hand, since δ^* is stationary, so is σ. And since δ^* is deterministic, σ is pure. This concludes the proof of the theorem. □

[4] This interpretation is the converse of that described in Section 8.2 to introduce Markov games with infinite state-spaces.

F.4.2 Proof of Theorem 8.3.1

Consider the set of N players to be a unique (distributed) decision-maker following a fixed policy δ given by

$$\delta(x, a) = \sigma(x, a),$$

for every state-action pair (x, a). Then, the game Γ can be recast as an MDP $(\mathcal{X}, \mathcal{A}, \mathsf{P}, r, \gamma)$, where $\mathcal{A} = \times_{k=1}^{N} \mathcal{A}^k$, and the theorem is simply a restatement of Theorem 4.5.2. □

F.4.3 Proof of Theorem 8.4.1

The proof of this proposition will require several intermediate results before being properly established. To simplify the exposition, we start by considering the case where Q^* is continuous in x. We then trivially extend this result to the case where Q^* is continuous ψ-a.e. We denote by μ^{Leb} the Lebesgue measure in $\mathbb{R}^p$.

Suppose then that Q^* is continuous in x. This means that each function $V_a^*(x) = Q^*(\cdot, a)$ is continuous for each fixed $a \in \mathcal{A}$. We now partition $\mathcal{X}$ in two sets C and D and analyze the properties of each of these sets.

Take an arbitrary point $x \in \mathcal{X}$ and an arbitrary action $a_0 \in \mathcal{A}$. Then, one of two statements below holds:

1. $V_{a_0}^*(x) < V^*(x)$. If this is the case, due to the continuity of V^* and $V_{a_0}^*$, the inequality above holds for some neighborhood U of x. In other words, there is a neighborhood U of x such that

   $$V_{a_0}^*(y) < V^*(y),$$

   for all $y \in U$.

   This has an interesting implication: for every point $x \in \mathcal{X}$ there is a neighborhood U such that

   $$\mathbf{opt}(y) \subset \mathbf{opt}(x), \tag{F.20}$$

   for all $y \in U$.

2. $V_{a_0}^*(x) = V^*(x)$. If this is the case, two possible situations can occur:

   (a) There is a neighborhood U of x such that $a_0 \in \mathbf{opt}(y)$ for all $y \in U$;

   (b) Given any neighborhood U of x there is a point $y \in U$ such that

   $$a_0 \notin \mathbf{opt}(y); \tag{F.21}$$

Denote by $D(a_0)$ the set of points $x \in \mathcal{X}$ verifying 2b and define the sets $D = \bigcup_{a \in \mathcal{A}} D(a)$ and $C = \mathcal{X} - D$. We now show that

Lemma F.4.3. *Given the sets C and D defined above, $D = \partial C$.*

Proof From 1 and 2a, we see that a point $x \in C$ has a neighborhood U such that $U \cap D = \emptyset$. Then $C = \mathbf{int}\,(C)$ and since $D = \mathcal{X} - C$, $\partial C \subset D$. Since C and D are complementary and $C = \mathbf{int}\,(C)$, the conclusion of the lemma follows. □

Since $D = \partial C$, it is immediate that D is closed in $\mathcal{X}$ and therefore measurable. In turn, C must be open (in the subspace topology) and also measurable. We now proceed with the following result.

Lemma F.4.4. *The set D defined above verifies $\mu^{\text{Leb}}(D) = 0$.*

Proof Since we take the function Q^* to be continuous in x, the function

$$V^*(x) = \max_{a \in \mathcal{A}} Q^*(x, a)$$

is also continuous and so is the function

$$V^a(x) = Q^*(x, a),$$

defined for some $a \in \mathcal{A}$. We define a new function G^a as

$$G^a(x) = V^a(x) - V^*(x).$$

Clearly, G^a is continuous and $G^a(x) \leq 0$ for all $x \in \mathcal{X}$. We will show the set of points

$$\zeta_{G^a} = \{x \in \mathcal{X} \mid G^a(x) < 0\}$$

to be a p-dimensional topological manifold. Clearly, such set is a subset of $\mathbb{R}^p$ and, hence, Hausdorff and second countable (in the subspace topology). On the other hand, any point $x \in \zeta_{G^a}$ has a neighborhood $U \subset \zeta_{G^a}$, due to the continuity of G^a. This neighborhood is a neighborhood in $\mathbb{R}^p$ and therefore ζ_{G^a} is locally Euclidean and a topological manifold of dimension p. Its boundary is a manifold of dimension $p - 1$ and its Lebesgue measure is therefore zero.

By construction, we have that

$$\partial C \subset \bigcup_{a \in \mathcal{A}} \partial \zeta_{G^a},$$

and the conclusion follows. □

We can now proceed with the proof of Theorem 8.4.1. For each point $x \in C$, there is a neighborhood U such that

$$\mathbf{opt}(x) = \mathbf{opt}(y),$$

for all $y \in U$. This can be seen by noticing that a point in C either verifies Condition 1 or Condition 2a for every action $a \in \mathcal{A}$. Therefore, given one such point x and corresponding neighborhood U, it is immediate that the virtual game obtained

by setting to 1 all optimal actions and to 0 all non-optimal actions is the same in every point in U. This implies that, if $\psi(U) > 0$, there is a time T_0 such that, w.p.1, $S_m(x, H_t) \subset U$ for $t > T_0$. Since, for all $t > T_0$ all K-samples are drawn from $S_m(x, H_t)$, convergence of standard BAP ensures that, for all points in C, ABAP coordinates in an optimal Nash equilibrium w.p.1.

Notice that, if the irreducibility measure ψ is absolutely continuous w.r.t. the Lebesgue measure, *i.e.*, it has a Radon-Nikodym derivative w.r.t. μ^{Leb}, then we can use Lemma F.4.4 to immediately establish the conclusion of Theorem 8.4.1 or, in general, if $\psi(D) = 0$ (notice that the former implies the latter). However, this is generally not so, and we need to further analyze the behavior of ABAP in the points in D.

Suppose that $\psi(D) > 0$. Take an arbitrary point $x \in D$. One of two statements surely holds:

- There is a neighborhood U of x such that $\mathbf{opt}(x) = \mathbf{opt}(y)$ for all $y \in U \cap D$;
- Given any neighborhood U of x, there is a point $y \in U \cap D$ such that $\mathbf{opt}(x) \neq \mathbf{opt}(y)$.

In the first statement holds for a point x, then either $\psi(U \cap D) = 0$ or ψ has an atom in $U \cap D$. This implies that this set is visited infinitely often, and the reasoning used to establish convergence of ABAP in C can be applied to the points in $U \cap D$. Denote the set of such points by D_0.

If the second statement holds, then we replicate the reasoning in the proof of Lemma F.4.3 to establish that the set of points $D_1 = D - D_0$ is a $(p-2)$-dimensional manifold. If $\psi(D_1) = 0$, the proof is complete. Otherwise, we successively repeat this process until we get a set D_n of isolated points (a 0-dimensional manifold). Then, for each point x in D_n, either $\psi(\{x\}) = 0$ or $\psi(\{x\}) > 0$, and the set $\{x\}$ is visited infinitely often. But in such points, convergence of ABAP is an immediate consequence of the convergence of BAP.

With this, we have (recurrently) established convergence of ABAP in all but a set of points of null ψ-measure. Now if Q^* is continuous in all but a ψ-null set of points $\mathcal{N}_Q$, the previous proof holds for every point x in which Q^* is continuous (with some care when defining the p-dimensional manifolds ζ_{G_a}), and the proof is complete. □

F.4.4 Proof of Theorem 8.4.2

This proof is basically similar to that of Theorems 7.3.2 and 7.3.3.

Convergence of θ_t^k to θ^* arises as an immediate consequence of Corollary 8.3.2. In fact, since the GLIE strategy σ_{θ_t} ensures sufficient exploration and the remaining conditions of Theorem 8.4.2 ensure the applicability of Corollary 8.3.2, then $\theta_t \to \theta^*$ w.p.1.

Furthermore, Theorem D.2.1 of Appendix D ensures that

$$\limsup_{t\to\infty} \frac{\|\theta_t - \theta^*\|}{\sqrt{\alpha_t \log\left(\sum_{\tau=1}^{t} \alpha_\tau\right)}} \leq K_0.$$

Therefore, (8.8) and Lemma 7.3.1 guarantee that, w.p.1, the sequence of virtual games VG_t obtained from the sequence of estimates $\{Q_{\theta_t}\}$ converges to the virtual game VG obtained from $\{Q_{\theta^*}\}$.

On the other hand, since r is continuous μ_X^θ-a.e., so is V^* and, therefore, Q^*. The fact that the chain (X, P_θ) is geometrically ergodic for every θ implies, in particular, that it is ψ-irreducible and Harris recurrent. But this means that we can apply Theorem 8.4.1 and replicate the reasoning in the proof of Theorem 7.3.2 to guarantee that, w.p.1, CAQL will converge in behavior to an optimal Nash equilibrium in μ_X^θ-almost all $x \in \mathcal{X}$. □

Bibliography

[1] Pieter Abbeel and Andrew Y. Ng. Apprenticeship learning via inverse reinforcement learning. In *Proceedings of the 21st International Conference on Machine Learning*, volume 69 of *ACM International Conference Proceeding Series*, pages 1–8, New York, NY, 2004. ACM Press.

[2] Douglas A. Aberdeen. A (revised) survey of approximate methods for solving partially observable Markov decision processes. Technical report, National ICT Australia, Canberra, Australia, 2003.

[3] Douglas A. Aberdeen. *Policy-gradient algorithms for partially observable Markov decision processes.* PhD thesis, Australian National University, April 2003.

[4] Jinane Abounadi, Dimitri P. Bertsekas, and Vivek S. Borkar. Learning algorithms for Markov decision processes with average cost. *SIAM Journal on Control and Optimization*, 40:681–698, 2001.

[5] Jinane Abounadi, Dimitri P. Bertsekas, and Vivek S. Borkar. Stochastic approximation for non-expansive maps: Application to Q-learning algorithms. *SIAM Journal on Control and Optimization*, 41:1–22, 2002.

[6] András Antos, Csaba Szepesvári, and Rémi Munos. Learning near-optimal policies with Bellman-residual minimization based fitted policy iteration and a single sample path. In *Proceedings of the 19th Annual Conference on Learning Theory (COLT'06)*, pages 574–588, 2006.

[7] Michael Athans. A tutorial on the LQG/LTR design method. Technical Report LIDS-P-1542, Laboratory for Information and Decision Systems, Massachussets Institute of Technology, March 1986.

[8] Christopher G. Atkeson, Andrew W. Moore, and Stefan Schaal. Locally weighted learning for control. *Artificial Intelligence Review*, 11(1-5):75–113, 1997.

[9] Tim Bailey and Eduardo M. Nebot. Localisation in large-scale environments. *Robotics and Autonomous Systems*, 37(4):261–281, 2001.

[10] Leemon C. Baird. *Reinforcement learning through gradient descent.* PhD thesis, Carnegie Mellon University, Pittsburgh, PA, May 1999.

[11] Leemon C. Baird. Residual algorithms: Reinforcement learning with function approximation. In *Proceedings of the 12th International Conference on Machine Learning (ICML'95)*, pages 30–37, San Francisco, CA, 1995. Morgan Kaufman Publishers.

[12] Leemon C. Baird and Andrew More. Gradient descent for general reinforcement learning. In David A. Cohn, editor, *Advances in Neural Information Processing Systems*, volume 11, pages 968–974, Cambridge, Massachussets, 1999. MIT Press.

[13] W. Baker. Learning via stochastic approximation in function space. PhD Thesis, 1997.

[14] David P. Barnes and John O. Gray. Behavior synthesis for cooperant mobile robot control. In *Proceedings of the International Conference on Control*, pages 1135–1140, 1991.

[15] Peter L. Bartlett and Jonathan Baxter. Estimation and approximation bounds for gradient-based reinforcement learning. In *Proceedings of the 13th Annual Conference on Computional Learning Theory (COLT'00)*, pages 133–141, San Francisco, CA, 2000. Morgan Kaufmann Publishers.

[16] Andrew G. Barto and Michael Duff. Monte-Carlo matrix inversion and reinforcement learning. In J.D. Cowan, G. Tesauro, and J. Alspector, editors, *Advances in Neural Information Processing Systems*, volume 6, pages 687–694, San Francisco, 1994. Morgan Kauffmann Publishers.

[17] Andrew G. Barto, Steven J. Bradtke, and Satinder P. Singh. Learning to act using real-time dynamic programming. Technical Report UM-CS-1993-002, Department of Computer Science, University of Massachusetts at Amherst, 1993.

[18] Jonathan Baxter and Peter L. Bartlett. Infinite-horizon policy-gradient estimation. *Journal of Artificial Intelligence Research*, 15:319–350, 2001.

[19] Valentina Bayer and Thomas Dietterich. A POMDP approximation algorithm that anticipates the need to observe. Technical Report 00-30-01, Department of Computer Science, Oregon State University, 2000.

[20] Graeme Bell and Michael Weir. Forward chaining for robot and agent navigation using potential fields. In *ACM International Conference Proceeding Series*, volume 56, pages 265–274, Darlinghurst, Australia, 2004. Australian Computer Society, Inc.

[21] Albert Benveniste, Michel Métivier, and Pierre Priouret. *Adaptive Algorithms and Stochastic Approximations*, volume 22 of *Applications of Mathematics*. Springer-Verlag, Berlin, 1990.

[22] Ulrich Berger. Fictitious play in $2 \times n$ games. Game Theory and Information 0303009, EconWPA, March 2003.

[23] Daniel S. Bernstein, Shlomo Zilberstein, and Neil Immerman. The complexity of decentralized control of Markov decision processes. In *Proceedings of the 16th Conference on Uncertainty in Artificial Intelligence (UAI'00)*, pages 32–37. AUAI Press, 2000.

[24] Daniel S. Bernstein, Shlomo Zilberstein, and Neil Immerman. The complexity of decentralized control of Markov decision processes. *Mathematics of Operations Research*, 27(4):819–840, 2002.

[25] Daniel S. Bernstein, Eric A. Hansen, and Shlomo Zilberstein. Bounded policy iteration for decentralized POMDPs. In *Proceedings of the 19th International Joint Conference on Artificial Intelligence (IJCAI'05)*, pages 1287–1292, 2005.

[26] Dimitri P. Bertsekas and Sergey Ioffe. Temporal differences-based policy iteration and applications in neuro-dynamic programming. Technical Report LIDS-P-2349, Lab. for Information and Decision Systems, MIT, 1996.

[27] Dimitri P. Bertsekas and John N. Tsitsiklis. *Neuro-Dynamic Programming*. Optimization and Neural Computation Series. Athena Scientific, Belmont, Massachusetts, 1996.

[28] Dimitri P. Bertsekas, Vivek S. Borkar, and Angelia Nedić. *Improved temporal difference methods with linear function approximation*, chapter 9, pages 235–260. Wiley Publishers, 2004.

[29] B. Bharath and Vivek S. Borkar. Stochastic approximation algorithms: overview and recent trends. *Sādhanā*, 24(4, 5):425–452, 1999.

[30] Blai Bonet. An ϵ-optimal grid-based algorithm for partially observable Markov decision process. In Claude Sammut and Achim G. Hoffmann, editors, *Proceedings of the 19th International Conference on Machine Learning (ICML'02)*, pages 51–58. Morgan Kaufmann, 2002.

[31] Vivek S. Borkar. Asynchronous stochastic approximations. *SIAM Journal on Control and Optimization*, 36(3):840–851, May 1998.

[32] Vivek S. Borkar. A learning algorithm for discrete-time stochastic control. *Probability in the Engineering and Informational Sciences*, 14:243–258, 2000.

[33] Vivek S. Borkar. Stochastic approximation with two time scales. *Systems & Control Letters*, 29(5):291–294, 1997.

[34] Vivek S. Borkar and Sean P. Meyn. The O.D.E. method for convergence of stochastic approximation and reinforcement learning. *SIAM Journal on Control and Optimization*, 38(2):447–469, 2000.

[35] Craig Boutilier. Sequential optimality and coordination in multiagent systems. In *Proceedings of the 16th International Joint Conference on Artificial Intelligence (IJCAI'99)*, pages 478–485, 1999.

[36] Craig Boutilier. Planning, learning and coordination in multiagent decision processes. In *Proceedings of the 6th Conference on Theoretical Aspects of Rationality and Knowledge (TARK-96)*, pages 195–210, 1996.

[37] Craig Boutilier and David Poole. Computing optimal policies for partially observable decision processes using compact representations. In *Proceedings of the 13th National Conference on Artificial Intelligence (AAAI'96)*, pages 1168–1175, Portland, Oregon, USA, 1996. AAAI Press/The MIT Press.

[38] Michael Bowling. Convergence problems of general-sum multiagent reinforcement learning. In *Proceedings of the 17th International Conference on Machine Learning (ICML'00)*, pages 89–94. Morgan Kaufman Publishers, 2000.

[39] Michael Bowling and Manuela Veloso. An analysis of stochastic game theory for multiagent reinforcement learning. Technical Report CMU-CS-00-165, School of Computer Science, Carnegie Mellon University, 2000.

[40] Michael Bowling and Manuela Veloso. Scalable learning in stochastic games. In *Proceedings of the AAAI Workshop on Game Theoretic and Decision Theoretic Agents (GTDT'02)*, pages 11–18. The AAAI Press, 2000. Published as AAAI Technical Report WS-02-06.

[41] Michael Bowling and Manuela Veloso. Rational and convergent learning in stochastic games. In *Proceedings of the 17th International Joint Conference on Artificial Intelligence (IJCAI'01)*, pages 1021–1026, 2001.

[42] Michael Bowling and Manuela Veloso. Rational learning of mixed equilibria in stochastic games. In *Proceedings of the 17th International Joint Conference on Artificial Intelligence (IJCAI'01)*, pages 1021–1026, 2001.

[43] Michael Bowling and Manuela Veloso. Simultaneous adversarial multi-robot learning. In *Proceedings of the 18th International Joint Conference on Artificial Intelligence (IJCAI'03)*, pages 699–704, 2003.

[44] Justin Boyan and Andrew Moore. Generalization in reinforcement learning: Safely approximating the value function. In G. Tesauro, D.S. Touretzky, and T.K. Lee, editors, *Neural Information Processing Systems 7*, pages 369–376, Cambridge, MA, 1995. The MIT Press.

[45] Justin A. Boyan. Least-squares temporal difference learning. In *Proceedings of the 16th International Conference on Machine Learning (ICML'99)*, pages 49–56, San Francisco, CA, 1999. Morgan Kaufmann.

[46] Justin A. Boyan. Technical update: Least-squares temporal difference learning. *Machine Learning*, 49:233–246, 2002.

[47] Steven J. Bradtke. *Incremental dynamic programming for on-line adaptive optimal control.* PhD thesis, University of Massachusetts at Amherst, September 1994. Available as CMPSCI Technical Report 94-62.

[48] Darius Braziunas and Craig Boutilier. Stochastic local search for POMDP controllers. In *Proceedings of the 19th National Conference on Artificial Intelligence (AAAI'04)*, pages 690–696. AAAI Press, 2004.

[49] George W. Brown. Some notes on computation of games solutions. Research Memoranda RM-125-PR, RAND Corporation, Santa Monica, California, 1949.

[50] Y. Uny Cao, Alex S. Fukunaga, and Andrew B. Kahng. Cooperative mobile robotics: Antecedents and directions. *Autonomous Robots*, 4(1):1–23, 1997.

[51] Anthony R. Cassandra. *Exact and approximate algorithms for partially observable Markov decision processes.* PhD thesis, Brown University, May 1998.

[52] Anthony R. Cassandra. Optimal policies for partially observable Markov decision processes. Technical Report CS-94-14, Department of Computer Sciences, Brown University, August 1994.

[53] Anthony R. Cassandra, Leslie Kaelbling, and James A. Kurien. Acting under uncertainty: Discrete Bayesian models for mobile-robot navigation. *Mathematics of Operations Research*, 12(3):441–450, 1987.

[54] Anthony R. Cassandra, Michael L. Littman, and Nevin L. Zhang. Incremental pruning: A simple, fast, exact method for partially observable Markov decision processes. In *Proceedings of the 13th Annual Conference on Uncertainty in Artificial Intelligence (UAI-99)*, pages 54–61, Providence, Rhode Island, 1997. Morgan Kaufmann Publishers.

[55] Christos G. Cassandras and Stéphane Lafortune. *Introduction to Discrete-Event Systems*. The Kluwer International Series on Discrete-Event Dynamic Systems. Kluwer Academic Publishers, 1999.

[56] Georgios Chalkiadakis and Craig Boutilier. Coordination in multiagent reinforcement learning: A Bayesian approach. In *Proceedings of the 2nd International Joint Conference on Autonomous Agents and Multiagent Systems (AAMAS'03)*, pages 709–716, New York, NY, 2003. ACM Press.

[57] Howie Choset and Keiji Nagatani. Topological simultaneous localization and mapping (SLAM): Toward exact localization without explicit localization. *IEEE Transactions on Robotics and Automation*, 17(2):125–137, April 2001.

[58] Kai Lai Chung. *A Course in Probability Theory*. Mathematics and Statistics. Academic Press, third edition, 2001.

[59] Caroline Claus and Craig Boutilier. The dynamics of reinforcement learning in cooperative multiagent systems. In *Proceedings of the 15th National Conference on Artificial Intelligence (AAAI'98)*, pages 746–752, 1998.

[60] Monica-Gabriela Cojocaru. *Projected dynamical systems on Hilbert spaces*. PhD thesis, Queen's University, 2002.

[61] Vincent Conitzer and Tuomas Sandholm. AWESOME: A general multiagent learning algorithm that converges in self-play and learns a best response against stationary opponents. In *Proceedings of the 20th International Conference on Machine Learning (ICML'03)*, pages 83–90, 2003.

[62] Robert H. Crites and Andrew G. Barto. Elevator group control using multiple reinforcement learning agents. *Machine Learning*, 33(2-3):235–262, 1998.

[63] Michael Csorba. *Simultaneous localisation and map building*. PhD thesis, Robotics Research Group, University of Oxford, 1997.

[64] Valdinei Freire da Silva, Anna Helena Reali Costa, and Pedro Lima. Inverse reinforcement learning with evaluation. In *Proceedings of the IEEE International Conference on Robotics and Automation (ICRA'06)*, pages 4246– 4251, 2006.

[65] Richard Dearden, Nir Friedman, and Stuart Russell. Bayesian Q-learning. In *Proceedings of the 15th National Conference on Artificial Intelligence (AAAI'98) and 10th Conference on Innovative Applications of Artificial Intelligence (IAAI'98)*, pages 761–768, Menlo Park, California, USA, 1998. AAAI Press.

[66] Bernard Delyon. General results on the convergence of stochastic algorithms. *IEEE Transactions on Automatic Control*, AC-41(9):1245–1256, 1996.

[67] Bernard Delyon. Stochastic approximation with decreasing gain: Convergence and asymptotic theory. Available at the URL http://name.math.univ-rennes1.fr/bernard.delyon/as_cours.ps, June 2000.

[68] Marco Dorigo and Hugues Bersini. A comparison of Q-learning and classifier systems. In *Proceedings of the Third International Conference on Simulation of Adaptive Behavior: From Animals to Animats 3*, pages 248–255, Cambridge, MA, USA, 1994. MIT Press.

[69] Prashant Doshi. *Optimal sequential planning in partially observable multiagent settings*. PhD thesis, University of Illinois at Chicago, 2005.

[70] Prashant Doshi and Piotr Gmytrasiewicz. Subjective equilibrium in interactive POMDPs: Theory and computational limitations. 19th International Joint Conference on Artificial Intelligence (IJCAI'05), Game Theoretic and Decision Theoretic Agents Workshop, 2005.

[71] Prashant Doshi and Piotr Gmytrasiewicz. On the difficulty of achieving equilibria in interactive POMDPs. Proceedings of the 21st National Conference on Artificial Intelligence (AAAI'06) (to appear), 2006.

[72] Prashant Doshi and Piotr J. Gmytrasiewicz. Approximating state estimation in multiagent settings using particle filters. In *Proceedings of the 4th International Joint Conference on Autonomous agents and Multiagent Systems (AAMAS'05)*, pages 320–327, New York, NY, 2005. ACM Press.

[73] Christophe Druet, Damien Ernst, and Louis Wehenkel. *Application of reinforcement learning to electrical power system closed-loop emergency control*, volume 1910 of *Lecture Notes In Computer Science*, pages 86–95. Springer-Verlag, 2000.

[74] Michael Duff. *Optimal learning: Computational procedures for Bayes-adaptive Markov decision processes*. PhD thesis, Department of Computer Science, University of Massachusetts Amherst, February 2002.

[75] Edmund H. Durfee, Victor R. Lesser, and Daniel D. Corkill. Coherent cooperation among communicating problem solvers. *IEEE Transactions on Computers*, 36(11): 1275–1291, 1987.

[76] Sašo Džeroski. Relational reinforcement learning for agents in worlds with objects. *Adaptive Agents and Multi-Agent Systems*, pages 306–321, 2003.

[77] Magnus Egerstedt. *Motion planning and control of mobile robots*. PhD thesis, Royal Institute of Technology, Stockholm, Switzerland, 2000.

[78] Alberto Elfes. Using occupancy grids for mobile robot perception and navigation. *Computer*, 22(6):46–57, 1989.

[79] Rosemary Emery-Montemerlo. *Game-theoretic control for robot teams.* PhD thesis, The Robotics Institute, Carnegie Mellon University, August 2005.

[80] Rosemary Emery-Montemerlo, Geoff Gordon, Jeff Schneider, and Sebastian Thrun. Approximate solutions for partially observable stochastic games with common payoffs. In *Proceedings of the 3rd International Joint Conference on Autonomous Agents and Multiagent Systems (AAMAS'04)*, volume 1, pages 136–143, New York, NY, 2004. ACM Press.

[81] Rosemary Emery-Montemerlo, Geoff Gordon, Jeff Schneider, and Sebastian Thrun. Game-theoretic control for robot teams. In *Proceedings of the 2005 IEEE International Conference on Robotics and Automation (ICRA'05)*, pages 1175–1181, 2004.

[82] Eyal Even-Dar and Yishay Mansour. Learning rates for Q-learning. *Journal of Machine Learning Research*, 5:1–25, 2003.

[83] Elisabetta Fabrizi and Alessandro Saffiotti. Augmenting topology-based maps with geometric information. *Robotics and Autonomous Systems*, 40:91–97, 2002.

[84] Nicholas V. Findler and Raphael M. Malyankar. Social structures and the problem of coordination in intelligent agent societies. Invited talk at the special session on "Agent-Based Simulation, Planning and Control", IMACS World Congress (2000), 2000. CD Paper 122-12.

[85] Arlington M. Fink. Equilibrium in a stochastic n-person game. *Journal of Science in Hiroshima University, Series A-I*, 28:89–93, 1969.

[86] Jeffrey R. Forbes. *Reinforcement learning for autonomous vehicles.* PhD thesis, University of California, Berkeley, 2002.

[87] Dieter Fox. *Markov localization: A probabilistic framework for mobile robot localization and navigation.* PhD thesis, University of Bonn, Germany, 1998.

[88] Dieter Fox, Wolfram Burgard, and Sebastian Thrun. Markov localization for reliable robot navigation and people detection. In *Selected Papers from the International Workshop on Sensor Based Intelligent Robots*, volume 1724 of *Lecture Notes In Computer Science*, pages 1–20, London, UK, 1998. Springer-Verlag.

[89] Dieter Fox, Wolfram Burgard, and Sebastian Thrun. Markov localization for mobile robots in dynamic environments. *Journal of Artificial Intelligence Research*, 11:391–427, 1999.

[90] Matthias O. Franz and Hanspeter A. Mallot. Biomimetic robot navigation. *Robotics and Autonomous Systems*, 30:133–153, 2000.

[91] Holger Friedrich, Michael Kaiser, Oliver Rogalla, and Rüdiger Dillmann. *Learning and communication in multi-agent systems*, volume 1221 of *Lecture Notes in Aartificial Intelligence*, chapter III, pages 259–275. Springer-Verlag, May 1997.

[92] Les Gasser. Social conceptions of knowledge and action: Distributed artificial intelligence and open systems semantics. *Artificial Intelligence*, 47(1-3):107–138, 1991.

[93] Hector Geffner and Blai Bonet. Solving large POMDPs by real time dynamic programming. Fall AAAI Symposium on POMDPS (Working Notes), 1998.

[94] Robert Givan, Sonia Leach, and Thomas Dean. Bounded parameter Markov decision processes. Technical Report CS-97-05, Department of Computer Science, Brown University, May 1997.

[95] Georg Glaeser and Ernst Strouhal. Von Kempelen's chess-playing pseudo-automaton and Zu Racknitz's explanation of its controls. In *Proceedings of the International Symposium on History of Machines - HMM'2000*, pages 351–360. Kluwer Academic Publishers, 2000.

[96] Robert Glaubius and William D. Smart. Manifold representations for value-function approximation in reinforcement learning. Technical Report 05-19, Department of Computer Science and Engineering, Washington University in St. Louis, 2005.

[97] Peter W. Glynn. A GSMP formalism for discrete event systems. In *Proceedings of the IEEE*, volume 77, No. 1, pages 14–23, 1989.

[98] Piotr Gmytrasiewicz and Prashant Doshi. A framework for sequential planning in multiagent settings. *Journal of Artificial Intelligence Research*, 24:49–79, 2005.

[99] Claudia V. Goldman and Jeffrey S. Rosenschein. Emergent coordination through the use of cooperative state-changing rules. In *Proceedings of the 12th International Workshop on Distributed Artificial Intelligence*, pages 171–185, Hidden Valley, Pennsylvania, 1993.

[100] Judy Goldsmith and M. Mundhenk. Complexity issues in Markov decision processes. In *Proceedings of the 13th Annual IEEE Conference on Computational Complexity*, pages 272–280, 1998.

[101] Geoffrey J. Gordon. Reinforcement learning with function approximation converges to a region. In *Proceedings of the Neural Information Processing Systems*, pages 1040–1046, 2000.

[102] Geoffrey J. Gordon. Chattering in SARSA(λ). CMU Learning Lab Internal Report, CMU Learning Lab, Carnegie Mellon University, 1996.

[103] Geoffrey J. Gordon. Stable function approximation in dynamic programming. Technical Report CMU-CS-95-103, School of Computer Science, Carnegie Mellon University, 1995.

[104] Amy Greenwald and Keith Hall. Correlated Q-learning. In *Proceedings of the 20th International Conference on Machine Learning (ICML'03)*, pages 242–249, 2003.

[105] Geoffrey Grimmet and David Stirzaker. *Probability and Random Processes*. Oxford University Press, third edition, 2001.

[106] Thomas Hakon Grönwall. Note on the derivative with respect to a parameter of the solutions of a system of differential equations. *Annals of Mathematics*, 20:292–296, 1919.

[107] Lars Grüne and Willi Semmler. Using dynamic programming with adaptive grid scheme for optimal control problems in economics. *Journal of Economic Dynamics and Control*, 28:2427–2456, 2004.

[108] Lars Grüne and Fabian Wirth. On linear convergence of discounted optimal control problems with vanishing discount rate. In *Proceedings of the Mathematical Theory of Networks and Systems, MTNS98*, pages 185–188, Padova, 1998.

[109] Carlos Guestrin, Daphne Koller, and Ronald Parr. Solving factored POMDPs with linear value functions. In *IJCAI-01 Workshop on Planning under Uncertainty and Incomplete Information*, pages 67–75, August 2001.

[110] Carlos Guestrin, Michail G. Lagoudakis, and Ronald Parr. Coordinated reinforcement learning. In *Proceedings of the 19th International Conference on Machine Learning (ICML'02)*, pages 227–234, 2002.

[111] José Guivant and Eduardo Nebot. Optimization of the simultaneous localization and map building algorithm for real time implementation. *IEEE Transaction of Robotic and Automation*, 17(3):242–257, 2991.

[112] José Guivant, Juan Nieto, Favio Masson, and Eduardo Nebot. Navigation and mapping in large unstructured environments. *International Journal of Robotics Research*, 23(4/5):449–472, 2003.

[113] Vijaykumar Gullapalli and Andrew G. Barto. Convergence of indirect adaptive asynchronous value iteration algorithms. In J. D. Cowan, G. Tesauro, and J Alspector, editors, *Advances in Neural Processing Information Systems*, volume 6, pages 695–702, San Francisco, CA, 1994. Morgan Kaufmann Publishers.

[114] AnYuan Guo and Victor Lesser. Planning for weakly-coupled partially observable stochastic games. In *Proceedings of the 19th International Joint Conference on Artificial Intelligence (IJCAI'05)*, pages 1715–1716, Edinburgh, Scotland, 2005.

[115] Kamran H-Sedighi, Kaveh Ashenayi, Theordore W. Manikas, Roger L. Wainwright, and Heng-Ming Tai. Autonomous local path planning for a mobile robot using a genetic algorithm. In *Proceedings of the 2004 IEEE Congress on Evolutionary Computation (CEC'2004)*, pages 1338–1345, 2004.

[116] Rodney D. Hale, Mohd Rokonuzzaman, and Raymond G. Gosine. Control of mobile robots in unstructured environments using discrete event modeling. In Douglas W. Gage and Howie M. Choset, editors, *Proceedings of SPIE: Mobile Robots XIV*, volume 3838, pages 275–282, 1999.

[117] Joseph Y. Halpern and Yoram Moses. Knowledge and common knowledge in a distributed environment. *Journal of the ACM*, 37(3):549–587, 1990.

[118] Eric A. Hansen. An improved policy iteration algorithm for partially observable MDPs. In *Proceedings of the 1997 Conference on Advances in Neural Information Processing Systems (NIPS'97)*, volume 10, pages 1015–1021, Cambridge, MA, USA, 1998. MIT Press.

[119] Eric A. Hansen. Solving POMDPs by searching in policy space. In Gregory F. Cooper and Serafin Moral, editors, *Proceedings of the 14th Conference on Uncertainty in Artificial Intelligence (UAI'98)*, pages 211–219. Morgan Kaufmann, 1998.

[120] Samuel W. Hasinoff. Reinforcement learning for problems with hidden state. Technical report, University of Toronto, Department of Computer Science, 2002.

[121] Milos Hauskrecht. Value-function approximations for partially observable Markov decision processes. *Journal of Artificial Intelligence Research*, 13:33–94, 2000.

[122] Qiming He and Mark A. Shayman. Solving POMDPs by on-policy linear approximate learning algorithm. In *Proceedings of the Conference on Information Sciences and Systems*. Princeton University, 2000.

[123] Aditia Hermanu, Theodore W. Manikas, Kaveh Ashenayi, and Roger L. Wainwright. Autonomous robot navigation using a genetic algorithm with an efficient genotype structure. In C. H. Dagli et al., editor, *Intelligent Engineering Systems Through Artificial Neural Networks: Smart Engineering Systems Design: Neural Networks, Fuzzy Logic, Evolutionary Programming, Complex Systems and Artificial Life*, pages 319–324, New York, 2004. ASME Press.

[124] Morris W. Hirsch. Convergent activation dynamics in continuous time networks. *Neural Networks*, 2:331–349, 1989.

[125] Josef Hofbauer and William H. Sandholm. On the global convergence of stochastic fictitious play. *Econometrica*, 70(6):2265–2294, November 2002.

[126] Ralph Howard. The Gronwall inequality. Class Notes, 1998.

[127] Junling Hu and Michael P. Wellman. Multiagent reinforcement learning: Theoretical framework and an algorithm. In *Proceedings of the 15th International Conference on Machine Learning (ICML'98)*, pages 242–250, 1998.

[128] Junling Hu and Michael P. Wellman. Multiagent reinforcement learning in stochastic games. Available online at http://citeseer.ist.psu.edu/hu99multiagent.html, 1999.

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

[130] Hideaki Itoh and Kiyohiko Nakamura. Partially observable Markov decision processes with imprecise parameters. *Artificial Intelligence*, 171(8-9):453–490, 2007.

[131] Tommi Jaakkola, Michael I. Jordan, and Satinder P. Singh. On the convergence of stochastic iterative dynamic programming algorithms. *Neural Computation*, 6(6): 1185–1201, 1994.

[132] Tommi Jaakkola, Satinder Singh, and Michael Jordan. Reinforcement learning algorithm for partially observable Markov decision problems. In *Advances in Neural Information Processing Systems*, volume 7, pages 345–352, Cambridge, MA, 1995. The MIT Press.

[133] Leslie P. Kaelbling. *Learning in Embedded Systems*. The MIT Press, 1993.

[134] Leslie P. Kaelbling, Michael L. Littman, and Andrew W. Moore. Reinforcement learning: A survey. *Journal of Artificial Intelligence Research*, 4:237–285, 1996.

[135] Thomas Kaijser. A limit theorem for partially observed Markov chains. *Annals of Probability*, 3(4):677–696, 1975.

[136] Spiros Kapetanakis and Daniel Kudenko. Improving on the reinforcement learning of coordination in cooperative multi-agent systems. In *Proceedings of the 2nd Symposium on Adaptive Agents and Multi-agent Systems*, pages 89–94. The Society for the Study of Artificial Intelligence and Simulation of Behaviour, 2002.

[137] Michael Kearns and Satinder Singh. Finite-sample convergence rates for Q-learning and indirect algorithms. In M. J. Kearns, S. A. Solla, and D. A. Cohn, editors, *Advances in Neural Information Processing Systems*, volume 11, pages 996–1002, 1999.

[138] Michael Kearns and Satinder Singh. Near-optimal reinforcement learning in polynomial time. In *Proceedings of the 15th International Conference on Machine Learning (ICML'98)*, pages 260–268, San Francisco, CA, 1998. Morgan Kaufmann Publishers.

[139] Philipp W. Keller, Shie Mannor, and Doina Precup. Automatic basis function construction for approximate dynamic programming and reinforcement learning. In *Proceedings of the 23rd International Conference on Machine Learning (ICML'06)*, pages 449–456, New York, NY, 2006. ACM Press.

[140] Jack Kiefer and Jacob Wolfowitz. Stochastic estimation of the maximum of a regression function. *Annals of Mathematical Statistics*, 23:462–466, 1952.

[141] Masaaki Kijima. *Markov Processes for Stochastic Modelling.* Stochastic Modelling Series. Chapman & Hall, Boundary Row, London, 1997.

[142] Sven Koenig and Reid Simmons. Unsupervised learning of probabilistic models for robot navigation. In *Proceedings of the 1996 IEEE International Conference on Robotics and Automation (ICRA'96)*, pages 2301–2308, 1996.

[143] Jelle R. Kok, Matthijs T. J. Spaan, and Nikos Vlassis. An approach to noncommunicative multiagent coordination in continuous domains. In Marco Wiering, editor, *Benelearn 2002: Proceedings of the Twelfth Belgian-Dutch Conference on Machine Learning*, pages 46–52, Utrecht, The Netherlands, 2002.

[144] Andrei Nikolaevich Kolmogorov. Zur theorie der Markoffschen ketten. *Mathematzsche Annalen*, 112:155–160, 1936.

[145] Vijay R. Konda and John N. Tsitsiklis. On actor-critic algorithms. *SIAM Journal on Control and Optimization*, 42(4):1143–1166, 2003.

[146] R. Mathew Kretchmar and Charles W. Anderson. Comparison of CMACs and radial basis functions for local function approximators in reinforcement learning. In *Proceedings of the 1997 IEEE International Conference on Neural Networks*, pages 834–837, 1997.

[147] Vijay Krishna and Tomas Sjöström. On the convergence of fictitious play. Harvard Institute of Economic Research Working Papers 1717, Institute of Economic Research, Harvard University, 1995.

[148] Sanjeev R. Kulkarni and C. S. Horn. An alternative proof for convergence of stochastic approximation algorithms. *IEEE Transactions on Automatic Control*, AC-41(3): 419–424, 1996.

[149] Sanjeev Kumar and Philip R. Cohen. Towards a fault-tolerant multi-agent system architecture. In *Proceedings of the 4th International Conference on Autonomous Agents (Agents - 00)*, pages 459–466, Barcelona, Spain, 2000. ACM Press.

[150] Philippe Künzle. *Building topological maps for robot navigation using neural networks*. PhD thesis, McGill University, Montréal, 2005.

[151] Harold J. Kushner and Dean S. Clark. *Stochastic Approximation for Constrained and Unconstrained Systems*. Springer-Verlag, New York, 1978.

[152] Harold J. Kushner and S. Lakshmivarahan. Numerical studies of stochastic approximation procedures for constrained problems. *IEEE Transactions on Automatic Control*, AC-22(3):428–439, 1977.

[153] Harold J. Kushner and G. George Yin. *Stochastic Approximation and Recursive Algorithms and Applications*, volume 35 of *Applications of Mathematics*. Springer-Verlag New York, Inc., second edition, 2003.

[154] Michail G. Lagoudakis and Ronald Parr. Value function approximation in zero-sum Markov games. In Adnan Darwiche and Nir Friedman, editors, *Proceedings of the 18th International Conference on Uncertainty in Artificial Intelligence (UAI'02)*, pages 283–292, 2002.

[155] Dimitrios Lambrinos, Ralph Möller, Thomas Labhart, Rolf Pfeifer, and Rudiger Wehner. A mobile robot employing insect strategies for navigation. *Robotics and Autonomous Systems*, 30:39–64, 2000.

[156] Martin Lauer and Martin Riedmiller. An algorithm for distributed reinforcement learning in cooperative multi-agent systems. In *Proceedings of the 17th International Conference on Machine Learning (ICML'00)*, pages 535–542, San Francisco, CA, 2000. Morgan Kaufmann.

[157] Martin Lauer and Martin Riedmiller. Reinforcement learning for stochastic cooperative multi-agent-systems. In *Proceedings of the 3rd International Joint Conference on Autonomous Agents and Multiagent Systems (AAMAS'04)*, pages 1516–1517, New York, NY, 2004. ACM Press.

[158] Steven M. LaValle. *A game-theoretic framework for robot motion planning*. PhD thesis, Department of Electrical and Computer Engineering, University of Illinois at Urbana, 1995.

[159] François Le Gland and Laurent Mevel. Geometric ergodicity in hidden Markov models. Technical Report 2991, Institut National de Recherche en Informatique et en Automatique (INRIA), September 1996.

[160] David LeRoux and Michael L. Littman. Friend-or-foe Q-learning in general-sum games. International Workshop on Learning Classifier Systems (extended abstract), 2001.

[161] David S. Leslie and E. J. Collins. Generalised weakened fictitious play. *Games and Economic Behavior*, 56:285–298, 2006.

[162] David S. Leslie and E. J. Collins. Individual Q-learning in normal form games. *SIAM Journal on Control and Optimization*, 44(2):495–514, 2005.

[163] Michael L. Littman. The Witness algorithm: Solving partially observable Markov decision processes. Technical Report CS-94-40, Department of Computer Sciences, Brown University, December 1994.

[164] Michael L. Littman. Memoryless policies: Theoretical limitations and practical results. In *From Animals to Animats 3: Proceedings of the Third International Conference on Simulation of Adaptive Behavior*, pages 1023–1028, Cambridge, MA, 1994. The MIT Press.

[165] Michael L. Littman. Markov games as a framework for multi-agent reinforcement learning. In Ramon López de Mántaras and David Poole, editors, *Proceedings of the 11th International Conference on Machine Learning (ICML'94)*, pages 157–163, San Francisco, CA, 1994. Morgan Kaufmann Publishers.

[166] Michael L. Littman. Value-function reinforcement learning in Markov games. *Journal of Cognitive Systems Research*, 2(1):55–66, 2001.

[167] Michael L. Littman. Friend-or-foe Q-learning in general-sum games. In *Proceedings of the 18th International Conference on Machine Learning (ICML'01)*, pages 322–328, San Francisco, CA, 2001. Morgan Kaufmann Publishers.

[168] Michael L. Littman and Csaba Szepesvári. A generalized reinforcement learning model: Convergence and applications. In *Proceedings of the 13th International Conference on Machine Learning (ICML'96)*, pages 310–318, San Francisco, CA, 1996. Morgan Kaufmann Publishers.

[169] Michael L. Littman, Anthony R. Cassandra, and Leslie P. Kaelbling. Learning policies for partially observable environments: Scaling up. In Armand Prieditis and Stuart Russell, editors, *Proceedings of the 12th International Conference on Machine Learning (ICML'95)*, pages 362–370, San Francisco, CA, 1995. Morgan Kaufmann Publishers.

[170] Michael L. Littman, Thomas L. Dean, and Leslie Pack Kaelbling. On the complexity of solving Markov decision problems. In *Proceedings of the 11th Conference on Uncertainty in Artificial Intelligence (UAI-95)*, pages 394–402, New York, NY, 1995. Elsevier Science Publishing Company, Inc.

[171] Lennart Ljung. Analysis of recursive stochastic algorithms. *IEEE Transactions on Automatic Control*, AC-22(4):551–575, August 1977.

[172] Lennart Ljung and Torsten Söderström. *Theory and Practice of Recursive Identification*. Signal Processing, Optimization and Control. The MIT Press, 1987.

[173] John Loch and Satinder Singh. Using eligibility traces to find the best memoryless policy in partially observable Markov decision processes. In Jude W. Shavlik, editor, *Proceedings of the 15th International Conference on Machine Learning (ICML'98)*, pages 323 – 331, San Francisco, CA, 1998. Morgan Kaufmann.

[174] Elena López, Luis M. Bergasa, Rafael Barea, and Maria S. Escudero. Topological robot navigation using multisensorial event-based POMDPs. In *Proceedings of the 11th International Conference on Advanced Robotics (ICAR'2003)*, pages 216–221, 2003.

[175] Christopher Lusena, Judy Goldsmith, and Martin Mundhenk. Nonapproximability results for partially observable Markov decision processes. *Journal of Artificial Intelligence Research*, 14:83–103, 2001.

[176] Xiaowei Ma, Xiaoli Li, and Hong Qiao. Fuzzy neural network-based real-time self-reaction of mobile robot in unknown environments. *Mechatronics*, 11:1039–1052, 2001.

[177] Omid Madani, Steve Hanks, and Anne Condon. On the undecidability of probabilistic planning and infinite-horizon partially observable Markov decision problems. In *Proceedings of the 16th National Conference on Artificial intelligence (AAAI'99)*, pages 541–548, Menlo Park, CA, USA, 1999. AAAI Press.

[178] Sridhar Mahadevan. Average reward reinforcement learning: Foundations, algorithms, and empirical results. *Machine Learning*, 22:159–196, 1996.

[179] Shie Mannor, Reuven Rubinstein, and Yohai Gat. The cross entropy method for fast policy search. In *Proceedings of the 20th International Conference on Machine Learning (ICML'03)*, pages 512–519, San Francisco, CA, 2003. Morgan Kaufmann Publishers.

[180] Andrey Andreyevich Markov. Rasprostranenie zakona bol'shih chisel na velichiny, zavisyaschie drug ot druga. *Izvestiya Fiziko-matematicheskogo obschestva pri Kazanskom universitete*, 2(15):135–156, 1906.

[181] Andrey Andreyevich Markov. Extension of the limit theorems of probability theory to a sum of variables connected in a chain. In R. Howard, editor, *Dynamic Probabilistic Systems*, volume 1, chapter B (App.). John Wiley and Sons, 1971. Reprinted.

[182] Bhaskara Marthi. Automatic shaping and decomposition of reward functions. In Zoubin Ghahramani, editor, *Proceedings of the 24th International Conference on Machine Learning (ICML'07)*. Omni Press, 2007.

[183] Richard J. Martin, Lena Valavani, and Michael Athans. Multivariable control of a submersible using the LQG/LTE design methodology. Technical Report LIDS-P-1548, Laboratory for Information and Decision Systems, Massachussets Institute of Technology, March 1986.

[184] Mario Mata, José M. Armingol, Arturo de la Escalera, and Miguel A. Salichs. A visual landmark recognition system for topological navigation of mobile robots. In *Proceedings of the 2001 IEEE International Conference on Robotics and Automation (ICRA'01)*, pages 1124–1129, 2001.

[185] Maja J. Matarić. Using communication to reduce locality in distributed multi-agent learning. *Journal of Experimental and Theoretical Artificial Intelligence*, 10(3):357–369, 1998.

[186] Maja J. Matarić. *Interaction and intelligent behavior*. PhD thesis, Massachusetts Institute of Technology, 1994. Also available as MIT AI Lab Tech. Report AITR-1495.

[187] David A. McAllester and Satinder Singh. Approximate planning for factored POMDPs using belief state simplification. In Kathryn B. Laskey and Henri Prade, editors, *Proceedings of the 15th Annual Conference on Uncertainty in Artificial Intelligence (UAI'99)*, pages 409–416, San Francisco, CA, 1999. Morgan Kaufmann Publishers.

[188] Francisco S. Melo and M. Isabel Ribeiro. Convergence of classical reinforcement learning algorithms and partial observability. Technical Report RT-601-06, Institute for Systems and Robotics, January 2006.

[189] Francisco S. Melo and M. Isabel Ribeiro. Rational and convergent model-free adaptive learning for team Markov games. Technical Report RT-601-07, Institute for Systems and Robotics, February 2007.

[190] Francisco S. Melo and M. Isabel Ribeiro. Q-learning with linear function approximation. Technical Report RT-602-07, Institute for Systems and Robotics, March 2007.

[191] Francisco S. Melo and M. Isabel Ribeiro. Transition entropy in partially observable Markov decision processes. In T. Balch T. Arai, R. Pfeifer and H. Yokoi, editors, *Proceedings of the 9th International Conference on Intelligent Autonomous Systems (IAS-9)*, pages 282–289. IOS Press, 2005.

[192] Francisco S. Melo and M. Isabel Ribeiro. Q-learning with linear function approximation. In *Learning Theory: Proceedings of the 20th Annual Conference on Learning Theory*, volume 4539 of *Lecture Notes in Artificial Intelligence*, pages 308–322. Springer-Verlag, 2007.

[193] Francisco S. Melo and M. Isabel Ribeiro. Convergence of Q-learning with linear function approximation. In *Proceedings of the 2007 European Control Conference (ECC'07)*, pages 2671–2678, 2007.

[194] Francisco S. Melo, Pedro A. Lima, and M. Isabel Ribeiro. Event-driven modelling and control of a mobile robot population. In *Proceedings of the 8th International Conference on Intelligent Autonomous Systems (IAS-8)*, pages 227–234. IOS Press, 2004.

[195] Francisco S. Melo, M. Isabel Ribeiro, and Pedro A. Lima. Blocking controllability of a mobile robot population. Technical Report RT-601-04, Institute for Systems and Robotics, May 2004.

[196] Francisco S. Melo, M. Isabel Ribeiro, and Pedro A. Lima. Navigation controllability of a mobile robot population. In *RoboCup 2004: Robot Soccer World Cup VIII*, volume 3276 of *Lecture Notes in Computer Science*, pages 499–506. Springer-Verlag, 2005.

[197] Ishai Menache, Shie Mannor, and Nahum Shimkin. Basis function adaptation in temporal difference reinforcement learning. *Annals of Operations Research*, 134(1): 215–238, February 2005.

[198] Michel Metivier and Pierre Priouret. Applications of a Kushner and Clark lemma to general classes of stochastic algorithms. *IEEE Transactions on Information Theory*, IT-30(2):140–151, 1984.

[199] Nicolas Meuleau, Leonid Peshkin, Kee-Eung Kim, and Leslie P. Kaelbling. Learning finite-state controllers for partially observable environments. In *Proceedings of the 15th International Conference on Uncertainty in Artificial Intelligence (UAI'99)*, pages 427–436, 1999.

[200] Jean-Arcady Meyer and David Filliat. Map-based navigation in mobile robots: II. A review of map-learning and path-planning strategies. *Cognitive Systems Research*, 4:283–317, 2003.

[201] Sean P. Meyn and Richard L. Tweedie. *Markov Chains and Stochastic Stability*. Communications and Control Engineering Series. Springer-Verlag, New York, 1993.

[202] Don Monderer and Lloyd S. Shapley. Fictitious play property for games with identical interests. *Journal of Economic Theory*, 68:258–265, 1996.

[203] Andrew W. Moore and Christopher G. Atkeson. Memory-based reinforcement learning: Efficient computation with prioritized sweeping. In S. J. Hanson, J. D. Cowan, and C. L. Giles, editors, *Advances in Neural Information Processing Systems*, volume 5, pages 263–270, San Francisco, CA, 1993. Morgan Kaufmann Publishers.

[204] Andrew W. Moore and Christopher G. Atkeson. Prioritized sweeping: Reinforcement learning with less data and less time. *Machine Learning*, 13:103–130, 1993.

[205] David E. Moriarty, Alan C. Schultz, and John J. Grefenstette. Evolutionary algorithms for reinforcement learning. *Journal of Artificial Intelligence Research*, 11: 241–276, 1999.

[206] Martin Mundhenk, Judy Goldsmith, and Eric Allender. The complexity of policy evaluation for finite-horizon partially observable Markov decision processes. In *Proceedings of the 22nd International Symposium on Mathematical Foundations of Computer Science*, pages 129–138, 1997.

[207] Rémi Munos. Performance bounds in L_p-norm for approximate value iteration. *SIAM Journal on Control and Optimization*, (to appear), 2007.

[208] Rémi Munos and Csaba Szepesvári. Finite-time bounds for sampling-based fitted value iteration. *Journal of Machine Learning Research*, page (submitted), 2007.

[209] Kevin P. Murphy. A survey of POMDP solution techniques. Technical report, Department of Computer Science, University of California, 2000.

[210] Wasif Naeem, Robert Sutton, and S. M. Ahmad. LQG/LTR control of an autonomous underwater vehicle using a hybrid guidance law. In *Proceedings of GCUV'03 Conference*, pages 35–40, 2003.

[211] Ranjit Nair, David Pynadath, Makoto Yokoo, Milind Tambe, and Stacy Marsella. Taming decentralized POMDPs: Towards efficient policy computation for multiagent settings. In *Proceedings of the 18th International Joint Conference on Artificial Intelligence (IJCAI'03)*, pages 705–711, 2003.

[212] Ranjit Nair, Maayan Roth, Makoto Yokoo, and Milind Tambe. Communication for improving policy computation in distributed POMDPs. In *Proceedings of the 3rd International Joint Conference on Agents and Multiagent Systems (AAMAS'04)*, pages 1098–1105, New York, NY, 2004. ACM Press.

[213] Jae Ho Nam, Seung Min Baek, and Tae-Yong Kuc. A robust nonlinear optimal controller for autonomous mobile robots. In *Proceedings of the IEEE International Conference on Systems, Man, and Cybernetics*, volume 2, pages 1453–1458, 1996.

[214] John F. Nash. Two-person cooperative games. *Econometrica*, 21(1):128–140, 1953.

[215] John F. Nash. The bargaining problem. *Econometrica*, 18(2):155–162, 1950.

[216] John F. Nash. Equilibrium points in n-person games. *Proceedings of the National Academy of Sciences*, 36:48–49, 1950.

[217] Gonçalo Neto, Hugo Costelha, and Pedro U. Lima. Topological navigation in configuration space applied to soccer robots. In *RoboCup 2003: Robot Soccer World Cup VII*, pages 551–558. Springer, 2003.

[218] Paul M. Newman. *On the structure and solution of the simultaneous localisation and map building problem*. PhD thesis, Australian Centre for Field Robotics, University of Sydney, 1999.

[219] Andrew Y. Ng and Michael I. Jordan. PEGASUS: A policy search method for large MDPs and POMDPs. In Craig Boutilier and Moisés Goldszmidt, editors, *Proceedings of the 16th Conference Uncertainty in Artificial Intelligence (UAI'00)*, pages 406–415. Morgan Kaufmann, 2000.

[220] Andrew Y. Ng and Stuart J. Russel. Algorithms for inverse reinforcement learning. In *Proceedings of the 17th International Conference on Machine Learning (ICML'00)*, pages 663–670, San Francisco, CA, 2000. Morgan Kaufmann Publishers.

[221] Andrew Y. Ng, Daishi Harada, and Stuart Russel. Policy invariance under reward transformations: Theory and application to reward shaping. In *Proceedings of the 16th International Conference on Machine Learning (ICML'99)*, pages 278–287, San Francisco, CA, 1999. Morgan Kaufmann.

[222] Illah R. Nourbakhsh, Rob Powers, and Stanley T. Birchfield. Dervish: An office navigating robot. *AI Magazine*, 16(2):53–60, 1995.

[223] Andrzej S. Novak and Eitan Altman. ϵ-equilibria for stochastic games with uncountable state-space and unbounded costs. *SIAM Journal of Control and Optimization*, 40(6):1821–1839, 2002.

[224] Takuya Ohko, Kazuo Hiraki, and Yuichiro Anzai. *Adressee learning and message interception for communication load reduction in multiple robot environments*, volume 1221 of *Lecture Notes in Artificial Intelligence*, chapter III, pages 242–258. Springer-Verlag, May 1997.

[225] Dirk Ormoneit and Śaunak Sen. Kernel-based reinforcement learning. *Machine Learning*, 49:161–178, 2002.

[226] Martin J. Osborne. *An Introduction to Game Theory*. Oxford University Press, 2004.

[227] Martin J. Osborne and Ariel Rubinstein. *A Course in Game Theory*. The MIT Press, 1994.

[228] Christos H. Papadimitriou and John N. Tsitsiklis. The complexity of Markov chain decision processes. *Mathematics of Operations Research*, 12(3):441–450, 1987.

[229] Il-Pyung Park and John R. Kender. Topological direction-giving and visual navigation in large environments. *Artificial Intelligence*, 78:355–395, 1995.

[230] K. H. Park, S. B. Cho, and Y. W. Lee. Optimal tracking control of a nonholonomic mobile robot. In *Proceedings of the IEEE International Symposium on Industrial Electronics, 2001 (ISIE'01)*, volume 3, pages 2073–2076, 2001.

[231] Ronald Parr and Stuart Russell. Approximating optimal policies for partially observable stochastic domains. In *Proceedings of the International Joint Conference on Artificial Intelligence*, pages 1088–1094, 1995.

[232] Ronald Parr, Christopher Painter-Wakefield, Lihong Li, and Michael Littman. Analyzing feature generation for value-function approximation. In Zoubin Ghahramani, editor, *Proceedings of the 24th International Conference on Machine Learning (ICML'07)*, pages 737–744. Omni Press, 2007.

[233] Stephen D. Patek and Michael Athans. Optimal robust H_∞ control. Technical Report LIDS-P-2235, Laboratory for Information and Decision Systems, Massachussets Institute of Technology, March 1994.

[234] Mariane Pelletier. On the almost sure asymptotic behaviour of stochastic algorithms. *Stochastic Processes and their Applications*, 78:217–244, 1998.

[235] Jing Peng and Ronald J. Williams. Incremental multi-step Q-learning. *Machine Learning*, 22(1-3):283–290, 1996.

[236] Jing Peng and Ronald J. Williams. Efficient learning and planning within the DYNA framework. *Adaptive Behavior*, 1(4):437–454, 1993.

[237] Theodore J. Perkins. Reinforcement learning for POMDPs based on action values and stochastic optimization. In 199-204, editor, *Proceedings of the 18th National Conference on Artificial Intelligence (AAAI'02)*, page AAAI Press/MIT Press, Menlo Park, CA 2002.

[238] Theodore J. Perkins and Doina Precup. A convergent form of approximate policy iteration. In S. Thrun S. Becker and K. Obermayer, editors, *Advances in Neural Information Processing Systems*, volume 15, pages 1595–1602, Cambridge, MA, 2003. MIT Press.

[239] Jan Peters, Sethu Vijayakumar, and Stefan Schaal:. Natural Actor-Critic. In P. Brazdil A. Jorge L. Torgo J. Gama, R. Camacho, editor, *Proceedings of the 16th European Conference on Machine Learning (ECML'05)*, volume 3720 of *Lecture Notes on Computer Science*, pages 280–291. Springer, 2005.

[240] Alberto Poncela, Eduardo J. Perez, Antonio Bandera, Cristina Urdiales, and Francisco Sandoval. Efficient integration of metric and topological maps for directed exploration of unknown environments. *Robotics and Autonomous Systems*, 41:21–39, 2002.

[241] Pascal Poupart and Craig Boutilier. Bounded finite state controllers. In Sebastian Thrun, Lawrence Saul, and Bernhard Schölkopf, editors, *Advances in Neural Information Processing Systems*, volume 16, Cambridge, MA, 2004. MIT Press.

[242] Pascal Poupart, Nikos Vlassis, Jesse Hoey, and Kevin Regan. An analytic solution to discrete Bayesian reinforcement learning. In *Proceedings of the 23rd International Conference on Machine Learning (ICML'06)*, pages 697–704, New York, NY, 2006. ACM Press.

[243] Rob Powers and Yoav Shoham. New criteria and a new algorithm for learning in multi-agent systems. In Lawrence K. Saul, Yair Weiss, and Léon Bottou, editors, *Advances in Neural Information Processing Systems*, volume 17, pages 1089–1096, Cambridge, MA, 2005. MIT Press.

[244] Doina Precup, Richard S. Sutton, and Sanjoy Dasgupta. Off-policy temporal-difference learning with function approximation. In *Proceedings of the 18th International Conference on Machine Learning (ICML'01)*, pages 417–424, San Francisco, CA, 2001. Morgan Kaufmann.

[245] Suparerk Premvuti and Shin'ichi Yuta. Consideration on the cooperation of multiple autonomous mobile robots. In *Proceedings of the 1990 IEEE International Workshop on Intelligent Robots and Systems (IROS'90)*, pages 59–63, 1990.

[246] David V. Pynadath and Milind Tambe. The communicative multiagent team decision problem: Analyzing teamwork theories and models. *Journal of Artficial Intelligence Research*, 16:389–423, 2002.

[247] David V. Pynadath and Milind Tambe. Multiagent teamwork: Analyzing the optimality and complexity of key theories and models. In *Proceedings of the 1st International Joint Conference on Autonomous Agents and Multiagent Systems (AAMAS'02)*, pages 873–880, New York, NY, 2002. ACM Press.

[248] L. R. Rabiner and B. Juang. An introduction to hidden Markov models. *IEEE Audio, Speech and Signal Processing (ASSP) Magazine*, 3(1):4–16, 1986.

[249] Deepak Ramachandran and Eyal Amir. Bayesian inverse reinforcement learning. In *Proceedings of the 20th International Joint Conference on Artificial Intelligence*, pages 2586–2591, 2007.

[250] P. Ranganathan, Jean-Bernard Hayet, Michel Devy, Seth Hutchinson, and Frederic Lerasle. Topological navigation and qualitative localization for indoor environment using multi-sensory perception. *Robotics and Autonomous Systems*, 41:137–144, 2002.

[251] Bharanee Rathnasabapathy, Prashant Doshi, and Piotr Gmytrasiewicz. Exact solutions for interactive POMDPs using behavioral equivalence. In *Proceedings of the 5th International Joint Conference on Autonomous Agents and Multiagent Systems (AAMAS'06)*, pages 1025–1032, New York, NY, 2006. ACM Press.

[252] Carlos Ribeiro and Csaba Szepesvári. Q-learning combined with spreading: Convergence and results. In *Proceedings of the ISRF-IEE International Conference: Intelligent and Cognitive Systems (Neural Networks Symposium)*, pages 32–36, 1996.

[253] Herbert Robbins and Sutton Monro. A stochastic approximation method. *Annals of Mathematical Statistics*, 22:400–407, 1951.

[254] Julia Robinson. An iterative method of solving a game. *Annals of Mathematics*, 54: 296–301, 1951.

[255] Armando A. Rodriguez and Michael Athans. Multivariable control of a twin lift helicopter system using the LQG/LTR design methodology. Technical Report LIDS-P-1551, Laboratory for Information and Decision Systems, Massachussets Institute of Technology, April 1986.

[256] Maayan Roth, Reid Simmons, and Manuela Veloso. Reasoning about joint beliefs for execution-time communication decisions. In *Proceedings of the 4th International Joint Conferences on Autonomous Agents and Multi-Agent Systems (AAMAS'05)*, pages 1098–1105, New York, NY, 2005. ACM Press.

[257] Maayan Roth, Reid Simmons, and Manuela Veloso. Decentralized communication strategies for coordinated multi-agent policies. In Frank E. Schneider Lynne E. Parker and Alan C. Shultz, editors, *Multi-Robot Systems: From Swarms to Intelligent Automata*, volume III, pages 93–106. Springer, 2005.

[258] Maayan Roth, Reid Simmons, and Manuela Veloso. What to communicate? Execution-time decision in multi-agent POMDPs. In *Proceedings of the 8th International Symposium on Distributed Autonomous Robotic Systems (DARS'06)*, 2006.

[259] Maayan Roth, Reid Simmons, and Manuela Veloso. Exploiting factored representations for decentralized execution in multi-agent teams. In *Proceedings of the 6th International Joint Conference on Autonomous Agents and Multiagent Systems (to appear)*, New York, NY, 2007. ACM Press.

[260] Michael Rovatsos, Felix A. Fischer, and Gerhard Weiß. Hierarchical reinforcement learning for communicating agents. In W. van der Hoek, editor, *Proceedings of the 2nd European Workshop on Multiagent Systems (EUMAS'04)*, pages 593–604, 2004.

[261] Nicholas Roy and Sebastian Thrun. Coastal navigation with mobile robot. In *Proceedings of 1999 Conference on Neural Information Processing Systems (NIPS'99)*, pages 1043–1049, 1999.

[262] Nicholas Roy, Geoffrey Gordon, and Sebastian Thrun. Finding approximate POMDP solutions through belief compression. *Journal of Artificial Intelligence Research*, 23: 1–40, 2005.

[263] Gavin A. Rummery and Mahesan Niranjan. On-line Q-learning using connectionist systems. Technical Report CUED/F-INFENG/TR 166, Cambridge University Engineering Department, 1994.

[264] Paat Rusmevichientong and Benjamin Van Roy. A tractable POMDP for a class of sequencing problems. In *Proceedings of the 17th Conference in Uncertainty in Artificial Intelligence (UAI'01)*, pages 480–487. Morgan Kaufmann, 2001.

[265] John Rust. Numerical dynamic programming in Economics. In H. Amman, D. Kendrick, and J. Rust, editors, *Handbook of Computational Economics*. Elsevier, North Holland, 1996.

[266] John Rust. Using randomization to break the curse of dimensionality. *Econometrica*, 65(3):487–516, 1997.

[267] Byeong-Soon Ryu and Hyun S. Yang. An enhanced topological map for efficient and reliable mobile robot navigation with imprecise sensors. *Robotics and Computer-Integrated Manufacturing*, 14:185–197, 1998.

[268] Arthur L. Samuel. Some studies in machine learning using the game of checkers. *IBM Journal of Research and Development*, 3(3):210–229, 1959. Reprinted in IBM J. Res. Devel. 44:1/2, pp. 206-226, 2000.

[269] Arthur L. Samuel. Some studies in machine learning using the game of checkers II: Recent progress. *IBM Journal of Research and Development*, 11:601–617, 1967.

[270] José Santos-Victor, Raquel Vassallo, and Hans Schneebeli. Topological maps for visual navigation. In *Proceedings of the 1st International Conference on Computer Vision Systems*, pages 21–36, London, UK, 1999. Springer-Verlag.

[271] Jamieson Schulte and Sebastian Thrun. A heuristic search algorithm for acting optimally in Markov decision processes with deterministic hidden state. Unpublished Manuscript, 2001.

[272] Sandip Sen and Gerhard Weiß. *Learning in multiagent systems*, chapter 6, pages 259–298. The MIT Press, 1999.

[273] Lloyd S. Shapley. Stochastig games. *Proceedings of the National Academy of Sciences*, 39:1095–1100, 1953.

[274] Hagit Shatkay and Leslie Pack Kaelbling. Learning topological maps with weak local odometric information. In *Proceedings of the 15th International Joint Conference on Artificial Intelligence (IJCAI'97)*, pages 920–929, 1997.

[275] Li Sheng, Ma Guoliang, and Hu Weili. Stabilization and optimal control of nonholonomic mobile robot. In *Proceedings of the 8th International Conference on Control, Automation, Robotics and Vision*, pages 1427–1430, 2004.

[276] Yoav Shoham, Rob Powers, and Trond Grenager. On the agenda(s) of research on multi-agent learning. In *Proceedings of the AAAI Fall Symposium on Artificial Multi-Agent Learning*, pages 89–95, 2004.

[277] Reid Simmons and Sven Koenig. Probabilistic robot navigation in partially observable environments. In *Proceedings of the 14th International Joint Conference on Artificial Intelligence (IJCAI'95)*, pages 1080–1087, 1995.

[278] Satinder P. Singh, Tommi Jaakkola, and Michael I. Jordan. Learning without state estimation in partially observable environments. In *Proceedings of the 11th International Conference on Machine Learning (ICML'94)*, pages 284–292, 1994.

[279] Satinder P. Singh, Tommi Jaakkola, and Michael I. Jordan. Reinforcement learning with soft state aggregation. In *Advances in Neural Information Processing Systems*, volume 7, pages 361–368, 1994.

[280] Satinder P. Singh, Tommi Jaakkola, Michael Littman, and Csaba Szepesvari. Convergence results for single-step on-policy reinforcement-learning algorithms. *Machine Learning*, 38(3):287–310, 2000.

[281] Satinder P. Singh, Michael Kearns, and Yishay Mansour. Nash convergence of gradient dynamics in general-sum games. In *Proceedings of the 16th Conference on Uncertainty in Artificial Intelligence (UAI'00)*, pages 541–548, 2000.

[282] William D. Smart and Leslie P. Kaelbling. Effective reinforcement learning for mobile robots. In *Proceedings of the 2002 IEEE International Conference on Robotics and Automation (ICRA'02)*, pages 3404–3410, 2002.

[283] Matthijs T. J. Spaan and Nikos Vlassis. Planning with continuous actions in partially observable environments. In *Proceedings of the IEEE International Conference on Robotics and Automation (ICRA'05)*, pages 3469–3474, 2005.

[284] Matthijs T. J. Spaan and Nikos Vlassis. Perseus: Randomized Point-based Value Iteration for POMDPs. *Journal of Artificial Intelligence Research*, 24:195–220, 2005.

[285] Péter Stefán. *Combined use of reinforcement learning and simulated annealing: Algorithms and applications*. PhD thesis, Department of Mechanical Engineering, University of Miskolc, 2003.

[286] Peter Stone and Richard S. Sutton. Scaling reinforcement learning toward RoboCup Soccer. In *Proceedings of the 18th International Conference on Machine Learning (ICML'01)*, pages 537–544, 2001.

[287] Peter Stone and Manuela Veloso. Multiagent systems: A survey from a machine learning perspective. *Autonomous Robots*, 8(3):345–383, 2000.

[288] Malcolm Strens. Efficient hierarchical MCMC for policy search. In *Proceedings of the 21st International Conference on Machine Learning (ICML'04)*, volume 69, pages 97–104, New York, NY, 2004. ACM Press.

[289] Richard S. Sutton. Learning to predict by the methods of temporal differences. *Machine Learning*, 3:9–44, 1988.

[290] Richard S. Sutton. Generalization in reinforcement learning: Successful examples using sparse coarse coding. *Advances in Neural Information Processing Systems*, 8: 1038–1044, 1996.

[291] Richard S. Sutton. DYNA, an integrated architecture for learning, planning, and reacting. *ACM SIGART Bulletin*, 2(4):160–163, 1991.

[292] Richard S. Sutton. Planning by incremental dynamic programming. In *Proceedings of the 8th International Workshop on Machine Learning*, pages 353–357. Morgan Kaufmann, 1991.

[293] Richard S. Sutton. Integrated architectures for learning, planning, and reacting based on approximated dynamic programming. In *Proceedings of the 7th International Conference on Machine Learning*, pages 216–224, San Francisco, CA, 1990. Morgan Kaufmann Publishers.

[294] Richard S. Sutton. First results with DYNA: An integrated architecture for learning, planning, and reacting. *Neural Networks for Control*, pages 179–189, 1990.

[295] Richard S. Sutton and Andrew G. Barto. *Reinforcement Learning: An Introduction*. Adaptive Computation and Machine Learning Series. MIT Press, Cambridge, Massachusetts, third edition, 1998.

[296] Richard S. Sutton, David McAllester, Satinder Singh, and Yishay Mansour. Policy gradient methods for reinforcement learning with function approximation. In *Advances in Neural Information Processing Systems*, volume 12, pages 1057–1063, Cambridge, Massachussets, 2000. MIT Press.

[297] Richard Szabo. Topological navigation of simulated robots using occupancy grid. *International Journal of Advanced Robotic Systems*, 1(3):201 – 206, 2004.

[298] Csaba Szepesvári. The asymptotic convergence rates for Q-learning. In *Proceedings of Neural Information Processing Systems (NIPS'97)*, volume 10, pages 1064–1070, 1997.

[299] Csaba Szepesvári and Michael L. Littman. Generalized Markov decision processes: Dynamic programming and reinforcement learning algorithms. Technical Report CS-96-11, Brown University, Providence - RI, USA, 1996.

[300] Csaba Szepesvári and Michael L. Littman. A unified analysis of value-function-based reinforcement learning algorithms. *Neural Computation*, 11(8):2017–2059, 1999.

[301] Csaba Szepesvári and Rémi Munos. Finite time bounds for sampling based fitted value iteration. In *Proceedings of the 22nd International Conference on Machine Learning (ICML'05)*, volume 119 of *ACM International Conference Proceeding Series*, pages 880–887. ACM Press, 2005.

[302] Csaba Szepesvári and William D. Smart. Interpolation-based Q-learning. In *Proceedings of the 21st International Conference on Machine learning (ICML'04)*, pages 100–107, New York, USA, July 2004. ACM Press.

[303] Daniel Szer, François Charpillet, and Shlomo Zilberstein. MAA*: A heuristic search algorithm for solving decentralized POMDPs. In *Proceedings of the 21st Conference on Uncertainty in Artificial Intelligence (UAI'05)*, pages 576–583, Arlington, Virginia, 2005. AUAI Press.

[304] Vladislav B. Tadić and Arnaud Doucet. Exponential forgetting and geometric ergodicity in state-space models. In *Proceedings of the 41st IEEE Conference on Decision and Control (CDC'02)*, volume 2, pages 2231–2235, 2002.

[305] Ming Tan. Multi-agent reinforcement learning: Independent vs. cooperative agents. In *Readings in Agents*, pages 487–494, San Francisco, CA, USA, 1997. Morgan Kaufmann Publishers Inc.

[306] Stephanus ten Hagen. *Continuous state-space Q-learning for control of nonlinear systems*. PhD thesis, Universiteit van Amsterdam, February 2001.

[307] Gerald Tesauro. TD-Gammon, a self-teaching backgammon program, achieves master-level play. *Neural Computation*, 6(2):215–219, 1994.

[308] Gerald Tesauro. Temporal difference learning and TD-Gammon. *Communications of the ACM*, 38(3):58–68, 1995.

[309] Georgios Theocharous, Khashayar Rohanimanesh, and Sridhar Maharlevan. Learning hierarchical partially observable Markov decision process models for robot navigation. In *Proceedings of the IEEE International Conference on Robotics and Automation (ICRA'01)*, volume 1, pages 511–516, 2001.

[310] Sebastian B. Thrun. Learning occupancy grids with forward models. In *Proceedings of the Conference on Intelligent Robots and Systems (IROS'2001),*, volume 3, pages 1676–1681, 2001.

[311] Sebastian B. Thrun. Monte-Carlo POMDPs. *Advances in Neural Information Processing Systems (NIPS'00)*, 12:1064–1070, 2000.

[312] Sebastian B. Thrun. The role of exploration in learning control. *Handbook of Intelligent Control: Neural, Fuzzy and Adaptive Approaches*, 1992.

[313] Sebastian B. Thrun, Wolfram Burgard, and Dieter Fox. A probabilistic approach to concurrent mapping and localization for mobile robots. *Machine Learning and Autonomous Robots (Joint Issue)*, 31(5):1–25, 1998.

[314] Franiois Tièche and Heinz Hügli. From topological knowledge to geometrical map. *Control Engineering Practice*, 7:797–802, 1999.

[315] Nicola Tomatis. *Hybrid, metric-topological, mobile robot navigation*. PhD thesis, École Polytechnique Fédérale de Lausanne, 2001.

[316] Hui Tong and Timothy X. Brown. Reinforcement learning for call admission control and routing under quality of service constraints in multimedia networks. *Machine Learning*, 49(2-3):111–139, 2000.

[317] Panos E. Trahanias, Savvas Velissaris, and Stelios C. Orphanoudakis. Visual recognition of workspace landmarks for topological navigation. *Autonomous Robots*, 7: 143–158, 1999.

[318] Olivier Trullier and Jean-Arcady Meyer. Animat navigation using a cognitive graph. *Biological Cybernetics*, 83:271–285, 2000.

[319] John N. Tsitsiklis and Michael Athans. On the complexity of decentralized decision making and detection problems. *IEEE Transactions on Automatic Control*, AC-30 (5):440–446, 1985.

[320] John N. Tsitsiklis and Benjamin Van Roy. Feature-based methods for large scale dynamic programming. *Machine Learning*, 22:59–94, 1996.

[321] John N. Tsitsiklis and Benjamin Van Roy. An analysis of temporal-difference learning with function approximation. *IEEE Transactions on Automatic Control*, 42(5):674–690, May 1996.

[322] John N. Tsitsiklis and Benjamin Van Roy. On average versus discounted reward temporal-difference learning. *Machine Learning*, 49(2):179–191, 2002.

[323] William Uther and Manuela Veloso. Adversarial reinforcement learning. Technical Report CMU-CS-03-107, School of Computer Science, Carnegie Mellon University, January 2003.

[324] Alberto Vale. *Mobile robot navigation in outdoor environments: A topological approach*. PhD thesis, Instituto Superior Técnico, February 2005.

[325] Alberto Vale and Maria Isabel Ribeiro. Environment mapping as a topological representation. In *Proceedings of the 11th International Conference on Advanced Robotics*, pages 29–34, 2003.

[326] Benjamin Van Roy. *Learning and value function approximation in complex decision processes*. PhD thesis, Department of Electrical Engineering and Computer Science, Massachusetts Institute of Technology, June 1998.

[327] Nikos Vlassis and Matthijs T. J. Spaan. A fast point-based algorithm for POMDPs. In *Benelearn 2004: Proceedings of the Annual Machine Learning Conference of Belgium and the Netherlands*, pages 170–176, 2004.

[328] Petr Švestka and Mark H. Overmars. Motion planning for car-like robots using a probabilistic learning approach. Technical Report UU-CS-1994-33, Department of Computer Science, Utrecht University, August 1994.

[329] Marilyn A. Walker. An application of reinforcement learning to dialogue strategy selection in a spoken dialogue system for e-mail. *Journal of Artificial Intelligence Research*, 12:387–416, 2000.

[330] Xiaofeng Wang and Tuomas Sandholm. Reinforcement learning to play an optimal Nash equilibrium in team Markov games. In S. Becker, S. Thrun, and K. Obermayer, editors, *Advances in Neural Information Processing Systems*, volume 15, pages 1571–1578. MIT Press, Cambridge, MA, 2003.

[331] Keigo Watanabe. Control of an omnidirectional mobile robot. In *Proceedings of the 2nd International Conferece on Knowledge-Based Intelligent Electronic Systems*, pages 51–60, 1998.

[332] Christopher Watkins and Peter Dayan. Technical note: Q-learning. *Machine Learning*, 8:279–292, 1992.

[333] Christopher J. C. H. Watkins. *Learning from delayed rewards*. PhD thesis, King's College, University of Cambridge, May 1989.

[334] Gerhard Weiß. A multiagent variant of DYNA-Q. In *Proceedings of the 4th International Conference on Multi-Agent Systems (ICMAS'00)*, pages 461–462. IEEE Computer Society, 2000.

[335] Gerhard Weiß. Achieving coordination through combining joint planning and joint learning. In *Proceedings of the 14th European Conference on Artificial Intelligence (ECAI'00)*, pages 388–392. IEEE Computer Society, IOS Press, 2000.

[336] Gerhard Weiß. A multiagent perspective of parallel and distributed machine learning. In *Proceedings of the 2nd International Conference on Autonomous Agents (Agents - 98)*, pages 226–230. ACM Press, 1998.

[337] Shimon Whiteson, Matthew E. Taylor, and Peter Stone. Adaptive tile coding for value function approximation. Technical Report AI-TR-07-339, University of Texas at Austin, 2007.

[338] Marco Wiering and Jürgen Schmidhuber. Efficient model-based exploration. In J. A. Meyer R. Pfeiffer, B. Blumberg and S. W. Wilson, editors, *Proceedings of the 5th International Conference on Simulation of Adaptive Behavior (SAB'98): From Animals to Animats*, volume 5, pages 223–228. MIT Press, 1998.

[339] Stefan B. Williams. *Efficient solutions to autonomous mapping and navigation problems.* PhD thesis, Australian Centre for Field Robotics, University of Sydney, 2001.

[340] Niall Winters, José Gaspar, Gerard Lacey, and José Santos-Victor. Omni-directional vision for robot navigation. In *Proceedings of the IEEE Workshop on Omnidirectional Vision*, pages 21–28, 2000.

[341] Alex Yahja, Sanjiv Singh, and Anthony Stentz. Recent results in path planning for mobile robots operating in vast outdoor environments. In *Proceedings of the 1998 Symposium on Image, Speech, Signal Processing and Robotics*, 1998.

[342] Alex Yahja, Anthony Stentz, Sanjiv Singh, and Barry L. Brumitt. Framed-quadtree path planning for mobile robots operating in sparse environments. In *Proceedings of the IEEE Conference on Robotics and Automation (ICRA'98)*, volume 1, pages 650–655, 1998.

[343] Erfu Yang and Donbing Gu. Multiagent reinforcement learning for multi-robot systems: A survey. Technical Report CSM-404, Department of Computer Science, University of Essex, 2004.

[344] Gary Yen and Travis Hickey. Reinforcement learning algorithms for robotic navigation in dynamic environments. In *Proceedings of the 2002 International Joint Conference on Neural Networks (IJCNN'02)*, volume 2, pages 1444–1449, 2002.

[345] H. Peyton Young. The evolution of conventions. *Econometrica*, 61(1):57–84, January 1993.

[346] Nevin Zhang and Weihong Zhang. Speeding up the convergence of value iteration in partially observable Markov decision processes. *Journal of Artificial Intelligence Research*, 14:29–51, 2001.

[347] Uwe R. Zimmer. Robust world-modelling and navigation in a real world. *Neurocomputing*, 13:247–260, 1996.

[348] Dominique Van Zwynsvoorde, Thierry Siméon, and Rachid Alami. Building topological models for navigation in large-scale environments. In *Proceedings of the 2001 IEEE International Conference on Robotics and Automation (ICRA'01)*, pages 160–165, 2001.

Notation

List of symbols

We now summarize the symbols used throughout the thesis. As there are quite a few of them, we grouped them into different clusters, mainly according to the chapter in which they are introduced.

General purpose

$\mathbb{N}$	Set of natural numbers $(1, 2, \ldots)$;
$\mathbb{R}$	Set of real numbers $(-1, 0, 1/2, \pi, \ldots)$;
$^\top$	Transpose;
$\mathcal{L}(U)$	Linear span of U;
$\langle \cdot, \cdot \rangle$	Inner product;
$\|\cdot\|_2$	Euclidean norm;
$\|\cdot\|_\infty$	Sup-norm;
$\mathcal{T}$	Time (index) set;
t	Time index;
ω	General element of the probability space $(\Omega, \mathcal{F}, \mathbb{P}[\cdot])$;
$\mathbb{P}[E]$	Probability of event E;
$\mathbb{P}_\mu[E]$	Probability of event E w.r.t. the measure μ;
$\mathbb{P}_x[E]$	Probability of event E w.r.t. the point-mass measure at x;
$\mathcal{F}$	General σ-field;
$\mathcal{F}(E)$	General σ-field generated by the event E;
$\mathcal{F}_t$	σ-field generated by the history $\mathcal{H}_t$ up to time t;
$\mathbb{E}[X]$	Expectation of the random variable X;
$\mathbb{E}_\mu[X]$	Expectation of the random variable X w.r.t. the measure μ;
$\mathcal{N}$	General null-measured set;
$\mathsf{N}(0, 1)$	Normal (Gaussian) distribution;
$\mathbf{I}$	Identity operator;
$\mathbb{I}_U$	Indicator function for the set U;
$\mathbb{I}_u$	Indicator function for the set $\{u\}$;
$\mathcal{H}_t$	History up to time instant t;
$\succsim$	Preference relation;
$\succsim^k$	Preference relation for player k;

$\bigotimes$	Generalized optimization operator;
$\mathbf{B}$	Best response function;
ϕ	Similarity function;
$\mathcal{U}$	Lyapunov function;

Markov chains

P	Transition probability kernel;
P^k	k-step transition probability kernel;
P_θ	Controlled transition probability kernel;
K_a	Sampling chain with sampling distribution a;
η_U	Number of visits to set U;
τ_U	First return time to set U;
φ	Irreducibility measure;
ψ	Maximal irreducibility measure;
μ, μ^*	Invariant probability measure;
μ^θ, μ_θ	Invariant probability measure w.r.t. the control parameter θ;
$\mathcal{B}(U)$	Countably generated σ-field on U/Borel σ-field on U;
$\mathcal{B}^+(U)$	Subset of $\mathcal{B}(U)$ consisting on all sets in $\mathcal{B}(U)$ with positive ψ-measure;
$\mathcal{D}$	d-cycle;
$\|\cdot\|$	Total variation norm;
ΔV	Drift operator;
ν	Solution of the Poisson equation;
ν_θ	Solution of the Poisson equation w.r.t. the control parameter θ;
$\mathcal{E}(i)$	Equivalence class of state i under the equivalence relation $\leftrightarrow$;

Chapter 2

$\mathcal{G}$	Graph;
$\mathcal{E}$	Graph edge set/event set;
V	Graph vertex set;
$\{\mathbf{v}_{e_1}, \ldots, \mathbf{v}_{e_n}\}$	Clock structure;
$\mathcal{X}, \mathcal{Y}$	State-spaces;
X_t	Chain state at time t;
$\{X_t\}$	Markov chain;
$\mathcal{A}$	Set of possible actions (action-space);
A_t	Action at time t;
$\{A_t\}$	Control sequence;
δ	General stationary policy;
δ_t	General non-stationary policy;
δ^*	Optimal policy;
P_e	Event-driven transition probability kernel;
P_a	Controlled transition probability kernel;
P_δ	Transition probability kernel w.r.t. the policy δ;

$\mathbb{E}_\delta[\cdot]$	Expectation w.r.t. the policy δ;
r	Reward function;
R	Random reward function;
$\mathcal{R}$	Reward bound;
γ	Discount factor;
V^*	Optimal value function;
V^δ	Value function given the policy δ;
Q^*	Optimal Q-function;

Chapter 3

$\mathbf{T}, \mathbf{T}^\delta$	Dynamic programming operators for V^* and V^δ;
$\mathbf{H}$	Dynamic programming operator for Q^*;
$n_t(i)$	Number of visits to state x up to time t;
$n_t(i,a)$	Number of visits to the state-action pair (i,a) up to time t;
$\hat{\mathsf{P}}$	Estimated transition probability kernel;
$\hat{r}$	Estimated reward function;
α_t	Step-size;
Δ_t	Temporal difference at time t;
δ_g	Greedy policy;
δ_g^Q	Greedy policy w.r.t. the function Q;

Chapter 4

$\hat{\mathbf{H}}$	Approximate dynamic programming operator in kernel-based learning;
θ	Parameter vector;
ξ_i	Basis function;
Ξ	Set of basis functions;
$\mathcal{V}$	Linear space of functions defined on $\mathcal{X}$ generated by a finite set Ξ of linearly independent functions;
$\mathcal{Q}$	Linear space of functions defined on $\mathcal{X} \times \mathcal{A}$ generated by a finite set Ξ of linearly independent functions;
$Q_\theta, Q(\theta)$	Approximate Q-function for the parameter vector θ;
δ_θ	θ-dependent stationary policy;
$(\delta_\theta)_t$	θ-dependent non-stationary policy;
μ_X, μ_Y	Invariant probability measure for the chain $\{X_t\}/\{Y_t\}$;
$\mu_X^\theta, \mu_X^\theta$	Invariant probability measure for the chain $\{X_t\}/\{Y_t\}$ w.r.t. the control parameter θ;
$\mathbb{E}_X[\cdot], \mathbb{E}_Y[\cdot]$	Expectation w.r.t. the invariant measure for the chain $\{X_t\}/\{Y_t\}$;
$\mathbb{E}^\theta[X]$	Expectation of the random variable X given the parameter θ;
$\mathbb{E}_\mu^\theta[X]$	Expectation of the random variable X w.r.t. measure μ and given the parameter θ;
$\mathcal{P}$	Projection operator;
$\mathcal{P}_\mathcal{Q}$	Orthogonal projection into the space $\mathcal{Q}$;
Σ	Normalization matrix;
Σ_{θ_t}	Normalization matrix w.r.t. the control parameter θ_t;

$\mathcal{Z}$	Observation space;
Z_t	Observation at time t;
$\{Z_t\}$	Observation sequence;
O	Observation probability function;
$\mathbb{S}^n$	$n-1$-dimensional probability simplex;
π	Belief state;
$\{\Pi_t\}$	Fully observable Markov chain in belief space;
$\bar{\mathsf{P}}$	Transition probability kernel for the associated fully observable chain;
$\bar{r}$	Reward function for the associated fully observable MDP;
$\mathbf{x}, \mathbf{y}$	Generic points in Euclidean space (contrasting with x, y, which are generic points in the set $\mathcal{X}$);

Chapter 6

N	Set of players $N = \{1, \ldots, N\}$;
$\mathcal{A}$	Set of possible joint actions (action-space);
$\mathcal{A}^k$	Set of possible individual actions for player k;
$\times_{k=1}^{N} \mathcal{A}^k$	Cartesian product of the sets of individual actions;
a^k	Individual action for player k;
a^{-k}	Reduced action profile;
σ	Joint strategy;
σ^k	Individual strategy for player k;
σ^{-k}	Reduced strategy profile;
$\mathsf{P}^{(k)}$	Individual transition probability kernel for player k;
P_σ	Transition probability kernel w.r.t. the strategy σ;
$\mathbb{E}_\sigma[\cdot]$	Expectation w.r.t. the strategy σ;
r	Joint payoff function;
r^k	Payoff function for player k;
$(V^\sigma)^k$	Value function for player k given the strategy σ;
$(V^*)^k$	Optimal value function for player k;

Chapter 7

Γ_i^*	State-game;
Γ_i	Estimated state-game;
VG	Virtual game;
VG_t	Auxiliary virtual game at time t;
$L(\Gamma)$	Maximum length of the shortest path between two vertices in the best response graph of Γ (see Appendix C);
$\mathbf{opt}(x)$	Set of optimal actions at state x;
$\mathbf{opt}^\varepsilon(x)$	Set of ε-optimal actions at x;
BR_t	Best response set for player k at time t;
EP_t	Expected payoff function for player k at time t;
rate	Convergence rate function;

Commonly used tuples

$(\mathcal{X}, \mathsf{P})$	Markov chain;
$(\mathcal{X}, \mathcal{A}, \mathsf{P})$	Controlled Markov chain;
$(\mathcal{X}, \mathcal{A}, \mathsf{P}, r, \gamma)$	Markov decision process;
$(\mathbb{S}^n, \mathcal{A}, \bar{\mathsf{P}}, \bar{r}, \gamma)$	Associated fully observable Markov decision process;
$(\mathcal{X}, \mathcal{Z}, \mathsf{P}, \mathsf{O})$	Partially observable Markov chain;
$(\mathcal{X}, \mathcal{A}, \mathcal{Z}, \mathsf{P}, \mathsf{O}, r, \gamma)$	Partially observable Markov decision process;
$(N, \mathcal{X}, (\mathcal{A}^k), (\mathcal{Z}^k), \mathsf{P}, (\mathsf{O}^k), (r^k), \gamma)$	Partially observable stochastic game;
$(N, \mathcal{X}, (\mathcal{A}^k), (\mathcal{Z}^k), \mathsf{P}, (\mathsf{O}^k), r, \gamma)$	Partially observable team stochastic game;
$\big(N, (\mathcal{A}^k), (\succsim^k)\big), \big(N, (\mathcal{A}^k), (r^k)\big)$	Strategic game;
$\big(\{1,2\}, (\mathcal{A}^k), (r^k)\big)$	Zero-sum strategic game;
$\big(N, (\mathcal{A}^k), r\big)$	Team strategic game;
$\big(N, \mathcal{X}, (\mathcal{A}^k), \mathsf{P}, (r^k), \gamma\big)$	Markov game;
$\big(N, \mathcal{X}, (\mathcal{A}^k), \mathsf{P}, r, \gamma\big)$	Team Markov game;
$\big(N, (\mathcal{A}^k), Q^*(x, \cdot)\big)$	State-game for a team Markov game;

Acronyms

ABAP	Approximate biased adaptive play;
AI	Artificial intelligence;
ARTDP	Adaptive real-time dynamic programming;
ARTQI	Adaptive real-time Q-iteration;
ARTVI	Adaptive real-time value-iteration;
BaGA	Bayesian game approximation;
BAP	Biased adaptive play;
CMAC	Cerebellar model articulation controller;
DEC-POMDP	Decentralized partially observable Markov decision process;
DP	Dynamic programming;
FF-Q	Friend-or-foe Q-learning;
GLIE	Greedy in the limit with infinite exploration;
GMDP	Generalized Markov decision process;
I-POMDP	Interactive partially observable Markov decision process;
JAL	Joint-action learners;
LP	Linear program;
LSTD	Least-squares temporal differencing;
MAS	Multi-agent systems;
MDP	Markov decision process;
MG	Markov game;
POMDP	Partially observable Markov decision process;
POSG	Partially observable stochastic game;
POTMG	Partially observable team Markov game;
RBF	Radial basis function;
RL	Reinforcement learning;

RTDP	Real-time dynamic programming;
SARSA	State-action-reward-state-action (on-policy learning algorithm);
SG	Stochastic game;
TD	Temporal difference;
TMG	Team Markov game;

Basis functions for function approximation

In the representation below, x denotes an element of $\mathcal{X}$; $\mathcal{X}$ is a compact subset of $\mathbb{R}^n$; $\{U_i, i = 1, \ldots, M\}$ is a partition of $\mathcal{X}$; $\{y_i, i = 1, \ldots, M\}$ is a set of points (random or not) in $\mathcal{X}$; Φ is a univariate, non-negative kernel (*e.g.*, the Gaussian kernel); b is an adjustable parameter.

FUNCTION APPROXIMATORS

Grid-based	$\xi_i(x) = \mathbb{I}_{U_i}(x)$;
Plane-based	$\xi_{2i}(x) = x(i); \xi_{2i-1} = 1 - x(i)$;
Exp-based	$\xi_{2i}(x) = e^{-x(i)}; \xi_{2i-1} = 1 - e^{-x(i)}$;
Kernel-based	$\xi_i(x) = \dfrac{\Phi(\|x - y_i\| / b)}{\sum_{j=1}^{M} \Phi(\|x - y_j\| / b)}$;

All functions ξ_i are normalized to yield $\sum_i \xi_i(x) = 1$.

Index