

Kyriakos G. Vamvoudakis  
Yan Wan  
Frank L. Lewis  
Derya Cansever *Editors*

# Handbook of Reinforcement Learning and Control

# **Studies in Systems, Decision and Control**

Volume 325

## **Series Editor**

Janusz Kacprzyk, Systems Research Institute, Polish Academy of Sciences,  
Warsaw, Poland

The series “Studies in Systems, Decision and Control” (SSDC) covers both new developments and advances, as well as the state of the art, in the various areas of broadly perceived systems, decision making and control—quickly, up to date and with a high quality. The intent is to cover the theory, applications, and perspectives on the state of the art and future developments relevant to systems, decision making, control, complex processes and related areas, as embedded in the fields of engineering, computer science, physics, economics, social and life sciences, as well as the paradigms and methodologies behind them. The series contains monographs, textbooks, lecture notes and edited volumes in systems, decision making and control spanning the areas of Cyber-Physical Systems, Autonomous Systems, Sensor Networks, Control Systems, Energy Systems, Automotive Systems, Biological Systems, Vehicular Networking and Connected Vehicles, Aerospace Systems, Automation, Manufacturing, Smart Grids, Nonlinear Systems, Power Systems, Robotics, Social Systems, Economic Systems and other. Of particular value to both the contributors and the readership are the short publication timeframe and the world-wide distribution and exposure which enable both a wide and rapid dissemination of research output.

Indexed by SCOPUS, DBLP, WTI Frankfurt eG, zbMATH, SCImago.

All books published in the series are submitted for consideration in Web of Science.

More information about this series at <http://www.springer.com/series/13304>

Kyriakos G. Vamvoudakis · Yan Wan ·  
Frank L. Lewis · Derya Cansever  
Editors

# Handbook of Reinforcement Learning and Control



Springer

*Editors*

Kyriakos G. Vamvoudakis  
The Daniel Guggenheim School  
of Aerospace Engineering  
Georgia Institute of Technology  
Atlanta, GA, USA

Frank L. Lewis  
Department of Electrical Engineering  
The University of Texas at Arlington  
Arlington, TX, USA

Yan Wan  
Department of Electrical Engineering  
The University of Texas at Arlington  
Arlington, TX, USA

Derya Cansever  
Army Research Office  
Durham, NC, USA

ISSN 2198-4182                    ISSN 2198-4190 (electronic)

Studies in Systems, Decision and Control

ISBN 978-3-030-60989-4            ISBN 978-3-030-60990-0 (eBook)

<https://doi.org/10.1007/978-3-030-60990-0>

© Springer Nature Switzerland AG 2021

This work is subject to copyright. All rights are reserved by the Publisher, whether the whole or part of the material is concerned, specifically the rights of translation, reprinting, reuse of illustrations, recitation, broadcasting, reproduction on microfilms or in any other physical way, and transmission or information storage and retrieval, electronic adaptation, computer software, or by similar or dissimilar methodology now known or hereafter developed.

The use of general descriptive names, registered names, trademarks, service marks, etc. in this publication does not imply, even in the absence of a specific statement, that such names are exempt from the relevant protective laws and regulations and therefore free for general use.

The publisher, the authors and the editors are safe to assume that the advice and information in this book are believed to be true and accurate at the date of publication. Neither the publisher nor the authors or the editors give a warranty, expressed or implied, with respect to the material contained herein or for any errors or omissions that may have been made. The publisher remains neutral with regard to jurisdictional claims in published maps and institutional affiliations.

This Springer imprint is published by the registered company Springer Nature Switzerland AG  
The registered company address is: Gewerbestrasse 11, 6330 Cham, Switzerland

# Preface

Teams of autonomous systems operate in complex dynamic and networked environments with spatiotemporal diverse threats generated by malicious attacks, functional failures, and human errors. With the wide availability of data, Reinforcement Learning (RL) enables adaptive autonomy where the agents automatically learn policies to optimize a reward through interactions with the environment. RL has demonstrated the capability to deal with incomplete information and uncertain environment, and realize full autonomy. However, there are still some critical challenges in the application of such learning and adaptation methods to solve academic and industrial problems, for instance, the curse of dimensionality; optimization in dynamic environments; convergence and performance analysis; safety; nonequilibrium learning; and online implementation. On the other hand, some emerging technologies such as deep learning and multiagent systems will also provide a potential opportunity to further tackle these challenges.

This book presents a variety of state-of-the-art methods for the RL and games in multiagent settings by bringing the leading researchers in the field. These methods cover the theory, future perspectives, and applications of learning-based systems. This book has seven sections. The first part provides an introduction to RL. The second part provides some theoretic perspectives for model-free and model-based learning-based control techniques. The third part incorporates constraints and discusses techniques for verification. A multiagent perspective is presented in the fourth part. Part Five discusses bounded rationality in games and the value of shared information. Finally, applications of RL and multidisciplinary connections are presented in part six and seven, respectively.

We summarize in this Preface the main contributions from each chapter to put them in context.

Part I starts with the Chap. 1, where Derya Cansever provides directions on the future of RL. Then Kiumarsi, Modares, and Lewis present “2” a variety of RL methods to successfully learn the solution to the optimal control and game problems online and using measured data along the system trajectories. Powell in Chap. 3 makes the case that the modeling framework of RL, inherited from discrete Markov decision processes, is quite limited and suggests a stochastic control framework that based on the core problem of optimizing over policies. Chapter 4 by Devraj, Bušić’, and Meyn contains a survey of the new class of Zap RL algorithms that can achieve convergence almost universally while also guaranteeing the optimal rate of convergence. Greene, Deptula, Kamalapurkar, and Dixon discuss in Chap. 5, mixed density RL-based approximate optimal control methods applied to deterministic systems. Chapter 6 and Mohammadi, Soltanolkotabi, and Jovanović review recent results on the convergence and sample complexity of the random search method for the infinite horizon continuous-time linear quadratic regulator problem with unknown model parameters.

Part II starts with Yang, Wunsch II, and Yin in Chap. 7, that present a Hamiltonian-driven framework of adaptive dynamic programming for continuous-time nonlinear systems. Rizvi, Wei, and Lin, in Chap. 8 provides RL-based methods for solving the optimal stabilization problem for time delay systems with unknown delays and unknown system dynamics. Chapter 9, written by Moghadam, Jagannathan, Narayanan, and Raghavan, presents a RL method for partially unknown linear continuous-time systems with state delays. Kanellopoulos, Vamvoudakis, Gupta, and Antsaklis in Chap. 10 investigate the problem of verifying desired properties of large-scale RL systems that operate in uncertain and adversarial environments. Chapter 11 and Benosman, Chakrabarty, and Borggaard show a method for RL-based model reduction for partial differential equations and specifically to Burgers equations.

Part III starts with Chap. 12 by Zhang, Yang, and Başar review the theoretical results of multiagent RL algorithms mainly within two representative frameworks, Markov/stochastic games and extensive-form games. Chapter 13 written by Liu, Wan, Lin, Lewis, Xie, and Jalaian describes the use of computationally effective uncertainty evaluation methods for adaptive optimal control, including learning control and differential games. Lin, Montúfar, and Osher in Chap. 14 demonstrate a method, to obtain decentralized multiagents through a top-down approach: first by obtaining a solution with a centralized controller, and then decentralizing using imitation learning. Chapter 15 from Yang and Hespanha provides a security perspective in Stackelberg games and for the case that the attack objective and capacity are unknown, they propose a learning-based approach that predicts the routing cost using a neural network and minimizes the predicted cost via projected gradient descent.

Part IV starts with Chap. 16 from Kokolakis, Kanellopoulos, and Vamvoudakis that present a unified framework of bounded rationality for control systems as this can be employed in a coordinated unmanned aerial vehicle tracking RL problem. Tsiotras in Chap. 17 utilizes bounded-rationality ideas for generating suitable hierarchical abstractions to handle demanding tasks under time and other resource constraints, when exact optimality/rationality may be elusive. Chapter 18 and authors, Zhang

and Liu, review existing literature on the fairness of data-driven sequential decision-making. Sledge and Príncipe in Chap. 19 provide a way to resolve the exploration-exploitation dilemma in RL.

Part V starts with Chap. 20 from Castagno and Atkins, that present the offline construction of a landing site database using publicly available data sources with a focus on rooftop sites. Surana in Chap. 21 presents an industrial perspective of RL. Chapter 22 with authors, Huang, Jiang, Malisoff, and Cui develop RL-based shared control designs for semi-autonomous vehicles with a human in the loop. Wang and Wang in Chap. 23 present a decision-making framework subject to uncertainties that are represented by a set of random variables injected to the system as a group of inputs in both performance and constraint equations.

Finally, Part VI starts with Chap. 24 from Poveda and Teel, that provided a framework for the analysis of RL-based controllers that can safely and systematically integrate the intrinsic continuous-time dynamics and discrete-time dynamics that emerge in cyber-physical systems. Haddad in Chap. 25 looks to systems biology, neurophysiology, and thermodynamics for inspiration in developing innovative architectures for control and learning. The last chapter of the book, written by Rajagopal, Zhang, Balakrishnan, Fakhari, and Busemeyer, reviews work on a new choice rule based on an amplitude amplification algorithm originally developed in quantum computing.

In summary, this book presents a variety of challenging theoretical problems coupled with real practical applications for RL systems. The future directions are also covered in individual chapters.

Atlanta, USA  
Arlington, USA  
Arlington, USA  
Durham, USA

Kyriakos G. Vamvoudakis  
Yan Wan  
Frank L. Lewis  
Derya Cansever

# Contents

## Part I Theory of Reinforcement Learning for Model-Free and Model-Based Control and Games

<b>1</b>	<b>What May Lie Ahead in Reinforcement Learning .....</b>	<b>3</b>
	Derya Cansever	
	References .....	5
<b>2</b>	<b>Reinforcement Learning for Distributed Control and Multi-player Games .....</b>	<b>7</b>
	Bahare Kiumarsi, Hamidreza Modares, and Frank Lewis	
2.1	Introduction .....	7
2.2	Optimal Control of Continuous-Time Systems .....	9
2.2.1	IRL with Experience Replay Learning Technique [12, 13] .....	11
	2.2.1.1 Off-Policy IRL Algorithm [14–16] .....	12
2.2.2	$\mathcal{H}_\infty$ Control of CT Systems .....	14
	2.2.2.1 Off-Policy IRL [15, 21] .....	16
2.3	Nash Games .....	17
2.4	Graphical Games .....	20
	2.4.1 Off-Policy RL for Graphical Games .....	22
2.5	Output Synchronization of Multi-agent Systems .....	23
2.6	Conclusion and Open Research Directions .....	26
	References .....	26
<b>3</b>	<b>From Reinforcement Learning to Optimal Control: A Unified Framework for Sequential Decisions .....</b>	<b>29</b>
	Warren B. Powell	
3.1	Introduction .....	29
3.2	The Communities of Sequential Decisions .....	31
3.3	Stochastic Optimal Control Versus Reinforcement Learning .....	33
	3.3.1 Stochastic Control .....	34
	3.3.2 Reinforcement Learning .....	37
	3.3.3 A Critique of the MDP Modeling Framework .....	41

3.3.4	Bridging Optimal Control and Reinforcement Learning .....	42
3.4	The Universal Modeling Framework .....	44
3.4.1	Dimensions of a Sequential Decision Model .....	45
3.4.2	State Variables .....	47
3.4.3	Objective Functions .....	49
3.4.4	Notes .....	51
3.5	Energy Storage Illustration .....	52
3.5.1	A Basic Energy Storage Problem .....	53
3.5.2	With a Time-Series Price Model .....	55
3.5.3	With Passive Learning .....	55
3.5.4	With Active Learning .....	56
3.5.5	With Rolling Forecasts .....	56
3.5.6	Remarks .....	57
3.6	Designing Policies .....	58
3.6.1	Policy Search .....	58
3.6.2	Lookahead Approximations .....	60
3.6.3	Hybrid Policies .....	63
3.6.4	Remarks .....	64
3.6.5	Stochastic Control, Reinforcement Learning, and the Four Classes of Policies .....	65
3.7	Policies for Energy Storage .....	67
3.8	Extension to Multi-agent Systems .....	69
3.9	Observations .....	71
	References .....	72
4	<b>Fundamental Design Principles for Reinforcement Learning Algorithms .....</b>	75
	Adithya M. Devraj, Ana Bušić, and Sean Meyn	
4.1	Introduction .....	76
4.1.1	Stochastic Approximation and Reinforcement Learning .....	77
4.1.2	Sample Complexity Bounds .....	78
4.1.3	What Will You Find in This Chapter? .....	79
4.1.4	Literature Survey .....	80
4.2	Stochastic Approximation: New and Old Tricks .....	81
4.2.1	What is Stochastic Approximation? .....	82
4.2.2	Stochastic Approximation and Learning .....	84
4.2.2.1	Monte Carlo .....	84
4.2.2.2	Temporal Difference Learning .....	86
4.2.3	Stability and Convergence .....	88
4.2.4	Zap–Stochastic Approximation .....	89
4.2.5	Rates of Convergence .....	91
4.2.5.1	White Noise Model .....	92
4.2.5.2	Markovian Model .....	93

4.2.5.3	Implications and Matrix Gain Stochastic Approximation .....	95
4.2.6	Optimal Convergence Rate .....	96
4.2.6.1	Stochastic Newton–Raphson .....	96
4.2.6.2	Zap Stochastic Approximation .....	97
4.2.6.3	Other Optimal Algorithms .....	98
4.2.7	TD and LSTD Algorithms .....	99
4.3	Zap Q-Learning: Fastest Convergent Q-Learning .....	101
4.3.1	Markov Decision Processes .....	101
4.3.2	Value Functions and the Bellman Equation .....	102
4.3.3	Q-Learning .....	104
4.3.4	Tabular Q-Learning .....	105
4.3.5	Convergence and Rate of Convergence .....	108
4.3.6	Zap Q-Learning .....	112
4.3.6.1	Main Results .....	113
4.3.6.2	Zap ODE and Policy Iteration .....	114
4.3.6.3	Overview of Proofs .....	115
4.4	Numerical Results .....	118
4.4.1	Finite State-Action MDP .....	119
4.4.2	Optimal Stopping in Finance .....	124
4.4.2.1	Approximations to the Optimal Stopping Time Problem .....	125
4.4.2.2	Experimental Results .....	126
4.4.2.3	Asymptotic Variance of the Discounted Reward .....	128
4.5	Zap-Q with Nonlinear Function Approximation .....	128
4.5.1	Choosing the Eligibility Vectors .....	130
4.5.2	Theory and Challenges .....	131
4.5.3	Regularized Zap-Q .....	131
4.6	Conclusions and Future Work .....	132
	References .....	134
5	<b>Mixed Density Methods for Approximate Dynamic Programming .....</b>	139
	Max L. Greene, Patryk Deptula, Rushikesh Kamalapurkar, and Warren E. Dixon	
5.1	Introduction .....	140
5.2	Unconstrained Affine-Quadratic Regulator .....	142
5.3	Regional Model-Based Reinforcement Learning .....	150
5.3.1	Preliminaries .....	151
5.3.2	Regional Value Function Approximation .....	151
5.3.3	Bellman Error .....	152
5.3.3.1	Extension to Unknown Dynamics .....	153
5.3.4	Actor and Critic Update Laws .....	155
5.3.5	Stability Analysis .....	156

5.3.6	Summary .....	158
5.4	Local (State-Following) Model-Based Reinforcement Learning .....	159
5.4.1	StaF Kernel Functions .....	160
5.4.2	Local Value Function Approximation .....	161
5.4.3	Actor and Critic Update Laws .....	162
5.4.4	Analysis .....	163
5.4.5	Stability Analysis .....	165
5.4.6	Summary .....	166
5.5	Combining Regional and Local State-Following Approximations .....	167
5.6	Reinforcement Learning with Sparse Bellman Error Extrapolation .....	168
5.7	Conclusion .....	168
	References .....	169
<b>6</b>	<b>Model-Free Linear Quadratic Regulator .....</b>	<b>173</b>
	Hesameddin Mohammadi, Mahdi Soltanolkotabi, and Mihailo R. Jovanović	
6.1	Introduction to a Model-Free LQR Problem .....	173
6.2	A Gradient-Based Random Search Method .....	175
6.3	Main Results .....	176
6.4	Proof Sketch .....	177
6.4.1	Controlling the Bias .....	179
6.4.2	Correlation of $\hat{\nabla}f(K)$ and $\nabla f(K)$ .....	181
6.5	An Example .....	182
6.6	Thoughts and Outlook .....	183
	References .....	185

## Part II Constraint-Driven and Verified RL

<b>7</b>	<b>Adaptive Dynamic Programming in the Hamiltonian-Driven Framework .....</b>	<b>189</b>
	Yongliang Yang, Donald C. Wunsch II, and Yixin Yin	
7.1	Introduction .....	190
7.1.1	Literature Review .....	190
7.1.2	Motivation .....	191
7.1.3	Structure .....	192
7.2	Problem Statement .....	192
7.3	Hamiltonian-Driven Framework .....	195
7.3.1	Policy Evaluation .....	195
7.3.2	Policy Comparison .....	198
7.3.3	Policy Improvement .....	201
7.4	Discussions on the Hamiltonian-Driven ADP .....	206
7.4.1	Implementation with Critic-Only Structure .....	206

7.4.2	Connection to Temporal Difference Learning .....	209
7.4.2.1	Continuous-Time Integral Temporal Difference .....	209
7.4.2.2	Continuous-Time Least Squares Temporal Difference .....	209
7.4.3	Connection to Value Gradient Learning .....	210
7.5	Simulation Study .....	210
7.6	Conclusion .....	213
	References .....	213
<b>8</b>	<b>Reinforcement Learning for Optimal Adaptive Control of Time Delay Systems .....</b>	<b>215</b>
	Syed Ali Asad Rizvi, Yusheng Wei, and Zongli Lin	
8.1	Introduction .....	216
8.2	Problem Description .....	218
8.3	Extended State Augmentation .....	219
8.4	State Feedback Q-Learning Control of Time Delay Systems .....	228
8.5	Output Feedback Q-Learning Control of Time Delay Systems .....	232
8.6	Simulation Results .....	238
8.7	Conclusions .....	240
	References .....	242
<b>9</b>	<b>Optimal Adaptive Control of Partially Uncertain Linear Continuous-Time Systems with State Delay .....</b>	<b>243</b>
	Rohollah Moghadam, S. Jagannathan, Vignesh Narayanan, and Krishnan Raghavan	
9.1	Introduction .....	244
9.2	Problem Statement .....	245
9.3	Linear Quadratic Regulator Design .....	246
9.3.1	Periodic Sampled Feedback .....	247
9.3.2	Event Sampled Feedback .....	249
9.4	Optimal Adaptive Control .....	252
9.4.1	Periodic Sampled Feedback .....	252
9.4.2	Event Sampled Feedback .....	257
9.4.3	Hybrid Reinforcement Learning Scheme .....	259
9.5	Perspectives on Controller Design with Image Feedback .....	260
9.6	Simulation Results .....	264
9.6.1	Linear Quadratic Regulator with Known Internal Dynamics .....	265
9.6.2	Optimal Adaptive Control with Unknown Drift Dynamics .....	265
9.7	Conclusion .....	267
	References .....	270

<b>10 Dissipativity-Based Verification for Autonomous Systems in Adversarial Environments .....</b>	273
Aris Kanellopoulos, Kyriakos G. Vamvoudakis, Vijay Gupta, and Panos Antsaklis	
10.1 Introduction .....	273
10.1.1 Related Work .....	275
10.1.2 Contributions .....	276
10.1.3 Structure .....	276
10.1.4 Notation .....	276
10.2 Problem Formulation .....	277
10.2.1 $(Q, S, R)$ -Dissipative and $L_2$ -Gain Stable Systems .....	278
10.3 Learning-Based Distributed Cascade Interconnection .....	279
10.4 Learning-Based $L_2$ -Gain Composition .....	281
10.4.1 Q-Learning for $L_2$ -Gain Verification .....	281
10.4.2 $L_2$ -Gain Model-Free Composition .....	285
10.5 Learning-Based Lossless Composition .....	286
10.6 Discussion .....	288
10.7 Conclusion and Future Work .....	289
References .....	290
<b>11 Reinforcement Learning-Based Model Reduction for Partial Differential Equations: Application to the Burgers Equation .....</b>	293
Mouhacine Benosman, Ankush Chakrabarty, and Jeff Borggaard	
11.1 Introduction .....	293
11.2 Basic Notation and Definitions .....	295
11.3 RL-Based Model Reduction of PDEs .....	296
11.3.1 Reduced-Order PDE Approximation .....	296
11.3.2 Proper Orthogonal Decomposition for ROMs .....	297
11.3.3 Closure Models for ROM Stabilization .....	298
11.3.4 Main Result: RL-Based Closure Model .....	299
11.4 Extremum Seeking Based Closure Model Auto-Tuning .....	304
11.5 The Case of the Burgers Equation .....	305
11.6 Conclusion .....	312
References .....	316

### Part III Multi-agent Systems and RL

<b>12 Multi-Agent Reinforcement Learning: A Selective Overview of Theories and Algorithms .....</b>	321
Kaiqing Zhang, Zhuoran Yang, and Tamer Başar	
12.1 Introduction .....	322
12.2 Background .....	324
12.2.1 Single-Agent RL .....	324
12.2.1.1 Value-Based Methods .....	325
12.2.1.2 Policy-Based Methods .....	326

12.2.2	Multi-Agent RL Framework .....	327
12.2.2.1	Markov/Stochastic Games .....	327
12.2.2.2	Extensive-Form Games .....	330
12.3	Challenges in MARL Theory .....	332
12.3.1	Non-unique Learning Goals .....	332
12.3.2	Non-stationarity .....	333
12.3.3	Scalability Issue .....	334
12.3.4	Various Information Structures .....	334
12.4	MARL Algorithms with Theory .....	336
12.4.1	Cooperative Setting .....	336
12.4.1.1	Homogeneous Agents .....	336
12.4.1.2	Decentralized Paradigm with Networked Agents .....	339
12.4.1.3	Partially Observed Model .....	344
12.4.2	Competitive Setting .....	345
12.4.2.1	Value-Based Methods .....	346
12.4.2.2	Policy-Based Methods .....	349
12.4.3	Mixed Setting .....	355
12.5	Application Highlights .....	358
12.5.1	Cooperative Setting .....	358
12.5.2	Competitive Setting .....	361
12.5.3	Mixed Settings .....	365
12.6	Conclusions and Future Directions .....	366
	References .....	368
<b>13</b>	<b>Computational Intelligence in Uncertainty Quantification for Learning Control and Differential Games .....</b>	<b>385</b>
	Mushuang Liu, Yan Wan, Zongli Lin, Frank L. Lewis, Junfei Xie, and Brian A. Jalaian	
13.1	Introduction .....	386
13.2	Problem Formulation of Optimal Control for Uncertain Systems .....	387
13.2.1	Optimal Control for Systems with Parameters Modulated by Multi-dimensional Uncertainties .....	387
13.2.1.1	Systems with Parameters Modulated by Multi-dimensional Uncertainties .....	387
13.2.1.2	Optimal Control for Systems with Parameters Modulated by Multi-dimensional Uncertainties .....	388
13.2.2	Optimal Control for Random Switching Systems .....	389
13.2.2.1	Random Switching Models .....	389
13.2.2.2	Optimal Control for Random Switching Systems .....	390
13.3	Effective Uncertainty Evaluation Methods .....	391
13.3.1	Problem Formulation .....	391

13.3.2	The MPCM .....	391
13.3.3	The MPCM-OFFD .....	393
13.4	Optimal Control Solutions for Systems with Parameter Modulated by Multi-dimensional Uncertainties .....	394
13.4.1	Reinforcement Learning-Based Stochastic Optimal Control .....	394
13.4.2	Q-Learning-Based Stochastic Optimal Control .....	396
13.5	Optimal Control Solutions for Random Switching Systems .....	397
13.5.1	Optimal Controller for Random Switching Systems .....	397
13.5.2	Effective Estimator for Random Switching Systems .....	399
13.6	Differential Games for Systems with Parameters Modulated by Multi-dimensional Uncertainties .....	401
13.6.1	Stochastic Two-Player Zero-Sum Game .....	401
13.6.1.1	On-Policy IRL .....	403
13.6.1.2	Off-Policy IRL .....	404
13.6.2	Multi-player Nonzero-Sum Game .....	405
13.6.2.1	On-Policy IRL .....	407
13.6.2.2	Off-Policy IRL .....	408
13.7	Applications .....	409
13.7.1	Traffic Flow Management Under Uncertain Weather .....	409
13.7.2	Learning Control for Aerial Communication Using Directional Antennas (ACDA) Systems .....	411
13.8	Summary .....	415
	References .....	416
14	<b>A Top-Down Approach to Attain Decentralized Multi-agents .....</b>	419
	Alex Tong Lin, Guido Montúfar, and Stanley J. Osher	
14.1	Introduction .....	420
14.2	Background .....	421
14.2.1	Reinforcement Learning .....	421
14.2.2	Multi-agent Reinforcement Learning .....	423
14.3	Centralized Learning, But Decentralized Execution .....	424
14.3.1	A Bottom-Up Approach .....	425
14.3.2	A Top-Down Approach .....	425
14.4	Centralized Expert Supervises Multi-agents .....	425
14.4.1	Imitation Learning .....	425
14.4.2	CESMA .....	426
14.5	Experiments .....	427
14.5.1	Decentralization Can Achieve Centralized Optimality .....	428
14.5.2	Expert Trajectories Versus Multi-agent Trajectories .....	428

14.6 Conclusion .....	429
References .....	430
<b>15 Modeling and Mitigating Link-Flooding Distributed Denial-of-Service Attacks via Learning in Stackelberg Games .....</b>	<b>433</b>
Guosong Yang and João P. Hespanha	
15.1 Introduction .....	433
15.2 Routing and Attack in Communication Network .....	436
15.3 Stackelberg Game Model .....	438
15.4 Optimal Attack and Stackelberg Equilibria for Malicious Adversaries .....	439
15.4.1 Optimal Attack and Stackelberg Equilibria for Networks with Identical Links .....	442
15.5 Mitigating Attacks via Learning .....	450
15.5.1 Predicting the Routing Cost .....	451
15.5.2 Minimizing the Predicted Routing Cost .....	451
15.6 Simulation Study .....	452
15.6.1 Discussion .....	459
15.7 Conclusion .....	461
References .....	461
<b>Part IV Bounded Rationality and Value of Information in RL and Games</b>	
<b>16 Bounded Rationality in Differential Games: A Reinforcement Learning-Based Approach .....</b>	<b>467</b>
Nick-Marios T. Kokolakis, Aris Kanellopoulos, and Kyriakos G. Vamvoudakis	
16.1 Introduction .....	467
16.1.1 Related Work .....	468
16.2 Problem Formulation .....	469
16.2.1 Nash Equilibrium Solutions for Differential Games .....	469
16.3 Boundedly Rational Game Solution Concepts .....	472
16.4 Cognitive Hierarchy for Adversarial Target Tracking .....	474
16.4.1 Problem Formulation .....	474
16.4.1.1 Vehicle Dynamics .....	475
16.4.1.2 Relative Kinematics .....	475
16.4.1.3 Differential Game Formulation .....	476
16.4.2 Zero-Sum Game .....	478
16.4.3 Cognitive Hierarchy .....	479
16.4.3.1 Level-0 (Anchor) Policy .....	479
16.4.3.2 Level- $k$ Policies .....	480
16.4.4 Coordination with Nonequilibrium Game-Theoretic Learning .....	481
16.4.5 Simulation .....	485
16.5 Conclusion and Future Work .....	486
References .....	488

<b>17 Bounded Rationality in Learning, Perception, Decision-Making, and Stochastic Games .....</b>	<b>491</b>
Panagiotis Tsiorras	
<b>17.1 The Autonomy Challenge .....</b>	<b>491</b>
17.1.1 The Case of Actionable Data .....	492
17.1.2 The Curse of Optimality .....	493
<b>17.2 How to Move Forward .....</b>	<b>494</b>
17.2.1 Bounded Rationality for Human-Like Decision-Making .....	495
17.2.2 Hierarchical Abstractions for Scalability .....	496
<b>17.3 Sequential Decision-Making Subject to Resource Constraints .....</b>	<b>497</b>
17.3.1 Standard Markov Decision Processes .....	498
17.3.2 Information-Limited Markov Decision Processes .....	500
<b>17.4 An Information-Theoretic Approach for Hierarchical Decision-Making .....</b>	<b>505</b>
17.4.1 Agglomerative Information Bottleneck for Quadtree Compression .....	506
17.4.2 Optimal Compression of Quadtrees .....	508
17.4.3 The Q-Tree Search Algorithm .....	509
<b>17.5 Stochastic Games and Bounded Rationality .....</b>	<b>511</b>
17.5.1 Stochastic Pursuit–Evasion .....	513
17.5.2 Level-k Thinking .....	515
17.5.3 A Pursuit–Evasion Game in a Stochastic Environment .....	517
<b>17.6 Conclusions .....</b>	<b>519</b>
<b>References .....</b>	<b>520</b>
<b>18 Fairness in Learning-Based Sequential Decision Algorithms: A Survey .....</b>	<b>525</b>
Xueru Zhang and Mingyan Liu	
<b>18.1 Introduction .....</b>	<b>525</b>
<b>18.2 Preliminaries .....</b>	<b>528</b>
18.2.1 Sequential Decision Algorithms .....	528
18.2.2 Notions of Fairness .....	528
<b>18.3 (Fair) Sequential Decision When Decisions Do Not Affect Underlying Population .....</b>	<b>529</b>
18.3.1 Bandits, Regret, and Fair Regret .....	529
18.3.2 Fair Experts and Expert Opinions .....	533
18.3.3 Fair Policing .....	535
<b>18.4 (Fair) Sequential Decision When Decisions Affect Underlying Population .....</b>	<b>535</b>
18.4.1 Two-Stage Models .....	536
18.4.1.1 Effort-Based Fairness .....	538

18.4.1.2	A Two-Stage Model in College Admissions .....	540
18.4.2	Long-Term Impacts on the Underlying Population .....	542
18.4.2.1	Effects of Decisions on the Evolution of Features .....	542
18.4.2.2	Fairness Intervention on Labor Market .....	547
18.4.2.3	Effects of Decisions on Group Representation .....	549
18.4.2.4	Combined Effects on Group Representation and Features .....	552
18.4.2.5	Fairness in Reinforcement Learning Problems .....	553
	References .....	553
<b>19</b>	<b>Trading Utility and Uncertainty: Applying the Value of Information to Resolve the Exploration–Exploitation Dilemma in Reinforcement Learning .....</b>	<b>557</b>
	Isaac J. Sledge and José C. Príncipe	
19.1	Introduction .....	558
19.2	Exploring Single-State, Multiple-Action Markov Decision Processes .....	564
19.2.1	Literature Survey .....	564
19.2.2	Methodology .....	567
19.2.2.1	Value of Information .....	568
19.2.2.2	Value of Information Optimization .....	571
19.2.3	Simulations and Analyses .....	575
19.2.3.1	Simulation Preliminaries .....	575
19.2.3.2	Value of Information Results and Analysis .....	576
19.2.3.3	Methodological Comparisons .....	581
19.2.4	Conclusions .....	586
19.3	Exploring Multiple-state, Multiple-Action Markov Decision Processes .....	587
19.3.1	Literature Survey .....	587
19.3.2	Methodology .....	590
19.3.2.1	Value of Information .....	592
19.3.2.2	Value of Information Optimization .....	595
19.3.3	Simulations and Analyses .....	598
19.3.3.1	Simulation Preliminaries .....	599
19.3.3.2	Value of Information Results and Analyses .....	600
19.3.3.3	Methodological Comparisons .....	605
19.3.4	Conclusions .....	606
	References .....	607

**Part V Applications of RL**

<b>20 Map-Based Planning for Small Unmanned Aircraft Rooftop Landing</b> .....	613
J. Castagno and E. Atkins	
20.1 Introduction .....	613
20.2 Background .....	615
20.2.1 Sensor-Based Planning .....	615
20.2.2 Map-Based Planning .....	616
20.2.3 Multi-goal Planning .....	617
20.2.4 Urban Landscape and Rooftop Landings .....	618
20.3 Preliminaries .....	619
20.3.1 Coordinates and Landing Sites .....	619
20.3.2 3D Path Planning with Mapped Obstacles .....	620
20.4 Landing Site Database .....	620
20.4.1 Flat-Like Roof Identification .....	621
20.4.2 Flat Surface Extraction for Usable Landing Area .....	622
20.4.3 Touchdown Points .....	623
20.4.4 Landing Site Risk Model .....	625
20.4.4.1 Vehicle Cost .....	625
20.4.4.2 Terrain Cost .....	625
20.4.4.3 Area Cost .....	626
20.4.4.4 Cumulative Area Cost .....	626
20.4.4.5 Property Cost .....	627
20.4.4.6 Human Occupancy Risk Mapping .....	627
20.5 Three-Dimensional Maps for Path Planning .....	628
20.6 Planning Risk Metric Analysis and Integration .....	630
20.6.1 Real-Time Map-Based Planner Architecture .....	630
20.6.2 Trade-Off Between Landing Site and Path Risk .....	631
20.6.3 Multi-goal Planner .....	632
20.6.3.1 Theory .....	632
20.6.3.2 Multi-goal Path Planning Algorithm .....	634
20.7 Maps and Simulation Results .....	635
20.7.1 Landing Sites and Risk Maps .....	636
20.7.2 Case Studies .....	637
20.7.3 Urgent Landing Statistical Analysis .....	641
20.7.3.1 Minimum Radius Footprint .....	641
20.7.3.2 Performance Benchmarks .....	641
20.8 Conclusion .....	644
References .....	644
<b>21 Reinforcement Learning: An Industrial Perspective</b> .....	647
Amit Surana	
21.1 Introduction .....	647
21.2 RL Applications .....	648

21.2.1	Sensor Management in Intelligence, Surveillance, and Reconnaissance .....	649
21.2.2	High Level Reasoning in Autonomous Navigation .....	649
21.2.3	Advanced Manufacturing Process Control .....	650
21.2.4	Maintenance, Repair, and Overhaul Operations .....	652
21.2.5	Human–Robot Collaboration .....	652
21.3	Case Study I: Optimal Sensor Tasking .....	653
21.3.1	Sensor Tasking as a Stochastic Optimal Control Problem .....	653
21.3.2	Multi-Arm Bandit Problem Approximation .....	654
21.3.2.1	Gittin’s Index in a Simplified Setting .....	656
21.3.2.2	Tracking with Multiple Sensors Combined with Search .....	656
21.3.3	Numerical Study .....	658
21.4	Case Study II: Deep Reinforcement Learning for Advanced Manufacturing Control .....	659
21.4.1	Cold Spray Control Problem .....	660
21.4.2	Guided Policy Search .....	662
21.4.3	Simulation Results .....	665
21.5	Future Outlook .....	667
	References .....	668
<b>22</b>	<b>Robust Autonomous Driving with Human in the Loop .....</b>	<b>673</b>
	Mengzhe Huang, Zhong-Ping Jiang, Michael Malisoff, and Leilei Cui	
22.1	Introduction .....	673
22.2	Mathematical Modeling of Human–Vehicle Interaction .....	676
22.2.1	Vehicle Lateral Dynamics .....	676
22.2.2	Interconnected Human–Vehicle Model .....	678
22.3	Model-Based Control Design .....	679
22.3.1	Discretization of Differential-Difference Equations .....	679
22.3.2	Formulation of the Shared Control Problem .....	681
22.3.3	Model-Based Optimal Control Design .....	682
22.4	Learning-Based Optimal Control for Cooperative Driving .....	683
22.5	Numerical Results .....	686
22.5.1	Algorithmic Implementation .....	686
22.5.2	Comparisons and Discussions for ADP-Based Shared Control Design .....	688
22.6	Conclusions and Future Work .....	690
	References .....	690
<b>23</b>	<b>Decision-Making for Complex Systems Subjected to Uncertainties—A Probability Density Function Control Approach .....</b>	<b>693</b>
	Aiping Wang and Hong Wang	
23.1	Introduction .....	693

23.2	Integrated Modeling Perspectives—Ordinary Algebra Versus {Max, +} Algebra .....	696
23.2.1	Process Level Modeling via Ordinary Algebra Systems .....	697
23.2.2	{Max, +} Algebra-Based Modeling .....	697
23.2.3	Learning Under Uncertainties—PDF Shaping of Modeling Error-Based Approach .....	699
23.3	Human-in-the-Loop Consideration: Impact of Uncertainties in Decision-Making Phase .....	702
23.4	Optimization Under Uncertainties Impacts .....	704
23.4.1	Formulation of Optimization as a Feedback Control Design Problem—Optimization is a Special Case of Feedback Control System Design ....	704
23.4.1.1	Source of Uncertainties in Decision-Making Phase .....	707
23.5	A Generalized Framework for Decision-Making Using PDF Shaping Approach .....	709
23.5.1	PDF Shaping for the Performance Function .....	709
23.5.2	Dealing with the Constraint .....	710
23.5.3	Dealing with Dynamic Constraint .....	713
23.5.4	A Total Probabilistic Solution .....	714
23.5.4.1	Relations to Chance Constrained Optimization .....	714
23.5.5	Uncertainties in Performance Function and Constraints .....	716
23.6	System Analysis: Square Impact Principle as a Mathematical Principle for Integrated IT with Infrastructure Design .....	717
23.6.1	Description of Operational Optimal Control .....	718
23.6.2	Square Impact Principle (SIP): Infrastructure Versus Control Performance .....	719
23.7	Conclusions .....	722
	References .....	723

## Part VI Multi-Disciplinary Connections

24	A Hybrid Dynamical Systems Perspective on Reinforcement Learning for Cyber-Physical Systems: Vistas, Open Problems, and Challenges .....	727
	Jorge I. Poveda and Andrew R. Teel	
24.1	Introduction .....	727
24.2	Hybrid Dynamical Systems .....	730
24.2.1	Non-uniqueness of Solutions and Set-Valued Dynamics .....	734

24.2.2	Hybrid Time Domains and Solutions of Hybrid Dynamical Systems .....	737
24.2.3	Graphical Convergence, Basic Assumptions and Sequential Compactness .....	738
24.2.4	Stability and Robustness .....	740
24.3	Reinforcement Learning via Dynamic Policy Gradient .....	742
24.3.1	Asynchronous Policy Iteration .....	744
24.3.2	Synchronous Policy Iteration: Online Training of Actor–Critic Structures .....	745
24.3.2.1	Learning Dynamics for the Critic .....	746
24.3.2.2	Learning Dynamics for the Actor .....	746
24.3.2.3	Closed-Loop System and Extensions to Other Settings .....	747
24.3.2.4	Extensions to Other Settings .....	749
24.4	Reinforcement Learning in Hybrid Dynamical Systems .....	749
24.4.1	Hybrid Learning Algorithms .....	750
24.4.1.1	Sampled-Data and Event-Triggered Architectures .....	750
24.4.1.2	Hybrid Coordination of Multi-agent Reinforcement Learning Algorithms .....	751
24.4.1.3	Hybrid Optimization and Estimation Algorithms .....	752
24.4.2	Hybrid Dynamic Environments .....	753
24.5	Conclusions .....	757
	References .....	757
25	<b>The Role of Systems Biology, Neuroscience, and Thermodynamics in Network Control and Learning .....</b>	763
	Wassim M. Haddad	
25.1	Introduction .....	763
25.2	Large-Scale Networks and Hybrid Thermodynamics .....	768
25.3	Multiagent Systems with Uncertain Interagent Communication .....	779
25.4	Systems Biology, Neurophysiology, Thermodynamics, and Dynamic Switching Communication Topologies for Large-Scale Multilayered Networks .....	789
25.5	Nonlinear Stochastic Optimal Control and Learning .....	792
25.6	Complexity, Thermodynamics, Information Theory, and Swarm Dynamics .....	798
25.7	Thermodynamic Entropy, Shannon Entropy, Bode Integrals, and Performance Limitations in Nonlinear Systems .....	801
25.8	Conclusion .....	809
	References .....	810

<b>26 Quantum Amplitude Amplification for Reinforcement Learning .....</b>	<b>819</b>
K. Rajagopal, Q. Zhang, S. N. Balakrishnan, P. Fakhari, and J. R. Busemeyer	
<b>26.1 Exploration and Exploitation in Reinforcement Learning .....</b>	<b>819</b>
<b>26.2 Quantum Probability Theory .....</b>	<b>820</b>
<b>26.3 The Original Quantum Reinforcement Learning (QRL)         Algorithm .....</b>	<b>822</b>
<b>26.4 The Revised Quantum Reinforcement Learning Algorithm .....</b>	<b>823</b>
<b>26.5 Learning Rate and Performance Comparisons .....</b>	<b>824</b>
<b>26.6 Other Applications of QRL .....</b>	<b>827</b>
<b>26.6.1 Example .....</b>	<b>830</b>
<b>26.7 Application to Human Learning .....</b>	<b>831</b>
<b>26.8 Concluding Comments .....</b>	<b>832</b>
<b>References .....</b>	<b>832</b>

## **Part I**

# **Theory of Reinforcement Learning for Model-Free and Model-Based Control and Games**

# Chapter 1

## What May Lie Ahead in Reinforcement Learning



Derya Cansever

The spectacular success enjoyed by machine learning (ML), primarily driven by deep neural networks can arguably be interpreted as only the tip of the iceberg. As neural network architectures, algorithmic methods, and computational power evolve, the scope of the problems that ML addresses will continue to grow. One such direction for growth materializes in reinforcement learning (RL) [1]. RL involves optimal sequential decision-making in uncertain/unknown environments where decisions at a given time take into account its impact on future events and decisions. As such, RL is akin to optimal adaptive control. In other words, it is a synthesis of dynamic programming and stochastic approximation methods [2]. Due to its sequential nature, RL is fundamentally different and broader than more commonly known deep neural network instantiations of ML, where the principal goal is to match the data with alternative hypotheses. By and large, classical neural networks tend to focus on one-shot, static problems. Ability to design deep neural networks to solve problems such as deciding whether an X-ray image corresponds to cancerous cells with previously unattainable accuracy is a glorious achievement, both in intellect and in consequence. However, it would fade in comparison with the challenges and potential rewards that could be attainable in the field of RL. The “curse of dimensionality” associated with sequential decision-making is well documented in the control literature. Having to deal with system uncertainties make the computational challenges even more daunting. Not only it geometrically increases the scope of the estimation problem but it also introduces a conflict between the need to exploit and the need to optimize, the solution of which is not well understood. Despite formidable computational challenges, the AlphaZero program achieved superhuman performance in the games of chess, shogi, and Go by reinforcement learning from self-play [3]. AlphaZero has shown that machines driven by RL can be the experts, not merely expert tools [4]. It appears

---

D. Cansever (✉)  
US Army Research Office, Adelphi, MD, USA  
e-mail: [derya.h.cansever.civ@mail.mil](mailto:derya.h.cansever.civ@mail.mil)

to have discovered novel fundamental principles about optimal strategies in chess, but it can't explicitly share that understanding with us as of yet [5]. Given the broad domain of applicability of optimal sequential decision making under uncertainty paradigm, it is tempting, and in fact plausible to imagine how RL in the future can radically shape fields that are instrumental in the functioning of society, including economics, business, politics, scientific investigations, etc. It is worthy to note that the first games (chess, shogi, and Go) that AlphaZero mastered occur in the open, fully observed by players. To an extent, this simplifies the computation of optimal strategies in RL. Subsequently, AlphaZero was extended to the game of poker, with uncertainty in the state observations [6]. This increases the dimensions of the affective state and strategy spaces and thus makes the problem even more challenging. The rules of the game, i.e., the system specification, are known and followed by all parties, which makes the problem more tractable than some other real-world problems where the rules of the evolution of the underlying system are not necessarily known in advance. In AlphaZero games, there are more than one decision-makers, which creates additional difficulties for the RL algorithm. For every fixed strategy of the opponent(s), AlphaZero needs to solve an RL induced optimization problem. As players need to take into account others' actions, and since other players' actions may take a plethora of possible values, the resulting set of optimization problems will be very challenging. The fact that players may have private information, e.g., closed cards, could induce more elaborate strategies such as deliberately missignaling their private information for better gains. Analysis and derivation of optimal strategies in games with private information is in general very difficult. Reinforcement learning-based approaches might be helpful with sufficient training data. AlphaZero focuses on zero-sum games, representing total conflict among the players. When all players share the same objective, i.e., a Team problem, players can collaborate to solve a common RL problem. Distributed RL and its corollary transfer learning are at the heart of the emerging topic of autonomy and will be instrumental in its realization. Coming back to the problems attacked by AlpaZero, the fact that the strategy space induced by the rules of the game is finite and reducible to smaller feasible subsets makes them relatively more tractable than other real-world multi-stage decision problems where the relevant strategy space may possibly take an unlimited number of values, and in fact may be infinite-dimensional, e.g., function spaces. AlphaZero enjoyed easily generated and virtually unbounded amounts of training data that are crucial for learning systems using self-play. Even with these advantages, AlphaZero requires considerable amounts of computational power, thus extending RL to more general problems remains a significant challenge. RL is built on a very solid foundation of optimal control theory [7] and stochastic approximation (Borkar [8]), so it should benefit from advances in these active fields. Some of the RL algorithms make use of deep neural network techniques as a component, a very popular research area of rapid advances. This robust mathematical and algorithmic foundation, along with Moore's Law, is expected to continue to fuel advances in RL research in the framework of optimal adaptive control theory.

For many problems of practical interest, computational power and training data requirements of RL can be formidable, and its convergence rate may be unacceptably

slow. In contrast, biological learning and decision making are in many ways more complex but can be very efficient. As AlphaZero demonstrates, in many ways, human brain is no match to sophisticated RL algorithms backed with enormous computing systems. But humans can decide quickly, discard irrelevant data, or defer the decision process to a slower time scale for more thorough processing [9]. The brain explicitly samples raw data from past memory episodes in decision making, which makes the state non-Markovian. The use of raw memory can accelerate decision-making, and also reduce processing burden by invoking strategically placed shortcuts. In contrast, many RL systems are formulated using Markovian assumptions, which may result in states that are too large to allow for efficient computations. Biological learning may involve building representations of the world from a few examples, filtering out superfluous data, and predicting events based on the history. A plausible alternate, or perhaps the complementary path to the optimal adaptive control formulation could emerge as being inspired by human brain, resulting in non-Markovian, hierarchical, and multiple time scales models. Obviously, this is a two-edged sword. It could motivate novel approaches and architectures for RL, but it could lead to some quirky features resulting from the long and windy road of evolution that are no longer relevant. Furthermore, some of the shortcut decision strategies in the brain may not necessarily be optimal in a high-powered computation environment for RL. In any case, it would be safe to postulate that one of the likely beneficiaries of the study of human brain is RL research.

## References

1. Barto, R.S.: Reinforcement Learning: An Introduction. The MIT Press, Cambridge, MA (2018)
2. Sutton, R., Barto, A., Williams, R.: Reinforcement learning is direct adaptive optimal control. *IEEE Control Syst. Mag.* **12**(2), 19–22 (1992)
3. Silver, D., Hubert, T., Schrittwieser, J., Antonoglou, I., Lai, M., Guez, A., et al.: A general reinforcement learning algorithm that masters chess, shogi, and go through self-play. *Science* **362**(6419), 1140–1144 (2018)
4. Kasparov, G.: Chess, a Drosophila of reasoning. *Science* **362**(6419), 1087 (2018)
5. Strogatz, S.: One Giant Step for a Chess-Playing Machine. *New York Times*, New York. (2018, December 26)
6. Noam Brown, T.S.: Superhuman AI for multiplayer poker. *Science* **365**(6456), 885–890 (2019)
7. Bertsekas, D.: Reinforcement Learning and Optimal Control. Athena Scientific, Nashua, NH (2019)
8. Borkar, V., Meyn, P.: The O.D.E. method for convergence of stochastic approximation and reinforcement learning. *SIAM J. Control Optim.* **38**(2), 447–469 (2000)
9. Neftci, E., Averbeck, B.: Reinforcement learning in artificial and biological systems. *Nat. Mach. Intell.* **1**, 133–143 (2019)

## Chapter 2

# Reinforcement Learning for Distributed Control and Multi-player Games



Bahare Kiumarsi, Hamidreza Modares, and Frank Lewis

**Abstract** This chapter presents the optimal control solution using reinforcement learning (RL). RL methods can successfully learn the solution to the optimal control and game problems online and using measured data along the system trajectories. However, one major challenge is that standard RL algorithms are data hungry in the sense that they must obtain a large number of samples from the interaction with the system to learn about the optimal policy. We discuss data-efficient RL algorithms using concepts of off-policy learning and experience replay and show how to solve  $H_2$  and  $H_\infty$  control problems, as well as graphical games using these approaches. Off-policy and experience replay-based RL algorithms allow reuse of data for learning and consequently lead to data-efficient RL algorithms.

## 2.1 Introduction

Optimal feedback control design has significantly contributed to the successful performance of engineered systems in aerospace, manufacturing, industrial processes, vehicles, ships, robotics, and elsewhere. The classical optimal control methods rely on a high-fidelity model of the system under control to solve Hamilton–Jacobi equations that are partial differential equations giving necessary and sufficient condition for optimally. However, a high-fidelity model of the system is not available for many real-world applications. Therefore, standard solutions to the optimal control problems require complete and accurate knowledge of the system dynamics and cannot

---

B. Kiumarsi (✉) · H. Modares

Michigan State University, 428 S Shaw Ln, East Lansing, MI 48824, USA

e-mail: [kiumarsi@msu.edu](mailto:kiumarsi@msu.edu)

H. Modares

e-mail: [modaresh@msu.edu](mailto:modaresh@msu.edu)

F. Lewis

University of Texas at Arlington, 701 S Nedderman Dr, Arlington, TX 76019, USA

e-mail: [lewis@uta.edu](mailto:lewis@uta.edu)

cope with uncertainties in dynamics. While adaptive control techniques can successfully cope with system uncertainties, they are generally far from optimal.

Reinforcement learning (RL) [1–5], a branch of machine learning (ML) inspired by learning in animals, bridges the gap between traditional optimal control and adaptive control algorithms by finding the solutions to the Hamilton–Jacobi equations online in real time for uncertain physical systems. RL algorithms are mainly based on iterating on two steps, namely policy evaluation and policy improvement [1]. The policy evaluation step evaluates the value of a given control policy by solving Bellman equations, and the policy improvement step finds an improved policy based on the evaluated value function. RL algorithms can be categorized into on-policy RL and off-policy RL algorithms depending on whether the policy under evaluation and the policy that is applied to the system are the same or not. That is, in on-policy RL algorithms, the policy that is applied to the system to collect data for learning, called behavior policy, must be the same as the policy under evaluation. This makes on-policy RL algorithms data hungry as for each policy to be evaluated a new set of data must be collected and the algorithm does not use any past data collected.

Off-policy and experience replay-based RL for learning feedback control policies take advantage of episodic memory. In off-policy RL algorithms, the data collected by applying an appropriate behavior policy to the system dynamics can be reused to evaluate as many policies as required, until an optimal policy is found. That is, the policy under evaluation, called target policy, reuses the data collected from applying the behavior policy to the system. This makes off-policy learning data-efficient and fast since a stream of experiences obtained from executing a behavior policy is reused to update several value functions corresponding to different target policies. Moreover, off-policy algorithms are more practical compare to on-policy RL approaches in the presence of disturbance since the disturbance input does not need to be adjusted in a specific manner as required in on-policy RL algorithms. They also take into account the effect of probing noise needed for exploration. On the other hand, experience replay-based RL algorithms have memories of high-value sample (past positive and negative experiences with large Bellman errors) and reuse them to not only achieve faster convergence but also relax the requirement on convergence to optimal feedback solutions. That is, instead of requiring standard persistence of excitation (PE) condition to guarantee convergence, which is generally impossible to check online, a condition on a rank of a matrix can guarantee convergence, which is easy to check.

In this chapter, we discuss how RL has been applied to find the optimal solutions for both single-agent and multi-agent problems without requiring complete knowledge about the system dynamics. In Sect. 2.2, we present the optimal control problems for continuous-time (CT) dynamical systems and its online solutions using both on-policy and off-policy RL algorithms. This includes optimal regulation problem and  $H_\infty$  problem as well as experience replay technique. In Sect. 2.3, we discuss Nash game problems and their online solutions. The RL solution to games on graphs is presented in Sect. 2.4. Output synchronization of distributed multi-agent systems using off-policy RL approach is discussed in Sect. 2.5. Finally, we provide open research directions in Sect. 2.6.

## 2.2 Optimal Control of Continuous-Time Systems

Consider the affine dynamical systems described by

$$\begin{aligned}\dot{x}(t) &= f(x(t)) + g(x(t)) u(t) \\ y(t) &= l(x(t)),\end{aligned}\tag{2.1}$$

where  $x(t) \in \mathbb{R}^n$ ,  $u(t) \in \mathbb{R}^m$ , and  $y(t) \in \mathbb{R}^p$  represent the state of the system, the control input, and the output of the system, respectively.  $f(x(t)) \in \mathbb{R}^n$  is the drift dynamics,  $g(x(t)) \in \mathbb{R}^{n \times m}$  is the input dynamics, and  $l(x(t)) \in \mathbb{R}^p$  is the output dynamics. It is assumed that  $f(0) = 0$  and  $f(x(t)) + g(x(t))u(t)$  is locally Lipschitz and the system is stabilizable.

The goal is to design a control input to assure the states of the system (2.1) converge to zero by minimizing a pre-defined performance function. The performance function is defined as [6]

$$J(x(0), u) = \int_0^\infty r(x, u) dt \equiv \int_0^\infty (Q(x) + u^T R u) dt,$$

where  $Q(x) \succeq 0$  and  $R = R^T > 0$ . The value function for the admissible control policy can be defined as

$$V(x, u) = \int_t^\infty r(x, u) d\tau \equiv \int_t^\infty (Q(x) + u^T R u) d\tau.\tag{2.2}$$

A differential equivalent to this is

$$r(x, u) + \frac{\partial V^T}{\partial x} (f(x) + g(x)u) = 0, \quad V(0) = 0,\tag{2.3}$$

and the Hamiltonian is given by

$$H\left(x, u, \frac{\partial V^T}{\partial x}\right) = r(x, u) + \frac{\partial V^T}{\partial x} (f(x) + g(x)u).$$

The optimal value is given by the Bellman optimality equation

$$r(x, u^*) + \frac{\partial V^*}{\partial x} (f(x) + g(x)u^*) = 0,\tag{2.4}$$

which is just the CT Hamilton–Jacobi–Bellman (HJB) equation. The optimal control is then given as

$$u^*(t) = \arg \min_u \left( r(x, u) + \frac{\partial V^*}{\partial x} (f(x) + g(x)u) \right) = -\frac{1}{2} R^{-1} g^T(x) \frac{\partial V^*}{\partial x}.\tag{2.5}$$

The HJB equation (2.4) is a nonlinear partial differential equation which is extremely difficult to solve. In the following, both on-policy and off-policy integral RL (IRL) algorithms are presented to approximate the solution to (2.4). We first present an on-policy RL algorithm that basically iterates on the Bellman equation (2.3) (which evaluate the value function for a given policy) and the update law (2.5) (which finds an improved control policy). This iterative algorithm converges to the solution to the HJB equation (2.4). Since the Bellman equation (2.3) requires complete knowledge of the system dynamics, a novel data-based policy evaluation step using an integral Bellman equation is presented in [7]. The basic idea is that the value function (2.2) can be rewritten as

$$V(x(t)) = \int_t^{t+T} (Q(x(\tau)) + u^T(\tau) R u(\tau)) d\tau + V(x(t+T)) \quad (2.6)$$

for any time  $t \geq 0$  and time interval  $T > 0$ . In [7], this equation is named the IRL Bellman equation. The advantage of the IRL Bellman equation over the standard Bellman equation in form of (2.3) is that it does not require any knowledge of the system dynamics and can be completely implemented data based on evaluation of a given control policy. By iteration on this Bellman equation and the policy improvement step in form of (2.5), the following policy iteration algorithm can be implemented to find the solution to the HJB equation (2.4).

**Algorithm 2.1** On-policy IRL algorithm to find the solution of HJB

1. Given an admissible policy  $u_0$
2. For  $j = 0, 1, \dots$  given  $u_j$ , solve for the value  $V_{j+1}(x)$  using Bellman equation

$$V_{j+1}(x(t)) = \int_t^{t+T} (Q(x) + u_j^T R u_j) d\tau + V_{j+1}(x(t+T)),$$

3. Update the control policy  $u_{j+1}(k)$  using

$$u_{j+1}(t) = -\frac{1}{2} R^{-1} g^T(x) \left( \frac{\partial V_{j+1}(x)}{\partial x} \right).$$

4. On convergence, set  $V_{j+1}(x) = V_j(x)$ . Otherwise, go to (2).

The IRL Algorithm 2.1 can be implemented online using value function approximation in a critic approximator network as [8–10]

$$\hat{V}(x) = \hat{W}^T \phi(x), \quad (2.7)$$

where  $\phi(x) : \mathbb{R}^n \rightarrow \mathbb{R}^N$  is the basis function vector and  $N$  is the number of basis functions. Using value function approximation in Step 2 of Algorithm 2.1 yields

$$\hat{W}_{j+1}^T [\phi(x(t)) - \phi(x(t+T))] = \int_t^{t+T} (Q(x) + u_j^T R u_j) d\tau. \quad (2.8)$$

Then, the temporal difference (TD) error is defined as

$$\epsilon^{u_j}(x(t), T) = \hat{W}_{j+1}^T [\phi(x(t+T)) - \phi(x(t))] + \int_t^{t+T} (Q(x) + u_j^T R u_j) d\tau.$$

The weights of critic network are updated to minimize the objective

$$S = \int_{\Omega} \epsilon^{u_j}(x, T) \epsilon^{u_j}(x, T) dx.$$

This amounts to  $\int_{\Omega} \frac{d\epsilon^{u_j}(x, T)}{d\hat{W}_j} \epsilon^{u_j}(x, T) dx = 0$ . Using the inner product notation for the Lebesgue integral, one can write

$$\left\langle \frac{d\epsilon^{u_j}(x, T)}{d\hat{W}_{j+1}}, \epsilon^{u_j}(x, T) \right\rangle_{\Omega} = 0,$$

which is

$$\Phi \hat{W}_{j+1} + \left\langle [\phi(x(t+T)) - \phi(x(t))], \int_t^{t+T} (Q(x(\tau)) + u_j^T(\tau) R u_j(\tau)) d\tau \right\rangle = 0,$$

where  $\Phi = \langle [\phi(x(t+T)) - \phi(x(t))], [\phi(x(t+T)) - \phi(x(t))]^T \rangle_{\Omega}$ . It is shown in [10] that  $\Phi$  is invertible. Then, one has

$$\hat{W}_{j+1} = -\Phi^{-1} \left\langle [\phi(x(t+T)) - \phi(x(t))], \int_t^{t+T} (Q(x(\tau)) + u_j^T(\tau) R u_j(\tau)) d\tau \right\rangle.$$

On convergence of the value parameters, the control policy is updated using

$$u_{j+1}(t) = -\frac{1}{2} R^{-1} g^T(x) \nabla \phi^T \hat{W}_{j+1}.$$

The control policy is implemented as a neural network with the same weights as the critic network, but its activation functions being replaced with the gradient of those used in the critic network. Synchronous update laws for actor and critic are introduced in [11] to update both actor and critic simultaneously while assuring system stability. The IRL Algorithm 2.1 does not require the knowledge of the drift dynamics. However, the knowledge about the input dynamic is required.

### 2.2.1 IRL with Experience Replay Learning Technique [12, 13]

In Algorithm 2.1, the persistence of excitation (PE) condition needs to be satisfied to assure the convergence of algorithm which is often difficult or impossible to verify online. Instead, the idea of the experience replay, which forms a history stack of

past experiences, is employed to update the weights of value function in which the recorded past experiences are used concurrently with current data for adaptation of the critic weights. This provides simplified conditions to check for PE-like requirements in real time by more efficient use of current and past data.

The approximate IRL Bellman equation becomes

$$e_B = \Delta\phi(t)^T \hat{W} + \int_t^{t+T} (Q(x) + \hat{u}^T R \hat{u}) d\tau,$$

where  $\Delta\phi(t) = \phi(t+T) - \phi(t)$  and  $e_B$  is TD error after using current critic approximator weights. Consider  $\Delta\phi(t_j)$  as the value of  $\Delta\phi$  at the time  $t_j$ . Then, a history of past data can be collected as  $[\Delta\phi(t_1), \dots, \Delta\phi(t_l)]$ . The TD error at the time  $t_j$  using the current critic weights'  $\hat{W}$  is defined as

$$(e_B)_j = \Delta\phi(t_j)^T \hat{W} + \int_{t_j}^{t_j+T} (Q(x(\tau)) + \hat{u}^T(\tau) R \hat{u}(\tau)) d\tau.$$

The weights of critic neural network are now updated based on experience replay gradient-decent algorithm as

$$\dot{\hat{W}} = -\alpha_c \frac{\Delta\phi(t)}{(\Delta\phi(t)^T \Delta\phi(t) + 1)^2} e_B - \alpha_c \sum_{j=1}^l \frac{\Delta\phi(t_j)}{(\Delta\phi(t_j)^T \Delta\phi(t_j) + 1)^2} (e_B)_j.$$

The first term is a gradient update law for the TD error and the last term minimizes its stored samples in the history stack.

**Condition 2.1** Let  $Z = [\Delta\bar{\phi}_1, \dots, \Delta\bar{\phi}_l]$  be the history stack, where  $\Delta\bar{\phi}_i = \frac{\Delta\phi(t_i)}{\Delta\phi(t_i)^T \Delta\phi(t_i) + 1}$ . Then,  $Z$  in the recorded data contains as many linearly independent elements as the number of neurons in (2.7). That is  $\text{rank}(Z) = N$  [12].

It is shown in [12] that the on-policy IRL with experience replay converges to the optimal solution if the recorded data satisfies Condition 2.1. Unlike PE, however, Condition 2.1 can easily be checked online in real time.

### 2.2.1.1 Off-Policy IRL Algorithm [14–16]

The on-policy IRL Algorithm 2.1 requires partial knowledge about the system dynamics in the policy improvement step. That is, it requires the knowledge of  $g(x)$ . Moreover, it does not use any memory of past experiences in learning new policies. This is because the learned policy at each iteration should be applied to the system to collect data for learning. The off-policy learning, on the other hand, reuses collected

data to improve data efficiency of RL. This is performed by separating the behavior policy, which is used to generate data for learning, and, the target policy, which is under evaluation. This greatly increases the information exploration ability during the learning process. Moreover, in off-policy RL, no dynamics model of the system is required.

To come up with a model-free off-policy IRL algorithm, first, we add and subtract a control input to the system dynamic (2.1) to get

$$\dot{x}(t) = f(x(t)) + g(x(t)) u_j(t) + g(x(t)) (u(t) - u_j(t)), \quad (2.9)$$

where the added and subtracted policy  $u_j(t)$  is the target policy, i.e., the policy to be updated. By contrast,  $u(t)$  is called the behavior policy that is actually applied to the system to collect required data for learning (i.e., evaluating target policies).

Differentiating  $V(x)$  along with the system dynamics (2.9) and using  $u_{j+1}(t) = -\frac{1}{2} R^{-1} g^T(x) (\frac{\partial V_j(x)}{\partial x})$  gives,

$$\begin{aligned} \dot{V}_j &= \left( \frac{\partial V_j(x)}{\partial x} \right)^T (f + g u_j) + \left( \frac{\partial V_j(x)}{\partial x} \right)^T g(u - u_j) \\ &= -Q(x) - u_j^T R u_j - 2 u_{j+1}^T R (u - u_j). \end{aligned}$$

Integrating from both sides of this equation results in the following off-policy IRL Bellman equation

$$V_j(x(t+T)) - V_j(x(t)) = \int_t^{t+T} (-Q(x) - u_j^T R u_j - 2 u_{j+1}^T R (u - u_j)) d\tau. \quad (2.10)$$

Note that for a given target policy  $u_j$ , the off-policy Bellman equation can be solved to simultaneously find the next (or improved) policy  $u_{j+1}$  and its corresponding value function  $V_j$ . Iterating on the IRL Bellman equation (2.10) results in the following model-free off-policy IRL algorithm.

### **Algorithm 2.2** Off-policy IRL algorithm to find the solution of HJB

1. Collect data using a fixed admissible policy  $u$
2. For  $j = 0, 1, \dots$  given  $u_j$ , solve for  $V_j(x)$  and  $u_{j+1}$  using Bellman equation

$$V_j(x(t+T)) - V_j(x(t)) = \int_t^{t+T} (-Q(x) - u_j^T R u_j - 2 u_{j+1}^T R (u - u_j)) d\tau.$$

3. On convergence, set  $V_j(x) = V_{j-1}(x)$ . Otherwise, go to (2).

To implement the off-policy IRL Algorithm 2.2, the actor-critic structure is used. The value function is approximated by a critic network as (2.7) and the control policy

is approximated by actor network

$$\hat{u}(t) = \hat{W}_a^T \psi(x).$$

The weights of critic and actor networks are updated simultaneously using least squares.

### 2.2.2 $\mathcal{H}_\infty$ Control of CT Systems

consider the following CT system dynamics in the presence of disturbance

$$\dot{x}(t) = f(x(t)) + g(x(t))u(t) + h(x(t))d(t), \quad (2.11)$$

where  $x(t) \in \mathbb{R}^n$  is the state vector,  $u(t) \in \mathbb{R}^m$  is the control input, and  $d(t) \in \mathbb{R}^q$  is the disturbance input.  $f(x(t)) \in \mathbb{R}^n$  is the drift dynamics,  $g(x(t)) \in \mathbb{R}^{n \times m}$  is the input dynamics, and  $h(x(t)) \in \mathbb{R}^{n \times q}$  is the disturbance input dynamics. The performance function is defined as

$$J(x(0), u, d) = \int_0^\infty (Q(x) + u^T R u - \beta^2 d^T d) d\tau,$$

where  $Q(x) \succeq 0$ ,  $R = R^T > 0$  and  $\beta \geq 0$  is a prescribed disturbance attenuation level. The value function can be defined as

$$V(x(t), u, d) = \int_t^\infty (Q(x) + u^T R u - \beta^2 d^T d) d\tau. \quad (2.12)$$

A differential equivalent to this is the Bellman equation

$$Q(x) + u^T R u - \beta^2 d^T d + \frac{\partial V}{\partial x} (f(x) + g(x)u + h(x)d) = 0 \quad V(0) = 0$$

and the Hamiltonian is given by

$$H(x, u, d, \frac{\partial V}{\partial x}) = Q(x) + u^T R u - \beta^2 d^T d + \frac{\partial V}{\partial x} (f(x) + g(x)u + h(x)d).$$

Define the two-player zero-sum differential game as [17]

$$V^*(x(t)) = \min_u \max_d J(x(0), u, d)$$

subject to (2.11).

The solution to this zero-sum game is found by Hamilton–Jacobi–Isaacs (HJI) equation

$$Q(x) + (u^*)^T R u^* - \beta^2 (d^*)^T d^* + \frac{\partial V^*}{\partial x} (f(x) + g(x)u^* + h(x)d^*) = 0.$$

Then, the optimal control input and worst case disturbance input are given as

$$u^* = -\frac{1}{2} R^{-1} g^T(x) \frac{\partial V^*}{\partial x},$$

$$d^* = \frac{1}{2\beta^2} h^T(x) \frac{\partial V^*}{\partial x}.$$

The value function (2.12) for an arbitrary control action  $u$  and disturbance signal  $d$ , and for any time interval  $T$  can be written as

$$V(x(t)) = \int_t^{t+T} (r(x(\tau), u(\tau), d(\tau)) d\tau + V(x(t+T)). \quad (2.13)$$

The following algorithm uses (2.13) to find the solution to the HJI equation.

**Algorithm 2.3** On-policy IRL for  $\mathcal{H}_\infty$  Control Problem [18–20]

1. Given an admissible policy  $u_0$
2. For  $j = 0, 1, \dots$  given  $u_j$
3. For  $i = 0, 1, \dots$  set  $d^0 = 0$ , solve for the value  $V_j^{(i)}(x)$  using Bellman equation

$$V_j^i(x(t)) = \int_t^{t+T} (Q(x) + (u_j)^T R u_j - \beta^2 (d^i)^T d^i + ) d\tau + V_j^i(x(t+T)),$$

and update the disturbance using

$$d^{i+1} = \frac{1}{2\beta^2} h^T(x) \frac{\partial V_j^i}{\partial x}.$$

- On convergence, set  $V_j(x) = V_{j-1}^i(x).$
4. Update the control policy  $u_{j+1}$  using

$$u_{j+1} = -\frac{1}{2} R^{-1} g^T(x) \frac{\partial V_j}{\partial x}.$$

5. Go to 3.

Actor–critic structure has been used to implement Algorithm 2.3 that simultaneously updates the value function and policies and does not require the knowledge of the drift dynamics.

The on-policy Algorithm 2.3 requires partial knowledge about the system dynamics. Moreover, the disturbance needs to be updated and applied to the system to collect data for the learning. However, this disturbance is not under our control. Then, in the next section, the off-policy IRL algorithm is presented that cover the disadvantages mentioned with Algorithm 2.3.

### 2.2.2.1 Off-Policy IRL [15, 21]

To come up with a model-free off-policy IRL algorithm for  $\mathcal{H}_\infty$  control problem, first, we add and subtract a control input and a disturbance input to the system dynamic (2.11) to get

$$\begin{aligned}\dot{x}(t) = & f(x(t)) + g(x(t)) u_j(t) + g(x(t)) (u(t) - u_j(t)) \\ & + h(x(t)) d_j(t) + g(x(t)) (d(t) - d_j(t)),\end{aligned}$$

where  $u_j(t)$  and  $d_j(t)$  are the target policies to be learned and updated, respectively. By contrast,  $u(t)$  and  $d(t)$  are the behavior policies that are actually applied to the system dynamics to collect data required for learning the control and disturbance target policies.

Differentiating  $V(x)$  along with the system dynamics (2.9) and using  $u_{j+1}(t) = -\frac{1}{2} R^{-1} g^T(x) (\frac{\partial V_j(x)}{\partial x})$  and  $d_{j+1}(t) = \frac{1}{2\beta^2} h^T(x) (\frac{\partial V_j(x)}{\partial x})$  gives

$$\begin{aligned}\dot{V}_j = & \left( \frac{\partial V_j(x)}{\partial x} \right)^T (f + g u_j + h d_j) + \left( \frac{\partial V_j(x)}{\partial x} \right)^T g (u - u_j) \\ & + \left( \frac{\partial V_j(x)}{\partial x} \right)^T h (d - d_j) = -Q(x) - u_j^T R u_j \\ & + \beta^2 d_j^T d_j - 2 u_{j+1}^T R (u - u_j) + 2 \beta^2 d_{j+1}^T (d - d_j).\end{aligned}$$

Integrating from both sides of this equation results in the  $\mathcal{H}_\infty$  off-policy IRL Bellman equation

$$\begin{aligned}V_j(x(t+T)) - V_j(x(t)) = & \int_t^{t+T} (-Q(x) - u_j^T R u_j + \beta^2 d_j^T d_j \\ & - 2 u_{j+1}^T R (u - u_j) + 2 \beta^2 d_{j+1}^T (d - d_j)) d\tau.\end{aligned}\tag{2.14}$$

Iterating on the IRL Bellman equation (2.14) results in the following off-policy IRL algorithm.

**Algorithm 2.4** Off-policy IRL for  $\mathcal{H}_\infty$  Control Problem

1. Collect data using admissible policy  $u$  and actual disturbance  $d$
2. For  $j = 0, 1, \dots$  given  $u_j$  and  $d_j$ , solve for the value  $V_j$  and  $u_{j+1}$  and  $d_{j+1}$  using Bellman equation

$$V_j(x(t+T)) - V_j(x(t)) = \int_t^{t+T} \left( -Q(x) - u_j^T R u_j + \beta^2 d_j^T d_j - 2 u_{j+1}^T R (u - u_j) + 2 \beta^2 d_{j+1}^T (d - d_j) \right) d\tau.$$

3. On convergence, set  $V_j(x) = V_{j-1}(x)$ .

Note that for a desired, possibly exploratory behavior policy  $u(t)$  and the actual external disturbance from the environment (the policies that are applied to the system), the off-policy IRL Bellman equation (2.14) learns the corresponding value function  $V_j$ , the control target policy  $u_{j+1}$  and disturbance target policy  $d_{j+1}$  simultaneously, and without requiring any knowledge about the system dynamics. The sequence of target disturbance policies converges to the worst-case disturbance policy, without requiring the disturbance applied to the system to be updated or determined, which is practically impossible. The actor-critic structure is used to approximate the value function, control, and disturbance policies [15].

## 2.3 Nash Games

Consider the  $N$ -player nonlinear differential game

$$\dot{x}(t) = f(x(t)) + \sum_{j=1}^N g_j(x(t))u_j(t),$$

where  $x \in \mathbb{R}^n$  is the state vector and  $u_j(t) \in \mathbb{R}^{m_j}$  is the control input. It is assumed that  $f(0) = 0$  and  $f(x) + \sum_{j=1}^N g_j(x)u_j$  is locally Lipschitz and that the system is stabilizable.

The performance function associated with each player is defined as

$$\begin{aligned} J_i(x(0), u_1, u_2, \dots, u_N) &= \int_0^\infty (r_i(x, u_1, \dots, u_N)) dt \\ &\equiv \int_0^\infty (Q_i(x) + \sum_{j=1}^N u_j^T R_{ij} u_j) dt, \quad \forall i \in \mathcal{N} := \{1, 2, \dots, N\}, \end{aligned}$$

where  $Q_i(\cdot) \succeq 0$  and  $R_{ii} = R_{ii}^T \succ 0, \forall i \in \mathcal{N}$ ,  $R_{ij} = R_{ij}^T \succeq 0, \forall j \neq i \in \mathcal{N}$  with  $\mathcal{N} := \{1, 2, \dots, N\}$ .

The value for each player can be defined as

$$V_i(x, u_1, u_2, \dots, u_N) = \int_t^\infty (r_i(x, u_1, \dots, u_N)) d\tau, \forall i \in \mathcal{N}, \forall x, u_1, u_2, \dots, u_N.$$

Differentiating the value function gives the following Bellman equations, we get

$$r_i(x, u_1, \dots, u_N) + \frac{\partial V_i}{\partial x}^T \left( f(x) + \sum_{j=1}^N g_j(x) u_j \right) = 0, V_i(0) = 0, \forall i \in \mathcal{N}.$$

The Hamiltonian functions are defined as

$$H_i \left( x, u_1, \dots, u_N, \frac{\partial V_i}{\partial x} \right) = r_i(x, u_1, \dots, u_N) + \frac{\partial V_i}{\partial x}^T \left( f(x) + \sum_{j=1}^N g_j(x) u_j \right), \forall i \in \mathcal{N}.$$

Applying the stationarity conditions gives the control policies

$$u_i^* = \arg \min_{u_i} H_i(x, u_1, \dots, u_N, \frac{\partial V_i^*}{\partial x}) = -\frac{1}{2} R_{ii}^{-1} g_i^T(x) \frac{\partial V_i}{\partial x}, \forall i \in \mathcal{N}.$$

Substituting the control policies into the Hamiltonian, the following coupled HJ equations are obtained

$$\begin{aligned} 0 &= Q_i(x) + \frac{1}{4} \sum_{j=1}^N \frac{\partial V_j}{\partial x}^T g_j(x) R_{jj}^{-T} R_{ij} R_{jj}^{-1} g_j^T(x) \frac{\partial V_j}{\partial x} \\ &\quad + \frac{\partial V_i}{\partial x}^T \left( f(x) - \frac{1}{2} \sum_{j=1}^N g_j(x) R_{jj}^{-1} g_j^T(x) \frac{\partial V_j}{\partial x} \right), \forall i \in \mathcal{N}. \end{aligned} \quad (2.15)$$

The solution to the coupled HJ equations can be approximated using RL. The value functions can be rewritten as

$$V_i(x(t)) = \int_t^{t+T} r_i(x(\tau), u_1(\tau), \dots, u_N(\tau)) d\tau + V_i(x(t+T)), \forall i \in \mathcal{N},$$

for any time  $t \geq 0$  and time interval  $T > 0$ .

**Algorithm 2.5** On-policy IRL algorithm to find the solution of coupled HJ equations [22, 23]

1. Given  $N$ -tuple of admissible policies  $u_i^0, \forall i \in \mathcal{N}$
2. For  $j = 0, 1, \dots$  given  $u_i^j$ , solve for the value  $V_i^{j+1}(x)$  using Bellman equation

$$V_i^{j+1}(x(t)) = \int_t^{t+T} (Q_i(x) + \sum_{i=1}^N (u_i^j)^T R_{ii} u_i^j) dt + V_i^{j+1}(x(t+T)),$$

3. Update the  $N$ -tuple of control policies  $u_i^{j+1}$ ,  $\forall i \in \mathcal{N}$  using

$$u_i^{j+1}(t) = -\frac{1}{2} R_{ii}^{-1} g_i^T(x) \left( \frac{\partial V_i^{j+1}(x)}{\partial x} \right).$$

4. On convergence, set  $V_i^{j+1}(x) = V_i^j(x)$ . Otherwise go to (2).

To implement Algorithm 2.5, assume that the value function is approximated by critic network as

$$\hat{V}_i(x) = \hat{W}_{ic}^T \phi_i(x), \quad \forall x, \quad i \in \mathcal{N}$$

and the control input is approximated by actor network as

$$\hat{u}_i(x) = -\frac{1}{2} R_{ii}^{-1} g^T(x) \nabla \phi_i^T \hat{W}_{iu}.$$

Then, the update law for critic network can be rewritten as

$$\begin{aligned} \dot{W}_{ic} &= -\alpha_i \frac{\Delta \phi_i(t)}{\left( (\Delta \phi_i(t))^T \Delta \phi_i(t) + 1 \right)^2} \left( \Delta \phi_i(t)^T \hat{W}_{ic} + \right. \\ &\quad \left. \int_t^{t+T} \left( Q_i(x) + \hat{u}_i^T R_{ii} \hat{u}_i + \sum_{j=1}^N \hat{u}_j^T R_{ij} \hat{u}_j \right) d\tau \right), \quad \forall i \in \mathcal{N}, \end{aligned}$$

and the update law for actor network is given as

$$\begin{aligned} \dot{W}_{iu} &= -\alpha_{iu} \left( (F_i \hat{W}_{iu} - L_i \Delta \phi_i(t)^T \hat{W}_{ic}) - \frac{1}{4} \sum_{j=1}^N \left( \frac{\partial \phi_i}{\partial x} g_i(x) R_{ii}^{-T} R_{ij} R_{ii}^{-1} g_i(x)^T \frac{\partial \phi_i}{\partial x}^T \right) \right. \\ &\quad \left. \hat{W}_{iu} \frac{\Delta \phi_i(t)^T}{\left( \Delta \phi_i(t)^T \Delta \phi_i(t) + 1 \right)^2} \hat{W}_{jc} \right), \quad \forall i \in \mathcal{N}, \end{aligned}$$

with  $\Delta \phi_i(t) := \phi_i(t) - \phi_i(t+T)$ .

## 2.4 Graphical Games

The goal here is to design local control inputs for a set of agents that seek to cooperatively track the state of a leader node. Consider the  $N$  agents with dynamics

$$\dot{x}_i(t) = Ax_i(t) + B_i u_i(t), \quad x_i(0) = x_{i0}, \quad t \geq 0,$$

where  $x_i(t) \in \mathbb{R}^n$  is the state vector and  $u_i(t) \in \mathbb{R}^{m_i}$ ,  $i \in \mathcal{N} := \{1, \dots, N\}$  is each control input (or player).

The leader node dynamics is

$$\dot{x}_0 = Ax_0.$$

The agents communicate with each other through a fixed strongly connected graph  $\mathcal{G} = (\mathcal{V}_{\mathcal{G}}, \mathcal{E}_{\mathcal{G}})$  defined by a finite set  $\mathcal{V}_{\mathcal{G}} = \{n_1, \dots, n_N\}$  of  $N$  agents and a set of edges  $\mathcal{E}_{\mathcal{G}} \subseteq \mathcal{V}_{\mathcal{G}} \times \mathcal{V}_{\mathcal{G}}$  that represent interagent information exchange links. The set of *neighbors* of a node  $n_i$  is defined as the set of nodes with edges incoming to  $n_i$  and is denoted by  $\mathcal{N}_i = \{n_j : (n_j, n_i) \in \mathcal{E}_{\mathcal{G}}\}$ . The adjacency matrix is  $\mathcal{A}_{\mathcal{G}} = [\alpha_{ij}]_{N \times N}$  with weights  $\alpha_{ij} > 0$  if  $(n_j, n_i) \in \mathcal{E}_{\mathcal{G}}$  and zero otherwise. The diagonal degree matrix  $D$  of the graph  $\mathcal{G}$  is defined as  $D = \text{diag}(d_i)$  with the weighted degree  $d_i = \sum_{j \in \mathcal{N}_i} \alpha_{ij}$ .

The neighborhood tracking error for every agent is defined as

$$\delta_i := \sum_{j \in \mathcal{N}_i} \alpha_{ij} (x_i - x_j) + g_i (x_i - x_0), \quad \forall i \in \mathcal{N}, \quad (2.16)$$

where  $g_i$  is the pinning gain.  $g_i \neq 0$  if agent is pinned to the leader node. The neighborhood tracking errors dynamics are given by

$$\dot{\delta}_i = A\delta_i + (d_i + g_i)B_i u_i - \sum_{j \in \mathcal{N}_i} \alpha_{ij} B_j u_j, \quad \forall i \in \mathcal{N}, \quad (2.17)$$

with  $\delta_i \in \mathbb{R}^n$ .

Define the local performance indices for each agent as

$$\begin{aligned} J_i(\delta_i(0); u_i, u_{\mathcal{N}_i}) &= \frac{1}{2} \int_0^\infty r_i(\delta_i, u_i, u_{\mathcal{N}_i}) dt, \quad \forall i \in \mathcal{N} \\ &\equiv \frac{1}{2} \int_0^\infty \left( \delta_i^T Q_i \delta_i + u_i^T R_{ii} u_i + \sum_{j \in \mathcal{N}_i} u_j^T R_{ij} u_j \right) dt, \quad \forall i \in \mathcal{N}, \end{aligned}$$

where matrices  $Q_i \succeq 0$ ,  $R_{ii} \succ 0$ ,  $R_{ij} \succeq 0$ ,  $\forall i, j \in \mathcal{N}$  of appropriate dimensions, and  $(\sqrt{Q_i}, A)$ ,  $\forall i \in \mathcal{N}$  are detectable.

The control objective of agent  $i$  in the graphical game is to determine

$$V_i^*(\delta_i(t)) := \min_{u_i} \int_t^\infty \frac{1}{2} \left( \delta_i^T Q_i \delta_i + u_i^T R_{ii} u_i + \sum_{j \in \mathcal{N}_i} u_j^T R_{ij} u_j \right) dt, \quad \forall t, \delta_i, i \in \mathcal{N}. \quad (2.18)$$

The Hamiltonians associated with each agent's neighborhood tracking error (2.17) and each  $V_i^*$  given in (2.18) can be defined as

$$\begin{aligned} H_i \left( \delta_i, u_i, u_{\mathcal{N}_i}, \frac{\partial V_i^*}{\partial \delta_i} \right) &= \frac{\partial V_i^*}{\partial \delta_i} \left( A \delta_i + (d_i + g_i) B_i u_i - \sum_{j \in \mathcal{N}_i} \alpha_{ij} B_j u_j \right) \\ &+ \frac{1}{2} \delta_i^T Q_i \delta_i + \frac{1}{2} u_i^T R_{ii} u_i + \frac{1}{2} \sum_{j \in \mathcal{N}_i} u_j^T R_{ij} u_j, \quad \forall \delta_i, u_i, \forall i \in \mathcal{N}. \end{aligned}$$

The optimal control for each  $i \in \mathcal{N}$  can be found to be

$$u_i^*(\delta_i) = \arg \min_{u_i} H_i(\delta_i, u_i, u_{\mathcal{N}_i}, \frac{\partial V_i^*}{\partial \delta_i}) = -(d_i + g_i) R_{ii}^{-1} B_i^T \frac{\partial V_i^*}{\partial \delta_i}, \quad \forall \delta_i, \quad (2.19)$$

that should satisfy the appropriate distributed coupled HJ equations

$$H_i(\delta_i, u_i^*, u_{\mathcal{N}_i}^*, \frac{\partial V_i^*}{\partial \delta_i}) = 0, \quad \forall i \in \mathcal{N}.$$

It is assumed that the value functions are quadratic

$$V_i^*(\delta_i) = \frac{1}{2} \delta_i^T P_i \delta_i, \quad \forall \delta_i, \forall i \in \mathcal{N}, \quad (2.20)$$

where  $P_i = P_i^T > 0 \in \mathbb{R}^{n \times n}$ ,  $\forall i \in \mathcal{N}$  are the unique matrices that solve the following complicated distributed coupled equations,

$$\begin{aligned} \delta_i^T P_i \left( A \delta_i - (d_i + g_i)^2 B_i R_{ii}^{-1} B_i^T P_i \delta_i + \sum_{j \in \mathcal{N}} \alpha_{ij} (d_j + g_j) B_j R_{jj}^{-1} B_j^T P_j \delta_j \right) + \\ \left( A \delta_i - (d_i + g_i)^2 B_i R_{ii}^{-1} B_i^T P_i \delta_i + \sum_{j \in \mathcal{N}} \alpha_{ij} (d_j + g_j) B_j R_{jj}^{-1} B_j^T P_j \delta_j \right)^T \times \\ P_i \delta_i + \sum_{j \in \mathcal{N}_i} (d_j + g_j)^2 \delta_j^T P_j B_j R_{jj}^{-T} R_{ij} R_{jj}^{-1} B_j^T P_j \delta_j + (d_i + g_i)^2 \delta_i^T P_i B_i R_{ii} B_i^T P_i \delta_i + \delta_i^T Q_i \delta_i = 0, \quad \forall i \in \mathcal{N}. \end{aligned}$$

Considering (2.20), the optimal control (2.19) for every player  $i \in \mathcal{N}$  can be written as

$$u_i^*(\delta_i) = -(d_i + g_i) R_{ii}^{-1} B_i^T P_i \delta_i, \quad \forall \delta_i.$$

The solution to the coupled HJ equations can be approximated using policy iteration algorithm.

**Algorithm 2.6** On-policy RL algorithm in Graphical Games [24, 25]

1. Given  $N$ -tuple of admissible policies  $u_i^0$ ,  $\forall i \in \mathcal{N}$
2. For  $s = 0, 1, \dots$ , solve for  $V_i^s(x)$  using the coupled Bellman equations,

$$\begin{aligned} & \delta_i^T Q_{ii} \delta_i + \frac{\partial V_i^s}{\partial \delta_i}^T \left( A \delta_i + (d_i + g_i) B_i u_i^s - \sum_{j \in \mathcal{N}_i} \alpha_{ij} B_j u_j^s \right) \\ & + u_i^{s+1} R_{ii} u_i^s + \sum_{j \in \mathcal{N}_i} u_j^{s+1} R_{ij} u_j^s = 0, \quad V_i^{s+1}(0) = 0. \end{aligned} \quad (2.21)$$

3. Update the control policies  $u_i^{s+1}$ ,  $\forall i \in \mathcal{N}$  using

$$u_i^{s+1} = -(d_i + g_i) R_{ii}^{-1} B_i^T \frac{\partial V_i^s}{\partial \delta_i}. \quad (2.22)$$

4. On convergence, set  $V_i^{s+1} = V_i^s$ . Otherwise, go to 2.

To implement Algorithm 2.6 online, actor–critic structure can be used. As can be seen, complete knowledge about the agent dynamics is required to find the solution to the coupled HJ equations using Algorithm 2.6. In the following, the off-policy algorithm will be presented that does not require any knowledge about the agent dynamics.

#### 2.4.1 Off-Policy RL for Graphical Games

Consider  $N$  agents with the identical dynamics as

$$\dot{x}_i(t) = Ax_i(t) + Bu_i(t), \quad x_i(0) = x_{i0}, \quad t \geq 0.$$

Introducing auxiliary variables  $u_i^s$  and  $u_{\mathcal{N}_i}^s$  for each agent, one has

$$\dot{\delta}_i = A \delta_i + (d_i + g_i) B u_i^s - \sum_{j \in \mathcal{N}_i} \alpha_{ij} B u_j^s + (d_i + g_i) B(u_i - u_i^s) - \sum_{j \in \mathcal{N}_i} \alpha_{ij} B(u_j - u_j^s), \quad \forall i \in \mathcal{N}, \quad (2.23)$$

where  $u_i$  are interpreted as behavior policies actually applied to the system. By contrast,  $u_i^s$  are the target policies.

Differentiating  $V_i^{s+1}$  with respect to (2.23) and using (2.21) and (2.22) yield

$$\begin{aligned} \frac{d V_i^{s+1}}{d t} = & -\delta_i^T Q_{ii} \delta_i - u_i^s R_{ii} u_i^s - 2u_i^{s+1 T} R_{ii} (u_i - u_i^s) \\ & + 2(d_i + g_i)^{-1} \cdot \sum_{j \in \mathcal{N}_i} \alpha_{ij} u_i^{s+1 T} R_{ii} (u_j - u_j^s). \end{aligned} \quad (2.24)$$

Integrating both sides of (2.24) yields the following off-policy Bellman equations:

$$\begin{aligned} V_i^{s+1}(\delta_i(t)) - V_i^{s+1}(\delta_i(t+T)) = & \int_t^{t+T} \left( \delta_i^T(\tau) Q_{ii} \delta_i(\tau) + u_i^s R_{ii} u_i^s + 2u_i^{s+1 T} R_{ii} (u_i - u_i^s) \right) d\tau \\ & - \int_t^{t+T} \left( 2(d_i + g_i)^{-1} \cdot \sum_{j \in \mathcal{N}_i} \alpha_{ij} u_i^{s+1 T} R_{ii} (u_j - u_j^s) \right) d\tau. \end{aligned} \quad (2.25)$$

### Algorithm 2.7 Off-policy RL algorithm in Graphical Games [26]

1. Collect data using admissible policy  $u_i$ ,  $\forall i \in \mathcal{N}$
2. For  $s = 0, 1, \dots$ , given  $u_i^s$ , solve for  $V_i^{s+1}$  and  $u_i^{s+1}$  using the off-policy Bellman equations (2.25)
4. On convergence, set  $V_i^{s+1} = V_i^s$ . Otherwise, go to 2.

To implement Algorithm 2.7 online, actor–critic structure is developed in [26] to approximate  $V_i^{s+1}$  and  $u_i^{s+1}$ .

## 2.5 Output Synchronization of Multi-agent Systems

The dynamics of  $N$  linear heterogeneous agents is given by

$$\begin{aligned} \dot{x}_i = & A_i x_i + B_i u_i \\ y_i = & C_i x_i, \end{aligned} \quad (2.26)$$

where  $x_i \in \mathbb{R}^{n_i}$  is the system state,  $u_i \in \mathbb{R}^{m_i}$  is the input, and  $y_i \in \mathbb{R}^{q_i}$  is the output for agent  $i$ .

Let the leader dynamics be given by

$$\begin{aligned} \dot{\zeta}_0 = & S \zeta_0 \\ y_0 = & R \zeta_0, \end{aligned}$$

where  $\zeta_0 \in \mathbb{R}^p$  is the reference trajectory to be followed by followers and  $S \in \mathbb{R}^{p \times p}$  is the leader's dynamic matrix. The agents communicate to each other through a graph defined in Sect. 2.4.

In output synchronization problem, the goal is to design local control protocols  $u_i$  such that the outputs of all agents synchronize to the output of the leader node. That is,  $y_i(t) - y_0(t) \rightarrow 0$ ,  $\forall i$ .

The standard methods to solve the output regulation problem require solving the output regulation equations given by

$$\begin{aligned} A_i \Pi_i + B_i \Gamma_i &= \Pi_i S \\ C_i \Pi_i &= R, \end{aligned} \quad (2.27)$$

where  $\Pi_i \in \mathbb{R}^{n_i \times p}$  and  $\Gamma_i \in \mathbb{R}^{m_i \times p}$  are solutions to (2.27). Based on these solutions, the following control protocol guarantees output synchronization:

$$u_i = \bar{K}_{1i}(x_i - \Pi_i \zeta_i) + \Gamma_i \zeta_i,$$

where  $K_{1i}$  is the state feedback gain which stabilizes  $A_i + B_i K_{1i}$  and  $\zeta_i$  is the estimate of the leader trajectory  $\zeta_0$  for agent  $i$  obtained by

$$\dot{\zeta}_i = S \zeta_i + c \left[ \sum_{j=1}^N \alpha_{ij} (\zeta_j - \zeta_i) + g_i (\zeta_0 - \zeta_i) \right]$$

with  $c$  as the coupling gain.

To find an online solution to the output regulator equation (2.27), a model-free distributed adaptive observer is first designed to estimate the leader's state  $\zeta_0$  and then off-policy RL algorithm is employed, while the agents' dynamics and leader dynamics are not known.

The state of leader for agent  $i$  is estimated using a distributed observer as

$$\begin{aligned} \dot{\hat{\zeta}}_i &= \hat{S}_i \hat{\zeta}_i + c \left[ \sum_{j=1}^N \alpha_{ij} (\hat{\zeta}_j - \hat{\zeta}_i) + g_i (\hat{\zeta}_0 - \hat{\zeta}_i) \right] \\ \hat{y}_{0i} &= \hat{R}_i \hat{\zeta}_i, \end{aligned} \quad (2.28)$$

where  $\hat{S}_i \in \mathbb{R}^{p \times p}$  and  $\hat{R}_i \in \mathbb{R}^{q \times p}$  are estimations of the leader's dynamics  $S$  and  $R$ , respectively, and  $\hat{\zeta}_i$  and  $\hat{y}_{0i}$  are the state and output of the adaptive observer for node  $i$ , respectively.

The update laws for  $\hat{S}_i$  and  $\hat{R}_i$  are designed as

$$\begin{aligned} \dot{\hat{S}}_{veci} &= \Gamma_{Si} (I_p \otimes \hat{\zeta}_i) p_i \left[ \sum_{j=1}^N \alpha_{ij} (\hat{\zeta}_j - \hat{\zeta}_i) + g_i (\hat{\zeta}_0 - \hat{\zeta}_i) \right] \\ \dot{\hat{R}}_i &= \alpha \left[ \sum_{j=1}^N \alpha_{ij} (\hat{R}_j \hat{\zeta}_j - \hat{R}_i \hat{\zeta}_i) + g_i (y_0 - \hat{R}_i \hat{\zeta}_i) \right], \end{aligned}$$

where  $\hat{S}_{veci}$  is a vector obtained by stacking rows of matrix  $\hat{S}_i$ ,  $\Gamma_{Si} > 0$ ,  $\alpha > 0$ , and  $p_i = \text{diag}(q_i) > 0$  with  $q = [q_1, \dots, q_N]^T = ((L + G) \otimes I_p)^{-T} 1_{Np}$ .

It is shown in [29] that the state and output of the distributed observer converge to the leaders' ones.

Define the discounted performance function for (2.26) as

$$V(x_i, u_i) = \int_t^\infty e^{-\gamma_i(\tau-t)} [(y_i - y_0)^T Q_i (y_i - y_0) + u_i^T W_i u_i] d\tau, \quad (2.29)$$

where  $Q_i \succeq 0$ ,  $W_i = W_i^T \succ 0$ , and  $\gamma_i > 0$  is the discount factor.

This is an optimal tracking control problem for each agent. Then the control input is

$$u_i = K_{1i} x_i + K_{2i} \zeta_0.$$

As discussed in [27], the optimal tracking problem (2.29) can be formulated as an optimal regulation problem with the augmented state  $X_i = [x_i^T, \zeta_0^T]^T$  and so the augmented system dynamic

$$\dot{X}_i = T_i X_i + B_{1i} u_i.$$

The off-policy algorithm presented in [28] can be used to solve this problem. However, since all agents do not have access to the state of the leader  $\zeta_0$ , the distributed adaptive observer (2.28) is used to estimate it for each agent. Therefore, the distributed adaptive observer combined with the off-policy algorithm is presented as follows to solve the output synchronization problem of multi-agent systems.

### Algorithm 2.8 Off-policy RL algorithm for output synchronization [29]

1. Collect data using admissible policy  $u_i$ ,  $\forall i \in \mathcal{N}$
2. Estimate the state of the leader for each agent using (2.28) and perform  $X_i = [x_i^T, \hat{\zeta}_i^T]^T$
3. For  $s = 0, 1, \dots$ , given  $u_i^s$ , solve for  $P_i^s$  and  $u_i^{s+1}$  using the off-policy Bellman equations,

$$e^{-\gamma_i T} X_i^T (t+T) P_i^s X_i(t+T) - X_i^T(t) P_i^s X_i(t) = \\ - \int_t^{t+T} e^{-\gamma_i(\tau-t)} [(y_i - y_0)^T Q_i (y_i - y_0)] d\tau + 2 \int_t^{t+T} e^{-\gamma_i(\tau-t)} (u_i - K_i^s X_i)^T W_i K_i^{s+1} X_i d\tau.$$

4. On convergence, set  $P_i^{s+1} = P_i^s$ . Otherwise, go to 3.

## 2.6 Conclusion and Open Research Directions

This chapter has reviewed RL techniques for solving optimal control problems in real time using data measured along the system trajectories. We have presented families of online RL-based solutions for optimal regulation, Nash, and graphical games. The off-policy and experience replay-based RL algorithms are also presented which take advantages of episodic memories. These algorithms are fast and data-efficient because of the reuse of the data stored in memory for learning. One interesting and important open research direction in the context of multi-agent and distributed RL is to develop new architectures so that the agents can learn a common objective and learn to collaborate with incorporating network constraints, safety constraints, and privacy constraints. Adding new metrics into learning to perform trade-off between different objectives, for example, safety and stability/performance can be another open research direction.

## References

1. Sutton, S., Barto, A.G.: Reinforcement Learning: An Introduction, 2nd edn, in preparation. MIT Press, Cambridge (2017)
2. Zhang, H., Liu, D., Luo, Y., Wang, D.: Adaptive Dynamic Programming for Control. Springer, London (2013)
3. Powell, W.B.: Approximate Dynamic Programming. Wiley, Hoboken (2007)
4. Lewis, F.L., Liu, D.: Reinforcement Learning and Approximate Dynamic Programming for Feedback Control. Wiley, Hoboken (2013)
5. Lewis, F.L., Vrabie, D.: Reinforcement learning and adaptive dynamic programming for feedback control. *IEEE Circuits Syst. Mag.* **9**, 32–50 (2009)
6. Lewis, F.L., Vrabie, D., Syrmos, V.L.: Optimal Control. Wiley, Hoboken (2012)
7. Vrabie, D., Pastravanu, O., Abu-Khalaf, M., Lewis, F.L.: Adaptive optimal control for continuous-time linear systems based on policy iteration. *Automatica* **45**, 477–484 (2009)
8. Werbos, P.J.: Approximate dynamic programming for real-time control and neural modeling. In: White, D.A., Sofge, D.A. (eds.) *Handbook of Intelligent Control: Neural, Fuzzy, and Adaptive Approaches*. Van Nostrand Reinhold, New York (1992)
9. Bertsekas, D.P., Tsitsiklis, J.N.: Neuro-dynamic Programming. Athena Scientific, MA (1996)
10. Vrabie, D., Lewis, F.L.: Neural network approach to continuous time direct adaptive optimal control for partially unknown nonlinear systems. *Neural Netw.* **22**, 237–246 (2009)
11. Vamvoudakis, K.G., Lewis, F.L.: Online actor-critic algorithm to solve the continuous-time infinite horizon optimal control problem. *Automatica* **46**, 878–888 (2010)
12. Modares, H., Lewis, F.L., Naghibi-Sistani, M.B.: Integral reinforcement learning and experience replay for adaptive optimal control of partially-unknown constrained-input continuous-time systems. *Automatica* **50**, 193–202 (2014)
13. Kamalapurkar, R., Walters, P., Dixon, W.E.: Model-based reinforcement learning for approximate optimal regulation. *Automatica* **64**, 94–104 (2016)
14. Jiang, Y., Jiang, Z.P.: Robust adaptive dynamic programming and feedback stabilization of nonlinear systems. *IEEE Trans. Neural Netw. Learn. Syst.* **25**, 882–893 (2014)
15. Luo, B., Wu, H.N., Huang, T.: Off-policy reinforcement learning for  $H_\infty$  control design. *IEEE Trans. Cybern.* **45**, 65–76 (2015)
16. Jiang, Y., Jiang, Z.P.: Computational adaptive optimal control for continuous-time linear systems with completely unknown dynamics. *Automatica* **48**, 2699–2704 (2012)

17. Başar, T., Bernard, P.:  $H_\infty$  Optimal Control and Related Minimax Design Problems. Birkhäuser, Boston (1995)
18. Abu-Khalaf, M., Lewis, F.L.: Neurodynamic programming and zero-sum games for constrained control systems. *IEEE Trans. Neural Netw.* **19**, 1243–1252 (2008)
19. Vamvoudakis, K.G., Lewis, F.L.: Online solution of nonlinear two-player zero-sum games using synchronous policy iteration. *Int. J. Robust Nonlinear Control* **22**, 1460–1483 (2012)
20. Modares, H., Lewis, F.L., Naghibi-Sistani, M.-B.: Online solution of nonquadratic two-player zero-sum games arising in the  $H_\infty$  control of constrained input systems. *Int. J. Adapt. Control Signal Process.* **28**, 232–254 (2014)
21. Li, L., Liu, D., Wang, D.: Integral reinforcement learning for linear continuous-time zero-sum games with completely unknown dynamics. *IEEE Trans. Autom. Sci. Eng.* **11**, 706–714 (2014)
22. Vamvoudakis, K.G., Lewis, F.L.: Multi-player non-zero-sum games: online adaptive learning solution of coupled Hamilton–Jacobi equations. *Automatica* **47**, 1556–1569 (2011)
23. Johnson, M., Kamalapurkar, R., Bhasin, S., Dixon, W.E.: Approximate N-player nonzero-sum game solution for an uncertain continuous nonlinear system. *IEEE Trans. Neural Netw. Learn. Syst.* **26**, 1645–1658 (2015)
24. Vamvoudakis, K.G., Lewis, F.L., Hudas, G.R.: Multi-agent differential graphical games: Online adaptive learning solution for synchronization with optimality. *Automatica* **48**, 1598–1611 (2012)
25. Vamvoudakis, K.G., Modares, H., Kiumarsi, B., Lewis, F.L.: Game theory-based control system algorithms with real-time reinforcement learning: how to solve multiplayer games online. *IEEE Control Syst. Mag.* **37**, 33–52 (2017)
26. Li, J., Modares, H., Chai, T., Lewis, F.L., Xie, L.: Off-policy reinforcement learning for synchronization in multiagent graphical games. *IEEE Trans. Neural Netw. Learn. Syst.* **28**, 2434–2445 (2017)
27. Modares, H., Lewis, F.L.: Linear quadratic tracking control of partially-unknown continuous-time systems using reinforcement learning. *IEEE Trans. Autom. Control* **59** (2014)
28. Modares, H., Lewis, F.L., Jiang, Z.P.:  $H_\infty$  tracking control of completely unknown continuous-time systems. *IEEE Trans. Neural Netw. Learn. Syst.* **26**, 2550–2562 (2015)
29. Modares, H., Nageshrao, S.P., Delgado Lopes, G.A., Babuska, R., Lewis, F.L.: Optimal model-free output synchronization of heterogeneous systems using off-policy reinforcement learning. *Automatica* **71**, 334–341 (2016)

# Chapter 3

# From Reinforcement Learning to Optimal Control: A Unified Framework for Sequential Decisions



Warren B. Powell

**Abstract** There are over 15 distinct communities that work in the general area of sequential decisions and information, often referred to as decisions under uncertainty or stochastic optimization. We focus on two of the most important fields: stochastic optimal control, with its roots in deterministic optimal control, and reinforcement learning, with its roots in Markov decision processes. Building on prior work, we describe a unified framework that covers all 15 different communities and note the strong parallels with the modeling framework of stochastic optimal control. By contrast, we make the case that the modeling framework of reinforcement learning, inherited from discrete Markov decision processes, is quite limited. Our framework (and that of stochastic control) is based on the core problem of optimizing over policies. We describe four classes of policies that we claim are universal and show that each of these two fields has, in their own way, evolved to include examples of each of these four classes.

## 3.1 Introduction

There is a vast range of problems that consist of the sequence: decisions, information, decisions, information, . . . Application areas span engineering, business, economics, finance, health, transportation, and energy. It encompasses active learning problems that arise in the experimental sciences, medical decision-making, e-commerce, and sports. It also includes iterative algorithms for stochastic search, as well as two-agent games and multi-agent systems. In fact, we might claim that virtually any human enterprise will include instances of sequential decision problems.

Given the diversity of problem domains, it should not be a surprise that a number of communities have emerged to address the problem of making decisions over time to optimize some metric. The reason that so many communities exist is a testament to

---

W. B. Powell (✉)

Department of Operations Research and Financial Engineering, Princeton University,  
Princeton, USA

e-mail: [wbpowell328@gmail.com](mailto:wbpowell328@gmail.com)

the variety of problems, but it also hints at the many methods that are needed to solve these problems. As of this writing, there is not a single method that has emerged to solve all problems. In fact, it is fair to say that all the methods that have been proposed are *fragile*: relatively modest changes can invalidate a theoretical result, or increase run times by orders of magnitude.

In [31], we present a unified framework for all sequential decision problems. This framework consists of a mathematical model (that draws heavily from the framework used widely in stochastic control), which requires optimizing over *policies* which are functions for making decisions given what we know at a point in time (captured by the state variable).

The significant advance of the unified framework is the identification of four (meta)classes of policies that encompass all the communities. In fact, whereas the solution approach offered by each community is fragile, we claim that the four classes are universal: any policy proposed for any sequential decision problem will consist of one of these four classes, and possibly a hybrid.

The contribution of the framework is to raise the visibility of all of the communities. Instead of focusing on a specific solution approach (for example, the use of Hamilton–Jacobi–Bellman (HJB) equations, which is one of the four classes), the framework encourages people to consider all four classes, and then to design policies that are best suited to the characteristics of a problem.

This chapter is going to focus attention on two specific communities: stochastic optimal control and reinforcement learning. Stochastic optimal control emerged in the 1950s, building on what was already a mature community for deterministic optimal control that emerged in the early 1900s and has been adopted around the world. Reinforcement learning, on the other hand, emerged in the 1990s building on the foundation of Markov decision processes which was introduced in the 1950s (in fact, the first use of the term “stochastic optimal control” is attributed to Bellman, who invented Markov decision processes). Reinforcement learning emerged from computer science in the 1980s and grew to prominence in 2016 when it was credited with solving the Chinese game of Go using AlphaGo.

We are going to make the following points:

- Both communities have evolved from a core theoretical/algorithm result based on HJB equations, transitioning from exact results (that were quite limited) to the use of algorithms based on approximating value functions/cost-to-go functions/Q-factors, to other strategies that do not depend on HJB equations. We will argue that each of the fields is in the process of recognizing all four classes of policies.
- We will present and contrast the canonical modeling frameworks for stochastic control and reinforcement learning (adopted from Markov decision processes). We will show that the framework for stochastic control is very flexible and scalable to real applications, while that used by reinforcement learning is limited to a small problem class.
- We will present a universal modeling framework for sequential decision analytics (given in [31]) that covers any sequential decision problem. The framework draws heavily from that used by stochastic control, with some minor adjustments. While

not used by the reinforcement learning community, we will argue that it is used implicitly. In the process, we will dramatically expand the range of problems that can be viewed as either stochastic control problems or reinforcement learning problems.

We begin our presentation in Sect. 3.2 with an overview of the different communities that work on sequential decisions under uncertainty, along with a list of major problem classes. Section 3.3 presents a side-by-side comparison of the modeling frameworks of stochastic optimal control and reinforcement learning.

Section 3.4 next presents our universal framework (taken from [31]), and argues that (a) it covers all 15+ fields (presented in Sect. 3.2) dealing with sequential decisions and uncertainty, (b) it draws heavily from the standard model of stochastic optimal control, and (c) the framework of reinforcement learning, inherited from discrete Markov decision processes, has fundamental weaknesses that limit its applicability to a very narrow class of problems. We then illustrate the framework using an energy storage problem in Sect. 3.5; this application offers tremendous richness and allows us to illustrate the flexibility of the framework.

The central challenge of our modeling framework involves optimizing over policies, which represents our point of departure with the rest of the literature, since it is standard to pick a class of policy in advance. However, this leaves open the problem of how to search over policies. In Sect. 3.6 we present four (meta)classes of policies which, we claim, are universal, in that *any* approach suggested in the literature (or in practice) is drawn from one of these four classes, or a hybrid of two or more. Section 3.7 illustrates all four classes, along with a hybrid, using the context of our energy storage application. These examples will include hybrid resource allocation/active learning problems, along with the overlooked challenge of dealing with rolling forecasts.

Section 3.8 briefly discusses how to use the framework to model multi-agent systems, and notes that this vocabulary provides a fresh perspective on partially observable Markov decision processes. Then, Sect. 3.9 concludes the chapter with a series of observations about reinforcement learning, stochastic optimal control, and our universal framework.

## 3.2 The Communities of Sequential Decisions

The list of potential applications of sequential decision problems is virtually limitless. Below, we list a number of major application domains. Ultimately, we are going to model all of these using the same framework.

**Discrete problems** These are problems with discrete states and discrete decisions (actions), such as stochastic shortest path problems.

**Control problems** These span controlling robots, drones, rockets, and submersibles, where states are continuous (location and velocity) as are controls (forces). Other examples include determining optimal dosages of medications, or continuous inventory (or storage) problems that arise in finance and energy.

**Dynamic resource allocation problems** Here, we are typically managing inventories (retail products, food, blood, energy, money, drugs), typically over space and time. It also covers discrete problems such as dynamically routing vehicles, or managing people or machines. It would also cover planning the movements of robots and drones (but not how to do it). The scope of “dynamic resource allocation problems” is almost limitless.

**Active learning problems** This includes any problem that involves learning, and where decisions affect what information is collected (laboratory experiments, field experiments, test marketing, computer simulations, medical testing). It spans multi-armed bandit problems, e-commerce (bidding, recommender systems), black-box simulations, and simulation-optimization.

**Hybrid learning/resource allocation problems** This would arise if we are managing a drone that is collecting information, which means we have to manage a physical resource while running experiments which are then used to update beliefs. Other problems are laboratory science experiments with setups (this is the physical resource), collecting public health information from field technicians, and any experimental learning setting with a budget constraint.

**Stochastic search** This includes both derivative-based and derivative-free stochastic optimization.

**Adversarial games** This includes any two-player (or multiplayer) adversarial games. It includes pricing in markets where price affects market behavior and military applications.

**Multi-agent problems** This covers problems with multiple decision-makers who might be competing or cooperating. They might be making the same decisions (but spatially distributed), or making different decisions that interact (as arises in supply chains, or where different agents play different roles but have to work together).

Given the diversity of problems, it should not be surprising that a number of different research communities have evolved to address them, each with their own vocabulary and solution methods, creating what we have called the “jungle of stochastic optimization” ([30], see also <https://jungle.princeton.edu>). A list of the different communities that address the problem of solving sequential decision information problems might be

- Stochastic search (derivative-based)
- Ranking and selection (derivative-free)
- (Stochastic) optimal control
- Markov decision processes/dynamic programming
- Simulation-optimization
- Optimal stopping
- Model predictive control
- Stochastic programming
- Chance-constrained programming
- Approximate/adaptive/neuro-dynamic programming
- Reinforcement learning

- Robust optimization
- Online computation
- Multi-armed bandits
- Active learning
- Partially observable Markov decision processes

*Each of these communities is supported by at least one book and over a thousand papers.*

Some of these fields include problem classes that can be described as static: make decision, see information (possibly make one more decision), and then the problem stops (stochastic programming and robust optimization are obvious examples). However, all of them include problems that are fully sequential, consisting of sequences of decision, information, decision, information, . . . , over a finite or infinite horizon. The focus of this chapter is on fully sequential problems.

Several of the communities offer elegant theoretical frameworks that lead to optimal solutions for specialized problems (Markov decision processes and optimal control are two prominent examples). Others offer asymptotically optimal algorithms: derivative-based and certain derivative-free stochastic optimization problems, simulation-optimization, and certain instances of approximate dynamic programming and reinforcement learning. Still, others offer theoretical guarantees, often in the form of regret bounds (that is, bounds on how far the solution is from optimal).

We now turn our attention to focus on the fields of stochastic optimal control and reinforcement learning.

### 3.3 Stochastic Optimal Control Versus Reinforcement Learning

There are numerous communities that have contributed to the broad area of modeling and solving sequential decision problems, but there are two that stand out: optimal control (which laid the foundation for stochastic optimal control), and Markov decision processes, which provided the analytical foundation for reinforcement learning. Although these fields have intersected at different times in their history, today they offer contrasting frameworks which, nonetheless, are steadily converging to common solution strategies.

We present the modeling frameworks of (stochastic) optimal control and reinforcement learning (drawn from Markov decision processes), which are polar opposites. Given the growing popularity of reinforcement learning, we think it is worthwhile to compare and contrast these frameworks. We then present our own universal framework which spans all the fields that deal with any form of sequential decision problems. The reader will quickly see that our framework is quite close to that used by the (stochastic) optimal control community, with a few adjustments.

### 3.3.1 Stochastic Control

The field of optimal control enjoys a long and rich history, as evidenced by the number of popular books that have been written focusing on deterministic control, including [20, 45, 48]. There are also a number of books on stochastic control (see [4, 22, 28, 39, 42, 44]) but these tend to be mathematically more advanced.

Deterministic optimal control problems are typically written

$$\min_{u_0, \dots, u_T} \sum_{t=0}^{T-1} L_t(x_t, u_t) + L_T(x_T), \quad (3.1)$$

where  $x_t$  is the state at time  $t$ ,  $u_t$  is the control (that is, the decision) and  $L_t(x_t, u_t)$  is a loss function with terminal loss  $L_T(x_T)$ . The state  $x_t$  evolves according to

$$x_{t+1} = f_t(x_t, u_t), \quad (3.2)$$

where  $f_t(x_t, u_t)$  is variously known as the transition function, system model, plant model (as in chemical or power plant), plant equation, and transition law. We write the control problem in discrete time, but there is an extensive literature where this is written in continuous time, and the transition function is written

$$\dot{x}_t = f_t(x_t, u_t).$$

The most common form of a stochastic control problem simply introduces additive noise to the transition function given by

$$x_{t+1} = f_t(x_t, u_t) + w_t, \quad (3.3)$$

where  $w_t$  is random at time  $t$ . This odd notation arose because of the continuous time formulation, where  $w_t$  would be disturbances (such as wind pushing against an aircraft) between  $t$  and  $t + dt$ . The introduction of the uncertainty means that the state variable  $x_t$  is a random variable when we are sitting at time 0. Since the control  $u_t$  is also a function of the state, this means that  $u_t$  is also a random variable. Common practice is to then take an expectation of the objective function in Eq. (3.1), which produces

$$\min_{u_0, \dots, u_T} \mathbb{E} \left\{ \sum_{t=0}^{T-1} L_t(x_t, u_t) + L_T(x_T) \right\}, \quad (3.4)$$

which has to be solved subject to the constraint in Eq. (3.3). This is problematic, because we have to interpret (3.3) recognizing that  $x_t$  and  $u_t$  are random since they depend on the sequence  $w_0, \dots, w_{t-1}$ .

In the deterministic formulation of the problem, we are looking for an optimal control vector  $u_0^*, \dots, u_T^*$ . When we introduce the random variable  $w_t$ , then the controls need to be interpreted as functions that depend on the information available at time  $t$ . Mathematicians handle this by saying that “ $u_t$  must be  $\mathcal{F}_t$ -measurable” which means, in plain English, that the control  $u_t$  is a function (not a variable) which can only depend on information up through time  $t$ . This leaves us the challenge of finding this function.

We start by relaxing the constraint in (3.3) and add it to the objective function, giving us

$$\min_{u_0, \dots, u_T} \mathbb{E} \left\{ \sum_{t=0}^{T-1} L_t(x_t, u_t) + L_T(x_T) + \lambda_t(f(x_t, u_t) + w_t - x_{t+1}) \right\}, \quad (3.5)$$

where  $(\lambda_t)_{t=0}^T$  is a vector of dual variables (known as costate variables in the controls community). Assuming that  $\mathbb{E}w_t = 0$ , then  $w_t$  drops out of the objective function.

The next step is that we restrict our attention to quadratic loss functions given by

$$L_t(x_t, u_t) = (x_t)^T Q_t x_t + (u_t)^T R_t u_t, \quad (3.6)$$

where  $Q_t$  and  $R_t$  are a set of known matrices. This special case is known as linear-quadratic regulation (or LQR).

With this special structure, we turn to the Hamilton–Jacobi equations (often called the HJB equations) where we solve for the “cost-to-go” function  $J_t(x_t)$  using

$$J_t(x_t) = \min_u (L_t(x_t, u) + \mathbb{E}_w J_{t+1}(f(x_t, u, w))). \quad (3.7)$$

$J_t(x_t)$  is the value of being in state  $x_t$  at time  $t$  and following an optimal policy from time  $t$  onward. In the language of reinforcement learning,  $J_t(x_t)$  is known as the value function, and is written  $V_t(S_t)$ .

For the special case of the quadratic objective function in (3.6), it is possible to solve the HJB equations analytically and show that the optimal control as a function of the state is given by

$$u_t = K_t x_t, \quad (3.8)$$

where  $K_t$  is a matrix that depends on  $Q_t$  and  $R_t$ .

Here,  $u_t$  is a *function* that we refer to as a *policy*  $\pi$ , but is known as a *control law* in the controls community. Some would write (3.8) as  $\pi_t(x_t) = K_t x_t$ . Later, we are going to adopt the notation  $U^\pi(x_t)$  for writing a policy, where  $\pi$  carries information about the structure of the policy. We note that when the policy depends on the state  $x_t$ , then the function is, by construction, “ $\mathcal{F}_t$ -measurable,” so we can avoid this terminology entirely.

The linear control law (policy) in Eq. (3.8) is very elegant, but it is a byproduct of the special structure, which includes the quadratic form of the objective function

(Eq. (3.6)), the additive noise (Eq. (3.3)), and the fact that there are no constraints on the controls. For example, a much more general way of writing the transition function is

$$x_{t+1} = f(x_t, u_t, w_t),$$

which allows the noise to enter the dynamics in any form. For example, consider an inventory problem where the state (the inventory level) is governed by

$$x_{t+1} = \max\{0, x_t + u_t - w_{t+1}\},$$

where  $w_{t+1}$  is the random demand for our product.

We are also interested in general state-dependent reward functions which are often written as  $g(x_t, u_t)$  (where  $g(\cdot)$  stands for gain), as well as the constraints, where we might write

$$\begin{aligned} A_t u_t &= b_t, \\ u_t &\geq 0, \end{aligned}$$

where  $b_t$  (and  $A_t$ ) may contain information from the state variable.

For these more general problems, we cannot compute (3.7) exactly, so the research community has developed a variety of methods for approximating  $J_t(x_t)$ . Methods for solving (3.7) approximately have been widely studied in the controls community under names such as heuristic dynamic programming, approximate dynamic programming, neuro-dynamic programming, and adaptive dynamic programming. However, even this approach is limited to special classes of problems within our universe of sequential decision problems.

Optimal control enjoys a rich history. Deterministic control dates to the early 1900s, while stochastic control appears to have been first introduced by Bellman in the 1950s (known as the father of dynamic programming). Some of the more recent books in optimal control are [20, 25, 39, 42, 44]. The most common optimal control problems are continuous, low dimensional, and unconstrained. Stochastic problems are most typically formulated with additive noise.

The field of stochastic control has tended to evolve using the more sophisticated mathematics that has characterized the field. Some of the most prominent books include [1, 4, 22, 28, 50] (note that some of the books on deterministic controls touch on the stochastic case).

We are going to see below that this framework for writing sequential decision problems is quite powerful, even if the classical results (such as the linear control policy) are very limited. It will form the foundation for our unified framework, with some slight adjustments.

### 3.3.2 Reinforcement Learning

The field known as reinforcement learning evolved from early work done by Rich Sutton and his adviser Andy Barto in the early 1980s. They addressed the problem of modeling the search process of a mouse exploring a maze, developing methods that would eventually help solve the Chinese game of Go, outperforming world masters (Fig. 3.1).

Sutton and Barto eventually made the link to the field of Markov decision processes and adopted the vocabulary and notation of this field. The field is nicely summarized in [36] which can be viewed as the capstone volume on 50 years of research into Markov decision processes, starting with the seminal work of Bellman [2]. Puterman [36][Chap. 3] summarizes the modeling framework as consisting of the following elements:

**Decision epochs**  $T = 1, 2, \dots, N$ .

**State space**  $\mathcal{S}$  = set of (discrete) states.

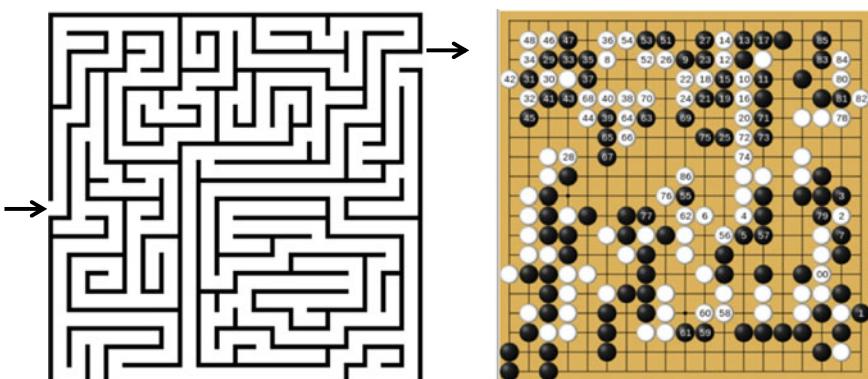
**Action space**  $\mathcal{A}$  = action space (set of actions when we are in state  $s$ ).

**Transition matrix**  $p(s'|s, a)$  = probability of transitioning to state  $s'$  given that we are in state  $s$  and take action  $a$ .

**Reward**  $r(s, a)$  = the reward received when we are in state  $s$  and take action  $a$ .

This notation (which we refer to below as the “MDP formal model”) became widely adopted in the computer science community where reinforcement learning evolved. It became standard for authors to define a reinforcement learning problem as consisting of the tuple  $(\mathcal{S}, \mathcal{A}, P, r)$ , where  $P$  is the transition matrix and  $r$  is the reward function.

Using this notation, Sutton and Barto (this work is best summarized in their original volume [46]) proposed estimating the value of being in a state  $s^n$  and taking an action  $a^n$  (at the  $n$ th iteration of the algorithm) using



**Fig. 3.1** From the mouse-in-the-maze problem to Chinese Go—a trajectory of reinforcement learning

$$\hat{q}^n(s^n, a^n) = r(s^n, a^n) + \gamma \max_{a' \in \mathcal{A}_{s'}} \bar{Q}^{n-1}(s', a'), \quad (3.9)$$

$$\bar{Q}^n(s^n, a^n) = (1 - \alpha_n) \bar{Q}^{n-1}(s^n, a^n) + \alpha_n \hat{q}^n(s^n, a^n), \quad (3.10)$$

where  $\gamma$  is a discount factor and  $\alpha_n$  is a smoothing factor that might be called a stepsize (the equation has roots in stochastic optimization) or learning rate. Equation (3.10) is the core of “reinforcement learning.”

We assume that when we are in state  $s^n$  and take action  $a^n$  that we have some way of simulating the transition to a state  $s'$ . There are two ways of doing this:

- Model-based—We assume we have the transition matrix  $P$  and then sample  $s'$  from the probability distribution  $p(s'|s, a)$ .
- Model-free—We assume we are observing a physical setting where we can simply observe the transition to state  $s'$  without a transition matrix.

Sutton and Barto named this algorithmic strategy “ $Q$ -learning” (after the notation). The appeal of the method is its sheer simplicity. In fact, they retained this style in their wildly popular book [46] which can be read by a high school student.

Just as appealing is the wide applicability of both the model and algorithmic strategy. Contrast the core algorithmic step described by Eqs.(3.9)–(3.10) to Bellman’s equation which is the foundation of Markov decision processes, which requires solving

$$V_t(s) = \max_a (r(s, a) + \gamma \sum_{s' \in \mathcal{S}} p(s'|s, a) V_{t+1}(s')), \quad (3.11)$$

for all states  $s \in \mathcal{S}$ . Equation (3.11) is executed by setting  $V_{T+1}(s) = 0$  for all  $s \in \mathcal{S}$ , and then stepping backward  $t = T, T - 1, \dots, 1$  (hence the reason that this is often called “backward dynamic programming”). In fact, this version was so trivial that the field focused on the stationary version which is written as

$$V(s) = \max_a (r(s, a) + \gamma \sum_{s' \in \mathcal{S}} p(s'|s, a) V(s')). \quad (3.12)$$

[Side note: The steady-state version of Bellman’s equation in (3.12) became the default version of Bellman’s equation, which explains why the default notation for reinforcement learning does not index variables by time. By contrast, the default formulation for optimal control is finite time, and variables are indexed by time in the canonical model.]

Equation (3.12) requires solving a system of nonlinear equations to find  $V(s)$ , which proved to be the foundation for an array of papers with elegant algorithms, where one of the most important is

$$V^{n+1}(s) = \max_a (r(s, a) + \gamma \sum_{s' \in \mathcal{S}} p(s'|s, a) V^n(s')). \quad (3.13)$$

Equation (3.13) is known as value iteration (note the similarity with Eq. (3.11)) and is the basis of  $Q$ -learning (compare to Eqs. (3.9)–(3.10)).

The problem with (3.11) is that it is far from trivial. In fact, it is quite rare that it can be computed due to the widely cited “curse of dimensionality.” This typically refers to the fact that for most problems, the state  $s$  is a vector  $s = (s_1, s_2, \dots, s_K)$ . Assuming that all the states are discrete, the number of states grows exponentially in  $K$ . It is for this reason that dynamic programming is widely criticized for suffering from the “curse of dimensionality.” In fact, the curse of dimensionality is due purely to the use of lookup table representations of the value function (note that the canonical optimal control model does not do this).

In practice, this typically means that one-dimensional problems can be solved in under a minute; two-dimensional problems might take several minutes (but possibly up to an hour, depending on the dimensionality and the planning horizon); three-dimensional problems easily take a week or a month; and four-dimensional problems can take up to a year (or more).

In fact, there are actually three curses of dimensionality: the state space, the action space, and the outcome space. It is typically assumed that there is a discrete set of actions (think of the roads emanating from an intersection), but there are many problems where decisions are vectors (think of all the ways of assigning different taxis to passengers). Finally, there are random variables (call them  $w$  for now) which might also be a vector. For example,  $w$  might be the set of riders calling our taxi company for rides in a 15-minute period. Or, it might represent all the attributes of a customer clicking on ads (age, gender, location).

The most difficult computational challenge in Eq. (3.11) is finding the one-step transition matrix  $P$  with element  $p(s'|s, a)$ . This matrix measures  $|\mathcal{S}| \times |\mathcal{S}| \times |\mathcal{A}|$  which may already be quite large. However, consider what it takes to compute just one element. To show this, we need to steal a bit of notation from the optimal control model, which is the *transition function*  $f(x, u, w)$ . Using the notation of dynamic programming, we would write this as  $f(s, a, w)$ . Assuming the transition function is known, the one-step transition matrix is computed using

$$p(s'|s, a) = \mathbb{E}_w \{\mathbb{1}_{\{s' = f(s, a, w)\}}\}. \quad (3.14)$$

This is not too hard if the random variable  $w$  is scalar, but there are many problems where  $w$  is a vector, in which case we encounter the third curse of dimensionality. There are many other problems where we do not even know the distribution of  $w$  (but have a way of observing outcomes).

Now return to the  $Q$ -learning Eqs. (3.9)–(3.10). At no time are we enumerating all the states, although we do have to enumerate all the actions (and the states these actions lead to), which is perhaps a reason why reinforcement learning is always illustrated in the context of relatively small, discrete action spaces (think of the Chinese game of Go). Finally, we do not need to take an expectation over the random variable  $w$ ; rather, we just simulate our way from state  $s$  to state  $s'$  using the transition function  $f(s, a, w)$ .

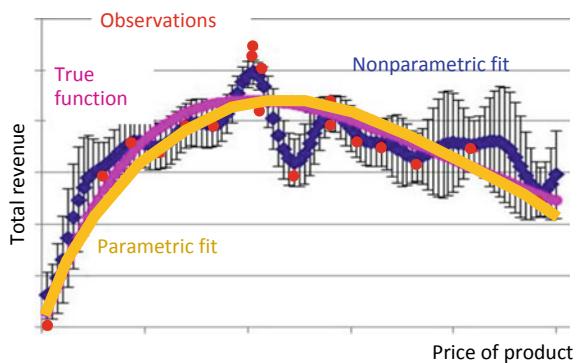
We are not out of the woods. We still have to estimate the value of being in state  $s$  and taking action  $a$ , captured by our  $Q$ -factors  $\bar{Q}(s, a)$ . If we use lookup tables, this means we need to estimate  $\bar{Q}(s, a)$  for each state that we *might* visit, and each action that we *might* take, which means we are back to the curse of dimensionality. However, we can use other approximation strategies:

- Lookup tables with hierarchical beliefs—Here, we use a family of lookup table models at different levels of aggregation.
- Parametric models, which might be linear (in the parameters) or nonlinear. We include shallow neural networks here. Parametric models transform the dimensionality of problems down to the dimensionality of the parameter vector, but we have to know the parametric form.
- Nonparametric models. Here, we include kernel regression, locally parametric, and flexible architectures such as support vector machines and deep neural networks.

Not surprisingly, considerable attention has been devoted to different methods for approximating  $\bar{Q}(s, a)$ , with recent attention focusing on using deep neural networks. We will just note that the price of higher dimensional architectures is that they come with the price of increased training (in fact, dramatically increased training). A deep neural network might easily require tens of millions of iterations, and yet still may not guarantee high-quality solutions.

The challenge is illustrated in Fig. 3.2, where we have a set of observations (red dots). We try to fit the observations using a nonparametric model (the blue line) which overfits the data, which we believe is a smooth, concave (approximately quadratic) surface. We would get a reasonable fit of a quadratic function with no more than ten data points, but any nonparametric model (such as a deep neural network) might require hundreds to thousands of data points (depending on the noise) to get a good fit.

**Fig. 3.2** Nonparametric fit (blue line) versus parametric fit (yellow line), compared to actual (purple line)



### 3.3.3 A Critique of the MDP Modeling Framework

For many years, the modeling framework of Markov decision processes lived within the MDP community which consisted primarily of applied probabilists, reflecting the limited applicability of the solution methods. Reinforcement learning, however, is a field that is exploding in popularity, while still clinging to the classical MDP modeling framework (see [23] for a typical example of this). What is happening, however, is that people doing computational work are adopting styles that overcome the limitations of the discrete MDP framework. For example, researchers will overcome the problem of computing the one-step transition matrix  $p(s'|s, a)$  by saying that they will “simulate” the process. In practice, this means that they are using the transition function  $f(s, a, w)$ , which means that they have to simulate the random information  $w$ , without explicitly writing out  $f(s, a, w)$  or the model of  $w$ . This introduces a confusing gap between the statement of the model and the software that captures and solves the model.

We offer the following criticisms of the classical MDP framework that the reinforcement learning community has adopted:

- The MDP/RL modeling framework models state *spaces*. The optimal control framework models state *variables*. We argue that the latter is much more useful since it more clearly describes the actual variables of the problem. Consider a problem with discrete states (perhaps with  $K$  dimensions). The state space could then be written  $\mathcal{S} = \mathcal{S}_1 \times \dots \times \mathcal{S}_K$  which produces a set of discrete states that we can write  $\{1, 2, \dots, |\mathcal{S}|\}$ . If we had a magical tool that could solve discrete Markov decision problems (remember we need to compute the one-step transition matrix), then we do not need to know anything about the state space  $\mathcal{S}$ , but this is rarely the case. Further, we make the case that just knowing that we have  $|\mathcal{S}|$  states provides no information about the problem itself, while a list of the variables that make up the state variable (as is done in optimal control) will map directly to the software implementing the model.
- Similarly, the MDP/RL community talks about action *spaces*, while the controls community uses control *variables*. There is a wide range of problems that are described by discrete actions, where the action space is not too large. However, there are also many problems where actions are continuous, and often are vector valued. The notation of an “action space”  $\mathcal{A}$  is simply not useful for vector-valued decisions/controls (the issue is the same as with state spaces).
- The MDP modeling framework does not explicitly model the exogenous information process  $w_t$ . Rather, it is buried in the one-step transition function  $p(s'|s, a)$ , as is seen in Eq. (3.14). In practical algorithms, we need to simulate the  $w_t$  process, so it helps to model the process explicitly. We would also argue that the model of  $w_t$  is a critical and challenging dimension of any sequential decision problem which is overlooked in the canonical MDP modeling framework.
- Transition functions, if known, are always computable since we just have to compute them for a single state, a single action, and a single observation of any exogenous information. We suspect that this is why the optimal control community

adopted this notation. One-step transition matrices (or one-step transition kernels if the state variable is continuous) are almost never computable.

- There is no proper statement of the objective function, beyond the specification of the reward function  $r(s, a)$ . There is an implied objective similar to Eq. (3.4), but as we are going to see below, objective functions for sequential decision problems come in a variety of styles. Most important, in our view, is the need to state the objective in terms of optimizing over *policies*.

We suspect that the reason behind the sharp difference in styles between optimal control and Markov decision processes (adopted by reinforcement learning) is that the field of optimal control evolved from engineering, while Markov decision processes evolved out of mathematics. The adoption of the MDP framework by reinforcement learning (which grew out of computer science and is particularly popular with less-mathematical communities) is purely historical—it was easier to make the connection between the discrete mouse-in-a-maze problem to the language of discrete Markov decision processes than stochastic optimal control.

In Sect. 3.4 below we are going to offer a framework that overcomes all of these limitations. This framework, however, closely parallels the framework widely used in optimal control, with a few relatively minor modifications (and one major one).

### 3.3.4 Bridging Optimal Control and Reinforcement Learning

We open our discussion by noting the remarkable difference between the canonical modeling framework for optimal control, which explicitly models state variables  $x_t$ , controls  $u_t$ , information  $w_t$ , and transition functions, and the canonical modeling framework for reinforcement learning (inherited from Markov decision processes) which uses constructs such as state spaces, action spaces, and one-step transition matrices. We will argue that the framework used in optimal control can be translated directly to software, whereas that used by reinforcement learning does not.

To illustrate this assertion, we note that optimal control and reinforcement learning are both addressing a sequential decision problem. In the notation of optimal control, we would write (focusing on discrete-time settings):

$$(x_0, u_0, w_0, x_1, u_1, w_1, \dots, x_T).$$

The reinforcement learning framework, on the other hand, never models anything comparable to the information variable  $w_t$ . In fact, the default setting is that we just observe the downstream state rather than modeling how we get there, but this is not universally true.

We also note that while the controls literature typically indexes variables by time, the RL community adopted the standard steady-state model (see Eq. (3.12)) which means their variables are not indexed by time (or anything else). Instead, they view

the system as evolving in steps (or iterations). For this reason, we are going to index variables by  $n$  (as in  $S^n$ ).

In addition, the RL community does not explicitly model an exogenous information variable. Instead, they tend to assume when you are in a state  $s$  and take action  $a$ , you then “observe” the next state  $s'$ . However, any simulation in a reinforcement learning model requires creating a transition function which may (but not always) involve some random information that we call “ $w$ ” (adopting, for the moment, the notation in the controls literature). This allows us to write the sequential decision problem

$$(S^0, a^0, (w^1), S^1, a^1, (w^2), S^2, \dots, S^N).$$

We put the  $(w^n)$  in parentheses because the RL community does not explicitly model the  $w^n$  process. However, when running an RL simulation, the software will have to model this process, even if we are just observing the next state. We use  $w^{n+1}$  after taking action  $a^n$  simply because it is often the case that  $S^{n+1} = w^{n+1}$ .

We have found that the reinforcement learning community likes to start by stating a model in terms of the MDP formal model, but then revert to the framework of stochastic control. A good example is the presentation by [23]; slide 22 presents the MDP formal model, but when the presentation turns to present an illustration (using a simple inventory problem), it turns to the style used in the optimal control community (see slide 29). Note that the presentation insists that the demand be stationary, which seems to be an effort to force it into the standard stationary model (see Eq. (3.12)). We use a much more complex inventory problem in this article, where we do not require stationarity (and which would not be required by the canonical optimal control framework).

So, we see that both optimal control and reinforcement learning are solving sequential decision problems, also known as Markov decision problems. Sequential decision problems (decision, information, decision, information, ...) span a truly vast range of applications, as noted in Sect. 3.2. We suspect that this space is much broader than has been traditionally viewed within either of these two communities. This is not to say that all these problems can be solved with  $Q$ -learning or even any Bellman-based method, but below we will identify four classes of policies that span any approach that *might* be used for any sequential decision problems.

The optimal control literature has its origins in problems with continuous states and actions, although the mathematical model does not impose any restrictions beyond the basic structure of sequential decisions and information (for stochastic control problems). While optimal control is best known for the theory surrounding the structure of linear-quadratic regulation which produces the linear policy in (3.8), it should not be surprising that the controls community branched into more general problems, requiring different solution strategies. These include

- Approximating the cost-to-go function  $J_t(x_t)$ .
- Determining a decision now by optimizing over a horizon  $t, \dots, t + H$  using a presumably known model of the system (which is not always available). This approach became known as *model predictive control*
- Specifying a parametric control law, which is typically linear in the parameters (following the style of (3.8)).

At the same time, the reinforcement learning community found that the performance of  $Q$ -learning (that is, Eqs. (3.9)–(3.10)), despite the hype, did not match early hopes and expectations. In fact, just as the optimal controls community evolved different solution methods, the reinforcement learning community followed a similar path (the same statement can be made of a number of fields in stochastic optimization). This evolution is nicely documented by comparing the first edition of Sutton and Barto's *Reinforcement Learning* [46], which focuses exclusively on  $Q$ -learning, with the second edition [47], which covers methods such as Monte Carlo tree search, upper-confidence bounding, and the policy gradient method.

We are going to next present (in Sect. 3.4) a universal framework which is illustrated in Sect. 3.5 on a series of problems in energy storage. Section 3.6 will then present four classes of policies that cover every method that has been proposed in the literature, which span all the variations currently in use in both the controls literature as well as the growing literature on reinforcement learning. We then return to the energy storage problems in Sect. 3.7 and illustrate all four classes of policies (including a hybrid).

### 3.4 The Universal Modeling Framework

We are going to present a universal modeling framework that covers all of the disciplines and application domains listed in Sect. 3.2. The framework will end up posing an optimization problem that involves searching over *policies*, which are functions for making decisions. We will illustrate the framework on a simple inventory problem using the setting of controlling battery storage (a classical stochastic control problem).

In Sect. 3.5, we will illustrate some key concepts by extending our energy storage application, focusing primarily on modeling state variables. Then, Sect. 3.6 describes a general strategy for designing policies, which we are going to claim covers *every* solution approach proposed in the research literature (or used in practice). Thus, we will have a path to finding solutions to *any* problem (but these are rarely optimal).

Before starting, we make a few notes on notation:

- The controls community uses  $x_t$  for state, while the reinforcement learning community adopted the widely used notation  $S_t$  for state. We have used  $S_t$  partly because of the mnemonics (making it easier to remember), but largely because  $x_t$  conflicts with the notation for decisions adopted by the field of math programming, which is widely used.

- There are three standard notational systems for decisions:  $a$  for action (typically discrete),  $u$  for control (typically a low-dimensional, continuous vector), and  $x$ , which is the notation used by the entire math programming community, where  $x$  can be continuous or discrete, scalar or vector. We adopt  $x$  because of how widely it is used in math programming, and because it has been used in virtually every setting (binary, discrete, continuous, scalar, or vector). It has also been adopted in the multi-armed bandit community in computer science.
- The controls community uses  $w_t$ , which is (sadly) random at time  $t$ , whereas all other variables are known at time  $t$ . We prefer the style that every variable indexed by time  $t$  (or iteration  $n$ ) is known at time  $t$  (or iteration  $n$ ). For this reason, we use  $W_t$  for the exogenous information that first becomes known between  $t - 1$  and  $t$ , which means it is known at time  $t$ . (Similarly,  $W^n$  would be information that becomes known between iterations/observations  $n - 1$  and  $n$ .)

### 3.4.1 Dimensions of a Sequential Decision Model

There are five elements to any sequential decision problem: state variables, decision variables, exogenous information variables, transition function, and objective function. We briefly describe each below, returning in Sect. 3.4.2 to discuss state variables in more depth. The description below is adapted from [31].

**State variables** —The state  $S_t$  of the system at time  $t$  (we might say  $S^n$  after  $n$  iterations) is a function of history which contains all the information that is necessary and sufficient to compute costs/rewards, constraints, and any information needed by the transition function. The state  $S_t$  typically consists of a number of dimensions which we might write as  $S_t = (S_{t1}, \dots, S_{tK})$ . This will be more meaningful when we illustrate it with an example below.

We distinguish between the initial state  $S_0$  and the dynamic state  $S_t$  for  $t > 0$ . The initial state contains all deterministic parameters, initial values of any dynamic parameters, and initial beliefs about unknown parameters in the form of the parameters of probability distributions. The dynamic state  $S_t$  contains only information that is evolving over time.

In Sect. 3.4.2, we will distinguish different classes of state variables, including physical state variables  $R_t$  (which might describe inventories or the location of a vehicle), other information  $I_t$  (which might capture prices, weather, or the humidity in a laboratory), and beliefs  $B_t$  (which includes the parameters of probability distributions describing unobservable parameters). It is sometimes helpful to recognize that  $(R_t, I_t)$  capture everything that can be observed perfectly, while  $B_t$  represents distributions of anything that is uncertain.

**Decision variables** —We use  $x_t$  for decisions, where  $x_t$  may be binary (e.g., for a stopping problem), discrete (e.g., an element of a finite set), continuous (scalar or vector), integer vectors, and categorical (e.g., the attributes of a patient). In some applications  $x_t$  might have hundreds of thousands, or even millions, of

*dimensions*, which makes the concept of “action spaces” fairly meaningless. We note that entire fields of research are sometimes distinguished by the nature of the decision variable.

We assume that decisions are made with a policy, which we might denote  $X^\pi(S_t)$ . We also assume that a decision  $x_t = X^\pi(S_t)$  is feasible at time  $t$ . We let “ $\pi$ ” carry the information about the type of function  $f \in \mathcal{F}$  (for example, a linear model with specific explanatory variables, or a particular nonlinear model), and any tunable parameters  $\theta \in \Theta^f$ .

**Exogenous information** —We let  $W_t$  be any new information that first becomes known at time  $t$  (that is, between  $t - 1$  and  $t$ ). This means any variable indexed by  $t$  is known at time  $t$ . When modeling specific variables, we use “hats” to indicate exogenous information. Thus,  $\hat{D}_t$  could be the demand that arose between  $t - 1$  and  $t$ , or we could let  $\hat{p}_t$  be the change in the price between  $t - 1$  and  $t$ . The exogenous information process may be stationary or nonstationary, purely exogenous or state (and possibly action) dependent.

As with decisions, the exogenous information  $W_t$  might be scalar, or it could have thousands to millions of *dimensions* (imagine the number of new customer requests for trips from zone  $i$  to zone  $j$  in an area that has 20,000 zones).

The distribution of  $W_{t+1}$  (given we are at time  $t$ ) may be described by a known mathematical model, or we may depend on observations from an exogenous source (this is known as “data driven”). The exogenous information may depend on the current state and/or action, so we might write it as  $W_{t+1}(S_t, x_t)$ . We will suppress this notation moving forward, but with the understanding that we allow this behavior.

**Transition function** —We denote the transition function by

$$S_{t+1} = S^M(S_t, x_t, W_{t+1}), \quad (3.15)$$

where  $S^M(\cdot)$  is also known by names such as system model, state equation, plant model, plant equation and transfer function. We have chosen not to use the standard notation  $f(s, x, w)$  used universally by the controls community simply because the letter  $f$  is also widely used for “functions” in many settings. The alphabet is very limited and the letter  $f$  occupies a valuable piece of real-estate.

An important problem class in both optimal control and reinforcement learning arises when the transition function is unknown. This is sometimes referred to as “model-free dynamic programming.” There are some classes of policies that do not need a transition function, both others do, introducing the dimension of trying to learn the transition function.

**Objective functions** —There are a number of ways to write objective functions in sequential decision problems. Our default notation is to let

$$C_t(S_t, x_t) = \text{the contribution of taking action } x_t \text{ given the information in state } S_t.$$

For now we are going to use the most common form of an objective function used in both dynamic programming (which includes reinforcement learning) and

stochastic control, which is to maximize the expected sum of contributions:

$$\max_{\pi} \mathbb{E}_{S_0} \mathbb{E}_{W_1, \dots, W_T | S_0} \left\{ \sum_{t=0}^T C_t(S_t, X_t^\pi(S_t)) | S_0 \right\}, \quad (3.16)$$

where

$$S_{t+1} = S^M(S_t, X_t^\pi(S_t), W_{t+1}), \quad (3.17)$$

and where we are given a source of the exogenous information process

$$(S_0, W_1, W_2, \dots, W_T). \quad (3.18)$$

We refer to Eq. (3.16) along with the state transition function (3.17) and exogenous information (3.18) as the *base model*. We revisit objective functions in Sect. 3.4.3.

An important feature of our modeling framework is that we introduce the concept of a policy  $X^\pi(S_t)$  when we describe decisions, and we search over policies in the objective function in Eq. (3.16), but we do not at this point specify what the policies might look like. Searching over policies is precisely what is meant by insisting that the control  $u_t$  in Eq. (3.4) be “ $\mathcal{F}_t$ -measurable.” In Sect. 3.6 we are going to make this much more concrete, and does not require mastering subtle concepts such as “measurability.” All that is needed is the understanding that a policy depends on the state variable (measurability is guaranteed when this is the case).

In other words (and as promised), we have modeled the problem without specifying how we would solve them (that is, we have not specified how we are computing the policy). This follows our “*Model first, then solve*” approach. Contrast this with the  $Q$ -learning Eqs. (3.9)–(3.10) which is basically an algorithm without a model, although the RL community would insist that the model is the canonical MDP framework given in Sect. 3.3.2.

### 3.4.2 State Variables

Our experience is that there is an almost universal misunderstanding of what is meant by a “state variable.” Not surprisingly, interpretations of the term “state variable” vary between communities. An indication of the confusion can be traced to attempts to define state variables. For example, Bellman introduces state variables with “we have a physical system characterized at any stage by a small set of parameters, the *state variables*” [2]. Puterman’s now classic text introduces state variables with “At each decision epoch, the system occupies a *state*.” Puterman [36][p. 18] (in both cases, the italicized text was included in the original text). As of this writing, Wikipedia offers “A state variable is one of the set of variables that are used to describe the

mathematical ‘state’ of a dynamical system.” Note that all three references use the word “state” in the definition of state variable.

It has also been our finding that most books in optimal control do, in fact, include proper definitions of a state variable (our experience is that this is the only field that does this). They all tend to say the same thing: a state variable  $x_t$  is all the information needed to model the system from time  $t$  onward.

Our only complaint about the standard definition used in optimal control books is that it is vague. The definition proposed in [32] (building on the definition in [29]) refines the basic definition with the following:

A **state variable** is:

- (a) **Policy-dependent version** A function of history that, combined with the exogenous information (and a policy), is necessary and sufficient to compute the decision function (the policy), the cost/contribution function, and the transition function.
- (b) **Optimization version** A function of history that, combined with the exogenous information, is necessary and sufficient to compute the cost or contribution function, the constraints, and the transition function.

There are three types of information in  $S_t$ :

- The physical state,  $R_t$ , which in most (but not all) applications is the state variables that are being controlled.  $R_t$  may be a scalar, or a vector with element  $R_{ti}$  where  $i$  could be a type of resource (e.g., a blood type) or the amount of inventory at location  $i$ . Physical state variables typically appear in the constraints. We make a point of singling out physical states because of their importance in modeling resource allocation problems, where the “state of the system” is often (and mistakenly) equated with the physical state.
- Other information,  $I_t$ , which is any information that is known deterministically not included in  $R_t$ . The information state often evolves exogenously, but may be controlled or at least influenced by decisions (e.g., selling a large number of shares may depress prices). Other information may appear in the objective function (such as prices), and the coefficients in the constraints.
- The belief state  $B_t$ , which contains distributional information about unknown parameters, where we can use frequentist or Bayesian belief models. These may come in the following styles:
  - Lookup tables—Here we have a set of discrete values  $x \in \mathcal{X} = \{x_1, \dots, x_M\}$ , and we have a belief about a function (such as  $f(x) = \mathbb{E}F(x, W)$ ) for each  $x \in \mathcal{X}$ .
  - Parametric belief models—We might assume that  $\mathbb{E}F(x, W) = f(x|\theta)$  where the function  $f(x|\theta)$  is known but where  $\theta$  is unknown. We would then describe  $\theta$  by a probability distribution.
  - Nonparametric belief models—These approximate a function at  $x$  by smoothing local information near  $x$ .

It is important to recognize that the belief state includes the parameters of a probability distribution describing unobservable parameters of the model. For example,

$B_t$  might be the mean and covariance matrix of a multivariate normal distribution, or a vector of probabilities  $p^n = (p_k^n)_{k=1}^K$  where  $p_k^n = \text{Prob}[\theta = \theta_k | S^n]$ .<sup>1</sup>

We feel that a proper understanding of state variables opens up the use of the optimal control framework to span the entire set of communities and applications discussed in Sect. 3.2.

### 3.4.3 Objective Functions

Sequential decision problems are diverse, and this is reflected in the different types of objective functions that may be used. Our framework is insensitive to the choice of objective function, but they all still require optimizing over policies.

We begin by making the distinction between state-independent problems, and state-dependent problems. We let  $F(x, W)$  denote a state-independent problem, where we assume that neither the objective function  $F(x, W)$ , nor any constraints, depends on dynamic information captured in the state variable. We let  $C(S, x)$  capture state-dependent problems, where the objective function (and/or constraints) may depend on dynamic information.

Throughout we assume problems are formulated over finite time horizons. This is the most standard approach in optimal control, whereas the reinforcement learning community adopted the style of Markov decision processes to model problems over infinite time horizons. We suspect that the difference reflects the history of optimal control, which is based on solving real engineering problems, and Markov decision processes, with its roots in mathematics and stylized problems.

In addition to the issue of state-dependency, we make the distinction between optimizing the cumulative reward versus the final reward. When we combine state dependency and the issue of final versus cumulative reward, we obtain four objective functions. We present these in the order: (1) State-independent, final reward, (2) state-independent, cumulative reward, (3) state-dependent, cumulative reward, and (4) state-dependent, final reward (the last class is the most subtle).

**State-independent functions** These are pure learning problems, where the *problem* does not depend on information in the state variable. The only state variable is the belief about an unknown function  $\mathbb{E}_W F(x, W)$ .

---

<sup>1</sup>It is not unusual for people to overlook the need to include beliefs in the state variable. The RL tutorial [23] does this when it presents the multi-armed bandit problem, insisting that it does not have a state variable (see slide 49). In fact, any bandit problem is a sequential decision problem where the state variable is the belief (which can be Bayesian or frequentist). This has long been recognized by the probability community that has worked on bandit problems since the 1950s (see the seminal text [14]). Bellman's equation (using belief states) was fundamental to the development of Gittins indices in [16] (see [17] for a nice introduction to this rich area of research). It was the concept of Gittins indices that laid the foundation for upper-confidence bounding, which is just a different form of index policy.

**(1) Final reward** This is the classical stochastic search problem. Here we go through a learning/training process to find a final design/decision  $x^{\pi, N}$ , where  $\pi$  is our search policy (or algorithm), and  $N$  is the budget. We then have to test the performance of the policy by simulating  $\widehat{W}$  using

$$\max_{\pi} \mathbb{E}_{S^0} \mathbb{E}_{W^1, \dots, W^N | S^0} \mathbb{E}_{\widehat{W} | S^0} F(x^{\pi, N}, \widehat{W}), \quad (3.19)$$

where  $x^{\pi, N}$  depends on  $S^0$  and the experiments  $W^1, \dots, W^N$ , and where  $\widehat{W}$  represents the process of testing the design  $x^{\pi, N}$ .

**(2) Cumulative reward** This describes problems where we have to learn in the field, which means that we have to optimize the sum of the rewards we earn, eliminating the need for a final testing pass. This objective is written

$$\max_{\pi} \mathbb{E}_{S^0} \mathbb{E}_{W^1, \dots, W^N | S^0} \sum_{n=0}^{N-1} F(X^{\pi}(S^n), W^{n+1}). \quad (3.20)$$

**State-dependent functions** This describes the massive universe of problems where the objective and/or the constraints depend on the state variable which may or may not be controllable.

**(3) Cumulative reward** This is the version of the objective function that is most widely used in stochastic optimal control (as well as Markov decision processes). We switch back to time-indexing here since these problems are often evolving over time (but not always). We write the contribution in the form  $C(S_t, x_t, W_{t+1})$  to help with the comparison to  $F(x, W)$ .

$$\max_{\pi} \mathbb{E}_{S^0} \mathbb{E}_{W_1, \dots, W_T | S^0} \left\{ \sum_{t=0}^T C(S_t, X^{\pi}(S_t), W_{t+1}) | S^0 \right\}. \quad (3.21)$$

**(4) Final reward** This is the objective function that describes optimization algorithms (represented as  $\pi^{lrn}$ ) optimizing a time-staged, state-dependent objective. This is the objective that should be used when finding the best algorithm for a dynamic program/stochastic control problem, yet has been almost universally overlooked as a sequential decision problem. The objective is given by

$$\max_{\pi^{lrn}} \mathbb{E}_{S^0} \mathbb{E}_{W^1, \dots, W^N | S^0} \mathbb{E}_{S|S^0} \mathbb{E}_{\widehat{W}|S^0} C(S, X^{\pi^{imp}}(S|\theta^{\pi^{imp}}), \widehat{W}). \quad (3.22)$$

where  $\pi^{lrn}$  is the learning policy (or algorithm), while  $\pi^{imp}$  is the implementation policy that we are learning through  $\pi^{lrn}$ . We note that we use the learning policy  $\pi^{lrn}$  to learn the parameters  $\theta^{\pi^{imp}}$  that govern the behavior of the implementation policy.

There are many problems that require more complex objective functions such as the best (or worst) performance in a time period, across all time periods. In these settings we cannot simply sum the contributions across time periods (or iterations). For this purpose, we introduce the operator  $\rho$  which takes as input the entire sequence of contributions. We would write our objective function as

$$\max_{\pi} \mathbb{E}_{S^0} \mathbb{E}_{W^1, \dots, W^N | S^0} \rho(C_0(S_0, X^\pi(S_0), W_1), C_2(S_2, X^\pi(S_2), W_3), \dots, C_t(S_t, X^\pi_t(S_t), W_{t+1}), \dots). \quad (3.23)$$

The objective in (3.23), through creative use of the operator  $\rho$ , subsumes all four objectives (3.19)–(3.22). However, we feel that generality comes at a cost of clarity.

The controls community, while also sharing an interest in risk, is also interested in stability, an issue that is important in settings such as controlling aircraft and rockets. While we do not address the specific issue of designing policies to handle stability, we make the case that the problem of searching over policies remains the same; all that has changed is the metric.

All of these objectives can be written in the form of *regret* which measures the difference between the solution we obtain and the best possible. Regret is popular in the learning community where we compare against the solution that assumes perfect information. A comparable strategy compares the performance of a policy against what can be achieved with perfect information about the future (widely known as a posterior bound).

### 3.4.4 Notes

It is useful to list some similarities (and differences) between our modeling framework and that used in stochastic optimal control:

- (1) The optimal control framework includes all five elements, although we lay these out more explicitly.
- (2) We use a richer understanding of state variables, which means that we can apply our framework to a much wider range of problems than has traditionally been considered in the optimal control literature. In particular, *all* the fields and problem areas in Sect. 3.2 fit this framework, which means we would say that all of these are “optimal control problems.”
- (3) The stochastic control modelling framework uses  $w_t$  as the information that will arrive between  $t$  and  $t + 1$ , which means it is random at time  $t$ . We let  $W_t$  be the information that arrives between  $t - 1$  and  $t$ , which means it is known at time  $t$ . This means we write our transition as

$$S_{t+1} = S^M(S_t, x_t, W_{t+1}).$$

This notation makes it explicitly clear that  $W_{t+1}$  is not known when we determine decision  $x_t$ .

- (4) We recognize a wider range of objective functions, which expands the problem classes to offline and online applications, active learning (bandit) problems, and hybrids.
- (5) We formulate the optimization problem in terms of optimizing over policies, without prejudging the classes of policies. We describe four classes of policies in Sect. 3.6 that we claim are universal: they cover all the strategies that have been proposed or used in practice. This also opens the door to creating hybrids that combine two or more classes of policies.

The first four items are relatively minor, highlighting our belief that stochastic control is fundamentally the most sound of all the modeling frameworks used by any of the communities listed in Sect. 3.2. However, the fifth item is a significant transition from how sequential decision problems are approached today.

Many have found that  $Q$ -learning often does not work well. In fact,  $Q$ -learning, as with all approximate dynamic programming algorithms, tend to work well only on a fairly small set of problems. Our experience is that approximate dynamic programming ( $Q$ -learning is a form of approximate dynamic programming) tends to work well when we can exploit the structure of the value function. For example, ADP has been very successful with some very complex, high-dimensional problems in fleet management (see [41] and [7]) where the value functions were convex. However, vanilla approximation strategies (e.g., using simple linear models for value function approximations) can work very poorly even on small inventory problems (see [19] for a summary of experiments which compare results against a rigorous benchmark). Furthermore, as we will see in Sect. 3.6 below, there are a range of policies that do not depend on value functions that are natural choices for many applications.

## 3.5 Energy Storage Illustration

We are going to illustrate our modeling framework using the energy system depicted in Fig. 3.3, which consists of a wind farm (where energy is free but with high variability in supply), the grid (which has unlimited supply but highly stochastic prices), a market (which exhibits very time-dependent, although relatively predictable, demands), and an energy storage device (we will assume it is a battery). While small, this rich system introduces a variety of modeling and algorithmic challenges.

We are going to demonstrate how to model this problem, starting with a simple model and then expanding to illustrate some modeling devices. We will translate each variation into the five core components: state variables, decision variables, exogenous information variables, transition function, and objective function.



**Fig. 3.3** Energy system consisting of wind farms, grid, market, and battery storage

### 3.5.1 A Basic Energy Storage Problem

**State variables** State  $S_t = (R_t, D_t, E_t, p_t)$  where

$R_t$  = Energy in the battery at time  $t$ ,

$E_t$  = Power being produced by the wind farms at time  $t$ ,

$D_t$  = Demand for power at time  $t$ ,

$p_t$  = Price of energy on the grid at time  $t$ .

Note that it is necessary to go through the rest of the model to determine which variables are needed to compute the objective function, constraints, and transition function.

**Decision variables**  $x_t = (x_t^{GB}, x_t^{GD}, x_t^{EB}, x_t^{ED}, x_t^{BD})$  where

$x_t^{GB}$  = Flow of energy from grid to battery ( $x_t^{GB} > 0$ ) or back ( $x_t^{GB} < 0$ ),

$x_t^{GD}$  = Flow of energy from grid to demand,

$x_t^{EB}$  = Flow of energy from wind farm to battery,

$x_t^{ED}$  = Flow of energy from wind farm to demand,

$x_t^{BD}$  = Flow of energy from battery to demand.

These decision variables have to obey the constraints:

$$x_t^{EB} + x_t^{ED} \leq E_t, \quad (3.24)$$

$$x_t^{GD} + x_t^{BD} + x_t^{ED} = D_t, \quad (3.25)$$

$$x_t^{BD} \leq R_t, \quad (3.26)$$

$$x_t^{GD}, x_t^{EB}, x_t^{ED}, x_t^{BD} \geq 0. \quad (3.27)$$

Finally, we introduce the policy (function)  $X^\pi(S_t)$  that will return a feasible vector  $x_t$ . We defer to later the challenge of designing a good policy.

**Exogenous information variables**  $W_{t+1} = (\hat{E}_{t+1}, \hat{D}_{t+1}, \hat{p}_{t+1})$ , where

$\hat{E}_{t+1}$  = The change in the power from the wind farm between  $t$  and  $t + 1$ ,

$\hat{D}_{t+1}$  = The change in the demand between  $t$  and  $t + 1$ ,

$\hat{p}_{t+1}$  = The price charged at time  $t + 1$  as reported by the grid.

We note that the first two exogenous information variables are defined as changes in values, while the last (price) is reported directly from an exogenous source.

**Transition function**  $S_t = S^M(S_t, x_t, W_{t+1})$ :

$$R_{t+1} = R_t + \eta(x_t^{GB} + x_t^{EB} - x_t^{BD}), \quad (3.28)$$

$$E_{t+1} = E_t + \hat{E}_{t+1}, \quad (3.29)$$

$$D_{t+1} = D_t + \hat{D}_{t+1}, \quad (3.30)$$

$$p_{t+1} = \hat{p}_{t+1}. \quad (3.31)$$

Note that we have illustrated here a controllable transition (3.28), two where the exogenous information is represented as the change in a process (Eqs. (3.29) and (3.30)), and one where we directly observe the updated price (3.31). This means that the processes  $E_t$  and  $D_t$  are first-order Markov chains (assuming that  $\hat{E}_t$  and  $\hat{D}_t$  are independent across time), while the price process would be described as “model free” or “data driven” since we are not assuming that we have a mathematical model of the price process.

**Objective function** We wish to find a policy  $X^\pi(S_t)$  that solves

$$\max_\pi \mathbb{E}_{S_0} \mathbb{E}_{W_1, \dots, W_T | S_0} \left\{ \sum_{t=0}^T C(S_t, X^\pi(S_t)) | S_0 \right\},$$

where  $S_{t+1} = S^M(S_t, x_t = X^\pi(S_t), W_{t+1})$  and where we are given an information process

$$(S_0, W_1, W_2, \dots, W_T).$$

Normally, we would transition at this point to describe how we are modeling the uncertainty in the information process  $(S_0, W_1, W_2, \dots, W_T)$ , and then describe how to design policies. For compactness, we are going to skip these steps now, and instead illustrate how to model a few problem variations that can often cause confusion.

### 3.5.2 With a Time-Series Price Model

We are now going to make a single change to the model above. Instead of assuming that prices  $p_t$  are provided exogenously, we are going to assume we can model them using a time series model given by

$$p_{t+1} = \theta_0 p_t + \theta_1 p_{t-1} + \theta_2 p_{t-2} + \varepsilon_{t+1}. \quad (3.32)$$

A common mistake is to say that  $p_t$  is the “state” of the price process, and then observe that it is no longer Markovian (it would be called “history dependent”), but “it can be made Markovian by expanding the state variable,” which would be done by including  $p_{t-1}$  and  $p_{t-2}$  (see [12] for an example of this). According to our definition of a state variable, the state is all the information needed to model the process from time  $t$  onward, which means that the state of our price process is  $(p_t, p_{t-1}, p_{t-2})$ . This means our system state variable is now

$$S_t = ((R_t, D_t, E_t), (p_t, p_{t-1}, p_{t-2})).$$

We then have to modify our transition function so that the “price state variable” at time  $t + 1$  becomes  $(p_{t+1}, p_t, p_{t-1})$ .

### 3.5.3 With Passive Learning

We implicitly assumed that our price process in Eq. (3.32) was governed by a model where the coefficients  $\theta = (\theta_0, \theta_1, \theta_2)$  were known. Now assume that the price  $p_{t+1}$  depends on prices over the last three time periods, which means we would write

$$p_{t+1} = \bar{\theta}_{t0} p_t + \bar{\theta}_{t1} p_{t-1} + \bar{\theta}_{t2} p_{t-2} + \varepsilon_{t+1}. \quad (3.33)$$

Here, we have to adaptively update our estimate  $\bar{\theta}_t$  which we can do using recursive least squares. To do this, let

$$\begin{aligned} \bar{p}_t &= (p_t, p_{t-1}, p_{t-2})^T, \\ \bar{F}_t(\bar{p}_t | \bar{\theta}_t) &= (\bar{p}_t)^T \bar{\theta}_t \end{aligned}$$

We perform the updating using a standard set of updating equations given by

$$\bar{\theta}_{t+1} = \bar{\theta}_t + \frac{1}{\gamma_t} M_t \bar{p}_t \varepsilon_{t+1}, \quad (3.34)$$

$$\varepsilon_{t+1} = \bar{F}_t(\bar{p}_t | \bar{\theta}_t) - p_{t+1}, \quad (3.35)$$

$$M_{t+1} = M_t - \frac{1}{\gamma_t} M_t (\bar{p}_t)^T M_t, \quad (3.36)$$

$$\gamma_t = 1 - (\bar{p}_t)^T M_t \bar{p}_t. \quad (3.37)$$

To compute these equations, we need the three-element vector  $\bar{\theta}_t$  and the  $3 \times 3$  matrix  $M_t$ . These then need to be added to our state variable, giving us

$$S_t = ((R_t, D_t, E_t), (p_t, p_{t-1}, p_{t-2}), (\bar{\theta}_t, M_t)).$$

We then have to include Eqs. (3.34)–(3.37) in our transition function.

### 3.5.4 With Active Learning

We can further generalize our model by assuming that our decision  $x_t^{GB}$  to buy or sell energy from or to the grid can have an impact on prices. We might propose a modified price model given by

$$p_{t+1} = \bar{\theta}_{t0} p_t + \bar{\theta}_{t1} p_{t-1} + \bar{\theta}_{t2} p_{t-2} + \bar{\theta}_{t3} x_t^{GB} + \varepsilon_{t+1}. \quad (3.38)$$

All we have done is introduce a single term  $\bar{\theta}_{t3} x_t^{GB}$  to our price model. Assuming that  $\theta_3 > 0$ , this model implies that purchasing power from the grid ( $x_t^{GB} > 0$ ) will increase grid prices, while selling power back to the grid ( $x_t^{GB} < 0$ ) decreases prices. This means that purchasing a lot of power from the grid (for example) means we are more likely to observe higher prices, which may assist the process of learning  $\theta$ . When decisions control or influence what we observe, then this is an example of *active learning*.

This change in our price model does not affect the state variable from the previous model, aside from adding one more element to  $\bar{\theta}_t$ , with the required changes to the matrix  $M_t$ . The change will, however, have an impact on the policy. It is easier to learn  $\theta_{t3}$  by varying  $x_t^{GB}$  over a wide range, which means trying values of  $x_t^{GB}$  that do not appear to be optimal given our current estimate of the vector  $\bar{\theta}_t$ . Making decisions partly just to learn (to make better decisions in the future) is the essence of *active learning*, best known in the field of multi-armed bandit problems.

### 3.5.5 With Rolling Forecasts

Forecasting is such a routine activity in operational problems, it may come as a surprise that we have been modelling these problems incorrectly.

Assume we have a forecast  $f_{t,t+1}^E$  of the energy  $E_{t+1}$  from wind, which means

$$E_{t+1} = f_{t,t+1}^E + \varepsilon_{t+1,1}, \quad (3.39)$$

where  $\varepsilon_{t+1,1} \sim N(0, \sigma_\varepsilon^2)$  is the random variable capturing the one-period-ahead error in the forecast.

Equation (3.39) effectively replaces Eq.(3.29) in the transition function for the base model. However, it introduces a new variable, the forecast  $f_{t,t+1}^E$ , which must now be added to the state variable. This means we now need a transition equation to describe how  $f_{t,t+1}^E$  evolves over time. We do this by using a two-period-ahead forecast,  $f_{t,t+2}^E$ , which is basically a forecast of  $f_{t+1,t+2}^E$ , plus an error, giving us

$$f_{t+1,t+2}^E = f_{t,t+2}^E + \varepsilon_{t+1,2}, \quad (3.40)$$

where  $\varepsilon_{t+1,2} \sim N(0, 2\sigma_\varepsilon^2)$  is the two-period-ahead error (we are assuming that the variance in a forecast increases linearly with time). Now we have to put  $f_{t,t+2}^E$  in the state variable, which generates a new transition equation. This generalizes to

$$f_{t+1,t'}^E = f_{t,t'}^E + \varepsilon_{t+1,t'-t}, \quad (3.41)$$

where  $\varepsilon_{t+1,t'-t} \sim N(0, (t' - t)\sigma_\varepsilon^2)$ .

This stops, of course, when we hit the planning horizon  $H$ . This means that we now have to add

$$f_t^E = (f_{tt'}^E)_{t'=t+1}^{t+H}$$

to the state variable, with the transition Eqs. (3.41) for  $t' = t + 1, \dots, t + H$ . Combined with the learning statistics, our state variable is now

$$S_t = ((R_t, D_t, E_t), (p_t, p_{t-1}, p_{t-2}), (\bar{\theta}_t, M_t), f_t^E).$$

It is useful to note that we have a nice illustration of the three elements of our state variable:

$(R_t, D_t, E_t)$  = The physical state variables (note that they all appear in the right hand side of constraints (24)–(25)),

$(p_t, p_{t-1}, p_{t-2})$  = other information,

$((\bar{\theta}_t, M_t), f_t^E)$  = the belief state, since these parameters determine the distribution of belief about variables that are not known perfectly.

### 3.5.6 Remarks

We note that all the models illustrated in this section are sequential decision problems, which means that all of them can be described as either stochastic control problems, or reinforcement learning problems. This is true whether state variables

or decision/control variables are scalar or vector, discrete or continuous (or mixed). We have, however, assume that time is either discrete or discretized.

Energy storage is a form of inventory problem, which is the original stochastic control problem used by Bellman to motivate his work on dynamic programming [3], and is even used today by the reinforcement learning community [23]. However, we have never seen the variations that we illustrated here solved by any of these communities.

In Sect. 3.6 we are going to present four classes of policies, and then illustrate, in Sect. 3.7, that each of the four classes (including a hybrid) can be applied to the full range of these energy storage problems. We are then going to show that both communities (optimal control and reinforcement learning) use methods that are drawn from each of the four classes, but apparently without an awareness that these are instances in broader classes, that can be used to solve complex problems.

## 3.6 Designing Policies

There are two fundamental strategies for creating policies:

**Policy search** —Here we use any of the objective functions (3.19)–(3.23) to search within a family of functions to find the policy that works best. This means we have to a) find a class of function and b) tune any parameters. The challenge is finding the right family, and then performing the tuning (which can be hard).

**Lookahead approximations** —Alternatively, we can construct policies by approximating the impact of a decision now on the future. The challenge here is designing and computing the approximation of the future (this is also hard).

Either of these approaches can yield optimal policies, although in practice this is rare. Below we show that each of these strategies can be further divided into two classes, creating four (meta)classes of policies for making decisions. We make the claim that these are universal, which is to say that *any* solution approach to any sequential decision problem will use a policy drawn from one of these four classes, or a hybrid of two or more classes.

### 3.6.1 Policy Search

Policy search involves tuning and comparing policies using the objective functions (3.19)–(3.23) so that they behave well when averaged over a set of sample paths. Assume that we have a class of functions  $\mathcal{F}$ , where for each function  $f \in \mathcal{F}$ , there is a parameter vector  $\theta \in \Theta^f$  that controls its behavior. Let  $X^\pi(S_t|\theta)$  be a function in class  $f \in \mathcal{F}$  parameterized by  $\theta \in \Theta^f$ , where  $\pi = (f, \theta)$ ,  $f \in \mathcal{F}, \theta \in \Theta^f$ . Policy search involves finding the best policy using

$$\max_{\pi \in (\mathcal{F}, \Theta^f)} \mathbb{E}_{S_0} \mathbb{E}_{W_1, \dots, W_T | S_0} \left\{ \sum_{t=0}^T C(S_t, X^\pi(S_t)) | S_0 \right\}. \quad (3.42)$$

In special cases, this can produce an optimal policy, as we saw for the case of linear-quadratic regulation (see Eq. (3.8)).

Since we can rarely find optimal policies using (3.42), we have identified two sub-classes within the policy search class:

**Policy function approximations (PFAs)** — Policy function approximations can be lookup tables, parametric or nonparametric functions, but the most common are parametric functions. This could be a linear function such as

$$X^\pi(S_t | \theta) = \theta_0 + \theta_1 \phi_1(S_t) + \theta_2 \phi_2(S_t) + \dots,$$

which parallels the linear control law in Eq. (3.8) (these are also known as “affine policies”). We might also use a nonlinear function such as an order-up-to inventory policy, a logistics curve, or a neural network. Typically there is no guarantee that a PFA is in the optimal class of policies. Instead, we search for the best performance within a class.

**Cost function approximations (CFAs)** — A CFA is

$$X^\pi(S_t | \theta) = \arg \max_{x \in \mathcal{X}_t^\pi(\theta)} \bar{C}_t^\pi(S_t, x | \theta),$$

where  $\bar{C}_t^\pi(S_t, x | \theta)$  is a parametrically modified cost function, subject to a parametrically modified set of constraints. A popular example known to the computer science community is interval estimation where a discrete alternative  $x \in \mathcal{X} = \{x_1, \dots, x_M\}$  is chosen which maximizes

$$X^{IE}(S^n | \theta^{IE}) = \arg \max_{x \in \mathcal{X}} (\bar{\mu}_x^n + \theta^{IE} \bar{\sigma}_x^n)$$

where  $\bar{\mu}_x^n$  is the current estimate of  $\mathbb{E}_W F(x, W)$  after  $n$  experiments, and where  $\bar{\sigma}_x^n$  is the standard deviation of the statistic  $\bar{\mu}_x^n$ . Here,  $\theta^{IE}$  is a parameter that has to be tuned.

CFAs are widely used for solving large scale problems such as scheduling an airline or planning a supply chain. For example, we might introduce slack into a scheduling problem, or buffer stocks for an inventory problem.

Policy search is best suited when the policy has clear structure, such as inserting slack in an airline schedule, or selling a stock when the price goes over some limit. Neural networks have become popular recently because they assume no structure, but the price of this generality is that extensive tuning is needed. We urge caution with the use of high-dimensional architectures such as neural networks. There are many problems where we expect the policy to exhibit structure, such as increasing the dosage of a drug with the weight of a patient, or setting the bid price of a stock as

a function of market indicators. Neural networks do not offer these guarantees, and would require a tremendous amount of training to produce this behavior.

### 3.6.2 Lookahead Approximations

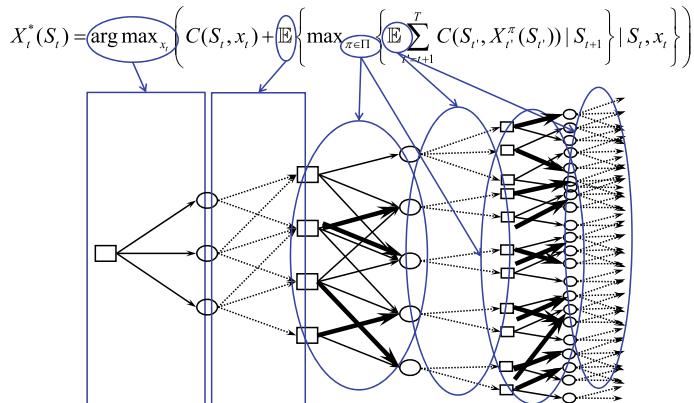
Just as we can, in theory, find an optimal policy using policy search, we can also find an optimal policy by modeling the downstream impact of a decision made now on the future. This can be written

$$X_t^*(S_t) = \arg \max_{x_t} \left( C(S_t, x_t) + \mathbb{E} \left\{ \max_{\pi} \mathbb{E} \left\{ \sum_{t'=t+1}^T C(S_{t'}, X_{t'}^\pi(S_{t'})) \mid S_{t+1} \right\} \mid S_t, x_t \right\} \right). \quad (3.43)$$

Equation (3.43) is daunting, but can be parsed in the context of a decision tree with discrete actions and discrete random outcomes (see Fig. 3.4). The states  $S_{t'}$  correspond to the square nodes in the decision tree. The state  $S_t$  is the initial node, and the actions  $x_t$  are the initial actions. The first expectation is over the first set of random outcomes  $W_{t+1}$  (out of the outcome nodes resulting from each decision  $x_t$ ). The imbedded policy  $\pi$  is the choice of decision for *each* decision node (state) over the horizon  $t+1, \dots, T$ . The second expectation is over  $W_{t+2}, \dots, W_T$ .

In practice, a stochastic lookahead policy is generally impossible to compute (decision trees grow exponentially). There are two broad strategies for approximating the lookahead model:

**Value function approximations (VFAs)** —Our first approach is to replace the entire term capturing the future in (3.43) with an approximation of the value



**Fig. 3.4** Relationship between the stochastic lookahead policy and a decision tree, showing initial decision, initial expectation, and then the decisions made for each state in the future (which is the lookahead policy  $\tilde{\pi}$ )

function (the controls community uses the term cost-to-go function). We can do this in two ways. The first is to replace the function starting at  $S_{t+1}$  with a value function  $V_{t+1}(S_{t+1})$  giving us

$$X_t^{VFA}(S_t) = \arg \max_{x_t} (C(S_t, x_t) + \mathbb{E} \{ \bar{V}_{t+1}(S_{t+1}) | S_t \}) \quad (3.44)$$

where  $S_{t+1} = S^M(S_t, x_t, W_{t+1})$ , and where the expectation is over  $W_{t+1}$  conditioned on  $S_t$  (some write the conditioning as dependent on  $S_t$  and  $x_t$ ). Since we generally cannot compute  $V_{t+1}(S_{t+1})$ , we can use various machine learning methods to replace it with some sort of approximation  $\bar{V}_{t+1}(S_{t+1})$  called the value function approximation.

The second way is to approximate the function around the post-decision state  $S_t^x$  (this is the state immediately after a decision is made), which eliminates the expectation (3.44), giving us

$$X_t^{VFA}(S_t) = \arg \max_{x_t} (C(S_t, x_t) + \bar{V}_t^x(S_t^x)). \quad (3.45)$$

The benefit of using the post-decision value function approximation is that it eliminates the expectation from within the max operator. This has proven to be especially useful for problems where  $x_t$  is a vector, and  $V_t^x(S_t^x)$  is a convex function of  $S_t^x$ .

There is by now an extensive literature on the use of value function approximations that have evolved under names such as heuristic dynamic programming [40], neuro-dynamic programming [5], adaptive dynamic programming ([24, 27]), approximate dynamic programming [29], and reinforcement learning [46]. While the use of value function approximations has tremendous appeal, it is no panacea. Our experience is that this approach works best only when we can exploit problem structure.

**Direct lookahead (DLAs)** There are many problems where it is just not possible to compute sufficiently accurate VFAs. When all else fails, we have to resort to a direct lookahead, where we replace the lookahead expectation and optimization in (3.43) with an *approximate lookahead model*.

The most widely used approximation strategy is to use a deterministic lookahead, often associated with model predictive control, although it is more accurate to refer to any policy based on solving a lookahead model as model predictive control. We can create an approximate (stochastic) lookahead model that we represent as the following sequence

$$(S_t, x_t, \tilde{W}_{t,t+1}, \tilde{S}_{t,t+1}, \tilde{x}_{t,t+1}, \dots, \tilde{W}_{tt'}, \tilde{S}_{tt'}, \tilde{x}_{tt'}, \tilde{W}_{t,t'+1}, \dots).$$

We use tilde-variables to indicate variables within the lookahead model. Each tilde-variable is indexed by  $t$  (the time at which the lookahead model is being formed) and  $t'$  (the time within the lookahead model). Our lookahead policy might then be written

$$X_t^{DLA}(S_t) = \arg \max_{x_t} \left( C(S_t, x_t) + \tilde{E} \left\{ \max_{\tilde{\pi}} \tilde{E} \left\{ \sum_{t'=t+1}^T C(\tilde{S}_{tt'}, \tilde{X}^{\tilde{\pi}}(\tilde{S}_{tt'})) | \tilde{S}_{t,t+1} \right\} | S_t, x_t \right\} \right). \quad (3.46)$$

Typically the approximate expectations  $\tilde{E}$  are computed using Monte Carlo sampling, although we can use a deterministic forecast. The real challenge is the lookahead policy  $\tilde{X}^{\tilde{\pi}}(\tilde{S}_{tt'})$  which may take any form. This policy is also known as a “rollout policy” where it is used in combinatorial optimization [6], and Monte Carlo tree search ([8, 11, 13]).

One possibility for the lookahead policy is to use a simpler parameterized policy that we might write  $\tilde{X}^{\tilde{\pi}}(\tilde{S}_{tt'}|\tilde{\theta})$ . In this case, the  $\max_{\tilde{\pi}}$  operator would be replaced with  $\max_{\tilde{\theta}}$ , but even this simpler problem means that we are finding the best parameter  $\tilde{\theta}$  for each state  $\tilde{S}_{t,t+1}$ , which means we are really looking for a function  $\tilde{\theta}(s)$  where  $s = \tilde{S}_{t,t+1}$ . A simpler alternative would be to fix a single parameter  $\theta$  which means we now have a parameterized lookahead policy given by

$$X_t^{DLA}(S_t|\theta) = \arg \max_{x_t} \left( C(S_t, x_t) + \tilde{E} \left\{ \sum_{t'=t+1}^T C(\tilde{S}_{tt'}, \tilde{X}^{\tilde{\pi}}(\tilde{S}_{tt'}|\theta)) | \tilde{S}_{t,t+1} \right\} | S_t, x_t \right). \quad (3.47)$$

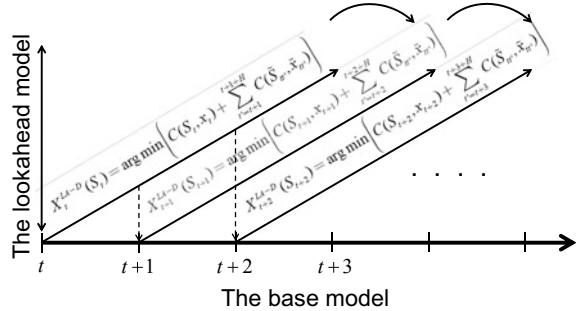
This version no longer has the imbedded  $\max_{\tilde{\theta}}$ , but we still have to tune  $\theta$  in the policy  $X_t^{DLA}(S_t|\theta)$ .

Another strategy for computing a stochastic lookahead policy is to use Monte Carlo tree search, a term coined by [13] but first proposed in [11] (see [8] for a tutorial in the context of deterministic problems). This strategy searches forward in time, using methods to limit the full enumeration of the tree. Monte Carlo tree search gained prominence from the role it played in creating AlphaGo for playing the Chinese game of Go, which was the first system to beat world class Go players (see [15] for a nice review of the history of MCTS and AlphaGo).

It is important to emphasize that designing a policy using a stochastic lookahead (even a simplified stochastic lookahead) means solving a stochastic optimization problem within the policy. Recall that our stochastic optimization problem is the base model given by any of the objective functions described earlier (Eqs. (3.19)–(3.23)). Equation (3.46) represents the simplified stochastic optimization problem, which has to be solved at each time period.

Figure 3.5 depicts the process of simulating a direct lookahead (the figure uses a deterministic lookahead, but the same process would be used with any direct lookahead). This is what is needed to do any parameter tuning for the DLA. Not surprisingly, stochastic lookaheads can be computationally difficult to solve, which makes it particularly difficult to run simulations to do parameter tuning.

**Fig. 3.5** Illustration of simulating a lookahead policy using a deterministic lookahead model



### 3.6.3 Hybrid Policies

A byproduct of identifying the four classes of policies is the ability to create hybrids that combine two or more classes. Some examples include:

- Lookahead policies plus VFAs—We can do an  $H$ -period lookahead, and then terminate with a value function approximation. This will likely simplify the task of coming up with a good value function approximation, while also providing better results for the same horizon (allowing us to shorten the horizon for the tree search).
- Value function approximations with parameter tuning. Imagine that we create a VFA-based policy that looks like

$$X^{VFA}(S_t|\theta) = \arg \max_x \left( C(S_t, x) + \sum_{f \in \mathcal{F}} \theta_f \phi_f(S_t, x_t) \right). \quad (3.48)$$

Assume we use ADP-based algorithms to determine an estimate of  $\theta$ . Now, using this value of  $\theta$  as an initial estimate, perform policy search by solving

$$\max_{\theta} \sum_{t=0}^T C(S_t, X^{VFA}(S_t|\theta)).$$

A nice illustration of this strategy is given in [26]. It is quite likely that performing this additional tuning (which can be expensive) will further improve the results. After performing parameter tuning, we can no longer view the linear term as an approximation of the value of being in a state. After tuning the policy, this is a form of CFA with cost function correction term.

- PFA with anything—A policy function approximation is any mapping of state to action without solving an optimization problem. PFAs are the simplest and easiest to control, but they cannot solve complex problems. The remaining three classes of policies are all cost-based, which allows them to be used for much more complex

problems (including problems where  $x_t$  is a high-dimensional vector). However, cost-based policies are harder to control.

It is possible to create a hybrid of a PFA and any cost-based policy. Assume we are using a VFA-based policy  $X^{VFA}(S_t|\theta^{VFA})$  (this could also be a direct lookahead or parametric CFA), which we would write as we did in Eq. (3.48), where we let  $\theta^{VFA}$  be the coefficients in the value function approximation. Now assume we are given some parametric function (a PFA) that we represent using  $X^{PFA}(S_t|\theta^{PFA})$ . We can write a hybrid policy using parameter vector  $\theta = (\theta^{VFA}, \theta^{PFA}, \theta^{PFA-VFA})$

$$X^{VFA-PFA}(S_t|\theta) = \arg \max_x \left( C(S_t, x) + \sum_{f \in \mathcal{F}} \theta_f \phi_f(S_t, x_t) + \theta^{PFA-VFA} \|x - X^{PFA}(S_t|\theta^{PFA})\| \right).$$

where  $\theta^{PFA-VFA}$  handles the scaling between the norm of the difference between  $x$  and the decision suggested by  $X^{PFA}(S_t|\theta^{PFA})$  and the rest of the cost-based objective function.

These hybrid policies help to emphasize the reason why we need to state the objective (as we did in Eqs. (3.19)–(3.23)) in terms of optimizing over *policies*.

### 3.6.4 Remarks

The academic literature is heavily biased toward the lookahead classes (VFAs and DLAs). These offer optimal policies, but computable optimal policies are limited to a very small class of problems: the linear control policy for LQR problems in optimal control, and lookup table value functions for problems with small state and action spaces, and a computable one-step transition matrix.

Approximating the value function has extended this framework, but only to a degree. Approximate/adaptive dynamic programming and  $Q$ -learning is a powerful tool, but again, algorithms that have been demonstrated empirically to provide near-optimal policies are rare. Readers have to realize that just because an algorithm enjoys asymptotic optimality (or attractive regret bounds) does not mean that it is producing near-optimal solutions in practice.

It is our belief that the vast majority of real-world sequential decision problems are solved with policies from the policy search class (PFAs and CFAs). PFAs have received some attention from the research literature (neural networks, linear/affine control policies). CFAs, on the other hand, are widely used in practice, yet have received minimal attention in the academic literature.

We would argue that PFAs and CFAs should have a place alongside parametric models in machine learning. A limitation is that they require a human to specify the structure of the parameterization, but this is also a feature: it is possible for domain experts to use their knowledge of a problem to capture structure. Most important is that PFAs and CFAs tend to be much simpler than policies based on value functions and lookaheads. But, *the price of simplicity is tunable parameters*, and tuning is hard.

### 3.6.5 Stochastic Control, Reinforcement Learning, and the Four Classes of Policies

The fields of stochastic control and reinforcement learning both trace their origins to a particular model that leads to an optimal policy. Stochastic control with additive noise (see Eq. (3.3)) produced an optimal policy from the original deterministic model with a quadratic objective function, given by  $u_t = K_t x_t$ . Reinforcement learning owes its origins to the field of Markov decision processes, which also produces an optimal policy for discrete problems, where the one-step transition matrix can be computed (see both (3.11) and (3.12)).

What then happened to both fields is the realization that these optimal policies can only be used in practice for a fairly narrow range of problems. For this reason, both communities evolved other strategies which can be viewed as being drawn from each of the four classes of policies we have described.

**Optimal control** The following policies can be found in the optimal control literature:

**Policy function approximations** These describe a wide range of simple rules used in every day problems. Some examples are:

- Buy low, sell high—These are simple rules for buying or selling assets.
- $(s, S)$  inventory policies—A widely used policy for inventory control is to place an order when the inventory  $R_t < s$ , in which case we order  $S - R_t$ .
- Linear control laws—Drawing on the optimal policy  $u_t = K_t x_t$  for the LQR problems, the controls community branched into general linear “control laws” which we might write as

$$U^\pi(x_t|\theta) = \sum_{f \in \mathcal{F}} \theta_f \phi_f(x_t),$$

where someone with domain knowledge needs to choose the features  $\phi_f(x_t)$ ,  $f \in \mathcal{F}$ , after which the parameters  $\theta$  need to be tuned.

**Value function approximation** The optimal control literature was using neural networks to approximate “cost-to-go” functions since the 1970s [49]. This strategy has been pursued in the controls literature under names including heuristic/neuro/approximate/adaptive dynamic programming. See [40] for a nice summary of this line of research.

**Direct lookahead** The controls community has referred to policies based on optimizing over a planning horizon as *model predictive control*. The most common strategy is to solve deterministic lookahead models. Most optimal control problems are deterministic, but using a deterministic approximation of the future is typical even when the underlying problem is stochastic (see [10] and [18] for thorough introductions).

**Parameterized MPC** Some authors in model predictive control have realized that you can obtain better results by introducing parameters as we did in our energy storage problem to handle the uncertainty in forecasts. This work has been done under the umbrella of “robust MPC” (see [21] and [37]). In our framework, this would be a hybrid direct lookahead-cost function approximation.

**Reinforcement learning** —The following policies are all contained in [47]:

**Policy function approximation** A popular policy for discrete action spaces is to choose an action based on the Boltzmann distribution given by

$$p(a|\theta, s) = \frac{e^{\theta \bar{\mu}_a}}{1 + \sum_{a' \in \mathcal{A}_s} e^{\theta \bar{\mu}_{a'}}}$$

where  $\bar{\mu}_a$  is the current estimate (contained in the state  $s$ ) of the value of action  $a$ . The policy is parameterized by  $\theta$ , which can be optimized using several methods, one of which is known as the “policy gradient method.”

In addition to using the policy gradient method on a Boltzmann policy, a number of papers approximate the policy with a neural network. If  $\theta$  is the weights of the neural network, then we have a high-dimensional parameter search problem that we can approach using stochastic gradient algorithms (see, e.g., [43]), although the problem is not easy; simulating policies is noisy, and the problem is not convex.

**Cost function approximation** Upper confidence bounding (for multi-armed bandit problems) is a classic CFA:

$$X^{CFA}(S^n|\theta) = \arg \max_a \left( \bar{\mu}_a^n + \theta \sqrt{\frac{\ln n}{N_a^n}} \right),$$

where  $N_a^n$  is the number of times we have tried action  $a$  after  $n$  iterations. UCB policies enjoy nice regret bounds [9], but it is still important to tune  $\theta$ .

**VFA-based policy** This would be  $Q$ -learning, where the policy is given by

$$X^{VFA}(S^n) = \arg \max_a \tilde{Q}^n(S^n, a).$$

**Direct lookaheads** Monte Carlo tree search is a classic direct lookahead. Since MCTS is a stochastic lookahead, it has a policy within the lookahead policy. This policy looks like

$$\tilde{X}_{tt'}^\pi(\tilde{S}_{tt'}|\theta^{UCT}) = \arg \max_{\tilde{x}_{tt'}} \left( \tilde{C}(\tilde{S}_{tt'}, \tilde{x}_{tt'}) + \tilde{V}_{tt'}^x(\tilde{S}_{tt'}^x) + \theta^{UCT} \sqrt{\frac{\log N_x^n}{N^n(\tilde{S}_{tt'}, \tilde{x}_{tt'})}} \right). \quad (3.49)$$

Note that this lookahead policy uses both a value function approximation as well as a bonus term from upper confidence bounding. This logic is known as “upper confidence bounding on trees,” abbreviated UCT. Thus, this is a hybrid policy (CFA with VFA) within a stochastic lookahead.

So, we see that both the optimal control and reinforcement learning communities are actively using strategies drawn from all four classes of policies. The same evolution has happened in the simulation-optimization community and the multi-armed bandit community. In the case of multi-armed bandit problems, there are actually distinct communities pursuing the different classes of policies:

- PFAs—Random sampling of experiments would constitute a PFA (this is a default policy, often used implicitly in statistics).
- CFAs—Upper confidence bounding is a popular policy for bandit problems in computer science [9].
- VFAs—The applied probability community has long used a decomposition technique to produce a series of dynamic programs which can be solved (one “arm” at a time) to obtain Gittins indices [17].
- DLAs—These include expected improvement, knowledge gradient and kriging developed in applied math, operations research and geosciences (see the tutorial in [33] or the book [35] for overviews).

We now return to our energy storage problem.

## 3.7 Policies for Energy Storage

We can illustrate all four classes of policies using our energy storage problem. Note that fully developing any of these classes would require a serious effort, so these are going to be little more than brief illustrations.

**Policy function approximation** —As a simple illustration of a policy function approximation, consider a buy-low, sell-high policy for handling the charging and discharging of a battery connected to the grid. This policy would be written

$$X^{PFA}(S_t|\theta) = \begin{cases} +1 & p_t < \theta^{charge} \\ 0 & \theta^{charge} \leq p_t \leq \theta^{discharge} \\ -1 & p_t > \theta^{discharge} \end{cases}. \quad (3.50)$$

Figure 3.6 shows an example of a parameterized PFA for the energy system in Fig. 3.3, where we have highlighted four tunable parameters. Designing these policies (especially the one in Fig. 3.6) is an art that requires an understanding of the structure of the problem. Tuning is an algorithmic exercise.

It is important to recognize that we have written our PFAs using parameters  $\theta$  that do not depend on time  $t$ , yet it is clear that in such a time-dependent setting (due to time-of-day patterns and rolling wind forecasts), the policy should be time dependent. However, tuning a two (or four)-dimensional parameter vector  $\theta$  is much easier than tuning a time-dependent parameter vector  $(\theta_\tau)_{\tau=1}^{24}$ .

**Cost function approximation** We are going to present a hybrid cost function approximation with a direct lookahead below.

**Fig. 3.6** Parameterized policy function approximation for energy system in Fig. 3.3, with tunable parameters highlighted

$$X_t^{PFA}(S_t|\theta) = \begin{cases} x_t^{EL} = \min\{L_t, E_t\} \\ x_t^{RL} = \begin{cases} \min\{L_t - x_t^{EL}, \min\{R_t(\rho^{dch})\}\} & \text{if } P_t > \theta^H \\ 0 & \text{if } P_t \leq \theta^H \end{cases} \\ x_t^{GL} = L_t - x_t^{EL} - x_t^{RL} \\ x_t^{ER} = \min\{E_t - x_t^{EL}(\rho^{ch}), R^{max} - R_t\} \\ x_t^{GR} = \begin{cases} \min\{\rho^{ch} - x_t^{ER}, R^{max} - R_t - x_t^{ER}\} & \text{if } P_t < \theta^L \\ 0 & \text{if } P_t \geq \theta^L \end{cases} \\ x_t^{RG} = \begin{cases} \min\{R_t - x_t^{RL}(\rho^{dch}) - x^{RL}\} & \text{if } P_t > \theta^H \\ 0 & \text{if } P_t \leq \theta^H \end{cases} \end{cases}$$

**Value function approximation** We apply the VFA-based policy in Eq. (3.48) with the policy

$$x^{VFA}(S_t|\theta) = \arg \max_x \left( C(S_t, x) + (\theta_1 R_t + \theta_2 R_t^2 + \theta_3 (x_t^{EB} + x_t^{ED})^2 + \theta_4 (x_t^{ED} + x_t^{BD} + x_t^{GD})^2) \right).$$

There are a variety of strategies for fitting the coefficients  $\theta$  that have been developed under headings of reinforcement learning [47], approximate dynamic programming (see e.g., [29]) and adaptive dynamic programming [40]. Jiang et al. [19] describes an extensive series of tests using value function approximations, where we found that VFA-based policies only worked well when we could exploit structure (such as concavity).

**Direct lookahead** For time-varying problems with a rolling forecast, a natural choice is to do a deterministic lookahead. We do this by setting up a time-staged linear programming model to optimize all decisions over a planning horizon. This is a deterministic lookahead model, so we let the decisions in the lookahead model created at time  $t$  be represented by  $\tilde{x}_{tt'}$ , which produces

$$X^{DLA}(S_t) = \arg \max_{x_t, (\tilde{x}_{tt'}, t'=t+1, \dots, t+H)} \left( p_t(x_t^{GB} + x_t^{GD}) + \sum_{t'=t+1}^{t+H} \tilde{p}_{tt'}(\tilde{x}_{tt'}^{GB} + \tilde{x}_{tt'}^{GD}) \right) \quad (3.51)$$

subject to the following constraints. First, for time  $t$ , we have

$$x_t^{BD} - x_t^{GB} - x_t^{EB} \leq R_t, \quad (3.52)$$

$$\tilde{R}_{t,t+1} - (x_t^{GB} + x_t^{EB} - x_t^{BD}) = R_t, \quad (3.53)$$

$$x_t^{ED} + x_t^{BD} + x_t^{GD} = D_t, \quad (3.54)$$

$$x_t^{EB} + x_t^{ED} \leq E_t, \quad (3.55)$$

$$x_t^{GD}, x_t^{EB}, x_t^{ED}, x_t^{BD} \geq 0., \quad (3.56)$$

Then, for  $t' = t + 1, \dots, t + H$  we have

$$\tilde{x}_{tt'}^{BD} - \tilde{x}_{tt'}^{GB} - \tilde{x}_{tt'}^{EB} \leq \tilde{R}_{tt'}, \quad (3.57)$$

$$\tilde{R}_{t,t'+1} - (\tilde{x}_{tt'}^{GB} + \tilde{x}_{tt'}^{EB} - \tilde{x}_{tt'}^{BD}) = \tilde{R}_{tt'}, \quad (3.58)$$

$$\tilde{x}_{tt'}^{ED} + \tilde{x}_{tt'}^{BD} + \tilde{x}_{tt'}^{GD} = f_{tt'}^D, \quad (3.59)$$

$$\tilde{x}_{tt'}^{EB} + \tilde{x}_{tt'}^{ED} \leq f_{tt'}^E. \quad (3.60)$$

**Hybrid DLA-CFA** The policy defined by the lookahead model given by Eqs. (3.51)–(3.60) does not make any provision for handling uncertainty. The most significant source of uncertainty is the forecast of wind, which is represented deterministically in Eq.(3.60). One idea is to parameterize this constraint by replacing it with

$$\tilde{x}_{tt'}^{EB} + \tilde{x}_{tt'}^{ED} \leq \theta_{t'-T} f_{tt'}^E. \quad (3.61)$$

Now we would write the policy as  $X^{DLA-CFA}(S_t|\theta)$ , where  $\theta = (\theta_\tau)_{\tau=1}^H$  is a set of coefficients for a rolling set of forecasts over a horizon of length  $H$ . It is very important to note that  $\theta$  is not time-dependent, which means that a policy that needs to behave differently at different times of day becomes a stationary policy, because the forecasts capture all the time-dependent information, and the forecasts are captured in the state variable.

The policies in the policy search class, given by  $X^{PFA}(S_t|\theta)$  in Eq.(3.50), and  $X^{DLA-CFA}(S_t|\theta)$  using the parameterized constraint (3.61), both need to be tuned by solving

$$\max_{\theta} F^\pi(\theta) = \mathbb{E}_{S_0} \mathbb{E}_{W_1, \dots, W_T | S_0} \sum_{t=0}^T C(S_t, X^\pi(S_t|\theta)), \quad (3.62)$$

where  $S_t$  is governed by the appropriate system model as illustrated in Sect. 3.5, and associated information process.

It is unlikely that anyone would test all four classes of policies to see which is best. A notable exception is [34] which showed that any of the four classes of policies (or a hybrid) can work best by carefully choosing the data. It is important to realize that the four classes of policies are meta-classes: simply choosing a class does not mean that your problem is solved. Each class is actually a path to an entire range of strategies.

## 3.8 Extension to Multi-agent Systems

We can easily extend our framework to multi-agent systems by using the framework to model the environment associated with each agent. Let  $\mathcal{Q}$  be the set of agents and let  $q \in \mathcal{Q}$  represent a specific agent. There are four types of agents:

- The ground truth agent—This agent cannot make any decisions or perform any learning (i.e., anything that implies intelligence). This is the agent that would know the truth about unknown parameters that we are trying to learn, or which performs the modeling of physical systems that are being observed by other agents. Controlling agents are, however, able to change the ground truth.
- Controlling agents—These are agents that make decisions that act on other agents, or the ground truth agent (acting as the environment). Controlling agents may communicate information to other controlling and/or learning agents.
- Learning agents—These agents do not make any decisions, but can observe and perform learning (about the ground truth and/or other controlling agents), and communicate beliefs to other agents.
- Combined controlling/learning agents—These agents perform learning through observations of the ground truth or other agents, as well as making decisions that act on the ground truth or other agents.

Now take every variable in our framework and introduce the index  $q$ . So,  $S_{tq}$  would be the state of the system for agent  $q$  at time  $t$ , which includes

$R_{tq}$  = The state of resources controlled by agent  $q$  at time  $t$ .

$I_{tq}$  = Any other information known to agent  $q$  at time  $t$ .

$B_{tq}$  = The beliefs of agent  $q$  about anything known to any other agent (and therefore not known to agent  $q$ ). This covers parameters in the ground truth,  
anything known by any other agent (e.g., the resources that an agent  $q'$  might be controlling),  
and finally, beliefs about how other agents make decisions.

Belief states are the richest and most challenging dimension of multi-agent systems, especially when there is more than one controlling agent, as would occur in competitive games.

Decisions for agent  $q$  are represented by  $x_{tq}$ . In addition to decisions that act on the environment, decisions in multi-agent systems can include both information collection and communication to other controlling and/or learning agents. Exogenous information arriving to agent  $q$  would be given by  $W_{tq}$ . The exogenous information may be observations of a ground truth, or decisions made by other agents. The transition function gives the equations for updating  $S_{tq}$  from decision  $x_{tq}$  and exogenous information  $W_{t+1,q}$ . The objective function captures the performance metrics for agent  $q$ .

The design of policies is drawn from the same four classes that we have described above.

One of the controlling agents may play the role of a central agent, but in this framework, a “central agent” is simply another agent who makes decisions that are communicated to “field agents” who then use these decisions in their planning.

There is a tendency in the literature on multi-agent systems to work with a “system state”  $S_t = (S_{tq})_{q \in Q}$ . We would take the position that this is meaningless since no agent ever sees all this information. We would approach the modeling of each agent

as its own system, with the understanding that a challenge of any intelligent agent is to develop models that help the agent to forecast the exogenous information process  $W_{tq}$ . Of course, this depends on the policy being used by the agent.

A careful treatment of the rich problem of multi-agent systems is beyond the scope of this chapter. However, we feel that the modeling of multi-agent systems using this approach, drawing on the four classes of policies, opens up new strategies for the modeling and control of distributed systems.

### 3.9 Observations

We are not the first to bridge optimal control with reinforcement learning. Recht [38] highlights recent successes of reinforcement learning in AlphaGo [15], and suggests that these methods should be adapted to control problems. We would argue that both fields have explored methods that could benefit the other, although we note that the controls community introduced the idea of direct lookaheads (model predictive control) in the 1950s (a form of direct lookahead), affine policies in the 1960s (a form of policy function approximation), and value function approximations in the 1970s. Both communities have addressed “model-free” and “model-based” settings, and both have explored methods from all four classes of policies (although neither has investigated parametric CFAs in depth). We think the biggest difference between optimal control and reinforcement learning is the core motivating applications of each field: optimal control grew originally out of continuous problems of controlling physical devices (aircraft, rockets, robots), while reinforcement learning grew out of problems with discrete action spaces.

We close with the following observations:

- (1) The fields of stochastic control and reinforcement learning address sequential decision problems. We feel that this perspective identifies a range of problems that is much wider than the problems that have been traditionally associated with these communities.
- (2) There seems to be considerable confusion about the meaning of “reinforcement learning.” In the 1990s and early 2000s, reinforcement learning was a method, called  $Q$ -learning. Today, it covers the entire range of methods described in Sect. 3.6. If we accept that the four classes of policies are universal, then it means that reinforcement learning covers *any* policy for a sequential decision problem (the same is true of stochastic control).
- (3) Parametric CFAs are often derided by the stochastic optimization community (“heuristics,” “deterministic approximations” are often heard), yet are widely used in practice. Properly designed parametric policies, however, can be surprisingly effective for two reasons: a) the parameterization can capture domain knowledge that is completely ignored with policies based on lookaheads (VFAs or stochastic DLAs) and b) tuning parameters in a realistic, stochastic base model which avoids the different approximations needed in stochastic lookaheads, can

- capture complex behaviors that would be overlooked using a simplified stochastic lookahead model.
- (4) Stochastic optimal control also addresses sequential decision problems, using a more flexible and scalable modeling framework. We have argued that the control community is also using instances of all four classes of policies. So, what is the difference between stochastic optimal control and reinforcement learning?
  - (5) Our universal framework, which draws heavily on the language used by the stochastic control community, broadens the scope of both of these fields to any sequential decision problem, which we would argue is broader than the problem classes considered by either community. Further, we have drawn on the four classes of policies identified in [31] which encompasses all the strategies already being explored by both communities. Since our classes are general (they are better described as meta-classes), they help guide the design of new strategies, including hybrids.

## References

1. Astrom, K.J.: Introduction to Stochastic Control Theory. Dover Publications, Mineola (1970)
2. Bellman, R.E.: Dynamic Programming. Princeton University Press, Princeton (1957)
3. Bellman, R.E., Glicksberg, I., Gross, O.: On the optimal inventory equation. *Manage. Sci.* **1**, 83–104 (1955)
4. Bertsekas, D.P., Shreve, S.E.: Stochastic Optimal Control: The Discrete Time Case. Academic, New York (1978)
5. Bertsekas, D.P., Tsitsiklis, J.N.: Neuro-Dynamic Programming. Athena Scientific, Belmont (1996)
6. Bertsekas, D.P., Tsitsiklis, J.N., Wu, C.: Rollout algorithms for combinatorial optimization. *J. Heuristics* **3**(3), 245–262 (1997)
7. Bouzaïene-Ayari, B., Cheng, C., Das, S., Fiorillo, R., Powell, W.B.: From single commodity to multiattribute models for locomotive optimization: a comparison of optimal integer programming and approximate dynamic programming. *Transp. Sci.* **50**(2), 1–24 (2016)
8. Browne, C.B., Powley, E.J., Whitehouse, D., Lucas, S.M., Cowling, P.I., Rohlfshagen, P., Tavener, S., Perez, D., Samothrakis, S., Colton, S.: A survey of Monte Carlo tree search methods. *IEEE Trans. Comput. Intell. AI Games* **4**(1), 1–49 (2012)
9. Bubeck, S., Cesa-Bianchi, N.: ‘Regret analysis of stochastic and nonstochastic multi-armed bandit problems’, *foundations and trends®. Mach. Learn.* **5**(1), 1–122 (2012)
10. Camacho, E., Bordons, C.: Model Predictive Control. Springer, London (2003)
11. Chang, H.S., Fu, M.C., Hu, J., Marcus, S.I.: An adaptive sampling algorithm for solving Markov decision processes. *Oper. Res.* **53**(1), 126–139 (2005)
12. Cinlar, E.: Probability and Stochastics. Springer, New York (2011)
13. Coulom, R.: Efficient selectivity and backup operators in Monte-Carlo tree search. *Computers and Games*, pp. 72–83. Springer, Berlin (2007)
14. DeGroot, M.H.: Optimal Statistical Decisions. Wiley, Hoboken (1970)
15. Fu, M.C.: Markov decision processes, AlphaGo, and Monte Carlo tree search: back to the future. *TutORials Oper. Res.*, pp. 68–88 (2017)
16. Gittins, J., Jones, D.: A dynamic allocation index for the sequential design of experiments. In: Gani, J. (ed.) *Progress in Statistics*, pp. 241–266. North Holland, Amsterdam (1974)
17. Gittins, J., Glazebrook, K.D., Weber, R.R.: Multi-Armed Bandit Allocation Indices. Wiley, New York (2011)

18. Rossiter, J.A.: *Model-Based Predictive Control*. CRC Press, Boca Raton (2004)
19. Jiang, D.R., Pham, T.V., Powell, W.B., Salas, D.F., Scott, W.R.: A comparison of approximate dynamic programming techniques on benchmark energy storage problems: does anything work? In: IEEE Symposium on Adaptive Dynamic Programming and Reinforcement Learning, pp. 1–8. IEEE, Orlando, FL (2014)
20. Kirk, D.E.: *Optimal Control Theory: An Introduction*. Dover, New York (2004)
21. Kothare, M.V., Balakrishnan, V., Morari, M.: Robust constrained model predictive control using linear matrix inequalities. *Automatica* **32**(10), 1361–1379 (1996)
22. Kushner, H.: *Introduction to Stochastic Control*. Holt, Rinehart and Winston, New York (1971)
23. Lazaric, A.: *Introduction to Reinforcement Learning* (2019). [https://www.irit.fr/cimi-machine-learning/sites/irit.fr.CMS-DRUPAL7.cimi\\_ml/files/pictures/lecture1\\_bandits.pdf](https://www.irit.fr/cimi-machine-learning/sites/irit.fr.CMS-DRUPAL7.cimi_ml/files/pictures/lecture1_bandits.pdf)
24. Lewis, F.L., Vrabie, D.: Reinforcement learning and adaptive dynamic programming for feedback control. *IEEE Circuits Syst. Mag.* **9**(3), 32–50 (2009)
25. Lewis, F.L., Vrabie, D., Syrmos, V.L.: *Optimal Control*, 3rd edn. Wiley, Hoboken (2012)
26. Maxwell, M.S., Henderson, S.G., Topaloglu, H.: Tuning approximate dynamic programming policies for ambulance redeployment via direct search. *Stoch. Syst.* **3**(2), 322–361 (2013)
27. Murray, J.J., Member, S., Cox, C.J., Lendaris, G.G., Fellow, L., Saeks, R.: Adaptive dynamic programming. *IEEE Trans. Syst., Man, Cybern.-Part C Appl. Rev.* **32**(2), 140–153 (2002)
28. Nisio, M.: *Stochastic Control Theory: Dynamic Programming Principle*. Springer, New York (2014)
29. Powell, W.B.: *Approximate Dynamic Programming: Solving the Curses of Dimensionality*, 2nd edn. Wiley, Hoboken (2011)
30. Powell, W.B.: Clearing the jungle of stochastic optimization. *Bridging Data and Decisions*, pp. 109–137 (2014) (January 2015)
31. Powell, W.B.: A unified framework for stochastic optimization. *Eur. J. Oper. Res.* **275**(3), 795–821 (2019)
32. Powell, W.B.: *Reinforcement Learning and Stochastic Optimization: A Unified Framework for Sequential Decisions*, Princeton
33. Powell, W.B., Frazier, P.I.: Optimal learning. *TutORials Oper. Res.*, pp. 213–246 (2008)
34. Powell, W.B., Meisel, S.: Tutorial on stochastic optimization in energy - part II: an energy storage illustration. *IEEE Trans. Power Syst.* (2016)
35. Powell, W.B., Ryzhov, I.O.: *Optimal Learning* (2012)
36. Puterman, M.L.: *Markov Decision Processes*, 2nd edn. Wiley, Hoboken (2005)
37. Rakovic, S.V., Kouvaritakis, B., Cannon, M., Panos, C., Findeisen, R.: Parameterized tube model predictive control. *IEEE Trans. Autom. Control* **57**(11), 2746–2761 (2012)
38. Recht, B.: A tour of reinforcement learning: the view from continuous control. *Annu. Rev. Control., Robot., Auton. Syst.* **2**(1), 253–279 (2019)
39. Sethi, S.P.: *Optimal Control Theory: Applications to Management Science and Economics*, 3rd edn. Springer, Boston (2019)
40. Si, J., Barto, A.G., Powell, W.B., Wunsch, D.: *Handbook of Learning and Approximate Dynamic Programming*. Wiley-IEEE Press (2004)
41. Simão, H., Day, J., George, A.P., Gifford, T., Nienow, J., Powell, W.B.: An approximate dynamic programming algorithm for large-scale fleet management: a case application. *Transp. Sci.* (2009)
42. Sontag, E.: *Mathematical Control Theory*, 2nd edn., pp. 1–544. Springer, Berlin (1998)
43. Spall, J.C.: *Introduction to Stochastic Search and Optimization: Estimation, Simulation and Control*. Wiley, Hoboken (2003)
44. Stengel, R.F.: *Stochastic Optimal Control: Theory and Application*. Wiley, Hoboken (1986)
45. Stengel, R.F.: *Optimal Control and Estimation*. Dover Publications, New York (1994)
46. Sutton, R.S., Barto, A.G.: *Reinforcement Learning: An Introduction*. MIT, Cambridge (1998)
47. Sutton, R.S., Barto, A.G.: *Reinforcement Learning: An Introduction*, 2nd edn. MIT Press, Cambridge (2018)
48. Vrabie, D., Lewis, F.: Neural network approach to continuous-time direct adaptive optimal control for partially unknown nonlinear systems. *Neural Netw.* **22**(3), 237–246 (2009)

49. Werbos, P.J.: Beyond regression: new tools for prediction and analysis in the behavioral sciences. PhD thesis, Harvard University (1974)
50. Yong, J., Zhou, X.Y.: Stochastic Controls: Hamiltonian Systems and HJB Equations. Springer, New York (1999)

# Chapter 4

## Fundamental Design Principles for Reinforcement Learning Algorithms



Adithya M. Devraj, Ana Bušić, and Sean Meyn

**Abstract** Along with the sharp increase in visibility of the field, the rate at which new reinforcement learning algorithms are being proposed is at a new peak. While the surge in activity is creating excitement and opportunities, there is a gap in understanding of two basic principles that these algorithms need to satisfy for any successful application. One has to do with guarantees for convergence, and the other concerns the convergence rate. The vast majority of reinforcement learning algorithms belong to a class of learning algorithms known as stochastic approximation (SA). The objective here is to review the foundations of reinforcement learning algorithm design based on recent and ancient results from SA. In particular, it was established in [Borkar and Meyn, 2000] that both stability and convergence of these algorithms are guaranteed by analyzing the stability of two associated ODEs. Moreover, if the linearized ODE passes a simple eigenvalue test, then an optimal rate of convergence is guaranteed. This chapter contains a survey of these concepts, along with a survey of the new class of Zap reinforcement learning algorithms introduced by the authors. These algorithms can achieve convergence almost universally, while also guaranteeing optimal rate of convergence.

---

A. M. Devraj · S. Meyn (✉)

Department of Electrical and Computer Engineering, University of Florida,

Gainesville, FL 32611, USA

e-mail: [meyn@ece.ufl.edu](mailto:meyn@ece.ufl.edu)

A. M. Devraj

e-mail: [adithyamdevraj@ufl.edu](mailto:adithyamdevraj@ufl.edu)

A. Bušić

Inria and the Computer Science Department of École Normale Supérieure,

75005 Paris, France

e-mail: [ana.busic@inria.fr](mailto:ana.busic@inria.fr)

## 4.1 Introduction

Reinforcement learning (RL) is an area of machine learning and stochastic optimal control that has received significant attention in the recent past. The credit for this, in part, is due to the emergence of deep learning, with applications to RL in particular [38, 48, 49]. The general objective of RL is simple: learn to make “good” decisions in an uncertain environment through experience.

Most RL algorithms can be considered to be parameter estimation techniques, where the goal is to recursively estimate the parameter vector  $\theta^* \in \mathbb{R}^d$  that directly, or indirectly yields an optimal decision-making rule within a parameterized family. The update equation for the  $d$ -dimensional parameter estimates  $\{\theta_n : n \geq 0\}$  can be expressed in the general form:

$$\theta_{n+1} = \theta_n + \alpha_{n+1}[\bar{f}(\theta_n) + \Delta_{n+1}], \quad n \geq 0, \quad (4.1)$$

in which  $\theta_0 \in \mathbb{R}^d$  is given,  $\{\alpha_n\}$  is a positive scalar *gain sequence* (also known as *learning rate* in the RL literature),  $\bar{f} : \mathbb{R}^d \rightarrow \mathbb{R}^d$  is a deterministic function, and  $\{\Delta_n\}$  is a “noise” sequence.

The recursion (4.1) is an example of stochastic approximation (SA), for which there is a vast research literature. Under standard assumptions, it can be shown that

$$\lim_{n \rightarrow \infty} \theta_n = \theta^*,$$

where  $\bar{f}(\theta^*) = 0$ . Theory for convergence of the SA recursion is surveyed in this chapter, and it is argued that the best algorithms achieve the optimal mean square error (MSE) convergence rate:

$$\mathbb{E}[\|\theta_n - \theta^*\|^2] = O(1/n), \quad (4.2)$$

It is known that RL algorithms such as TD- and Q-learning can be written in the form (4.1). In these algorithms,  $\{\theta_n\}$  represents the sequence of parameter estimates that are used to approximate a *value function* or *Q-function*. It is also widely recognized that these algorithms can be slow to converge.

It was first established in our work [21, 22] that the convergence rate of the MSE  $\mathbb{E}[\|\theta_n - \theta^*\|^2]$  of Watkins’ Q-learning can be slower than  $O(1/n^{2(1-\beta)})$ , if the discount factor  $\beta \in (0, 1)$  satisfies  $\beta > \frac{1}{2}$ . It was also shown that the optimal convergence rate (4.2) is obtained by using a step-size of the form  $\alpha_n = g/n$ , for  $g > 0$  sufficiently large. In earlier work, [64] obtained a sample path *upper bound* on the rate of convergence that is roughly consistent with the mean square rate established for  $\beta > \frac{1}{2}$  in [21, 22].

While only an upper bound, [64] provided early warning of the poor convergence rate for the standard Q-learning algorithm.

Since the publication of [64], many papers have appeared with proposed improvements to the algorithm. It is argued in [26] that a “polynomial” learning rate of the

form  $\alpha_n = 1/n^\rho$ , with  $\rho \in (0.5, 1)$ , results in a faster convergence rate than using a “linear” learning rate of the form  $\alpha_n = 1/n$ . Theory surveyed in this chapter shows that a polynomial learning rate *cannot* lead to the optimal rate (4.2).

In [28], the authors suggest that the main reason for slow convergence of Q-learning is due to “overestimating” the Q-values, and introduce a *double Q-learning* algorithm to resolve this. The “speedy” Q-learning algorithm was introduced in [2], along with finite-time error bounds that seem to imply fast convergence. We do not know if the algorithms in [28] or [2] can achieve the optimal rate of convergence (4.2).

The point of view of this chapter is based on the SA representation (4.1) for RL algorithms. Both ancient and recent SA theories are surveyed and applied to analyze RL algorithms and optimize their rate of convergence.

### 4.1.1 Stochastic Approximation and Reinforcement Learning

Asymptotic statistical theory for SA is extremely rich. Large deviations or central limit theorem (CLT) limits hold under very general assumptions for both SA and related Monte Carlo techniques [3, 8, 33, 35, 47].

The CLT will be a guide to algorithm design in this chapter. For a typical SA algorithm, this takes the following form: denote the *error sequence* by

$$\tilde{\theta}_n := \theta_n - \theta^*. \quad (4.3)$$

Under general conditions, the scaled sequence  $\{\sqrt{n}\tilde{\theta}_n : n \geq 0\}$  converges in distribution to a Gaussian  $\mathcal{N}(0, \Sigma_\theta)$ . Typically, the covariance of this scaled sequence is also convergent:

$$\Sigma_\theta = \lim_{n \rightarrow \infty} n \mathbb{E}[\tilde{\theta}_n \tilde{\theta}_n^\top]. \quad (4.4)$$

The limit is known as *asymptotic covariance*. Provided  $\Sigma_\theta$  is finite, this implies (4.2), which is the fastest possible rate [3, 8, 35, 55, 57].

An asymptotic bound such as (4.4) may not be satisfying for practitioners of RL or optimization, given the success of finite-time performance bounds in prior research. There are, however, good reasons to apply this asymptotic theory in algorithm design:

- (i) The asymptotic covariance  $\Sigma_\theta$  has a simple representation as the solution to a Lyapunov equation. We will see that it can be improved or optimized by algorithm design.
- (ii) As shown in examples in this chapter, the asymptotic covariance is often a good predictor of finite-time performance, since the CLT approximation is accurate for reasonable values of  $n$  (see Sect. 4.4, and in particular Fig. 4.7).
- (iii) The MSE convergence is refined in [12]: For some  $\delta > 0$ ,

$$\mathbb{E}[\tilde{\theta}_n \tilde{\theta}_n^\top] = n^{-1} \Sigma_\theta + O(n^{-1-\delta}).$$

- (iv) The asymptotic covariance lies beneath the surface in the theory of finite-time error bounds. Here is what can be expected from the theory of large deviations [17, 34]: on denoting the *rate function* by

$$I_i(\varepsilon) := -\lim_{n \rightarrow \infty} \frac{1}{n} \log \mathsf{P}\{|\theta_n(i) - \theta^*(i)| > \varepsilon\}, \quad (4.5)$$

the second-order Taylor series approximation holds,

$$I_i(\varepsilon) = \frac{1}{2\Sigma_\theta(i, i)} \varepsilon^2 + O(\varepsilon^3). \quad (4.6)$$

Equally important is that the asymptotic theory surveyed in this chapter provides insight into the slow convergence of well-known algorithms, and this insight will lead to rules for algorithm design.

Two approaches were previously known for optimizing the asymptotic covariance. One is the remarkable averaging technique of Ruppert [58] and Polyak and Juditsky [53, 54] ([32] provides an accessible treatment in a simplified setting). A lesser known approach is the Newton–Raphson variant of the SA algorithm (SqNR) of Ruppert [57]. The more recent Zap SA algorithm can be regarded as a significant extension of Ruppert’s SqNR [20–22] (see also the dissertation [18]). The Zap–SA approach is the basis for Zap Q-learning [21].

We are not aware of *theory* that distinguishes the performance of Polyak–Ruppert averaging as compared to the Newton–Raphson-based methods. However, it is noted in [50], that the averaging approach often leads to very large transients, so that the algorithm should be modified (such as through projection of parameter updates). This may explain why averaging is not very popular in practice. In our own numerical experiments, it is observed that the rate of convergence of the CLT is slow when compared to the Zap algorithms. Most important: *the Zap–SA approach leads to a stable algorithm under conditions far more general than other techniques*. This is discussed in Sects. 4.2.4 and 4.5, based on the recent paper [11].

### 4.1.2 Sample Complexity Bounds

An emphasis in much of the RL literature is computation of a finite-time error bound of the form

$$\mathsf{P}\{\|\theta_n - \theta^*\| > \varepsilon\} \leq \bar{b} \exp(-n\bar{I}(\varepsilon)), \quad (4.7)$$

where  $\bar{b}$  is a constant and  $\bar{I}(\varepsilon) > 0$  for  $\varepsilon > 0$ . The bound is usually inverted: for a given  $\delta > 0$ , denote

$$\bar{n}(\varepsilon, \delta) = \frac{1}{\bar{I}(\varepsilon)} [\log(\bar{b}) + \log(\delta^{-1})]. \quad (4.8)$$

Then, (4.7) implies the *sample complexity bound*:  $\mathsf{P}\{\|\theta_n - \theta^*\| > \varepsilon\} \leq \delta$  for all  $n \geq \bar{n}(\varepsilon, \delta)$ . Explicit bounds were obtained in [26, 72] for Watkins' algorithm, and in [2] for the “speedy” Q-learning algorithm. General theory for SA algorithms is presented in [12, 50, 60].

The value of (4.8) depends on the size of the constants. Ideally, the function  $\bar{I}$  is quadratic as a function of  $\varepsilon$ . Theorem 6 of [26] asserts that this ideal is not in general possible for Q-learning: an example is given for which the best bound requires  $\bar{I}(\varepsilon) = O(\varepsilon^{1/(1-\beta)})$  when the discount factor satisfies  $\beta > \frac{1}{2}$ .

We conjecture that the sample-path complexity bound (4.8) is possible with  $\bar{I}$  quadratic, provided a sufficiently large scalar gain is introduced on the right-hand side of the update equation (4.1). This conjecture is rooted in large deviations theory, for which a quadratic approximation typically holds. In view of the limit (4.5), and its approximation (4.6), we can expect (4.7) with quadratic  $\bar{I}$  only when the asymptotic covariance  $\Sigma_\theta$  is finite. In the very recent preprint [72], the finite- $n$  bound (4.8) with  $\bar{I}$  quadratic was obtained for Watkins' algorithm in a special synchronous setting, subject to a specific scaling of the step-size:

$$\alpha_n = \frac{1}{1 + (1 - \beta)n}. \quad (4.9)$$

This result is consistent with our conjecture: it was shown in [20] that the asymptotic covariance is finite for the equivalent step-size sequence,  $\alpha_n = [(1 - \beta)n]^{-1}$  (see Theorem 4.7 for details).

In the models considered in prior works [2, 26, 50, 64, 72], the update equation for the  $d$ -dimensional parameter estimates  $\{\theta_n\}$  can be expressed in the form (4.1) in which the “noise”  $\{\Delta_n\}$  is a martingale difference sequence. This representation is critical in analysis, but unfortunately is not typical in RL applications outside of a few versions of TD- and Q-learning. For Markovian models, the usual transformation used to obtain a representation similar to (4.1) results in a noise sequence  $\{\Delta_n\}$  that is the sum of a martingale difference sequence and a *telescoping sequence* [12, 40, 42]. It is the telescoping sequence that prevents easy analysis of Markovian models.

This gap in the research literature carries over to the general theory of Markov chains. Examples of concentration bounds for noise being independent and identically distributed (i.i.d.) sequences or martingale-difference sequences include the finite-time bounds of Hoeffding and Bennett. Extensions to Markovian models either offer very crude bounds [46], or restrictive assumptions [27, 39]; this remains an active area of research [5, 12, 52, 60].

### 4.1.3 What Will You Find in This Chapter?

**SA theory:** Section 4.2 contains a summary of SA theory, with a focus on algorithm stability, and optimizing the convergence rate. The Zap-SA algorithm of [21] is

defined here: a *two-time-scale* SA algorithm, constructed so that the matrix gain tracks the gain that would appear in a continuous-time Newton–Raphson algorithm.

The contents of Sect. 4.2 is mainly a review of well-known theory. It also provides new interpretations, along with implications to RL algorithm design:

- While TD-learning algorithms are convergent under mild conditions, the optimal rate of convergence (4.2) fails in most cases. This is particularly true when the discount factor is large. Explanations rooted in SA theory are contained in Sect. 4.2.7.
- In [33, Chap. 6], it is established that the LSTD algorithm achieves optimal asymptotic covariance. The close relationship between Newton–Raphson techniques and LSTD algorithms is explained in Sect. 4.2, which is largely unrecognized in the RL literature.

**RL theory:** Theory supporting the Zap–SA algorithm leads naturally to the *Zap Q-learning* algorithm that is a focus of Sect. 4.3. A full convergence analysis is presented for the special case of a “tabular representation” of the Q-function (similar to the setting of Watkins’ original algorithm [74, 75]). Extensions to non-ideal parameterized settings, such as neural networks, are discussed in Sect. 4.5.

**What’s left out?** The topic of *exploration* is the most notable gap in this chapter. Excellent recent references include [37, 51, 59].

Zap Q-learning shows promise of obtaining both convergence and optimal rate of convergence under very general assumptions: two basic principles that any RL algorithm has to satisfy for successful applications, and the theme of this chapter. A third basic principle, *notably left out of this chapter*, is total computation time to obtain a good estimate of  $\theta^*$ . An important topic for future research concerns implementations of Zap Q-learning that preserve the first two properties while reducing complexity of the basic algorithm. Further discussion is contained in Sect. 4.6.

#### 4.1.4 Literature Survey

What follows is a brief survey of the vast literature on SA and RL algorithms.

**History of SA:** The stochastic approximation algorithm was introduced by Robbins and Monro for scalar functions [55]. Blum in [6] extended the theory to vector-valued recursions. Soon after the introduction of the algorithms, Chung [14] identified the optimal asymptotic variance, and techniques to obtain the optimum for scalar recursions. This can be cast as a form of *stochastic Newton–Raphson* (SNR) [20–22], that is discussed in Sect. 4.2.6 of this chapter. Gradient-free methods [or *stochastic quasi-Newton–Raphson* (SqNR)] appeared in later work: The first such algorithm was proposed by Venter in [71], which was shown to obtain the optimal variance for a one-dimensional SA recursion. The algorithm obtains estimates of the SNR gain  $-A^{-1}$  (see Sect. 4.2.6), through a procedure similar to the Kiefer–Wolfowitz algorithm [29]. Ruppert proposed an extension of Venter’s algorithm for vector-valued functions in [57]. The averaging technique of Ruppert and Polyak [53, 54, 58] is a two-time-scale

algorithm that is also designed to achieve the optimal asymptotic covariance, without a matrix gain.

**History of RL:** Sutton’s TD-learning algorithm appears to be the first RL algorithm [62]. Soon after, the Q-learning algorithm was introduced by Watkins and Dayan [74, 75]. A thorough theoretical analysis of TD-learning algorithms came much later during the late 90s [68, 69], and for the actor–critic algorithms in the early mid-2000s [30–32, 63], both for the case of linear function approximation. More recently, the emergence of deep learning, and applications to RL in particular, has led to series of works that demonstrate the ability of RL algorithms to effectively solve complicated tasks [38, 48, 49]. Recent theoretical work in the setting of a non-linear function approximation architecture (such as neural networks) is contained in [11, 15].

**Finite-Time Bounds in SA and RL:** Finite-time performance of single-time-scale SA algorithms with application to TD-learning was studied in [36, 66], and bounds for two-time-scale SA algorithms were obtained in [16]. However, these works rest on a critical assumption: that the noise is a martingale difference sequence.

The literature on finite-time error bounds for SA recursions with Markovian noise has been recent. Bounds are obtained in [5] for both vanishing step-sizes, and for (carefully selected) constant step-size TD-learning, with projection. Finite-time bounds for SA with constant step-size are obtained in [60] by considering the drift of an appropriately chosen Lyapunov function. In the recent work [12] (briefly surveyed in Sect. 4.2.5), the limit (4.4) is refined to obtain something approaching a finite-time error bound, which provides further motivation for optimizing the asymptotic covariance  $\Sigma_\theta$ . All of this theory is encouraging: these approximations justify the use of asymptotic bounds for algorithm design.

**Organization:** The remainder of the chapter is organized as follows. Section 4.2 contains a background on general SA theory, with a focus on optimal convergence rates of the mean square error. Based on this theory, new Zap-SA algorithms are proposed, with the goal of optimizing the convergence rate, *and* obtaining stable algorithms. Extension of the analysis and algorithmic techniques to Q-learning and Zap Q-learning is contained in Sect. 4.3. The theory is illustrated with results from numerical experiments contained in Sects. 4.4, 4.5, and 4.6 contain discussion on extensions and directions for future research.

## 4.2 Stochastic Approximation: New and Old Tricks

Stochastic approximation (SA) is the foundation of many well-known RL algorithms, including Q-learning, TD-learning, and Actor–Critic algorithms. This section contains a survey of SA techniques, which forms the foundation for the remainder of this chapter.

The stability theory contained in Sect. 4.2.3 is taken from [9], which is expanded upon in Borkar’s monograph [8]. The approach to variance analysis in Sect. 4.2.5 is

based on [8, 32], and also our own interpretation [20, 21]. The *Zap-SA* algorithm introduced in Sect. 4.2.4 was first formalized in our works [20–22]. This algorithm and the analysis in this section lay the foundation for the Zap Q-learning algorithm.

### 4.2.1 What is Stochastic Approximation?

Consider a parameterized family of  $\mathbb{R}^d$ -valued functions  $\{\bar{f}(\theta) : \theta \in \mathbb{R}^d\}$  that can be expressed as an expectation,

$$\bar{f}(\theta) := \mathbb{E}[f(\theta, \Phi)], \quad \theta \in \mathbb{R}^d, \quad (4.10)$$

with  $\Phi \in \mathbb{R}^m$  a random vector,  $f : \mathbb{R}^d \times \mathbb{R}^m \rightarrow \mathbb{R}^d$ , and the expectation is with respect to the distribution of the random vector  $\Phi$ . It is assumed throughout this section that there exists a unique vector  $\theta^* \in \mathbb{R}^d$  satisfying  $\bar{f}(\theta^*) = 0$ . Under this assumption, the goal of SA is to estimate  $\theta^*$ .

If the distribution of  $\Phi$  is known, then estimates of  $\theta^*$  can be computed using successive approximation:

$$\theta_{n+1} = \theta_n + \alpha \bar{f}(\theta_n) \quad (4.11)$$

where  $\alpha$  is a non-negative scalar. Under stronger assumptions, the Newton–Raphson algorithm can be applied to obtain much faster convergence:

$$\theta_{n+1} = \theta_n - [A(\theta_n)]^{-1} \bar{f}(\theta_n), \quad A(\theta_n) = [\partial_\theta \bar{f}(\theta)]_{\theta=\theta_n}, \quad (4.12)$$

where the notation  $\partial_\theta$  denotes the partial derivative with respect to  $\theta$ . Convergence of either of these recursions, from each initial condition  $\theta_0$ , requires further conditions on  $\bar{f}$ .

However, in RL, and many other SA applications, the distribution of  $\Phi$  is unknown, and even if it is known, computing the expectation that defines  $\bar{f}$  is computationally infeasible. The SA algorithm was created by Robbins and Monro to break this computational curse [55]. The basic algorithm is in fact an approximation of successive approximation, in which the expectation in (4.11) is removed:

#### Stochastic Approximation

For initialization  $\theta_0 \in \mathbb{R}^d$ , obtain the sequence of estimates  $\{\theta_n : n \geq 0\}$  recursively:

$$\theta_{n+1} = \theta_n + \alpha_{n+1} f(\theta_n, \Phi_{n+1}), \quad (4.13)$$

where  $\Phi_n$  has the same distribution as  $\Phi$  for each  $n \geq 0$  (or its distribution converges to that of  $\Phi$  as  $n \rightarrow \infty$ ), and  $\{\alpha_n\}$  is a non-negative scalar step-size sequence.

Further assumptions on the step-size sequence are summarized in Sect. 4.2.3; we simply note here that the following is essential for convergence:

$$\lim_{n \rightarrow \infty} \alpha_n = 0, \quad \sum_{n=1}^{\infty} \alpha_n = \infty. \quad (4.14)$$

Analysis of the SA recursion begins with the suggestive representation

$$\theta_{n+1} = \theta_n + \alpha_{n+1}[\bar{f}(\theta_n) + \Delta_{n+1}], \quad (4.15)$$

in which  $\Delta_{n+1} = f(\theta_n, \Phi_{n+1}) - \bar{f}(\theta_n)$ . The difference  $f(\theta, \Phi_{n+1}) - \bar{f}(\theta)$  has zero mean for any (deterministic)  $\theta \in \mathbb{R}^d$  when  $\Phi_{n+1}$  is distributed according to  $\Phi$  (recall (4.10)). In much of the chapter it is assumed that  $\Phi := \{\Phi_n : n \geq 0\}$  is a geometrically ergodic Markov chain. This together with bounds on the function  $f$  imply that the “noise” sequence  $\Delta := \{\Delta_n : n \geq 0\}$  can be approximated by a *martingale difference sequence*. No approximation is required if the sequence  $\Phi$  is i.i.d. (independent and identically distributed):

$$\mathbb{E}[\Delta_{n+1} \mid \Delta_i : i \leq n] = 0, \quad n \geq 0.$$

Stability and convergence theory based on (4.15) is summarized in Sect. 4.2.3.

Our primary interest regards the rate of convergence of the error sequence  $\tilde{\theta}_n = \theta_n - \theta^*$ , measured by the error covariance

$$\Sigma_n = \mathbb{E}[\tilde{\theta}_n \tilde{\theta}_n^T] \quad (4.16)$$

and  $\sigma_n^2 = \text{trace}(\Sigma_n) = \mathbb{E}[\|\tilde{\theta}_n\|^2]$ . We say that  $\sigma_n^2$  tends to zero at rate  $1/n^\mu$  (with  $\mu > 0$ ) if for each  $\varepsilon > 0$ ,

$$\lim_{n \rightarrow \infty} n^{\mu-\varepsilon} \sigma_n^2 = 0 \quad \text{and} \quad \lim_{n \rightarrow \infty} n^{\mu+\varepsilon} \sigma_n^2 = \infty. \quad (4.17)$$

It is known that the maximal value is  $\mu = 1$ , and we will show that when this optimal rate is achieved, there is an associated limiting covariance

$$\Sigma_\theta = \lim_{n \rightarrow \infty} n \Sigma_n. \quad (4.18)$$

The matrix  $\Sigma_\theta$  is known as the *asymptotic covariance*, and under the conditions imposed here, to ensure that the limit (4.18) exists, we can also expect a CLT:

$$\sqrt{n} \tilde{\theta}_n \xrightarrow{d} W, \quad (4.19)$$

where the convergence is in distribution and  $W$  is Gaussian  $\mathcal{N}(0, \Sigma_\theta)$  [3, 35].

It is possible to optimize the asymptotic covariance over all algorithms in a prescribed class. Three “ancient” approaches are, in chronological order

- (i) SNR algorithm of Chung [14].
- (ii) Newton–Raphson variant of the SA algorithm (SqNR) of Ruppert [57].
- (iii) The averaging technique of Ruppert [58], and Polyak and Juditsky [53, 54].

Two recent techniques to optimize the asymptotic covariance are

- (iv) Zap–SA [20–22].
- (v) Matrix momentum algorithms [19].

Of all five approaches, the Zap–SA algorithm stands out because it is stable under minimal assumptions on the model. Full details on methods (i)–(iii) are contained in Sect. 4.2.5, Zap–SA algorithms are defined in Sect. 4.2.4, analyzed in Sect. 4.2.5, and applied to RL in Sect. 4.3. Momentum techniques are beyond the scope of this chapter.

## 4.2.2 Stochastic Approximation and Learning

Motivation for the SA method is provided through two canonical examples: Monte Carlo for estimation of moments, and the TD-learning algorithms.

### 4.2.2.1 Monte Carlo

The goal of this discussion is to introduce the subtleties regarding the convergence rate in the simplest possible setting.

Suppose we wish to estimate the mean  $\theta^* = \mathbb{E}[c(\Phi)]$ , where  $c: \mathbb{R}^m \rightarrow \mathbb{R}^d$  and the random variable  $\Phi$  has density  $\varrho$ . That is,

$$\theta^* = \int c(x) \varrho(x) dx.$$

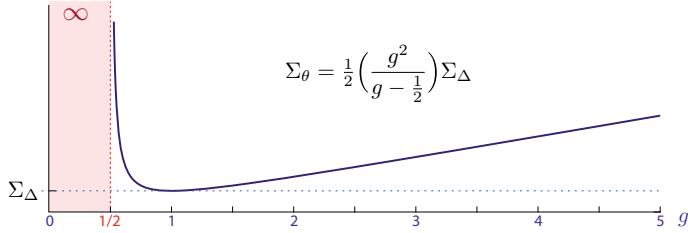
MCMC methods have been developed to construct a Markov chain  $\Phi$  whose steady-state distribution is equal to  $\varrho$ . Computation of the mean is an SA problem: find  $\theta^*$  solving

$$0 = \bar{f}(\theta^*) = \mathbb{E}[f(\theta^*, \Phi)] = \mathbb{E}[c(\Phi) - \theta^*].$$

Consider the SA recursion (4.13) in which  $\alpha_n = g/(n + 1)$ , with  $g > 0$ :

$$\theta_{n+1} = \theta_n + \frac{g}{n + 1} [c(\Phi_{n+1}) - \theta_n], \quad n \geq 0. \quad (4.20)$$

The special case  $g = 1$  reduces to the standard estimate,



**Fig. 4.1** Asymptotic covariance for MCMC estimates (4.20) for the scalar recursion

$$\theta_n = n^{-1} S_n, \quad \text{where, } S_n = \sum_{k=1}^n c(\Phi_k).$$

Under mild conditions on the Markov chain, the recursion (4.20) converges to  $\theta^*$ . However, the rate of convergence depends on the value of  $g$ .

Recall the definition of  $\Sigma_n$  in (4.16). In the scalar recursion, we have

$$\sigma_n^2 = \text{trace}(\Sigma_n) = \Sigma_n = E[\tilde{\theta}_n^2].$$

In this case, *the mean square error is guaranteed to converge to zero at rate  $1/n$  only if  $g > \frac{1}{2}$* . A plot of  $\Sigma_\theta$  as a function of  $g$  is displayed in Fig. 4.1. The definition of  $\Sigma_\Delta$  and the derivation of the variance formula is postponed to Sect. 4.2.5.

The M/M/1 queue in discrete time is perhaps the simplest Markov chain on a state space that is not finite. Its evolution can be expressed as the reflected random walk on the set of non-negative integers:

$$\Phi_{n+1} = \max(0, \Phi_n + D_{n+1}),$$

in which the sequence  $\{D_n\}$  is i.i.d., taking values  $\pm 1$ . On denoting  $\mu = P\{D_n = 1\}$ ,  $\alpha = P\{D_n = -1\}$ , it is assumed that  $\alpha + \mu = 1$ . The standard assumption  $\alpha < \mu$  is usually expressed as  $\rho = \alpha/\mu < 1$ . In this case, the Markov chain is reversible and geometrically ergodic (as defined in Eq. (4.49) below). With  $c(\Phi_n) = \Phi_n$ , the value  $\theta^*$  is the steady-state mean

$$\theta^* = \rho/(1 - \rho).$$

Statistics of the error sequence  $\tilde{\theta}_n$  using  $g = 1$  can be found in [44, Sect. 11.3]: the CLT holds, with  $\Sigma_\theta = \Sigma_\Delta = O((1 - \rho)^{-4})$ , and the asymptotic covariance for  $g > \frac{1}{2}$  is given by the formula in Fig. 4.1.

Given that this example has such desirable properties, many would expect the validity of the finite- $n$  bound (4.7). It was first demonstrated in [43, Prop. 1.1] that a bound of this form *cannot* be obtained for this model. There exist only (highly asymmetric) one-sided bounds: for functions  $I_-$ ,  $I_+ : \mathbb{R}_+ \rightarrow \mathbb{R}_+$ , and all  $\varepsilon > 0$ ,

- (i)  $\lim_{n \rightarrow \infty} \frac{1}{n} \log \mathsf{P}\{\tilde{\theta}_n \leq -\varepsilon\} = -I_-(\varepsilon) < 0$
- (ii)  $\lim_{n \rightarrow \infty} \frac{1}{n} \log \mathsf{P}\{\tilde{\theta}_n \geq n\varepsilon\} = -I_+(\varepsilon) < 0.$

The first limit is what is expected, based on the discussion surrounding (4.5). The probability in the second limit is not a typo: it is the probability that the error exceeds  $n$  times  $\varepsilon$ ! Much more on this topic can be found in [24, 25, 44].

The performance criterion (4.7) is focused exclusively on rare events. That is, we know that  $\mathsf{P}\{\|\tilde{\theta}_n\| \geq \varepsilon\}$  will go to zero very quickly. The asymptotic covariance is more grounded in typical performance, which helps to explain why it is finite and the CLT holds under very general conditions [47].

#### 4.2.2.2 Temporal Difference Learning

Our interest lies in the behavior of a Markov chain  $X$ , that is interpreted as the state process for a system to be controlled; say, position and velocity of a moving vehicle. If the state space  $X$  is finite, then the distribution of the stochastic process  $X$  is determined by its initial distribution, and the transition matrix  $P$ :

$$P(x, y) := \mathsf{P}\{X_{n+1} = y \mid X_n = x\}, \quad x, y \in X.$$

Given a cost function  $c : X \rightarrow \mathbb{R}$ , and a discount factor  $\beta \in (0, 1)$ , the value function is denoted

$$h(x) = \sum_{n=0}^{\infty} \beta^n \mathsf{E}[c(X_n) \mid X_0 = x], \quad x \in X. \quad (4.21)$$

The goal in TD-learning is to approximate  $h$  within a parameterized function class. The goal of SARSA algorithms is the same, but with  $X$  replaced by the joint state-input process  $(X, U)$ , and the cost is a function of these two variables.

The following Bellman equation is the starting point in the construction of TD-learning algorithms:

$$h(x) = c(x) + \beta \mathsf{E}[h(X_{n+1}) \mid X_n = x]. \quad (4.22)$$

The functional equation (4.22) can be recast:

$$\mathsf{E}[\mathcal{D}(h, \Phi_{n+1}) \mid \Phi_1 \dots \Phi_n] = 0, \quad n \geq 1, \quad (4.23)$$

where  $\Phi_n \equiv (X_n, X_{n-1})$  and

$$\mathcal{D}(h, \Phi_{n+1}) = c(X_n) + \beta h(X_{n+1}) - h(X_n). \quad (4.24)$$

Consider a parameterized family of candidate approximations  $\{h^\theta : \theta \in \mathbb{R}^d\}$ , where  $h^\theta : X \rightarrow \mathbb{R}$  for each  $\theta$ , and  $h^\theta(x)$  is a continuous function of  $\theta \in \mathbb{R}^d$  for

each  $x$ . In a neural network function approximation architecture, the vector  $\theta$  represents the “weight matrix”.

A common formulation of TD-learning is to solve the *Galerkin relaxation* of (4.23), (4.24): find  $\theta^* \in \mathbb{R}^d$  such that

$$\mathbb{E}[\mathcal{D}(h^{\theta^*}, \Phi_{n+1})\zeta_n] = 0, \quad (4.25)$$

where the  $d$ -dimensional random vector  $\zeta_n$  is called the *eligibility vector*, and the expectation is with respect to the steady-state distribution of the underlying Markov chain. In order that (4.25) be a true relaxation of (4.23), (4.24), it is essential that the sequence  $\{\zeta_n\}$  be *adapted to  $\Phi$* ; that is, for a sequence of functions  $\{z_n\}$ ,

$$\zeta_n = z_n(\Phi_1, \dots, \Phi_n).$$

In the TD(0)-learning algorithm, the eligibility vector is defined to be the gradient of the current value function estimate:

$$\zeta_n = \nabla_\theta h^\theta(\Phi_n)|_{\theta=\theta_n}. \quad (4.26)$$

In the neural network setting, the gradient is computed using backpropagation.

There are many other choices for the sequence of eligibility vectors [61]. Once this is specified, the basic TD-learning algorithm is the SA recursion (4.13) applied to solve (4.25):

### TD-learning

For initialization  $\theta_0 \in \mathbb{R}^d$ , and given a sequence of eligibility vectors  $\{\zeta_n\}$ , obtain the sequence of estimates  $\{\theta_n : n \geq 0\}$  recursively:

$$\begin{aligned} \theta_{n+1} &= \theta_n + \alpha_{n+1} d_{n+1} \zeta_n \\ d_{n+1} &= c(X_n) + \beta h^{\theta_n}(X_{n+1}) - h^{\theta_n}(X_n), \end{aligned} \quad (4.27)$$

where  $\{\alpha_n : n \geq 0\}$  denotes the non-negative step-size sequence.

Consider the special case of *linear function approximation* in which  $h^\theta$  is a linear function of  $\theta$ . Hence,  $h^\theta(x) = \theta^\top \psi(x)$  for each  $x$  and  $\theta$ , where  $\psi : \mathcal{X} \rightarrow \mathbb{R}^d$  is called the vector of *basis functions*. The expression (4.26) reduces to  $\zeta_n = \psi(X_n)$  in this special case, so that in particular the eligibility vector does not depend on the parameter estimates. Denoting

$$\begin{aligned} A_{n+1} &:= \psi(X_n)(\beta \psi(X_{n+1}) - \psi(X_n))^\top \\ b_{n+1} &:= -c(X_n)\psi(X_n) \end{aligned} \quad (4.28)$$

the algorithm (4.27) can be rewritten as the linear stochastic approximation algorithm:

$$\theta_{n+1} = \theta_n + \alpha_{n+1} [A_{n+1}\theta_n - b_{n+1}]. \quad (4.29)$$

Under general assumptions on the Markov chain  $X$ , and the basis functions  $\psi$ , it is known that matrix  $A = [\partial_\theta \mathcal{D}(h^\theta, \Phi_{n+1})]_{\theta=\theta^*} = \mathbb{E}[A_{n+1}]$  is Hurwitz (the real part of each eigenvalue is negative), and that the sequence of estimates converges to  $\theta^*$  [33, 69]. These conclusions follow from the stability theory to be developed in the remainder of this section.

The sub-optimal rate of convergence of Monte Carlo carries over to TD-learning algorithms (see Proposition 4.4 and Theorem 4.6). Criteria for an optimal rate of convergence are presented in Sect. 4.2.7.

### 4.2.3 Stability and Convergence

Associated with the general SA recursion (4.13) are two ODEs,

$$\frac{d}{dt} \xi(t) = \bar{f}(\xi(t)) \quad (4.30)$$

$$\frac{d}{dt} \xi(t) = \bar{f}_\infty(\xi(t)). \quad (4.31)$$

The first is the core foundation of the stochastic approximation method: Algorithm (4.13) is designed to mimic (4.30) [9] (also see Sect. 4.3.6.3). The second ODE (4.31) is defined with respect to the scaled function  $\bar{f}_\infty : \mathbb{R}^d \rightarrow \mathbb{R}^d$ : For each  $x \in \mathbb{R}^d$ ,

$$\bar{f}_\infty(x) := \lim_{r \rightarrow \infty} r^{-1} \bar{f}(rx), \quad (4.32)$$

Stability of the algorithm (4.13) (i.e., boundedness of the iterates  $\theta$ ) is established in [9], under the assumption that the ODE (4.31) is stable. This is formalized in Theorem 4.1 below.

Recursion (4.15) is the focus of discussion through the remainder of this section. The following assumptions are made throughout:

- (A1)  $\bar{f}$  is Lipschitz continuous and the limit  $\bar{f}_\infty(x)$  exists for each  $x \in \mathbb{R}^d$ .
- (A2) The sequence  $\{\Delta_n : n \geq 1\}$  is a martingale difference sequence. Moreover, for some  $\sigma_\Delta^2 < \infty$  and any initial condition  $\theta_0 \in \mathbb{R}^d$ ,

$$\mathbb{E}[\|\Delta_{n+1}\|^2 | \Delta_1, \dots, \Delta_n] \leq \bar{\sigma}_\Delta^2(1 + \|\theta_n\|^2), \quad n \geq 0.$$

- (A3) The sequence  $\{\alpha_n : n \geq 1\}$  is deterministic, satisfies  $0 < \alpha_n \leq 1$ , and

$$\sum_{n \geq 1} \alpha_n = \infty, \quad \sum_{n \geq 1} \alpha_n^2 < \infty. \quad (4.33)$$

Much of the theory in this chapter is restricted to the special case:

(A3\*)  $\alpha_n = 1/n, n \geq 1.$

□

Assumption (A2) is imposed here for simplicity, but it is not required for the convergence rate analysis in Sect. 4.2.5, nor is it essential for the convergence theory for SA [3, 8, 35]. As remarked below (4.15), (A2) does hold when  $\Phi$  is i.i.d.. It also holds for Q-learning in special cases [2, 64].

It is shown in [9, Theorem 2.1] that stability of the ODE (4.31) implies stability of the SA algorithm (4.15):

**Theorem 4.1** Suppose (A1)–(A3) hold, and moreover the origin in  $\mathbb{R}^d$  is an asymptotically stable equilibrium for the ODE (4.31). Then, for any initial condition  $\theta_0 \in \mathbb{R}^d$ ,

$$\sup_n \|\theta_n\| < \infty \quad a.s.. \quad \square$$

We can also guarantee convergence, subject to stability of the natural ODE (4.30). The following Theorem 4.2 is taken from [9, Theorem 2.2].

**Theorem 4.2** Suppose that the assumptions of Theorem 4.1 hold. Moreover, suppose that the ODE (4.30) has a unique globally asymptotically stable equilibrium  $\theta^*$ . Then  $\theta_n \rightarrow \theta^*$  a.s. for any initial condition  $\theta_0 \in \mathbb{R}^d$ .

Under more simplifying assumptions, we can also obtain convergence results for the fixed step-size algorithm ( $\alpha_n \equiv \alpha$  for all  $n \geq 0$ ); the reader is referred to [9] for details (see Theorem 2.3). The focus throughout this chapter is on SA algorithms with vanishing step-size.

#### 4.2.4 Zap–Stochastic Approximation

The results of Sect. 4.2.3 may appear to fall outside of application to RL because of the “white noise” assumption (A2). Even for the special case of TD(0) with linear function approximation (4.29), it will be seen in Sect. 4.2.5 that (A2) is violated. The white noise assumption is not required for convergence—far more general results can be found in [8]. We believe that Theorem 4.1 admits extension to a far larger class of SA recursions.

A more significant challenge is verification of stability of the ODE (4.30). Consider, for example, the TD(0) algorithm (4.27) with nonlinear function approximation. The associated ODE is given by

$$\frac{d}{dt} \xi(t) = \bar{f}(\xi(t)) = \mathbb{E} \left[ \left. \left\{ c(X_n) + \beta h^\theta(X_{n+1}) - h^\theta(X_n) \right\} \nabla_\theta h^\theta(\Phi_n) \right|_{\theta=\xi(t)} \right], \quad (4.34)$$

where the expectation is in steady state. Imagine the complexity of analysis of this ODE when  $h^\theta$  is defined using a neural network?

The Zap–SA algorithm is a two-time-scale algorithm introduced in [20–22] to optimize the rate of convergence:

**Zap–Stochastic Approximation**

For initialization  $\theta_0 \in \mathbb{R}^d$ , and  $\widehat{A}_1 \in \mathbb{R}^{d \times d}$ , obtain the sequence of estimates  $\{\theta_n : n \geq 0\}$  recursively:

$$\theta_{n+1} = \theta_n - \alpha_{n+1} \widehat{A}_{n+1}^{-1} f(\theta_n, \Phi_{n+1}), \quad n \geq 0 \quad (4.35a)$$

$$\widehat{A}_{n+1} = \widehat{A}_n + \gamma_{n+1} [A_{n+1} - \widehat{A}_n], \quad A_{n+1} := \partial_\theta f(\theta_n, \Phi_{n+1}), \quad n \geq 1. \quad (4.35b)$$

The two gain sequences  $\{\alpha_n\}$  and  $\{\gamma_n\}$  each satisfy (A3), and the second sequence is relatively large:

$$\lim_{n \rightarrow \infty} \frac{\alpha_n}{\gamma_n} = 0. \quad (4.36)$$

Under the “high gain” assumption (4.36) we expect that  $\widehat{A}_{n+1}$  is a reasonable estimate of  $A(\theta_n)$  for large  $n$ , with  $A(\theta_n)$  defined in (4.12). This intuition is supported by the general theory of two-time-scale SA (a focus of [8]). Subject to verification of the assumptions in [8], the ODE approximation for the Zap–SA algorithm (4.35) is

$$\frac{d}{dt} \xi(t) = -[A(\xi(t))]^{-1} \bar{f}(\xi(t)). \quad (4.37)$$

The following result appears in [21, 22], and was anticipated by [57] for a related algorithm. The ODE (4.37) also appears in [73], with applications to control.

**Theorem 4.3** *Suppose  $A(\xi)$  is non-singular for every  $\xi \in \mathbb{R}^d$ . Then, the solution to the ODE (4.37) satisfies*

$$\frac{d}{dt} \bar{f}(\xi(t)) = -\bar{f}(\xi(t)).$$

Consequently, for each initial condition,

$$\lim_{t \rightarrow \infty} \|\bar{f}(\xi(t))\| = \lim_{t \rightarrow \infty} \|\bar{f}(\xi(0))\| e^{-t} = 0. \quad (4.38)$$

□

This result is remarkable: we started off in [20–22] with the goal of optimizing the rate of convergence of SA and RL algorithms, and ended up obtaining an algorithm that can *also* achieve global convergence. This amazing property of the Zap SA algorithm has led to the extension of the theory of Zap Q-learning to obtain convergent Q-learning algorithms in a function-approximation setting [11]. These RL algorithms and results from numerical experiments are contained in Sects. 4.3 and 4.4 of this chapter.

### 4.2.5 Rates of Convergence

We now explain how to predict the convergence rate of the SA algorithm (4.13). Specifically, we are interested in the limit  $\Sigma_\theta$  defined in (4.18), wherein the step-size sequence is assumed to satisfy (4.33). The limit (4.18) is refined in Theorem 4.4 to something close to an explicit finite-time mean square error bound.

Much of the SA literature is concerned with the CLT (4.19) (see in particular [3, 35], [33, Theorem 6.1]). Numerical results contained in Sect. 4.4 show that the CLT is often a good predictor of algorithm performance. The analysis in this prior work is based on a “linearized” approximation of the SA recursion (4.13):

$$\theta_{n+1} = \theta_n + \alpha_{n+1} [A_{n+1}\theta_n - b_{n+1}], \quad (4.39)$$

where  $A_{n+1} = \mathcal{A}(\Phi_{n+1})$  is a  $d \times d$  matrix,  $b_{n+1} = b(\Phi_{n+1})$  is a  $d \times 1$  vector, and in reference to (4.13),

$$f(\theta, \Phi_{n+1}) = A_{n+1}\theta - b_{n+1}.$$

*In some cases, no approximation is needed:* recall the recursion (4.28), (4.29) for the TD(0) algorithm.

It is not difficult, but tedious to extend the analysis here to general Lipschitz continuous functions: see discussion in [33, Sect. 7.6].

Let  $A$  and  $b$  denote the respective means,

$$A = \mathbb{E}[\mathcal{A}(\Phi)], \quad b = \mathbb{E}[b(\Phi)]. \quad (4.40)$$

Assumption (A3\*) is assumed throughout this section. In addition,

(A4) The  $d \times d$  matrix  $A$  is Hurwitz.

Assumption (A4) implies that  $A$  is invertible, and that  $\theta^* = A^{-1}b$ . It is also a necessary and sufficient condition for global asymptotic stability of the ODE (4.30).

The recursion (4.39) can be rewritten in the form (4.15):

$$\theta_{n+1} = \theta_n + \alpha_{n+1} [A\theta_n - b + \Delta_{n+1}], \quad (4.41)$$

in which  $\{\Delta_n\}$  is the noise sequence:

$$\begin{aligned} \Delta_{n+1} &= A_{n+1}\theta_n - b_{n+1} - [A\theta_n - b] \\ &= A_{n+1}\theta^* - b_{n+1} + \tilde{A}_{n+1}\tilde{\theta}_n \end{aligned} \quad (4.42)$$

with  $\tilde{A}_{n+1} = A_{n+1} - A$ .

The parameter error sequence is the focus of this subsection. It evolves as the simple linear recursion,

$$\tilde{\theta}_{n+1} = \tilde{\theta}_n + \alpha_{n+1} [A\tilde{\theta}_n + \Delta_{n+1}]. \quad (4.43)$$

The asymptotic covariance (4.18) exists under special conditions, and under these conditions it satisfies the Lyapunov equation

$$(A + \frac{1}{2}I)\Sigma_\theta + \Sigma_\theta(A + \frac{1}{2}I)^T + \Sigma_\Delta = 0. \quad (4.44)$$

The “noise covariance matrix”  $\Sigma_\Delta$  is defined through the following two steps. First, the noise sequence  $\{\Delta_n\}$  in (4.42) is approximated by ignoring the vanishing term  $\tilde{A}_{n+1}\tilde{\theta}_n$  (justification of this approximation will be provided). The resulting sequence is denoted

$$\Delta_n^\infty = A_{n+1}\theta^* - b_{n+1}. \quad (4.45)$$

The covariance then has the following two equivalent forms (see [47, Theorem 17.5.3]):

$$\Sigma_\Delta = \lim_{T \rightarrow \infty} \frac{1}{T} \mathbb{E}[S_T S_T^T] = \sum_{k=-\infty}^{\infty} R_\Delta(k), \quad (4.46)$$

in which  $S_T = \sum_{k=1}^T \Delta_k^\infty$  and

$$R_\Delta(k) = R_\Delta(-k)^T = \mathbb{E}[\Delta_0^\infty \Delta_k^{\infty T}], \quad k \geq 0,$$

where the expectations are in steady state. If  $d = 1$ , so that  $\theta^*$  is a scalar, then the Lyapunov equation admits the explicit solution shown in Fig. 4.1.

We first explain (4.44) for the “white noise” model:  $\Sigma_\Delta = R_\Delta(0) = \mathbb{E}[\Delta_0^\infty \Delta_0^{\infty T}]$ .

#### 4.2.5.1 White Noise Model

We begin with a high-level treatment in the simplest setting in which  $\Phi$  is i.i.d. so that the noise sequence  $\{\Delta_n\}$  in (4.42) is a martingale difference sequence. We relax this condition in Sect. 4.2.5.2 to allow a general class of Markov chains.

Denote for  $n \geq 1$ ,

$$Z_n := \sqrt{n}\tilde{\theta}_n, \quad \text{and} \quad \Sigma_\theta^n := \mathbb{E}[Z_n Z_n^T] = n\mathbb{E}[\tilde{\theta}_n \tilde{\theta}_n^T]. \quad (4.47)$$

The sequence  $Z$  appears in the CLT (4.19), and  $\Sigma_\theta^n$  appears in (4.18).

**Proposition 4.1** *For the i.i.d. model, subject to the step-size assumption (A3\*),*

$$\Sigma_\theta^{n+1} = \Sigma_\theta^n + \frac{1}{n+1} \left\{ (A + k_n I)\Sigma_\theta^n + \Sigma_\theta^n(A + k_n I)^T + \Sigma_\Delta + o\left(\frac{1}{n}\right) \right\}, \quad (4.48)$$

where  $k_n = \frac{1}{2} + o(1/n)$ . If in addition the matrix  $A + \frac{1}{2}I$  is Hurwitz, then  $\Sigma_\theta^n$  converges to  $\Sigma_\theta$ , and this limit is the solution to the Lyapunov equation (4.44).

**Proof (Sketch)** The approximation (4.48) begins with the Taylor series approximation

$$\sqrt{n+1} = \sqrt{n} + \frac{1}{2\sqrt{n}} + o\left(\frac{1}{n}\right).$$

Multiplying both sides of the recursion (4.43) by  $\sqrt{n+1}$  results in a recursion for  $Z$ :

$$Z_{n+1} = Z_n + \frac{1}{n+1} \left\{ (A + \frac{1}{2}I)Z_n + \frac{1}{\sqrt{n+1}} \Delta_{n+1} + o\left(\frac{1}{n}\right) \right\}.$$

Taking outer products on both sides of the recursion, and then taking expectations, we obtain the recursion (4.48) after simplifications. It is in this step that we use the fact that  $\Delta$  is an uncorrelated sequence when  $\Phi$  is i.i.d.

The recursion (4.48) can be viewed as a *linear* stochastic approximation algorithm of the form (4.39), subject to additive noise and multiplicative noise that is  $o(1/n)$ . Convergence follows under the Hurwitz assumption on  $A + \frac{1}{2}I$ .  $\square$

It is not obvious how the recursion (4.48) can be obtained when  $\Phi$  is a Markov chain, but the convergence of the normalized mean square error can be established using a more elaborate analysis.

#### 4.2.5.2 Markovian Model

We adopt the assumptions of [47], which concerns a Markov chain  $\Phi$  on a state space  $Z$ . Some restrictions are required: in [47] it is assumed that  $Z$  is “Polish”, but the reader may assume that  $Z$  is a closed subset of Euclidean space. Don’t be afraid of unfamiliar language from measure theory: the functions  $g : Z \rightarrow \mathbb{R}$  that we come across are typically continuous, which immediately implies “measurable”.

We assume that there is a unique invariant measure  $\pi$  with the defining property that the expectation  $E[g(\Phi_k)]$  is independent of  $k$  provided  $\Phi_0 \sim \pi$ . This *steady-state* mean is also denoted

$$\pi(g) = \int g(z) \pi(dz).$$

Convergence to this steady-state from arbitrary initial conditions requires further assumptions on the chain and the functions of interest. We denote by  $V : Z \rightarrow [1, \infty)$  a function that bounds the growth rate of a measurable function  $g : Z \rightarrow \mathbb{R}$ . Denote by  $L_\infty^V$  the set of such measurable functions satisfying

$$\|g\|_V := \sup_{z \in Z} \frac{|g(z)|}{V(z)} < \infty.$$

The Markov chain  $\Phi$  is assumed to be *V-uniformly ergodic*: there exists  $\delta \in (0, 1)$ , and  $B_V < \infty$  such that for each  $g \in L_\infty^V$ , and  $z \in Z$ ,

$$|E[g(\Phi_n) | \Phi_0 = z] - \pi(g)| \leq B_V \|g\|_V V(z) \delta^n, \quad n \geq 0. \quad (4.49)$$

This holds under a simple Lyapunov drift condition [47, Thm. 16.0.1].

Under the assumption of  $V$ -uniform ergodicity (and stronger assumptions to guarantee stability of the algorithm, and to ensure that  $f$  is appropriately bounded by  $V$  as in (A6) that follows), it can be shown that  $\theta$  obtained using (4.39) converges with probability one to  $\theta^*$  [3, 8, 35]. Convergence of the recursion will be assumed henceforth.

Theorem 4.4 generalizes Proposition 4.1 by relaxing the i.i.d. assumption on  $\Phi$ .

We first collect together assumptions:

**(A5)** The Markov process  $\Phi$  is  $V$ -uniformly ergodic, with unique invariant measure denoted  $\pi$ .

**(A6)** For each  $\theta \in \mathbb{R}^d$ , the function  $f(\theta, \cdot) : \mathcal{Z} \rightarrow \mathbb{R}^d$  defined as

$$f(\theta, \Phi_{n+1}) = A_{n+1}\theta - b_{n+1}$$

satisfies  $\|(f_i(\theta, \cdot))^2\|_V < \infty$ .

We have  $\pi(f_i^{\theta^*}) = 0$  for each  $i$ , with  $f^{\theta^*}(\cdot) = f(\theta^*, \cdot)$ , since  $\theta^* = A^{-1}b$ .

Fixing the initial condition  $(\Phi_0, \tilde{\theta}_0)$ , recall the definition of  $\Sigma_n$  in (4.16):

$$\Sigma_n = \mathbf{E}[\tilde{\theta}_n \tilde{\theta}_n^\top].$$

Also, recall (4.17) for the definition of convergence rate  $1/n^\mu$ . The following result from [12] summarizes bounds on the convergence rate.

**Theorem 4.4** Suppose (A3\*), and (A4)–(A6) hold. Then, for the linear recursion (4.43),

(i) If  $\text{Real}(\lambda) < -\frac{1}{2}$  for every eigenvalue  $\lambda$  of  $A$ , then

$$\Sigma_n = n^{-1} \Sigma_\theta + O(n^{-1-\delta}),$$

where  $\delta = \delta(A, \Sigma_\Delta) > 0$  and  $\Sigma_\theta \geq 0$  is the solution to the Lyapunov equation (4.44). Consequently,  $\mathbf{E}[\|\tilde{\theta}_n\|^2]$  converges to zero at rate  $1/n$ .

(ii) Suppose there is an eigenvalue  $\lambda$  of  $A$  that satisfies  $-\varrho_0 = \text{Real}(\lambda) > -\frac{1}{2}$ . Let  $v \neq 0$  denote a corresponding left eigenvector, and suppose that  $\Sigma_\Delta v \neq 0$ . Then,  $\mathbf{E}[|v^\top \tilde{\theta}_n|^2]$  converges to 0 at a rate  $1/n^{2\varrho_0}$ .  $\square$

The following negative result is a direct corollary of Theorem 4.4 (ii):

**Corollary 4.1** Suppose (A3\*), and (A4)–(A6) hold. Moreover, suppose there is an eigenvalue  $\lambda$  of  $A$  that satisfies  $-\varrho_0 = \text{Real}(\lambda) > -\frac{1}{2}$ , with corresponding left eigenvector  $v^\top$  such that  $\Sigma_\Delta v \neq 0$ . Then,  $\mathbf{E}[\|\tilde{\theta}_n\|^2]$  converges to zero at a rate no faster than  $1/n^{2\varrho_0}$ .  $\square$

The null space of  $\Sigma_\Delta$  has an interesting characterization that can simplify the verification of these assumptions:

**Lemma 4.1** Suppose that (A4)–(A6) hold, and let  $\Sigma_\Delta$  be defined in (4.46) with  $\Delta_{n+1} = A_{n+1}\theta^* - b_{n+1}$ . Then, the following are equivalent for any vector  $v \in \mathbb{R}^d$ :

- (i)  $\Sigma_\Delta v = 0$
- (ii) *The representation  $v^T \Delta_n = g_v(\Phi_{n+1}) - g_v(\Phi_n)$  holds with probability one for each  $n$ , for some function  $g_v$  satisfying  $g_v^2 \in L_\infty^V$ .*  $\square$

#### 4.2.5.3 Implications and Matrix Gain Stochastic Approximation

Theorem 4.4 indicates that the convergence rate of  $\Sigma_n$  is determined jointly by the matrix  $A$ , and the martingale difference component of the noise sequence  $\{\Delta_n\}$ . Convergence of  $\{\tilde{\theta}_n\}$  can be slow if the matrix  $A$  has eigenvalues close to zero. These conclusions explain in part the slow convergence of some reinforcement learning algorithms – see Proposition 4.4 and Theorem 4.6 for further details.

A finite asymptotic covariance  $\Sigma_\theta$  can be guaranteed by scaling the step-size. Under the assumption that  $A$  is Hurwitz, we can choose  $\alpha_n = g/n$  in (4.39), with  $g > 0$  sufficiently large so that each eigenvalue  $\lambda(gA)$  satisfies  $\text{Re}(\lambda) < -\frac{1}{2}$ . Theorem 4.4 then implies that the asymptotic covariance is finite, and can be obtained as a solution to the Lyapunov equation (4.44), with  $A$  replaced by  $gA$ , and  $\Sigma_\Delta$  replaced by  $g^2 \Sigma_\Delta$ . Of course, we are not restricted to a scalar gain.

Introducing a  $d$ -dimensional “matrix gain”  $G \in \mathbb{R}^{d \times d}$  in (4.39), i.e.,  $\alpha_n = G/n$ , we obtain the “matrix gain SA algorithm”:

$$\theta_{n+1} = \theta_n + \frac{1}{n+1} G [A_{n+1} \theta_n - b_{n+1}]. \quad (4.50)$$

Provided  $\frac{1}{2}I + GA$  is Hurwitz, the corresponding asymptotic covariance  $\Sigma_\theta^G$  is finite, and solves a modified Lyapunov equation:

$$(GA + \frac{1}{2}I)\Sigma_\theta^G + \Sigma_\theta^G(GA + \frac{1}{2}I)^T + G\Sigma_\Delta G^T = 0. \quad (4.51)$$

The choice  $G^* = -A^{-1}$  is analogous to the gain used in the Newton–Raphson algorithm (4.12). The asymptotic covariance is finite in this case, and given by

$$\Sigma_\theta^* := A^{-1} \Sigma_\Delta (A^{-1})^T. \quad (4.52)$$

It is a remarkable fact that this choice is optimal in the strongest possible statistical sense: for any other gain  $G$ , the two asymptotic covariance matrices satisfy

$$\Sigma_\theta^G \geq \Sigma_\theta^*$$

That is, the difference  $\Sigma_\theta^G - \Sigma_\theta^*$  is positive semi-definite [3, 8, 35].

The following theorem summarizes these conclusions.

**Theorem 4.5** Suppose (A4) and (A6) hold, and moreover, suppose that the matrix  $\frac{1}{2}I + GA$  is Hurwitz. Then, for the linear recursion (4.50),

- (i)  $\Sigma_n = n^{-1} \Sigma_\theta^G + O(n^{-1-\delta})$ , where  $\delta = \delta(GA, \Sigma_\Delta) > 0$ , and  $\Sigma_\theta^G \geq 0$  is the solution to the Lyapunov equation (4.51).
- (ii) The asymptotic covariance admits the lower bound

$$\Sigma_\theta^G \geq \Sigma_\theta^* := A^{-1} \Sigma_\Delta (A^{-1})^T$$

This bound is achieved using  $G^* := -A^{-1}$ . □

Theorem 4.5 inspires improved algorithms in many settings, one of them being the Zap–SA algorithm of Sect. 4.2.4.

The extensions of the above results to the more general case of  $\alpha_n = 1/n^\rho$ ,  $\rho \in (0.5, 1)$  (which also satisfies (A3)) is not difficult, and leads to an interesting conclusion: under general conditions, the convergence rate of  $E[\|\tilde{\theta}_n\|^2]$  to 0 is *slower* than the optimal rate  $1/n$ , no matter what matrix gain we use.

#### 4.2.6 Optimal Convergence Rate

Theorem 4.5 establishes that the optimal convergence rate of (4.50) is obtained using the matrix gain  $G = -A^{-1}$ . In the two algorithms that follow, this quantity is replaced by a recursive estimate.

##### 4.2.6.1 Stochastic Newton–Raphson

Since the matrix  $A$  is an unknown quantity, the Stochastic Newton–Raphson (SNR) algorithm recursively estimates  $A$  in parallel to the estimation of  $\theta^*$ :

###### Stochastic Newton–Raphson

For initialization  $\theta_0 \in \mathbb{R}^d$ , and  $\widehat{A}_1 \in \mathbb{R}^{d \times d}$ , obtain the sequence of estimates  $\{\theta_n : n \geq 0\}$  recursively:

$$\begin{aligned} \theta_{n+1} &= \theta_n - \alpha_{n+1} \widehat{A}_{n+1}^{-1} [A_{n+1} \theta_n - b_{n+1}], & n \geq 0 \\ \widehat{A}_{n+1} &= \widehat{A}_n + \alpha_{n+1} [A_{n+1} - \widehat{A}_n], \quad \alpha_{n+1} = \frac{1}{n+1}, & n \geq 1. \end{aligned} \tag{4.53}$$

If the steady-state mean  $A$  (defined in (4.40)) is invertible, then  $\widehat{A}_n$  is invertible for all  $n$  sufficiently large.

The sequence  $\{n \widehat{A}_n \theta_n : n \geq 0\}$  admits a simple recursive representation that gives the following alternative representation for the SNR parameter estimates:

**Proposition 4.2** Suppose  $\widehat{A}_n$  is invertible for each  $n \geq 1$ . Then, the sequence of estimates  $\theta$  obtained using (4.53) are identical to the direct estimates:

$$\theta_n = \widehat{A}_n^{-1} \widehat{b}_n, \quad \text{where } \widehat{A}_n = \frac{1}{n} \sum_{i=1}^n A_i, \quad \widehat{b}_n = \frac{1}{n} \sum_{i=1}^n b_i, \quad n \geq 1.$$

□

Based on the proposition, it is obvious that the SNR algorithm (4.53) is consistent whenever the Law of Large Numbers holds for the sequence  $\{A_n, b_n\}$ . Under additional assumptions, Theorem 4.5 can be strengthened to show that the resulting asymptotic covariance is optimal, and given by (4.52).

Algorithm design in this linear setting is simplified in part because  $\bar{f}$  is an affine function of  $\theta$ , so that the gain  $G_n$  appearing in the standard Newton–Raphson algorithm (4.12) does not depend upon the parameter estimates  $\theta$ . However, an ODE analysis of the SNR algorithm suggests that even in this linear setting, the dynamics are very different from its deterministic counterpart: The ODE approximation for (4.53) can be written as follows:

$$\begin{aligned} \frac{d}{dt} \xi(t) &= -\mathcal{A}_t^{-1} [A\xi(t) - b], \\ \frac{d}{dt} \mathcal{A}_t &= -\mathcal{A}_t + A \end{aligned} \tag{4.54}$$

whereas the ODE for the Newton–Raphson algorithm (4.12), for an affine  $\bar{f}$  is

$$\frac{d}{dt} \xi(t) = -A^{-1} [A\xi(t) - b] = -\xi(t) + \theta^*, \tag{4.55}$$

While evidently  $\mathcal{A}_t$  converges to  $A$  exponentially fast in (4.54), with a poor initial condition we might expect poor transient behavior.

In addition, it is not obvious how to obtain a stable algorithm for nonlinear SA, since the linearization matrix  $A(\theta) = \partial_\theta \bar{f}(\theta)$  will depend on  $\theta$ . This was part of the motivation for the two-time-scale Zap SA, whose ODE is analyzed in Theorem 4.3.

#### 4.2.6.2 Zap Stochastic Approximation

The Zap SA algorithm for linear SA is a variant of (4.53):

$$\begin{aligned} \theta_{n+1} &= \theta_n - \alpha_{n+1} \widehat{A}_{n+1}^{-1} [A_{n+1} \theta_n - b_{n+1}] \\ \widehat{A}_{n+1} &= \widehat{A}_n + \gamma_{n+1} [A_{n+1} - \widehat{A}_n] \end{aligned} \tag{4.56}$$

in which  $\{\alpha_n\}$  and  $\{\gamma_n\}$  satisfy (A3) and (4.36). The ODE approximation (4.37) exactly matches the usual Newton–Raphson dynamics (4.55).

Under the assumption that  $\alpha_n = 1/n$ , and  $\gamma_n = (1/n)^\rho$  for  $\rho \in (0.5, 1)$ , it is possible to show that the asymptotic covariance is again optimal. This however

requires conditions to ensure that the sequence  $\widehat{A}_n^{-1}$  is bounded (e.g.,  $-A_n$  is positive semi-definite for each  $n$ , a.s., and  $-\widehat{A}_0 > 0$ ). Alternatively, the inverse may be approximated, in which case we give up a fraction of optimality for greater numerical stability. The matrix gain considered in [11] is the approximate pseudoinverse  $G_{n+1} = -[\varepsilon I + \widehat{A}_{n+1}^\top \widehat{A}_{n+1}]^{-1} \widehat{A}_{n+1}^\top$ , with  $\varepsilon > 0$  a small parameter.

#### 4.2.6.3 Other Optimal Algorithms

Before moving on to the application of the above theory and algorithms to RL, we briefly discuss two other well-known SA algorithms that also achieve the optimal asymptotic covariance  $\Sigma_\theta^*$  defined in (4.52). In this prior work, the goal is to obtain the CLT (4.19) with covariance  $\Sigma_\theta^*$ , rather than in the sense of minimizing the normalized mean square error (4.18) that is the focus here.

#### I. Stochastic Quasi-Newton–Raphson

We discovered only recently that Ruppert in [57] had proposed an algorithm that also intends to mimic the ODE (4.37). However, the algorithm itself is very different from (4.35) in two key aspects:

- (i) Ruppert does not assume access to the gradients  $\{A_{n+1}\}$  of  $f(\theta_n, \Phi_{n+1})$ . Instead, the algorithm uses a Keifer–Wolfowitz like procedure to approximate the gradients with finite differences [29].
- (ii) The mean  $A(\theta_n)$  is *not* estimated recursively. Instead, each iteration of the algorithm assumes access to multiple function evaluations (i.e.,  $f(\theta_n, \Phi_{n+1})$ ) to estimate  $A(\theta_n)$ . In-fact, for convergence of the algorithm, the number of function evaluations required grows with the number of iterations of the algorithm. This is not suitable for “online” applications wherein we have a single sample path of the Markov chain, as in most variants of RL.

#### II. Ruppert–Polyak Averaging

The Polyak–Ruppert averaging technique is also a special case of two-time-scale stochastic approximation [53, 54, 58]. The faster time-scale of the algorithm is a recursion of the form (4.13), with a *larger* step-size. The slower time-scale of the algorithm essentially averages the parameter estimates obtained from the fast time-scale: with initialization  $\theta_0^\bullet \in \mathbb{R}^d$  and  $\theta_0 \in \mathbb{R}^d$ , the averaging technique for the linear model is given by the following pair of recursions: For  $n \geq 0$ ,

$$\begin{aligned} \theta_{n+1}^\bullet &= \theta_n^\bullet + \gamma_{n+1} [A_{n+1} \theta_n^\bullet - b_{n+1}] \\ \theta_{n+1} &= \theta_n + \alpha_{n+1} [\theta_{n+1}^\bullet - \theta_n]. \end{aligned} \tag{4.57}$$

The step-sizes are chosen just as in the Zap SA algorithm. It is common to choose  $\alpha_n = 1/n$  and  $\gamma_n = 1/n^\rho$ ,  $\rho \in (0.5, 1)$ .

In [32, 53, 58], it is shown that the algorithm achieves the optimal asymptotic covariance (4.52). However, we observe in experiments in Sect. 4.3 that the transient

performance of the algorithm can be very bad, especially when the matrix  $A$  is poorly conditioned.

### 4.2.7 TD and LSTD Algorithms

Recall the variance plot in Fig. 4.1 for the simple linear recursion (4.20). It is not surprising that infinite variance is also typical for TD-learning algorithms used for value function approximation. There is a larger danger because the value function grows as  $1/(1 - \beta)$  as the discount factor  $\beta$  approaches unity. Suppose that the Markov chain is irreducible, with unique invariant measure  $\pi$ . Denote  $\tilde{c}(x) = c(x) - \eta$  for  $x \in \mathcal{X}$ , with  $\eta = \int c(x) \pi(dx)$  the steady-state mean.

**Proposition 4.3** *The value function  $h$  defined in (4.21) admits the decomposition*

$$h(x) = \frac{1}{1 - \beta} \eta + \tilde{h}(x), \quad \tilde{h}(x) = \sum_{n=0}^{\infty} \beta^n \mathbb{E}[\tilde{c}(X_n) \mid X_0 = x], \quad x \in \mathcal{X}. \quad (4.58)$$

As  $\beta \uparrow 1$ , the function  $\tilde{h}$  is convergent to  $\hat{h}$ , a solution to Poisson's equation:

$$\hat{h}(x) = \tilde{c}(x) + \mathbb{E}[\hat{h}(X_{n+1}) \mid X_n = x]. \quad (4.59)$$

Consequently, an approximation architecture is likely to suffer from numerical challenges if it doesn't take into account the fact that  $h$  is large only because of the constant  $\eta/(1 - \beta)$ .  $\square$

Proposition 4.4 that follows illustrates this challenge for TD(0)-learning, framed within the vector-space setting of this section. Recall from Sect. 4.2.2 that the function  $\bar{f}$  associated with the TD(0)-learning algorithm is linear in  $\theta$  when the parameterization is linear. Applying (4.28) and (4.29) gives  $\bar{f}(\theta) = A\theta - b$  with

$$A = \mathbb{E}[A_{n+1}], \quad A_{n+1} = \psi(X_n)(\beta\psi(X_{n+1}) - \psi(X_n))^T, \quad b = -\mathbb{E}[c(X_n)\psi(X_n)].$$

We write  $A^\beta$  below to emphasize dependency on the discount factor  $\beta$ .

It is frequently found in examples that  $A^\beta + \frac{1}{2}I$  fails to be Hurwitz when the discount factor is close to 1. A general class of models for which this is the case is provided next, under the assumption that there is a vector  $v \in \mathbb{R}^d$  such that

$$1 = \sum_i v_i \psi_i(x), \quad x \in \mathcal{X}. \quad (4.60)$$

Under this assumption, (4.61) follows from (4.25), with  $\zeta_n = \psi(X_n)$ :

$$v^T \Delta_{n+1} := v^T f(\theta^*, \Phi_{n+1}) = \sum_i v_i \mathcal{D}(h^{\theta^*}, \Phi_{n+1}) \psi_i(X_n) = \mathcal{D}(h^{\theta^*}, \Phi_{n+1})$$

(4.61)

*where*  $\mathcal{D}(h^{\theta^*}, \Phi_{n+1}) = c(X_n) + \beta h^{\theta^*}(X_{n+1}) - h^{\theta^*}(X_n)$ .

This representation combined with Lemma 4.1 implies that the assumption  $\Sigma_\Delta v \neq 0$  can be expected to hold in the majority of examples.

**Proposition 4.4** Suppose that (4.60) holds for some  $v \in \mathbb{R}^d$ , and that  $\Sigma_\Delta v \neq 0$ . Then,

- (i)  $\lambda_{\min}\{(A^\beta)^T A^\beta\} \rightarrow 0$  as  $\beta \uparrow 1$ .
- (ii) There is  $\beta_0 < 1$  such that  $A^\beta + \frac{1}{2}I$  is not Hurwitz for all  $\beta \in [\beta_0, 1]$
- (iii) For  $\beta \in (\beta_0, 1)$ , the convergence rate of  $\mathbb{E}[\|\tilde{\theta}_n\|^2]$  to zero is  $1/n^\varepsilon$  for some  $\varepsilon < 1$ , and  $\varepsilon \rightarrow 0$  as  $\beta \uparrow 1$ . □

**Proof (Sketch)** We have  $[A^\beta v]_i = -(1-\beta)\pi(\psi_i)$  for each  $i$ , which implies (i). Also, since  $A^\beta v = 0$  for  $\beta = 1$ , it can be shown that there is a continuous path  $\{\lambda(\beta)\}$ , where  $\lambda(\beta)$  is an eigenvalue for  $A^\beta$  for each  $\beta$  in a neighborhood of  $\beta = 1$ , and  $\lambda(\beta) \rightarrow 0$  as  $\beta \uparrow 1$ . This implies (ii). Part (iii) follows from (ii) and Theorem 4.4. □

Assumption (4.60) holds in the “tabular” setting for which the span contains all functions on  $Z$  (in this case the state space is necessarily finite). Watkins’ Q-learning algorithm is posed in this special setting, and in fact we find that Proposition 4.4 admits an extension to this algorithm, with  $\beta_0 = \frac{1}{2}$  (see Theorem 4.6 of Sect. 4.3).

The SNR algorithm (4.53) can be applied to TD(0)-learning to obtain the optimal convergence rate of  $1/n$ :

$$\theta_{n+1} = \theta_n - \alpha_{n+1} \widehat{A}_{n+1}^{-1} d_{n+1} \psi(X_n) \quad (4.62a)$$

$$d_{n+1} = c(X_n) + \beta h^{\theta_n}(X_{n+1}) - h^{\theta_n}(X_n)$$

$$\widehat{A}_{n+1} = \widehat{A}_n + \alpha_{n+1} [A_{n+1} - \widehat{A}_n]. \quad (4.62b)$$

Under the assumption that the sequence of matrices  $\{\widehat{A}_n : n \geq 0\}$  is invertible for each  $n$ , Proposition 4.2 implies that the sequence of estimates obtained using (4.62) are identical to the parameter estimates obtained using the LSTD(0) algorithm:

$$\theta_{n+1} = \widehat{A}_{n+1}^{-1} \widehat{b}_{n+1} \quad (4.63a)$$

$$\text{where: } \widehat{A}_{n+1} = \widehat{A}_n + \alpha_{n+1} [A_{n+1} - \widehat{A}_n] \quad (4.63b)$$

$$\widehat{b}_{n+1} = \widehat{b}_n + \alpha_{n+1} [b_{n+1} - \widehat{b}_n]. \quad (4.63c)$$

Consequently, the LSTD(0) algorithm achieves optimal asymptotic covariance (which was also shown in [33, Theorem 6.3]).

The Zap–SA algorithm can also be applied to obtain the optimal convergence rate. The parameter estimates approximate the Newton–Raphson ODE (4.55) in this case.

### 4.3 Zap Q-Learning: Fastest Convergent Q-Learning

The Zap Q-learning algorithm is the Zap–SA algorithm (4.35), applied to Q-learning. A full analysis of the algorithm is presented here for the special case of a “tabular representation” (similar to the setting of Watkins’ Q-learning algorithm [74, 75]). It is found that the associated ODE has a simple representation, which implies consistency under suitable assumptions. The ODE approximation suggests an extension of the theory for convergence of the algorithm in non-ideal parameterization settings, which is briefly discussed in Sect. 4.5.

#### 4.3.1 Markov Decision Processes

Consider a Markov Decision Process (MDP) model with state space  $X$ , action space  $U$ , cost function  $c: X \times U \rightarrow \mathbb{R}$ , and discount factor  $\beta \in (0, 1)$ . It is assumed throughout this section that the state and action spaces are finite: denote  $\ell = |X|$  and  $\ell_u = |U|$ . In the following, the terms “action”, “control”, and “input” are used interchangeably.

Along with the state–action process  $(X, U)$  is an i.i.d. sequence  $I = \{I_1, I_2, \dots\}$  used to model a randomized policy. We assume without loss of generality that each  $I_n$  is real-valued, with uniform distribution on the interval  $[0, 1]$ . An input sequence  $U$  is called *non-anticipative* if

$$U_n = z_n(X_0, U_0, I_1, \dots, U_{n-1}, X_n, I_n), \quad n \geq 0,$$

where  $\{z_n\}$  is a sequence of functions. The input sequence is *admissible* if it is non-anticipative, and if it is feasible in the sense that  $X_{n+1}$  remains in the state space for each  $n$ .

Under the assumption that the state and action spaces are finite, it follows that there are a finite number of deterministic stationary policies  $\{\phi^{(i)} : 1 \leq i \leq \ell_\phi\}$ , where each  $\phi^{(i)} : X \rightarrow U$ , and  $\ell_\phi = (\ell_u)^\ell$ . A randomized stationary policy is defined by a probability mass function (pmf)  $\mu$  on the integers  $\{1 \leq i \leq \ell_\phi\}$  such that

$$U_n = \sum_{k=1}^{\ell_\phi} \iota_n(k) \phi^{(k)}(X_n) \tag{4.64}$$

with  $\mu(k) = \mathbb{P}\{\iota_n(k) = 1 \mid X_0, \dots, X_n\}$  for each  $n \geq 0$  and  $1 \leq k \leq \ell_\phi$ . It is assumed that  $\iota_n$  is a fixed function of  $(I_n, X_n)$  for each  $n$  so that this input sequence is non-anticipative.

It is convenient to use the following operator-theoretic notation. The controlled transition matrix  $P_u$  acts on functions  $h: X \rightarrow \mathbb{R}$  via

$$\begin{aligned} P_u h(x) &:= \sum_{x' \in X} P_u(x, x') h(x') \\ &= \mathbb{E}[h(X_{n+1}) \mid X_n = x, U_n = u; X_k, I_k, U_k : k < n], \quad x \in X, u \in U, \end{aligned} \tag{4.65}$$

where the second equality holds for any non-anticipative input sequence  $U$ .

For any deterministic stationary policy  $\phi$ , let  $S_\phi$  denote the substitution operator, defined for any function  $q : X \times U \rightarrow \mathbb{R}$  by

$$S_\phi q(x) := q(x, \phi(x)).$$

If the policy  $\phi$  is randomized, of the form (4.64), then we define

$$S_\phi q(x) = \sum_k \mu(k) q(x, \phi^{(k)}(x)).$$

With  $P$  viewed as a single matrix with  $\ell \cdot \ell_u$  rows and  $\ell$  columns, and  $S_\phi$  viewed as a matrix with  $\ell$  rows and  $\ell \cdot \ell_u$  columns, the following interpretations hold:

**Lemma 4.2** *Suppose that  $U$  is defined using a stationary policy  $\phi$  (possibly randomized). Then, both  $X$  and the pair process  $(X, U)$  are Markovian, and*

- (i)  $P_\phi := S_\phi P$  is the transition matrix for  $X$ .
- (ii)  $PS_\phi$  is the transition matrix for  $(X, U)$ .

□

### 4.3.2 Value Functions and the Bellman Equation

For any (possibly randomized) stationary policy  $\phi$ , we consider two value functions

$$h_\phi(x) := \sum_{n=0}^{\infty} (\beta P_\phi)^n S_\phi c(x) \tag{4.66a}$$

$$Q_\phi(x, u) := \sum_{n=0}^{\infty} (\beta P S_\phi)^n c(x, u). \tag{4.66b}$$

The function  $h_\phi : X \rightarrow \mathbb{R}$  in (4.66a) is the same as  $h$  defined in (4.21), with transition probability matrix  $P_\phi$ , and cost function  $S_\phi c$ . The function  $Q_\phi : X \times U \rightarrow \mathbb{R}$  is the fixed-policy Q-function considered in the SARSA algorithm [45, 56, 65].

Denoting  $h^*$  to be the minimal (optimal) value function

$$h^*(x) := \min_{\phi} h_\phi(x),$$

it is known that  $h^*$  is the unique solution to the following Bellman equation:

$$h^*(x) = \min_u \left\{ c(x, u) + \beta \sum_{x' \in \mathcal{X}} P_u(x, x') h^*(x') \right\}, \quad x \in \mathcal{X}. \quad (4.67)$$

The minimizer defines a deterministic stationary policy  $\phi^*: \mathcal{X} \rightarrow \mathcal{U}$  that is optimal over all input sequences [4]:

$$\phi^*(x) = \arg \min_u \left\{ c(x, u) + \beta \sum_{x' \in \mathcal{X}} P_u(x, x') h^*(x') \right\}, \quad x \in \mathcal{X}. \quad (4.68)$$

The Q-function associated with (4.67) is defined to be the term within the brackets:

$$Q^*(x, u) := c(x, u) + \beta P_u h^*(x), \quad x \in \mathcal{X}, u \in \mathcal{U}.$$

The Bellman equation (4.67) implies a similar fixed point equation for the Q-function:

$$Q^*(x, u) = c(x, u) + \beta \sum_{x' \in \mathcal{X}} P_u(x, x') \underline{Q}^*(x'), \quad (4.69)$$

in which  $\underline{Q}(x) := \min_u Q(x, u)$  for any function  $Q: \mathcal{X} \times \mathcal{U} \rightarrow \mathbb{R}$ .

For any function  $q: \mathcal{X} \times \mathcal{U} \rightarrow \mathbb{R}$ , let  $\phi^q: \mathcal{X} \rightarrow \mathcal{U}$  denote an associated policy that satisfies

$$\phi^q(x) \in \arg \min_u q(x, u), \quad x \in \mathcal{X}. \quad (4.70)$$

It is assumed to be specified *uniquely* as follows:

$$\phi^q := \phi^{(\kappa)} \text{ such that } \kappa = \min\{i : \phi^{(i)}(x) \in \arg \min_u q(x, u), \text{ for all } x \in \mathcal{X}\}. \quad (4.71)$$

Using the above notations, the fixed point equation (4.69) can be rewritten as

$$Q^*(x, u) = c + \beta P S_\phi Q^*(x, u), \quad \text{with } \phi = \phi^q, q = Q^*. \quad (4.72)$$

In the analysis that follows it is necessary to consider the Q-function associated with all possible cost functions simultaneously: given any function  $\varsigma: \mathcal{X} \times \mathcal{U} \rightarrow \mathbb{R}$ , let  $Q(\varsigma)$  denote the corresponding solution to the fixed point equation (4.69), with  $c$  replaced by  $\varsigma$ . That is, the function  $q = Q(\varsigma)$  is the solution to the fixed point equation,

$$q(x, u) = \varsigma(x, u) + \beta \sum_{x'} P_u(x, x') \min_{u'} q(x', u'), \quad x \in \mathcal{X}, u \in \mathcal{U}. \quad (4.73)$$

For a pmf  $\mu$  defined on the set of policy indices  $\{1 \leq i \leq \ell_\phi\}$ , denote

$$\partial Q_\mu := \left( \sum \mu(i) [I - \beta P S_{\phi^{(i)}}] \right)^{-1}, \quad (4.74)$$

so that  $\partial Q_\mu c = Q_\phi$  defined in (4.66b), where  $\phi$  is the randomized policy defined by  $\mu$ . It follows from (4.73) that the functional  $Q$  can be expressed as the minimum over all pmf's:

$$Q(\varsigma) = \min_{\mu} \partial Q_\mu \varsigma. \quad (4.75)$$

It is known that there exists a single degenerate pmf that attains the minimum for each  $(x, u)$  (the optimal stationary policy is deterministic, and is given by (4.68)) [4].

**Lemma 4.3** *The mapping  $Q$  is a bijection on the set of real-valued functions on  $X \times U$ . It is also piece-wise linear, monotone, and denoting  $q_\varsigma = Q(\varsigma)$ ,  $q_\varsigma(x, u)$  is concave in  $\varsigma$  for each  $x \in X$  and  $u \in U$ .*

**Proof** The fixed point equation (4.73) defines the  $Q$ -function with respect to the cost function  $\varsigma$ . Concavity and monotonicity hold because  $q_\varsigma = Q(\varsigma)$  as defined in (4.75) is the minimum of linear, monotone functions. The existence of an inverse  $q_\varsigma \mapsto \varsigma$  follows from (4.73).  $\square$

### 4.3.3 *Q-Learning*

The goal in Q-learning is to approximately solve the fixed point equation (4.69), without assuming knowledge of the controlled transition matrix. We consider a linear parameterization for the  $Q$ -function:  $Q^\theta(x, u) = \theta^\top \psi(x, u)$ , where  $\theta \in \mathbb{R}^d$  denotes the parameter vector, and  $\psi : X \times U \rightarrow \mathbb{R}^d$  denotes the vector of basis functions.

Following along the lines of Sect. 4.2.2.2, a *Galerkin approach* to approximating  $Q^*$  is formulated as follows: Obtain a non-anticipative input sequence  $\mathbf{U}$  (using a randomized stationary policy  $\phi$ ), and a  $d$ -dimensional stationary stochastic process  $\zeta$  that is adapted to  $(X, \mathbf{U})$ ; The goal in *Galerkin relaxation* of the problem of solving (4.69) is to find  $\theta^*$  such that

$$\bar{f}_i(\theta^*) := \mathbb{E}[\{c(X_n, U_n) + \beta \underline{Q}^{\theta^*}(X_{n+1}) - Q^{\theta^*}(X_n, U_n)\}\zeta_n(i)] = 0, \quad 1 \leq i \leq d, \quad (4.76)$$

where  $\underline{Q}^\theta(x) = \min_u Q^\theta(x, u)$ , and the expectation is with respect to the steady-state distribution of the Markov chain  $(X, \mathbf{U})$ . This is clearly a special case of the general root-finding problem that was the focus of Sect. 4.2. The choice for the sequence of eligibility vectors  $\{\zeta_n\}$  in (4.77c) below is inspired by the TD( $\lambda$ ) algorithm [62, 69]. The following Q( $\lambda$ ) algorithm is the SA algorithm (4.13), applied to estimate  $\theta^*$  that solves (4.76):

**$Q(\lambda)$  Algorithm**

For initialization  $\theta_0, \zeta_0 \in \mathbb{R}^d$ , define the sequence of estimates recursively:

$$\theta_{n+1} = \theta_n + \alpha_{n+1} \zeta_n d_{n+1} \quad (4.77a)$$

$$d_{n+1} = c(X_n, U_n) + \beta \underline{Q}^{\theta_n}(X_{n+1}) - Q^{\theta_n}(X_n, U_n) \quad (4.77b)$$

$$\zeta_{n+1} = \lambda \beta \zeta_n + \psi(X_{n+1}, U_{n+1}). \quad (4.77c)$$


---

Matrix gain Q-learning algorithms are also popular. For a sequence of  $d \times d$  matrices  $\mathbf{G} = \{G_n\}$  and  $\lambda \in [0, 1]$ , the *matrix-gain Q( $\lambda$ ) algorithm* is described as follows:

 **$G$ - $Q(\lambda)$  Algorithm**

For initialization  $\theta_0, \zeta_0 \in \mathbb{R}^d$ , the sequence of estimates are defined recursively:

$$\theta_{n+1} = \theta_n + \alpha_{n+1} G_{n+1} \zeta_n d_{n+1} \quad (4.78a)$$

$$d_{n+1} = c(X_n, U_n) + \beta \underline{Q}^{\theta_n}(X_{n+1}) - Q^{\theta_n}(X_n, U_n) \quad (4.78b)$$

$$\zeta_{n+1} = \lambda \beta \zeta_n + \psi(X_{n+1}, U_{n+1}). \quad (4.78c)$$


---

A common choice when  $\lambda = 0$  is

$$G_n = \left( \frac{1}{n} \sum_{k=1}^n \psi(X_k, U_k) \psi^\top(X_k, U_k) \right)^{-1}. \quad (4.79)$$

Examples will follow shortly, and again in Sect. 4.4.2.1.

The success of these algorithms has been demonstrated in a few restricted settings, such as optimal stopping [13, 70, 77], deterministic optimal control [41], and the tabular setting discussed next.

#### 4.3.4 Tabular Q-Learning

The basic Q-learning algorithm of Watkins [74, 75] (also known as “*tabular*” Q-learning) is a particular instance of the Galerkin approach (77), with  $\lambda = 0$ . The basis functions are taken to be indicator functions:

$$\psi_i(x, u) = \mathbb{I}\{x = x^i, u = u^i\}, \quad 1 \leq i \leq d, \quad (4.80)$$

where  $\{(x^k, u^k) : 1 \leq k \leq d\}$  is an enumeration of all state-input pairs, with  $d = \ell \cdot \ell_u$ . The goal of this approach is to *exactly* compute the function  $Q^*$ . Substituting  $\zeta_n \equiv \psi(X_n, U_n)$  with  $\psi$  defined in (4.80), the objective (4.76) can be rewritten as follows: Find  $\theta^* \in \mathbb{R}^d$  such that, for each  $1 \leq i \leq d$ ,

$$0 = \mathbb{E}\left[\{c(X_n, U_n) + \beta \underline{Q}^{\theta^*}(X_{n+1}) - Q^{\theta^*}(X_n, U_n)\}\psi_i(X_n, U_n)\right] \quad (4.81)$$

$$= \left[c(x^i, u^i) + \beta \mathbb{E}\left[\underline{Q}^{\theta^*}(X_{n+1}) | X_n = x^i, U_n = u^i\right] - Q^{\theta^*}(x^i, u^i)\right] \varpi(x^i, u^i), \quad (4.82)$$

where the expectation in (4.81) is in steady state, and  $\varpi$  in (4.82) denotes the invariant distribution of the Markov chain  $(X, U)$ . The conditional expectation in (4.82) is

$$\mathbb{E}\left[\underline{Q}^{\theta^*}(X_{n+1}) | X_n = x^i, U_n = u^i\right] = \sum_{x' \in \mathcal{X}} P_{u^i}(x^i, x') \underline{Q}^{\theta^*}(x').$$

Consequently, (4.82) can be rewritten as

$$0 = \left[c(x^i, u^i) + \beta \sum_{x' \in \mathcal{X}} P_{u^i}(x^i, x') \underline{Q}^{\theta^*}(x') - Q^{\theta^*}(x^i, u^i)\right] \varpi(x^i, u^i). \quad (4.83)$$

If  $\varpi(x^i, u^i) > 0$  for each  $1 \leq i \leq d$ , then the function  $Q^{\theta^*}$  that solves (4.83) is identical to the optimal Q-function in (4.69).

There are three flavors of Watkins' Q-learning that are popular in the literature. We discuss each of them below.

**Asynchronous Q-learning:** The SA algorithm applied to solve (4.81) coincides with the most basic version of Watkins' Q-learning algorithm:

#### Asynchronous Q-learning

For initialization  $\theta_0 \in \mathbb{R}^d$ , define the sequence of estimates  $\{\theta_n : n \geq 0\}$  recursively:

$$\theta_{n+1} = \theta_n + \alpha_{n+1} [c(X_n, U_n) + \beta \underline{Q}^{\theta_n}(X_{n+1}) - Q^{\theta_n}(X_n, U_n)] \psi(X_n, U_n), \quad (4.84)$$

where  $\{\alpha_n : n \geq 0\}$  denotes the non-negative step-size sequence.

Based on the choice of basis functions (4.80), a single entry of  $\theta$  is updated at each iteration, corresponding to the state-input pair  $(X_n, U_n)$  observed (hence the term “asynchronous”). Observing that  $\theta$  is identified with the estimate  $Q^\theta$ , a more familiar form of (4.84) is

$$Q^{n+1}(X_n, U_n) = Q^n(X_n, U_n) + \alpha_{n+1} [c(X_n, U_n) + \beta \underline{Q}^n(X_{n+1}) - Q^n(X_n, U_n)]. \quad (4.85)$$

With  $\alpha_n = 1/n$ , the ODE approximation of (4.84) takes the following form:

$$\frac{d}{dt} q_t(x, u) = \varpi(x, u) \left[ c(x, u) + \beta P_u \underline{q}_t(x) - q_t(x, u) \right], \quad (4.86)$$

in which  $\underline{q}_t(x) = \min_u q_t(x, u)$  as defined below (4.69). In Sect. 4.3.5, we discuss the conditions under which the above ODE is stable. We also see that we cannot expect a finite asymptotic covariance if  $\varpi(x, u)$  is very small for some  $(x, u)$ .

A second and perhaps more popular “Q-learning flavor” is defined using a particular “state-action dependent” step-size [20, 26, 64]. For each  $(x, u)$ , we set  $\alpha_n(x, u) = 0$  if the pair  $(x, u)$  has not been visited up until time  $n - 1$ . Otherwise,

$$\alpha_n(x, u) = [n(x, u)]^{-1}, \quad n(x, u) := \sum_{j=0}^{n-1} \mathbb{I}\{X_j = x, U_j = u\}. \quad (4.87)$$

At stage  $n + 1$  of the algorithm, once  $(x, u) = (X_n, U_n)$  and  $X_{n+1}$  are observed, then a single entry of the Q-function is updated as in (4.85):

$$Q^{n+1}(x, u) = Q^n(x, u) + \alpha_{n+1}(x, u) [c(x, u) + \beta \underline{Q}^n(X_{n+1}) - Q^n(x, u)].$$

With the above step-size rule, the ODE approximation simplifies:

$$\frac{d}{dt} q_t(x, u) = c(x, u) + \beta P_u \underline{q}_t(x) - q_t(x, u). \quad (4.88)$$

Conditions for a finite asymptotic covariance are also greatly simplified (see Theorem 4.7).

The asynchronous variant of Watkins’ Q-learning algorithm (4.84) with step-size (4.87) can be viewed as the  $G$ -Q(0) algorithm defined in (4.78), with the matrix gain sequence (4.79), and step-size  $\alpha_n = 1/n$ . On substituting the Watkins’ basis defined in (4.80), we find that this matrix is diagonal:

$$G_n = \widehat{\Pi}_n^{-1}, \quad \widehat{\Pi}_n(i, i) = \frac{1}{n} \sum_{k=1}^n \mathbb{I}\{X_k = x^i, U_k = u^i\}, \quad 1 \leq i \leq d. \quad (4.89)$$

By the Law of Large Numbers, we have

$$\lim_{n \rightarrow \infty} G_n = \lim_{n \rightarrow \infty} \widehat{\Pi}_n^{-1} = \Pi^{-1}, \quad (4.90)$$

where  $\Pi$  is a diagonal matrix with entries  $\Pi(i, i) = \varpi(x^i, u^i)$ . It is easy to see why the ODE approximation (4.86) simplifies to (4.88) with this matrix gain.

**Synchronous Q-learning:** In this final flavor, each entry of the Q-function approximation is updated in each iteration. It is very popular in the literature because the analysis is greatly simplified.

The algorithm assumes access to an “oracle” that provides the next state of the Markov chain, conditioned on any given current state–action pair: let  $\{X_n^i : n \geq 1, 1 \leq i \leq d\}$  denote a collection of mutually independent random variables taking values in  $\mathbf{X}$ . Assume moreover that for each  $i$ , the sequence  $\{X_n^i : n \geq 1\}$  is i.i.d. with common distribution  $P_{u^i}(x^i, \cdot)$ . The *synchronous Q-learning* algorithm is then obtained:

#### Synchronous Q-learning

For initialization  $\theta_0 \in \mathbb{R}^d$ , define the sequence of estimates  $\{\theta_n : n \geq 0\}$  recursively:

$$\theta_{n+1} = \theta_n + \alpha_{n+1} \sum_{i=1}^d [c(x^i, u^i) + \beta \underline{Q}^{\theta_n}(X_{n+1}^i) - Q^{\theta_n}(x^i, u^i)] \psi(x^i, u^i). \quad (4.91)$$

Once again, based on the choice of basis functions (4.80), and observing that  $\theta$  is identified with the estimate  $\underline{Q}^\theta$ , an equivalent form of the update rule (4.91) is

$$Q^{n+1}(x^i, u^i) = Q^n(x^i, u^i) + \alpha_{n+1} [c(x^i, u^i) + \beta \underline{Q}^n(X_{n+1}^i) - Q^n(x^i, u^i)], \quad 1 \leq i \leq d.$$

Using the step-size  $\alpha_n = 1/n$ , we obtain the simple ODE approximation (4.88).

### 4.3.5 Convergence and Rate of Convergence

Convergence of the tabular Q-learning algorithms requires the following assumptions:

(Q1) The input  $\mathbf{U}$  is defined by a randomized stationary policy of the form (4.64). The joint process  $(\mathbf{X}, \mathbf{U})$  is an irreducible Markov chain. That is, it has a unique invariant pmf  $\varpi$  satisfying  $\varpi(x, u) > 0$  for each  $x, u$ .

(Q2) The optimal policy  $\phi^*$  is unique. □

Both ODEs (4.86) and (4.88) are stable under assumption (Q1) [9], which then (based on the discussion in Sect. 4.2.3) implies that  $\theta$  converges to  $Q^*$  a.s.. Obtaining the rates of convergence requires an examination of the linearization of the ODEs at their equilibrium.

Linearization is justified under Assumption (Q2), which implies the existence of  $\varepsilon > 0$  such that

$$\phi^*(x) = \arg \min_{u \in \mathbf{U}} Q^\theta(x, u), \quad x \in \mathbf{X}, \theta \in \mathbb{R}^d, \|Q^\theta - Q^*\| < \varepsilon. \quad (4.92)$$

**Lemma 4.4** Under Assumptions (Q1) and (Q2) the following approximations hold

(i) When  $\|q_t - Q^*\| < \varepsilon$ , the ODE (4.86) reduces to

$$\frac{d}{dt}q_t = -\Pi[I - \beta PS_{\phi^*}]q_t - b,$$

where  $\Pi$  is defined below (4.89), and  $b(x, u) = -\varpi(x, u)c(x, u)$ , expressed as a  $d$ -dimensional column vector.

(ii) When  $\|q_t - Q^*\| < \varepsilon$ , the ODE (4.88) reduces to

$$\frac{d}{dt}q_t = -[I - \beta PS_{\phi^*}]q_t - b,$$

where  $b(x, u) = -c(x, u)$ .

**Proof** To prove (i) we recall the definition of the ODE (4.86):  $\frac{d}{dt}q_t = \bar{f}(q_t)$ , where for any  $q: \mathcal{X} \times \mathcal{U} \rightarrow \mathbb{R}$ , and  $1 \leq i \leq d$ ,

$$\bar{f}_i(q) = \mathbb{E}\left[\{c(X_n, U_n) + \beta q(X_{n+1}) - q(X_n, U_n)\}\psi_i(X_n, U_n)\right].$$

Substituting  $\underline{q}(X_{n+1}) = q(X_{n+1}, \phi^*(X_{n+1}))$  for  $\|q - Q^*\| < \varepsilon$  gives

$$\begin{aligned} \bar{f}_i(q) &= \mathbb{E}[c(X_n, U_n)\psi_i(X_n, U_n)] \\ &\quad + \mathbb{E}[\psi_i(X_n, U_n)\{\beta q(X_{n+1}, \phi^*(X_{n+1})) - q(X_n, U_n)\}] \\ &= \varpi(x^i, u^i)c(x^i, u^i) + \varpi(x^i, u^i)\{\beta \sum_j P_{u^i}(x^i, x^j)q(x^j, \phi^*(x^j)) - q(x^i, u^i)\}, \end{aligned}$$

where the second identity follows from the tabular basis. This establishes (i).

Part (ii) is immediate, given the similarity of the two ODEs.  $\square$

Recall the defintion of the linearization matrix

$$A = \partial_\theta \bar{f}(\theta) \Big|_{\theta=\theta^*}.$$

The crucial take away from Lemma 4.4 are the linearization matrices that correspond to the different tabular Q-learning algorithms:

$$A = -\Pi[I - \beta PS_{\phi^*}] \quad \text{in case (i) of Lemma 4.4} \quad (4.93a)$$

$$A = -[I - \beta PS_{\phi^*}] \quad \text{in case (ii) of Lemma 4.4.} \quad (4.93b)$$

Since  $PS_{\phi^*}$  is a transition probability matrix of an irreducible Markov chain (see Lemma 4.2), it follows that both matrices are Hurwitz.

It was remarked in Sect. 4.2.5 that the CLT and rates of convergence for the MSE that were obtained for the linear recursions of the form  $\bar{f}(\theta) = A\theta - b$  can be extended to general Lipschitz continuous functions, but such an extension requires tedious computations. An example where it is not so tedious is the analysis of the

tabular Q-learning algorithm [64]. In [64], an approximation for the SA recursion (4.84) is introduced, that is truly linear:

$$\theta_{n+1}^* = \theta_n^* + \alpha_{n+1} [c(X_n, U_n) + \beta Q^{\theta_n^*}(X_{n+1}, \phi^*(X_{n+1})) - Q^{\theta_n^*}(X_n, U_n)] \psi(X_n, U_n). \quad (4.94)$$

Asymptotic error bounds for the original SA recursion (4.84) were then obtained by establishing that  $\|\theta_n - \theta_n^*\|$  vanishes *quickly* as  $n \rightarrow \infty$ .

Consider the step-size  $\alpha_n = g[n(X_n, U_n)]^{-1}$ , with  $g > 0$  a constant. Recall from our previous discussion around (4.87) and (4.89), that this is equivalent to the matrix gain version of the asynchronous algorithm, with matrix gain (4.89), and step-size  $\alpha_n = g/n$ . Lemma 4.4 implies that the resulting linearization matrix for (4.94) is  $gA$ , with  $A$  defined in (4.93b). Suppose we choose  $g > 0$  such that  $\frac{1}{2}I + gA$  is Hurwitz. It then follows that the CLT will hold for  $\{\theta_n^*\}$ , following arguments in [3]. Convergence of the normalized MSE (4.18) follows from Theorem 4.4, since (4.94) is a linear SA recursion. For the (nonlinear) asynchronous Q-learning algorithm (4.84), with the same step-size and matrix gain, the following conclusions hold:

- (i) The CLT holds provided  $\lim_{n \rightarrow \infty} \sqrt{n}\|\theta_n - \theta_n^*\| = 0$  a.s..
- (ii) The limit (4.18) holds provided  $\lim_{n \rightarrow \infty} n\mathbb{E}[\|\theta_n - \theta_n^*\|^2] = 0$ .

The first coupling bound can be obtained following the arguments in [64], and the second holds in this context using the same arguments.

We have not verified (i) or (ii) for all of the algorithms discussed in this chapter. Rather, we design algorithms for which the linearized algorithm satisfies the eigenvalue conditions for the limit (4.18), with  $\Sigma_\theta$  finite. We begin by providing general conditions under which this limit is *not* finite.

**Theorem 4.6** *Suppose that assumptions (Q1) and (Q2) hold, and  $\alpha_n \equiv 1/n$ . Then, the sequence of parameters  $\{\theta_n\}$  obtained using the asynchronous Q-learning algorithm (4.84) converges to  $Q^*$  a.s.. Suppose moreover that the conditional variance of  $h^*(X_n)$  is positive:*

$$\sum_{x, x', u} \varpi(x, u) P_u(x, x') [h^*(x') - P_u h^*(x)]^2 > 0 \quad (4.95)$$

and

$$(1 - \beta) \max_{x, u} \varpi(x, u) < \frac{1}{2}. \quad (4.96)$$

Then,

- (i) *The asymptotic variance of the algorithm is infinite:*

$$\lim_{n \rightarrow \infty} n\mathbb{E}[\|\theta_n - \theta^*\|^2] = \infty,$$

- (ii)  *$\mathbb{E}[\|\theta_n - \theta^*\|^2]$  converges to zero at a rate no faster than  $1/n^{2(1-\beta)}$ .  $\square$*

The inequality (4.96) is satisfied whenever the discount factor satisfies  $\beta \geq \frac{1}{2}$ .

Theorem 4.6 explains why the Q-learning algorithm can be terribly slow: If the discount factor is close to 1, which is typical in most applications, using a step-size of the form  $\alpha_n = 1/n$  results in an MSE convergence rate that is much slower than the optimal rate  $1/n$ .

Similar conclusions hold for the other flavors of tabular Q-learning, for which the algorithm admits the ODE approximation (4.88). Based on Lemma 4.4, the linearization matrix for these algorithms is defined in (4.93b). This poses problems when  $\beta > \frac{1}{2}$ , but for these algorithms there is a simple remedy:

**Theorem 4.7** *For asynchronous Q-learning with the step-size rule (4.87), or synchronous Q-learning with step-size  $\alpha_n = 1/n$ , the matrix shown in (4.93b) is equal to the linearization matrix  $A = \partial_\theta f(\theta)|_{\theta=\theta^*}$ . It has one eigenvalue  $\lambda = -(1 - \beta)$ , and  $\text{Re}(\lambda(A)) \leq -(1 - \beta)$  for every other eigenvalue. Consequently,*

- (i) *Subject to (4.95), the asymptotic variance is infinite whenever  $\beta > \frac{1}{2}$*
- (ii) *Suppose that the step-sizes are scaled: use  $\alpha_n(x, u) = [(1 - \beta)n(x, u)]^{-1}$  for asynchronous Q-learning, or  $\alpha_n = [(1 - \beta)n]^{-1}$  for synchronous Q-learning (4.91). Then the eigenvalue test passes: for each eigenvalue  $\lambda = \lambda(A)$ ,*

$$\text{Re}(\lambda) = \text{Re}(\lambda(-(1 - \beta)^{-1}[I - \beta P S_{\phi^*}])) \leq -1.$$

□

The step-size rule  $\alpha_n = [(1 - \beta)n]^{-1}$  is equivalent to  $\alpha_n = [1 + (1 - \beta)n]^{-1}$  appearing in (4.9), in the sense that each algorithm will share the same asymptotic covariance.

**Overview of proofs:** We begin with Theorem 4.6. The proof of convergence can be found in [9, 67, 75]. The proof of infinite asymptotic variance is given in Appendix A.2 of [21], and relies on the theory developed for general SA algorithms in Sect. 4.2.5 (in particular, Corollary 4.1 to Theorem 4.4). A brief overview follows.

To establish the slow convergence rate, an eigenvector for  $A$  (defined in (4.93a)) is constructed with strictly positive entries, and with real parts of the eigenvalues satisfying  $\text{Real}(\lambda) \geq -1/2$ . Interpreted as a function  $v: \mathcal{X} \times \mathcal{U} \rightarrow \mathbb{C}$ , this eigenvector satisfies

$$v^\dagger \Sigma_\Delta v = \beta^2 \sum_{x, x', u} \varpi(x, u) |v(x, u)|^2 P_u(x, x') [h^*(x') - P_u h^*(x)]^2, \quad (4.97)$$

where  $\Sigma_\Delta$  is the noise covariance matrix (recall (4.46)). Assumption (4.95) ensures that the right hand side of (4.97) is strictly positive, as required in Corollary 4.1.

Theorem 4.7 is based on the simple structure of the eigenvalues of the linearization matrix  $A = -[I - \beta P S_{\phi^*}]$  defined in (4.93b). Because  $P S_{\phi^*}$  is the transition matrix for an irreducible Markov chain, it follows that all of its eigenvalues are in the closed unit disk in the complex plane, with a single eigenvalue at  $\lambda = 1$ . Consequently, A

has a single eigenvalue at  $\lambda = -(1 - \beta)$ , and  $\text{Re}(\lambda(A)) < -(1 - \beta)$  for all other eigenvalues. This implies both (i) and (ii) of the theorem.  $\square$

Theorems 4.6 and 4.7 motivate the introduction of *Zapped Q*-learning algorithms.

### 4.3.6 Zap Q-Learning

The special case of  $G\text{-}Q(\lambda)$  (4.78) based on Zap SA (4.56) is called the *Zap-Q*( $\lambda$ ) algorithm. As in (4.56), the step-size sequences  $\alpha$  and  $\gamma$  in the algorithm are assumed to satisfy (4.33) and (4.36):

$$\sum_n \alpha_n = \sum_n \gamma_n = \infty, \quad \sum_n \alpha_n^2 + \gamma_n^2 < \infty, \quad \lim_{n \rightarrow \infty} \frac{\alpha_n}{\gamma_n} = 0. \quad (4.98)$$

#### Zap-Q ( $\lambda$ ) Algorithm

**Input:**  $\theta_0 \in \mathbb{R}^d$ ,  $\zeta_0 = \psi(X_0, U_0)$ ,  $\widehat{A}_0 \in \mathbb{R}^{d \times d}$ ,  $n = 0$ ,  $T \in \mathbf{Z}^+$   $\triangleright$  Initialization

- 1: **repeat**
- 2:    $\phi_n(X_{n+1}) := \arg \min_u Q^{\theta_n}(X_{n+1}, u);$
- 3:    $d_{n+1} := c(X_n, U_n) + \beta Q^{\theta_n}(X_{n+1}, \phi_n(X_{n+1})) - Q^{\theta_n}(X_n, U_n);$   $\triangleright$  TD term
- 4:    $A_{n+1} := \zeta_n [\beta \psi(X_{n+1}, \phi_n(X_{n+1})) - \psi(X_n, U_n)]^\top;$
- 5:    $\widehat{A}_{n+1} = \widehat{A}_n + \gamma_{n+1} [A_{n+1} - \widehat{A}_n];$   $\triangleright$  Matrix gain update rule
- 6:    $\theta_{n+1} = \theta_n - \alpha_{n+1} \widehat{A}_{n+1}^{-1} \zeta_n d_{n+1};$   $\triangleright$  Zap-Q update rule
- 7:    $\zeta_{n+1} := \lambda \beta \zeta_n + \psi(X_{n+1}, U_{n+1});$   $\triangleright$  Eligibility vector update rule
- 8:    $n = n + 1$
- 9: **until**  $n \geq T$   $\square$

---

It is assumed that a projection is employed to ensure that  $\{\widehat{A}_n^{-1}\}$  is a bounded sequence—this is most easily achieved using the matrix inversion lemma (Sherman–Morrison–Woodbury formula). See the end of Sect. 4.2.6.2 for alternative techniques.

The analysis that follows is specialized to  $\lambda = 0$ , and the basis defined in (4.80) that is used in Watkins’ algorithm. The resulting *Zap-Q* algorithm is defined as follows: after identifying  $Q^\theta$  with  $\theta$ :

$$\theta_{n+1} = \theta_n + \alpha_{n+1} \widehat{G}_{n+1}^* [c(X_n, U_n) + \beta \underline{\theta}_n(X_{n+1}) - \theta_n(X_n, U_n)] \psi(X_n, U_n) \quad (4.99a)$$

$$\widehat{A}_{n+1} = \widehat{A}_n + \gamma_{n+1} [A_{n+1} - \widehat{A}_n] \quad (4.99b)$$

$$A_{n+1} = \psi(X_n, U_n) [\beta \psi(X_{n+1}, \phi_n(X_{n+1})) - \psi(X_n, U_n)]^\top$$

$$\phi_n(X_{n+1}) = \phi^{\theta_n}(X_{n+1}) = \arg \min_u Q^{\theta_n}(X_{n+1}, u),$$

where  $\widehat{G}_n^* = -[\widehat{A}_n]^{-1}$ ,  $[\cdot]$  denoting a projection operator chosen so that  $\{\widehat{G}_n^*\}$  is a bounded sequence. In Theorem 4.8, it is established that the projection is required only for a finite number of iterations:  $\{\widehat{A}_n^{-1} : n \geq n_\bullet\}$  is a bounded sequence, where  $n_\bullet < \infty$  a.s..

An equivalent representation for the parameter recursion (4.99a) is

$$\theta_{n+1} = \theta_n + \alpha_{n+1} \widehat{G}_n^* \{ \Psi_n c + A_{n+1} \theta_n \}, \quad (4.100)$$

in which  $c$  and  $\theta_n$  are treated as  $d$ -dimensional vectors, and  $\Psi_n = \psi(X_n, U_n) \psi(X_n, U_n)^\top$ . It would seem that the analysis is complicated by the fact that the sequence  $\{A_n\}$  depends upon  $\{\theta_n\}$  through the policy sequence  $\{\phi_n\}$ . Part of the analysis is simplified by obtaining a recursion for the following  $d$ -dimensional sequence:

$$\widehat{C}_n = -\Pi^{-1} \widehat{A}_n \theta_n, \quad n \geq 1. \quad (4.101)$$

This admits a very simple recursion in the special case  $\gamma \equiv \alpha$ . In the other case considered, wherein the step-size sequence  $\gamma$  satisfies (4.98), the recursion for  $\widehat{C}$  is more complex, but the ODE analysis is simplified (see [21, Proposition 3.7]) for discussion.

#### 4.3.6.1 Main Results

Conditions for convergence of the Zap-Q algorithm (4.98) are summarized in Theorem 4.8. The following assumption (Q3) is used to address the discontinuity in the recursion for  $\{\widehat{A}_n\}$  resulting from the dependence of  $A_{n+1}$  on  $\phi_n$ :

**(Q3)** The sequence of policies  $\{\phi_n\}$  satisfies:

$$\sum_{n=1}^{\infty} \gamma_n \mathbb{I}\{\phi_{n+1} \neq \phi_n\} < \infty, \quad a.s.. \quad (4.102)$$

□

**Theorem 4.8** Suppose that assumptions (Q1)–(Q3) hold, with the gain sequences  $\alpha$  and  $\gamma$  satisfying

$$\alpha_n = 1/n, \quad \gamma_n = 1/n^\rho, \quad n \geq 1 \quad (4.103)$$

for some fixed  $\rho \in (\frac{1}{2}, 1)$ . Then,

- (i) The parameter sequence  $\theta$  obtained using the Zap-Q algorithm (4.98) converges to  $Q^*$  a.s..
- (ii) The asymptotic covariance (4.18) is minimized over all G-Q(0) versions of Watkins' Q-learning algorithm (4.78).
- (iii) An ODE approximation holds for the sequence  $\{\theta_n, \hat{C}_n\}$ , by continuous functions  $(q, \varsigma)$  satisfying

$$q_t = Q(\varsigma_t), \quad \frac{d}{dt} \varsigma_t = -\varsigma_t + c, \quad (4.104)$$

This ODE approximation is exponentially asymptotically stable, with

$$\lim_{t \rightarrow \infty} q_t = Q^*.$$

□

Section 4.3.6.3 contains an overview of the proof, where we also discuss the precise meaning of the ODE approximation (4.104). The ODE analysis in the proof of Theorem 4.8 suggests that the dynamics of the Zap-Q algorithm closely matches the Newton–Raphson ideal (4.55). Moreover, the algorithm has the best performance in all of the numerical experiments surveyed in Sect. 4.4.

#### 4.3.6.2 Zap ODE and Policy Iteration

The ODE approximation (4.104) can be expressed

$$\frac{d}{dt} q_t = -q_t + \partial Q_{\mu_t} c, \quad (4.105)$$

where  $\partial Q_{\mu_t}$  is defined in (4.74),  $\mu_t$  is any pmf satisfying  $\partial Q_{\mu_t} \varsigma_t = q_t$ , and the derivative exists for a.e.  $t$  (see Lemma A.10 of [21] for full justification). This has an interesting geometric interpretation. Without loss of generality, assume that the cost function is non-negative, so that  $q$  evolves in the positive orthant  $\mathbb{R}_+^d$  whenever its initial condition lies in this domain.

A typical solution to the ODE is shown in Fig. 4.2: the trajectory is piecewise linear, with changes in a direction corresponding to changes in the policy  $\phi^{q_t}$ . Each set  $\Theta^k$  shown in the figure corresponds to a deterministic policy:

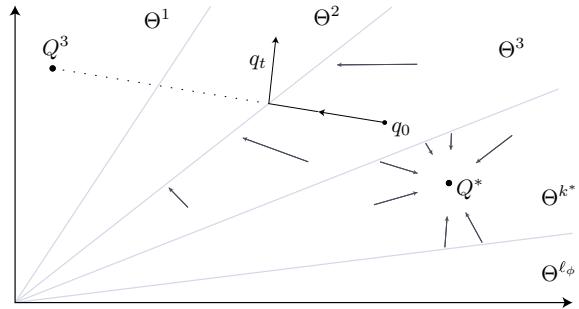
$$\Theta^k = \{q \in \mathbb{R}_+^d : \phi^q = \phi^{(k)}\}.$$

**Lemma 4.5** For each  $k$ , the set  $\Theta^k$  is a convex polyhedron, and also a positive cone. When  $q_t \in \text{interior}(\Theta^k)$  then

$$\partial Q_{\mu_t} c = Q^k := Q_{\phi^{(k)}} = c + \beta P h_{\phi^{(k)}}.$$

□

**Fig. 4.2** ODE for Zap Q-Learning. The light arrows show typical vectors in the vector field that defines the ODE (4.105). The solution starting at  $q_0 \in \Theta^3$  initially moves in a straight line towards  $Q^3$



**Proof** The power series expansion holds:

$$\partial Q_{\mu_t} c = [I - \beta P S_{\phi_t}]^{-1} c = c + \sum_{n=1}^{\infty} \beta^n [P S_{\phi_t}]^n c,$$

where  $\phi_t = \phi^{(k)}$ . For each  $n \geq 1$ , we have  $[P S_{\phi}]^n = P P_{\phi}^{n-1} S_{\phi}$ , which together with (4.66a) implies the desired result.

The function  $Q^k$  is the fixed-policy Q-function considered in the SARSA algorithm [45, 56, 65]. While  $q_t$  evolves in the interior of the set  $\Theta^k$ , it moves in a straight line toward the function  $Q^k$ . On reaching the boundary, it then moves in a straight line to the next Q-function. This is something like a policy iteration recursion since the policy  $\phi^{q_t}$  is obtained as the argmin over  $u$  of  $q_t(\cdot, u)$ .

Of course, it is far easier to establish stability of the equivalent ODE (4.104).  $\square$

#### 4.3.6.3 Overview of Proofs

It is assumed throughout the remainder of this section that assumptions (Q1)–(Q3) hold. Proofs of technical results are contained in [21].

We require the usual probabilistic foundations: There is a probability space  $(\Omega, \mathcal{F}, \mathbb{P})$  that supports all random variables under consideration. The probability measure  $\mathbb{P}$  may depend on initialization of the Markov chain. All stochastic processes under consideration are assumed adapted to a filtration denoted  $\{\mathcal{F}_n : n \geq 0\}$ .

The construction of an approximating ODE involves first defining a continuous time process. Denote

$$t_n = \sum_{i=1}^n \alpha_i, \quad n \geq 1, \quad t_0 = 0, \quad (4.106)$$

and define  $\bar{q}_{t_n} = \theta_n$  for these values, with the definition extended to  $\mathbb{R}_+$  via linear interpolation. For some function  $\bar{f} : \mathbb{R}^d \rightarrow \mathbb{R}^d$ , we say that the ODE approximation  $\frac{d}{dt} q_t = \bar{f}(q_t)$  holds, if we have the approximation,

$$\bar{q}_{T_0+t} = \bar{q}_{T_0} + \int_{T_0}^{T_0+t} \bar{f}(\bar{q}_\tau) d\tau + \mathcal{E}_{T_0, T_0+t}, \quad t, T_0 \geq 0 \quad (4.107)$$

where the error process satisfies, for each  $T > 0$ :

$$\lim_{T_0 \rightarrow \infty} \sup_{0 \leq t \leq T} \|\mathcal{E}_{T_0, T_0+t}\| = 0 \quad a.s.. \quad (4.108)$$

Such approximations will be represented using the more compact notation:

$$\bar{q}_{T_0+t} = \bar{q}_{T_0} + \int_{T_0}^{T_0+t} \bar{f}(\bar{q}_\tau) d\tau + o(1), \quad T_0 \rightarrow \infty. \quad (4.109)$$

An ODE approximation holds for Watkins' Q-learning algorithm, with  $\bar{f}(q_t)$  defined by the right-hand side of (4.86), or in more compact notation:

$$\frac{d}{dt} q_t = \Pi[c - \varsigma_t], \quad \varsigma_t = [I - \beta P S_{\phi^n}] q_t. \quad (4.110)$$

The significance of this representation is that  $q_t$  is the Q-function associated with the “cost function”  $\varsigma_t$ :  $q_t = Q(\varsigma_t)$ .

The same notation will be used in the following treatment of Zap Q-learning. Along with the piece-wise linear continuous-time process  $\{\bar{q}_t : t \geq 0\}$ , denote by  $\{\bar{\mathcal{A}}_t : t \geq 0\}$  the piece-wise linear continuous-time process defined similarly, with  $\bar{\mathcal{A}}_{t_n} = \bar{A}_n$ ,  $n \geq 1$ , and  $\bar{c}_t = Q^{-1}(\bar{q}_t)$  for  $t \geq 0$ .

To construct an ODE, it is convenient first to obtain an alternative and suggestive representation for the pair of equations (4.99a), (4.99b). A vector-valued sequence of random variables  $\{\mathcal{E}_k\}$  will be called *ODE-friendly* if it admits the decomposition,

$$\mathcal{E}_k = \Delta_k + \mathcal{T}_k - \mathcal{T}_{k-1} + \varepsilon_k, \quad k \geq 1 \quad (4.111)$$

in which  $\{\Delta_k : k \geq 1\}$  is a martingale-difference sequence satisfying

$$\mathbb{E}[\|\Delta_{k+1}\|^2 | \mathcal{F}_k] \leq \bar{\sigma}_\Delta^2 \quad a.s.$$

for some  $\bar{\sigma}_\Delta^2 < \infty$  and all  $k$ ,  $\{\mathcal{T}_k : k \geq 1\}$  is a bounded sequence, and  $\{\varepsilon_k\}$  is a bounded sequence and satisfies

$$\sum_{k=1}^{\infty} \gamma_k \|\varepsilon_k\| < \infty \quad a.s.. \quad (4.112)$$

**Lemma 4.6** *The pair of equations (4.99a), (4.99b) can be expressed,*

$$\theta_{n+1} = \theta_n + \alpha_{n+1} \widehat{G}_{n+1}^* \left[ -\Pi[I - \beta P S_{\phi^{\theta_n}}] \theta_n + \Pi c + \mathcal{E}_{n+1}^A \theta_n + \mathcal{E}_{n+1}^q \right] \quad (4.113a)$$

$$\widehat{G}_{n+1}^* = -[\widehat{A}_{n+1}]^{-1} \quad (4.113b)$$

$$\widehat{A}_{n+1} = \widehat{A}_n + \gamma_{n+1} \left[ -\Pi[I - \beta P S_{\phi^{\theta_n}}] - \widehat{A}_n + \mathcal{E}_{n+1}^A \right], \quad (4.113c)$$

in which the sequence  $\{\mathcal{E}_n^q : n \geq 1\}$  is ODE-friendly, and  $\{\mathcal{E}_n^A\}$  is ODE-friendly under assumption (Q3).  $\square$

The assertion that  $\{\mathcal{E}_n^q, \mathcal{E}_n^A\}$  are ODE-friendly follows from standard arguments based on solutions to Poisson's equation for zero-mean functions of the Markov chain  $(X, U)$ . This idea first appeared in [42], and has been a basic tool ever since [3, 40].

The representation in Lemma 4.6 appears similar to an Euler approximation of the solution to an ODE:

$$\begin{pmatrix} \theta_{n+1} \\ \widehat{A}_{n+1} \end{pmatrix} = \begin{pmatrix} \theta_n \\ \widehat{A}_n \end{pmatrix} + \begin{pmatrix} \alpha_{n+1} \bar{f}_\Theta(\theta_n, \widehat{A}_n) \\ \gamma_{n+1} \bar{f}_A(\theta_n, \widehat{A}_n) \end{pmatrix} + \mathcal{E}_{n+1}. \quad (4.114)$$

It is the discontinuity of the function  $\bar{f}_A$  that presents the most significant challenge in analysis of the algorithm—this violates standard conditions for existence and uniqueness of solutions to the ODE without disturbance. Fortunately, there is a special structure that will allow the construction of an ODE approximation. Some of this structure is highlighted in Lemma 4.7.

**Lemma 4.7** *For each  $t, T_0 \geq 0$ ,*

$$\bar{q}_{T_0+t} = \bar{q}_{T_0} - \int_{T_0}^{T_0+t} \bar{\mathcal{A}}_\tau^{-1} \Pi \{c - \bar{c}_\tau\} d\tau + o(1) \quad (4.115)$$

$$\bar{\mathcal{A}}_{T_0+t} = \bar{\mathcal{A}}_{T_0} - \int_{T_0}^{T_0+t} \{\Pi[I - \beta P S_{\phi^{\bar{q}_\tau}}] + \bar{\mathcal{A}}_\tau\} g_\tau d\tau + o(1), \quad T_0 \rightarrow \infty. \quad (4.116)$$

where  $g_t := \gamma_n / \alpha_n$  when  $t = t_n$ , and the definition is extended to all  $t \in \mathbb{R}_+$  by linear interpolation.  $\square$

The “gain”  $g_t$  appearing in (4.116) converges to infinity rapidly as  $t \rightarrow \infty$ : Based on the definitions in (4.103), it follows from (4.106) that  $t_n \approx \log(n)$  for large  $n$ , and consequently  $g_t \approx \exp((1 - \rho)t)$  for large  $t$ . This suggests that the integrand  $\Pi[I - \beta P S_{\phi^{\bar{q}_t}}] + \bar{\mathcal{A}}_t$  should converge to zero rapidly as  $t \rightarrow \infty$ . This intuition is made precise in the full proofs contained in [21]. Through several subsequent transformations, these integral equations are shown to imply the ODE approximation in Theorem 4.8.

## 4.4 Numerical Results

Results from numerical experiments are surveyed here, with focus on the Zap Q-learning algorithm (4.99a), (4.99b). Comparisons are made with several existing algorithms, including Watkins' Q-learning (4.84), Watkins' Q-learning with Ruppert–Polyak–Juditsky (RPJ) averaging [53, 54, 58] (see Sect. 4.2.6), Watkins' Q-learning with a “polynomial learning rate” [26], and the *Speedy Q-learning* algorithm [2].

In addition, the Watkins' algorithm with a step-size  $\alpha_n = g/n$  is considered, with  $g$  chosen so that the algorithm has finite asymptotic covariance. When the value of  $g$  is optimized and numerical conditions are favorable (e.g., the condition number of  $A$  is not too large) it is found that the performance is nearly as good as the Zap-Q algorithm. However, there is no free lunch:

- (i) Design of the scalar gain  $g$  depends on the approximation of  $A$ , and hence  $\theta^*$ . While it is possible to estimate  $A$  via Monte Carlo in Zap Q-learning, it is not known how to efficiently update approximations for an optimal scalar gain.
- (ii) A reasonable asymptotic covariance required a large value of  $g$ . Consequently, the scalar gain algorithm had massive transients, resulting in a poor performance in practice.
- (iii) Transient behavior could be tamed through projection to a bounded set. However, this again requires prior knowledge of the region in the parameter space to which  $\theta^*$  belongs.

Projection of parameters was also necessary for RPJ averaging.

The following batch mean method was used to estimate the asymptotic covariance.

**Batch Mean Method:** At stage  $n$  of the algorithm we are interested in the distribution of a vector-valued random variable of the form  $F_n(\theta_n)$ , where  $F_n: \mathbb{R}^d \rightarrow \mathbb{R}^m$  is possibly dependent on  $n$ . The batch mean method is used to estimate its statistics: For each algorithm,  $N$  parallel simulations are run with  $\theta_0$  initialized i.i.d. according to some distribution. Denoting  $\theta_n^i$  to be the vector  $\theta_n$  corresponding to the  $i^{th}$  simulation, the distribution of the random variable  $F_n(\theta_n)$  is estimated based on the histogram of the independent samples  $\{F_n(\theta_n^i) : 1 \leq i \leq N\}$ .

An important special case is  $F_n(\theta_n) = W_n = \sqrt{n}(\theta^n - \theta^*)$ . However, since the limit  $\theta^*$  is not available, the empirical mean is substituted:

$$W_n^i = \sqrt{n}[\theta_n^i - \bar{\theta}_n], \quad \bar{\theta}_n := \frac{1}{N} \sum_{i=1}^N \theta_n^i. \quad (4.117)$$

The estimate of the covariance of  $F_n(\theta_n)$  is then obtained as the sample covariance of  $\{W_n^i, 1 \leq i \leq N\}$ . This corresponds to the estimate of the asymptotic covariance  $\Sigma_\theta$  defined in (4.18).

The value  $N = 10^3$  is used in all of the experiments surveyed here.

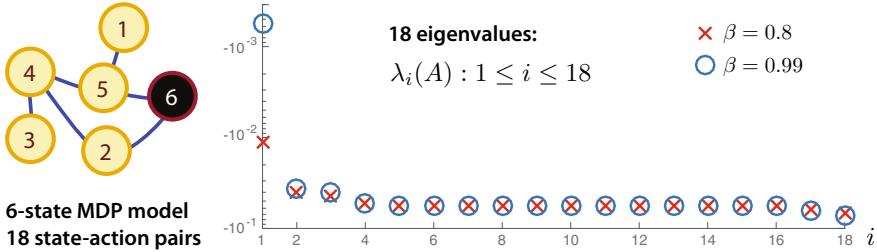


Fig. 4.3 Graph for MDP and eigenvalues of  $A$

#### 4.4.1 Finite State-Action MDP

Consider first a simple stochastic shortest path problem. The state space  $X = \{1, \dots, 6\}$  coincides with the six nodes on the undirected graph shown on the left-hand side of Fig. 4.3. The action space  $U = \{e_{x,x'}\}, x, x' \in X$ , consists of all feasible edges along which an agent can travel, including each “self-loop”,  $u = e_{x,x}$ . The number of state–action pairs for this example coincides with the number of nodes plus twice the number of edges:  $d = 18$ .

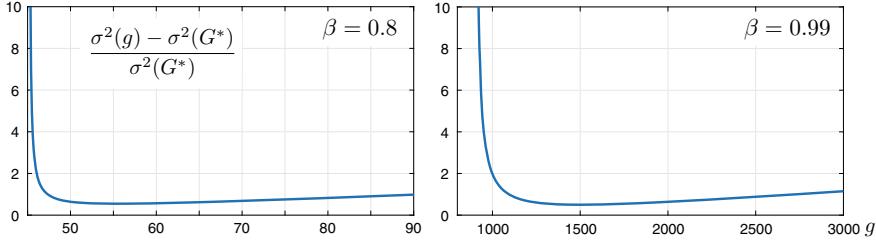
The controlled transition matrix is defined as follows: If  $X_n = x \in X$ , and  $U_n = e_{x,x'} \in U$ , then  $X_{n+1} = x'$  with probability 0.8, and with probability 0.2, the next state is randomly chosen between all neighboring nodes. The goal is to reach the state  $x^* = 6$  and maximize the time spent there. This is modeled through a discounted-reward optimality criterion with discount factor  $\beta \in (0, 1)$ . The one-step reward is defined as follows:

$$r(x, u) = \begin{cases} 0 & u = e_{x,x}, x \neq 6 \\ 100 & u = e_{x,6} \\ -100 & u = e_{4,5} \\ -5 & \text{otherwise.} \end{cases} \quad (4.118)$$

The solution to the discounted-cost optimal control problem can be computed numerically for this model; the optimal policy is unique and independent of  $\beta$ .

Six different variants of Q-learning were tested: Watkins’ algorithm with scalar gain  $g$ , so that  $\alpha_n \equiv g/n$ , Watkins’ algorithm using RPJ averaging, with  $\gamma_n \equiv (\alpha_n)^{0.6} \equiv n^{-0.6}$ , Watkins’ algorithm with the polynomial learning rate  $\alpha_n \equiv n^{-0.6}$ , Speedy Q-learning, Zap Q-learning with  $\alpha \equiv \gamma$ , and Zap Q-learning with  $\gamma_n \equiv (\alpha_n)^{0.85} \equiv n^{-0.85}$ . The basis was taken to be the same as in Watkins Q-learning algorithm. In each case, the randomized policy was taken to be uniform: feasible transitions were sampled uniformly at each time.

The randomized exploratory policy was chosen to be uniform over all feasible actions. For example, if the current state is  $X_n = 4$ , then  $U_n$  takes the value  $e_{4,2}$ ,  $e_{4,3}$ , or  $e_{4,5}$  with equal probability. The steady state distribution of the Markov chain  $(X, U)$  is uniform in this case:  $\Pi(i, i) = 1/18$  for each  $1 \leq i \leq 18$ .



**Fig. 4.4** Normalized MSE  $\sigma^2(g) = \text{trace}(\Sigma_\theta)$  for a range of scalar gains  $g$

Discount factors  $\beta = 0.8$  and  $\beta = 0.99$  were considered. In each case, the unique optimal parameter  $\theta^* = Q^*$  was obtained numerically.

**Asymptotic Covariance:** Speedy Q-learning cannot be represented as a standard stochastic approximation algorithm, so standard theory cannot be applied to obtain its asymptotic covariance. The Watkins' algorithm with polynomial learning rate has infinite asymptotic covariance.

For the other four algorithms, the asymptotic covariance  $\Sigma_\theta$  was computed by solving the Lyapunov equation (4.51) based on the matrix gain  $G$  that is particular to each algorithm. Recall that  $G = -A^{-1}$  in the case of either of the Zap-Q algorithms.

The matrices  $A$  and  $\Sigma_\Delta$  appearing in (4.51) are defined with respect to Watkins' Q-learning algorithm with  $\alpha_n = 1/n$ . The first matrix is  $A = -\Pi[I - \beta P S_{\phi^*}]$  under the standing assumption that the optimal policy is unique. The proof that this is a linearization comes first from the representation of the ODE approximation (4.86) in vector form:

$$\frac{d}{dt} q_t = \bar{f}(q_t) = A_\phi q_t - b, \quad \text{where } A_\phi = -\Pi[I - \beta P S_\phi], \quad b = -\Pi c. \quad (4.119)$$

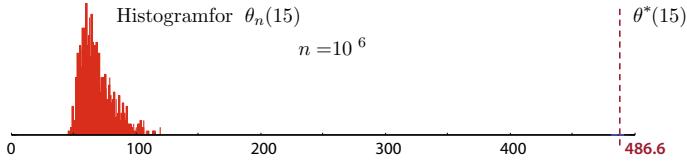
Uniqueness of the optimal policy implies that  $\bar{f}$  is locally linear: there exists  $\varepsilon > 0$  such that

$$\bar{f}(\theta) - \bar{f}(\theta^*) = A(\theta - \theta^*), \quad \|\theta - \theta^*\| \leq \varepsilon.$$

The matrix  $\Sigma_\Delta$  was also obtained numerically, without resorting to simulation.

The eigenvalues of the  $18 \times 18$  matrix  $A$  are real in this example: they are plotted in Fig. 4.3 for each value of  $\beta$ . To ensure that the eigenvalues of  $gA$  are all strictly less than  $-1/2$  in a scalar gain algorithm requires the lower bounds  $g > 45$  for  $\beta = 0.8$ , and  $g > 900$  for  $\beta = 0.99$ . Theorem 4.5 implies that the asymptotic covariance  $\Sigma_\theta(g)$  is finite for this range of  $g$  in the Watkins algorithm with  $\alpha_n \equiv g/n$ . Remarkably, the theorem also implies that if  $\alpha_n = 1/n$ , then we can expect  $\text{trace}(\Sigma_\theta) = \infty$  whenever  $\beta > 1/36$ . Fig. 4.4 shows the normalized trace of the asymptotic covariance as a function of  $g > 0$ , and the significance of  $g \approx 45$  and  $g \approx 900$ .

Based on this analysis, or on Theorem 4.6, it follows that the asymptotic covariance is not finite for the standard Watkins' algorithm with  $\alpha_n \equiv 1/n$ . In simulations, it was found that the parameter estimates are not close to  $\theta^*$  even after many millions of



**Fig. 4.5** Histogram of  $10^3$  estimates of  $\theta_n(15)$ , with  $n = 10^6$  for the Watkins algorithm applied to the six-state example with discount factor  $\beta = 0.8$

samples. This is illustrated for the case  $\beta = 0.8$  in Fig. 4.5, which shows a histogram of  $10^3$  estimates of  $\theta_n(15)$  with  $n = 10^6$  (other entries showed similar behavior).

It was found that the algorithm performed very poorly in practice for any scalar gain algorithm. For example, more than half of the  $10^3$  experiments using  $\beta = 0.8$  and  $g = 70$  resulted in values of  $\theta_n(15)$  exceeding  $\theta^*(15)$  by  $10^4$  (with  $\theta^*(15) \approx 500$ ), even with  $n = 10^6$ . The algorithm performed well with the introduction of projection in the case  $\beta = 0.8$ . *With  $\beta = 0.99$ , the performance was unacceptable for any scalar gain, even with projection.*

The next set of results provide variance estimates for Watkins algorithm in the special case of  $\beta = 0.8$ , with gain  $g = 70$ , and with projection of each parameter estimate to the interval  $(-\infty, 1000]$ . The first row of Fig. 4.6 shows histograms of  $\{W_n^i(k) : 1 \leq i \leq N\}$ , as defined in (4.117), for  $k = 10$  and  $18$ . The Central Limit Theorem holds:  $W_n$  is expected to be approximately normally distributed:  $\mathcal{N}(0, \Sigma_\theta(g))$ , when  $n$  is large. Of the  $d = 18$  entries of the vector  $W_n$ , with  $n \geq 10^4$ , it was found that the asymptotic variance matched the histogram nearly perfectly for  $k = 10$ , while  $k = 18$  showed the worst fit.

These experiments were repeated for each of the Zap-Q algorithms, for which the asymptotic covariance  $\Sigma_\theta^*$  is obtained using the formula (4.52):

$$\Sigma_\theta^* := A^{-1} \Sigma_\Delta (A^{-1})^\top.$$

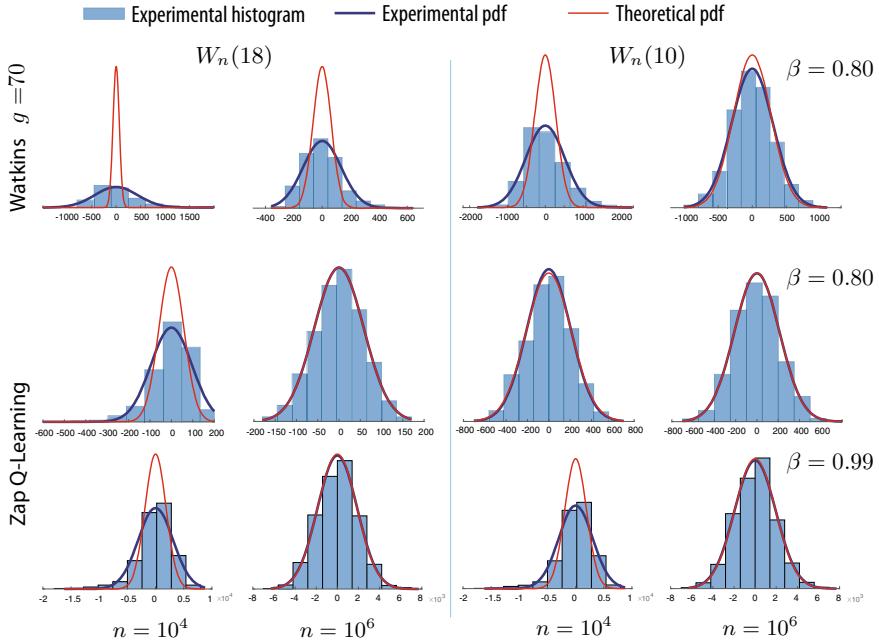
Plots are shown in the second and third rows of Fig. 4.6 for Zap Q-learning, with  $\gamma_n = (\alpha_n)^{0.85} = 1/n^{0.85}$ . Data in the third row was obtained with the discount factor  $\beta = 0.99$ . The covariance estimates and the Gaussian approximations match the theoretical predictions remarkably well for  $n \geq 10^4$ .

**Bellman Error:** The Bellman error at iteration  $n$  is denoted

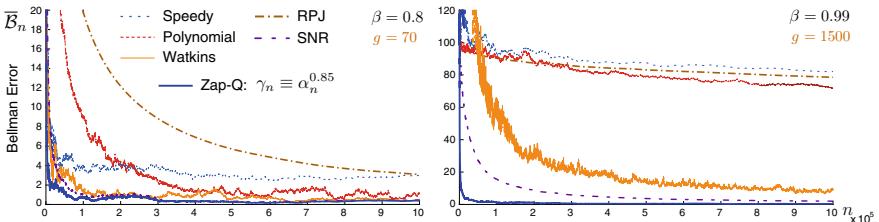
$$\mathcal{B}_n(x, u) = \theta_n(x, u) - r(x, u) - \beta \sum_{x' \in \mathcal{X}} P_u(x, x') \max_{u'} \theta_n(x', u'). \quad (4.120)$$

This is identically zero if and only if  $\theta_n = Q^*$ . If  $\{\theta_n\}$  converges to  $Q^*$ , then  $\mathcal{B}_n = \tilde{\theta}_n - \beta P S_{\phi^*} \tilde{\theta}_n$  for all sufficiently large  $n$ , and the CLT holds for  $\{\mathcal{B}_n\}$  whenever it holds for  $\{\theta_n\}$ . Moreover, on denoting the maximal error

$$\bar{\mathcal{B}}_n = \max_{x, u} |\mathcal{B}_n(x, u)|, \quad (4.121)$$



**Fig. 4.6** Comparison of theoretical and empirical asymptotic variances for Q-learning applied to the six-state example. Row 1: Watkins' algorithm with gain  $g = 70$ ,  $\beta = 0.8$ . Row 2: Zap Q-learning with  $\beta = 0.8$ . Row 3: Zap Q-learning with  $\beta = 0.99$

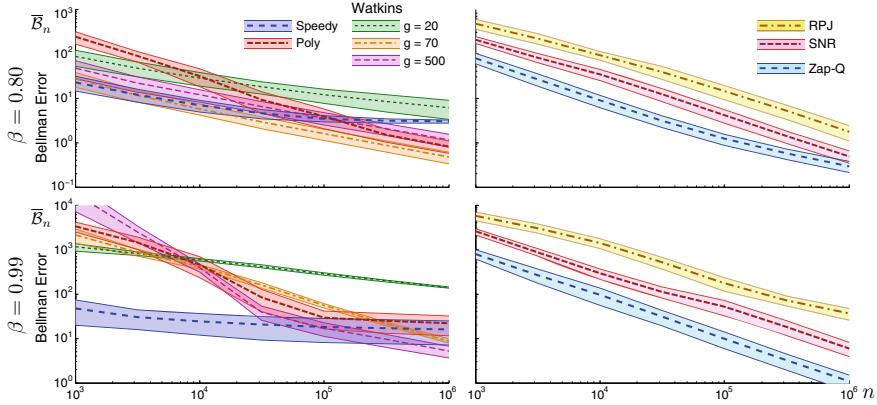


**Fig. 4.7** Maximum Bellman error  $\{\bar{B}_n : n \geq 0\}$  for the six Q-learning algorithms

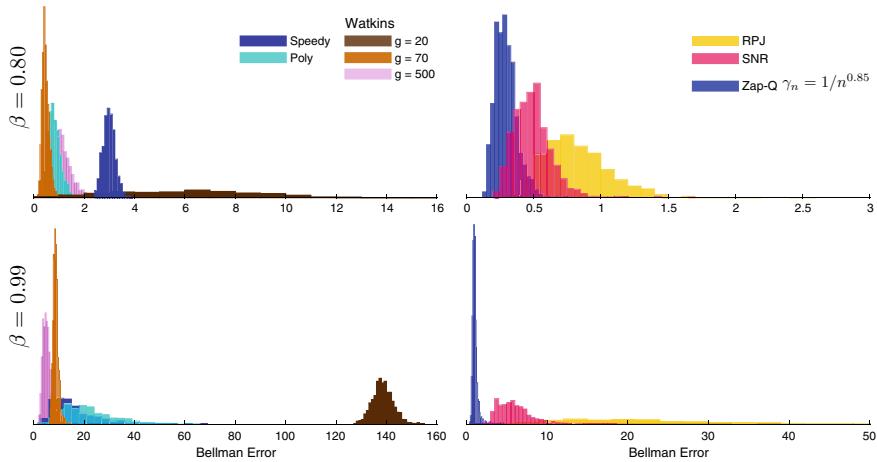
the sequence  $\{\sqrt{n} \bar{B}_n\}$  also converges in distribution as  $n \rightarrow \infty$ . Fig. 4.7 contains plots of  $\{\bar{B}_n\}$  for the six different Q-learning algorithms.

Each of the six algorithms performs reasonably well when  $\beta = 0.8$ . Only SNR and Zap Q-learning achieve near zero Bellman error within  $n = 10^6$  iterations when the discount factor is increased to  $\beta = 0.99$ . The performance of the two-time-scale algorithm is far superior to the one-time-scale SNR algorithm for larger values of  $\beta$ .

Figure 4.7 shows only the typical behavior—repeated trials were run to investigate the range of possible outcomes. For each algorithm, the outcomes of  $N = 1,000$



**Fig. 4.8** Simulation-based  $2\sigma$  confidence intervals for the six Q-learning algorithms



**Fig. 4.9** Histograms of the maximal Bellman error at iteration  $n = 10^6$

independent simulations resulted in samples  $\{\bar{B}_n^i, 1 \leq i \leq N\}$ , with  $\theta_0$  uniformly distributed on the interval  $[-10^3, 10^3]$  for  $\beta = 0.8$  and  $[-10^4, 10^4]$  for  $\beta = 0.99$ .

The batch means method was used to obtain estimates of the mean and variance of  $\bar{B}_n$  for a range of values of  $n$ . Plots of the mean and  $2\sigma$  confidence intervals are shown in Fig. 4.8, with  $\beta = 0.8$  in row 1, and  $\beta = 0.99$  in row 2.

Figure 4.9 shows histograms of  $\{\bar{B}_n^i, 1 \leq i \leq N\}$ ,  $n = 10^6$ , for all the six algorithms; this corresponds to the data shown in Fig. 4.8 at  $n = 10^6$ .

#### 4.4.2 Optimal Stopping in Finance

Optimal stopping is one of a few parameterized Q-learning settings for which convergence is guaranteed [70].

An example from [13, 70] is chosen to illustrate the application of the techniques surveyed in this section to this setting, and to show how Zap-Q-learning algorithm can be used with a more general basis. The reader is referred to these references for complete details of the problem setup and the reinforcement learning architecture used in this prior work.

The Markovian state process considered in [13, 70] is the vector of ratios:

$$X_n = (\tilde{p}_{n-99}, \tilde{p}_{n-98}, \dots, \tilde{p}_n)^T / \tilde{p}_{n-100} , \quad n = 0, 1, 2, \dots$$

in which  $\{\tilde{p}_t : t \in \mathbb{R}\}$  is a geometric Brownian motion (derived from an exogenous price process). This uncontrolled Markov chain is positive Harris recurrent on the state space  $X \equiv \mathbb{R}^{100}$  [47].

The “time to exercise” is modeled as a stopping time  $\tau \in \mathbb{Z}^+$ . The associated expected reward is defined as  $E[\beta^\tau r(X_\tau)]$ , with  $r(X_n) := X_n(100) = \tilde{p}_n / \tilde{p}_{n-100}$  and  $\beta \in (0, 1)$  fixed. The objective of finding a policy that maximizes the expected reward is modeled as an optimal stopping time problem.

The value function is defined to be the supremum over all stopping times:

$$h^*(x) = \sup_{\tau > 0} E[\beta^\tau r(X_\tau) | X_0 = x]. \quad (4.122)$$

This solves the Bellman equation:

$$h^*(x) = \max(r(x), \beta E[h^*(X_{n+1}) | X_n = x]) \quad x \in X. \quad (4.123)$$

The associated Q-function is denoted  $Q^*(x) := \beta E[h^*(X_{n+1}) | X_n = x]$ , which solves a similar fixed point equation:

$$Q^*(x) = \beta E[\max(r(X_{n+1}), Q^*(X_{n+1})) | X_n = x].$$

A stationary policy  $\phi: X \rightarrow \{0, 1\}$  assigns an action for each state  $x \in X$  as

$$\phi(x) = \begin{cases} 0 & \text{Do not exercise} \\ 1 & \text{Exercise.} \end{cases}$$

Each policy  $\phi$  defines a stopping time and associated average reward denoted

$$\tau := \min\{\phi(X_n) = 1\}, \quad h_\phi(x) := E[\beta^\tau r(X_\tau) | X_0 = x].$$

The optimal policy is expressed as

$$\phi^*(x) = \mathbb{I}\{r(x) \geq Q^*(x)\}.$$

The corresponding optimal stopping time that solves the supremum in (4.122) is achieved using this policy:  $\tau^* = \min\{n : \phi^*(X_n) = 1\}$  [70].

The objective here is to find an approximation for  $Q^*$  in a parameterized class  $\{Q^\theta := \theta^\top \psi : \theta \in \mathbb{R}^d\}$ , where  $\psi : \mathcal{X} \rightarrow \mathbb{R}^d$  is a vector of basis functions. For a fixed parameter vector  $\theta$ , the associated value function is denoted

$$h_{\phi^\theta}(x) = \mathbb{E}[\beta^{\tau_\theta} r(X_{\tau_\theta}) \mid x_0 = x], \quad (4.124)$$

where  $\tau_\theta = \min\{n : \phi^\theta(X_n) = 1\}$ ,  $\phi^\theta(x) = \mathbb{I}\{r(x) \geq Q^\theta(x)\}$ .

The function  $h_{\phi^\theta}$  was estimated using Monte Carlo in the numerical experiments surveyed below.

#### 4.4.2.1 Approximations to the Optimal Stopping Time Problem

To obtain the optimal parameter vector  $\theta^*$ , in [70] the authors apply the Q(0)-learning algorithm:

$$\theta_{n+1} = \theta_n + \alpha_{n+1} \psi(X_n) \left[ \beta \max(X_{n+1}(100), Q^{\theta_n}(X_{n+1})) - Q^{\theta_n}(X_n) \right], \quad n \geq 0. \quad (4.125)$$

In [13], the authors attempt to improve the performance of the Q(0) algorithm through a special choice for  $\{\alpha_n\}$ , and the matrix gain (4.79):

$$G_n = \left( \frac{1}{n} \sum_{k=1}^n \psi(X_k) \psi^\top(X_k) \right)^{-1}, \quad \alpha_n = \frac{g}{b+n}, \quad (4.126)$$

where  $g$  and  $b$  are positive constants. The resulting recursion is the  $G$ -Q(0) algorithm:

$$\theta_{n+1} = \theta_n + \alpha_n G_n \psi(X_n) \left[ \beta \max(X_{n+1}(100), Q^{\theta_n}(X_{n+1})) - Q^{\theta_n}(X_n) \right].$$

Through trial and error the authors find that  $g = 10^2$  and  $b = 10^4$  gives good performance. These values were also used in the experiments described in the following.

The limiting matrix gain is given by

$$G = \left( \mathbb{E}[\psi(X_k) \psi^\top(X_k)] \right)^{-1} g,$$

where the expectation is in steady state. The asymptotic covariance  $\Sigma_\theta^G$  is the unique positive semi-definite solution to the Lyapunov equation (4.51), provided all eigenvalues of  $GA$  satisfy  $\text{Re}(\lambda) < -\frac{1}{2}$ . The constant  $b$  in (4.126) has no impact on  $\Sigma_\theta^G$ .

The Zap Q-learning algorithm for this example is defined by

$$\begin{aligned}\theta_{n+1} &= \theta_n - \alpha_{n+1} \widehat{A}_{n+1}^{-1} \psi(X_n) \left[ \beta \max(X_{n+1}(100), Q^{\theta_n}(X_{n+1})) - Q^{\theta_n}(X_n) \right] \\ \widehat{A}_{n+1} &= \widehat{A}_n + \gamma_n [A_{n+1} - \widehat{A}_n],\end{aligned}\tag{4.127}$$

where

$$\begin{aligned}A_{n+1} &= \psi(X_n) \varphi^\top(\theta_n, X_{n+1}) \\ \varphi(\theta_n, X_{n+1}) &= \beta \psi(X_{n+1}) \mathbb{I}\{Q^{\theta_n}(X_{n+1}) \geq X_{n+1}(100)\} - \psi(X_n).\end{aligned}$$

It is conjectured that the asymptotic covariance  $\Sigma_\theta^*$  is obtained using (4.52), where the matrix  $A$  is the limit of  $\widehat{A}_n$ :

$$A = \mathbf{E} \left[ \psi(X_n) (\beta \psi(X_{n+1}) \mathbb{I}\{Q^{\theta^*}(X_{n+1}) \geq X_{n+1}(100)\} - \psi(X_n))^\top \right].$$

Justification is provided in [10], subject to additional assumptions.

#### 4.4.2.2 Experimental Results

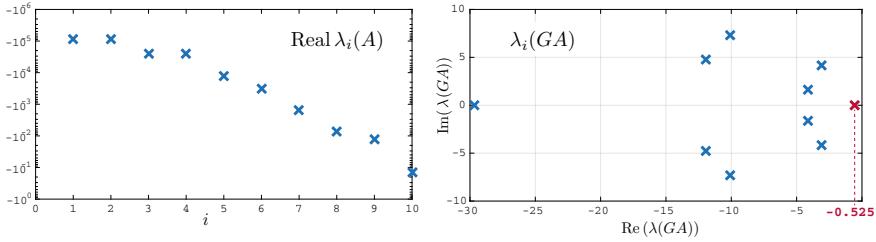
The experimental setting of [13, 70] is used to define the set of basis functions and other parameters. The dimension of the parameter vector  $d$  was chosen to be 10, with the basis functions defined in [13]. The objective here is to compare the performances of  $G$ -Q(0) and the Zap-Q algorithms in terms of both parameter convergence, and with respect to the resulting average reward (4.124).

The asymptotic covariance matrices  $\Sigma_\theta^*$  and  $\Sigma_\theta^G$  were estimated through the following steps: The matrices  $A$  and  $G$  were estimated via Monte Carlo. Estimation of  $A$  requires an estimate of  $\theta^*$ ; this was taken to be  $\theta_n$ , with  $n = 2 \times 10^6$ , obtained using the Zap-Q two-time-scale algorithm with  $\alpha_n \equiv 1/n$  and  $\gamma_n \equiv \alpha_n^{0.85}$ . This estimate of  $\theta^*$  was also used to estimate the covariance matrix  $\Sigma_\Delta$  defined in (4.46) using the batch means method. The matrices  $\Sigma_\theta^G$  and  $\Sigma_\theta^*$  were then obtained using (4.51) and (4.52), respectively.

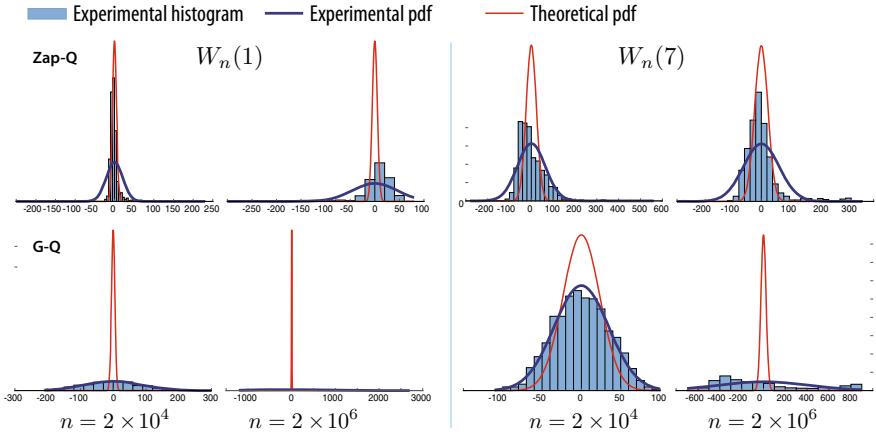
It was found that the trace of  $\Sigma_\theta^G$  was about 15 times greater than that of  $\Sigma_\theta^*$ .

**High performance despite ill-conditioned matrix gain:** The real part of the eigenvalues of  $A$  are shown on a logarithmic scale on the left-hand side of Fig. 4.10. The eigenvalues of the matrix  $A$  have a widespread: The condition-number is of the order  $10^4$ . This presents a challenge in applying any method. In particular, it was found that the performance of any scalar-gain algorithm was extremely poor, even with projection of parameter estimates.

The eigenvalues corresponding to the matrix  $GA$  are shown on the right-hand side of Fig. 4.10. Observe that one of these eigenvalues is very close to  $-0.5$ , so that the sufficient condition for  $\text{trace}(\Sigma_\theta^G) < \infty$  is barely satisfied. It is worth stressing that the finite asymptotic covariance was not a design goal in [13]: we discovered only recently that the sufficient condition  $\lambda < -\frac{1}{2}$  is satisfied [21, 22].



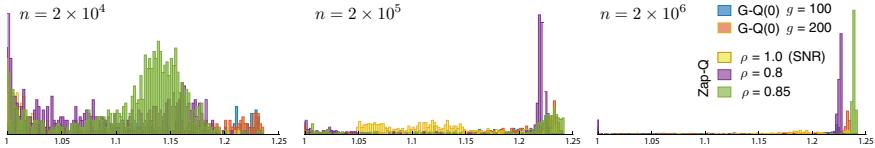
**Fig. 4.10** Eigenvalues of  $A$  and  $GA$  for the finance example



**Fig. 4.11** Theoretical and empirical variance for the finance example

In applying the Zap Q-learning algorithm it was found that the estimates  $\{\hat{A}_n\}$  in (4.127) are nearly singular. Despite the unfavorable setting for this approach, the performance of the algorithm was much better than any alternative that was tested. The upper row of Fig. 4.11 contains histograms of  $\{W_n^i(k) = \sqrt{n}(\theta_n^i(k) - \bar{\theta}_n(k)) : 1 \leq i \leq N\}$  for the Zap-Q algorithm. The variance for finite  $n$  is close to the theoretical predictions based on the asymptotic covariance  $\Sigma_{\theta}^*$ . The histograms were generated for two values of  $n$ , and  $k = 1, 7$ . Of the  $d = 10$  possibilities, the histogram for  $k = 1$  had the worst match with theoretical predictions, and  $k = 7$  was the closest.

The lower row of Fig. 4.11 contains the histograms of  $\{W_n^i(k) = \sqrt{n}(\theta_n^i(k) - \bar{\theta}_n(k)) : 1 \leq i \leq N\}$  for the G-Q(0) algorithm for  $n = 2 \times 10^4$  and  $2 \times 10^6$ , and  $k = 1, 7$ , along with the theoretical predictions based on the asymptotic covariance  $\Sigma_{\theta}^G$ .



**Fig. 4.12** Histograms of the average reward obtained using the  $G\text{-}Q(0)$  learning and the Zap-Q-learning,  $\gamma_n \equiv \alpha_n^{-\rho} \equiv n^{-\rho}$ . The case  $\rho = 1$  reduces to SNR

#### 4.4.2.3 Asymptotic Variance of the Discounted Reward

Denote  $h_n = h_\phi$ , with  $\phi = \phi^{\theta_n}$ . Histograms of the average reward  $h_n(x)$  were obtained for  $x(i) = 1$ ,  $1 \leq i \leq 100$ , and various values of  $n$ , based on  $N = 1000$  independent simulations. The plots shown in Fig. 4.12 are based on  $n = 2 \times 10^k$ , for  $k = 4, 5, 6$ . Omitted in this figure are *outliers*: values of the reward in the interval  $[0, 1]$ . Tables 4.1c, 4.1d, 4.1a and 4.1b lists the number of outliers for each  $n$  and each algorithm.

Recall that the asymptotic covariance of the  $G\text{-}Q(0)$  algorithm was not far from optimal (its trace was about 15 times larger than obtained using Zap-Q-learning). Moreover, this algorithm suffered from much larger outliers. It was found that doubling the scalar gain  $g$  (causing the largest eigenvalue of  $GA$  to be  $\approx -1$ ) resulted in better performance.

## 4.5 Zap-Q with Nonlinear Function Approximation

The theory presented for the Zap-Q algorithm in Sect. 4.3 is for a very special setting: the state and action spaces are finite, and the basis functions are indicator functions, as in Watkins' algorithm [75]. We discuss here how this can be generalized to a nonlinear function approximation setting, and the technical challenges that arise. Extended discussion is contained in [11, 18].

For simplicity, we restrict to the finite state and action space setting of Sect. 4.3. The objective remains the same: Find  $\theta^* \in \mathbb{R}^d$  such that

$$\bar{f}_i(\theta^*) = \mathbb{E}\left[\{c(X_n, U_n) + \beta \underline{Q}^{\theta^*}(X_{n+1}) - Q^{\theta^*}(X_n, U_n)\}\zeta_n(i)\right] = 0, \quad 1 \leq i \leq d, \quad (4.128)$$

where  $\zeta$  is a  $d$ -dimensional stationary stochastic process that is adapted to  $(X, U)$ , and  $\{Q^\theta : \theta \in \mathbb{R}^d\}$ , is a parameterized family of functions, with  $Q^\theta : \mathcal{X} \times \mathcal{U} \rightarrow \mathbb{R}$  for each  $\theta$ .

Algorithm design begins with specification of the family  $\{Q^\theta : \theta \in \mathbb{R}^d\}$  of candidate approximations, and the sequence of *eligibility vectors*  $\{\zeta_n\}$ . We begin with assumptions required for the Q-function approximation.

**Table 4.1** Outliers observed in  $N = 1000$  runs. Each table represents the number of runs which resulted in an average reward below a certain value (a)  $h_n(x) < 1$ , (b)  $h_n(x) < 0.95$ , (c)  $h_n(x) < 0.75$ , (d)  $h_n(x) < 0.5$

$n$	$2 \times 10^4$	$2 \times 10^5$	$2 \times 10^6$	$n$	$2 \times 10^4$	$2 \times 10^5$	$2 \times 10^6$
$G\text{-}Q(0)$ , $g = 100$	827	775	680	$G\text{-}Q(0)$ , $g = 100$	811	755	654
$G\text{-}Q(0)$ , $g = 200$	824	725	559	$G\text{-}Q(0)$ , $g = 200$	806	706	537
SNR	820	541	625	SNR	55	0	0
Zap-Q, $\rho = 0.8$	236	737	61	Zap-Q, $\rho = 0.8$	0	0	0
Zap-Q, $\rho = 0.85$	386	516	74	Zap-Q, $\rho = 0.85$	0	0	0
$n$	$2 \times 10^4$	$2 \times 10^5$	$2 \times 10^6$	$n$	$2 \times 10^4$	$2 \times 10^5$	$2 \times 10^6$
$G\text{-}Q(0)$ , $g = 100$	774	727	628	$G\text{-}Q(0)$ , $g = 100$	545	497	395
$G\text{-}Q(0)$ , $g = 200$	789	688	525	$G\text{-}Q(0)$ , $g = 200$	641	518	390
SNR	4	0	0	SNR	0	0	0
Zap-Q, $\rho = 0.8$	0	0	0	Zap-Q, $\rho = 0.8$	0	0	0
Zap-Q, $\rho = 0.85$	0	0	0	Zap-Q, $\rho = 0.85$	0	0	0

The only requirement on the parameterized family is that  $Q^\theta$  is continuously differentiable, and Lipschitz continuous with respect to  $\theta$ . In the context of deep-RL,  $Q^\theta$  itself is a deep neural network, with  $\theta \in \mathbb{R}^d$  the weights of the neural network. The input to the neural network is the state-action pair  $(x, u) \in \mathcal{X} \times \mathcal{U}$ , and the output is the Q-function estimate  $Q^\theta(x, u)$ .

As was the case in the tabular setting, the function  $\bar{f}(\theta)$  in (4.128) is only piecewise smooth. At points of differentiability, the linearization of  $\bar{f}$  has a simple form:

$$A(\theta) := \partial_\theta \bar{f}(\theta) = \mathbb{E}[\zeta_n(\beta \partial_\theta Q^\theta(X_{n+1}, \phi^\theta(X_{n+1})) - \partial_\theta Q^\theta(X_n, U_n))], \quad (4.129)$$

where  $\phi^\theta$  denotes the greedy policy induced by  $Q^\theta$ :

$$\phi^\theta := \phi^{(\kappa)}, \quad \text{where } \kappa := \min\{i : \phi^{(i)}(x) \in \arg \min_u Q^\theta(x, u), \text{ for all } x \in \mathcal{X}\} \quad (4.130)$$

with  $\{\phi^{(i)} : 1 \leq i \leq \ell_\phi\}$ , being the enumeration of all stationary policies. The definition of  $A(\theta)$  is extended to all of  $\mathbb{R}^d$  through (4.129), in which  $\phi^\theta$  is uniquely determined using (4.130).

It is straightforward to extend the Zap-Q algorithm in Sect. 4.3.6:

**Zap-Q-Learning with Nonlinear Function Approximation**


---

**Input:**  $\theta_0 \in \mathbb{R}^d$ ,  $\hat{A}_0 \in \mathbb{R}^{d \times d}$ ,  $n = 0$ ,  $T \in \mathbb{Z}^+$  ▷ Initialization

1: **repeat**

- 2:    $\phi_n(X_{n+1}) := \arg \min_u Q^{\theta_n}(X_{n+1}, u);$
- 3:    $d_{n+1} := c(X_n, U_n) + \beta Q^{\theta_n}(X_{n+1}, \phi_n(X_{n+1})) - Q^{\theta_n}(X_n, U_n);$  ▷ TD term
- 4:    $A_{n+1} := \zeta(X_n, U_n) [\beta \partial Q^{\theta_n}(X_{n+1}, \phi_n(X_{n+1})) - \partial Q^{\theta_n}(X_n, U_n)]^\top;$
- 5:    $\hat{A}_{n+1} = \hat{A}_n + \gamma_{n+1} [A_{n+1} - \hat{A}_n];$  ▷ Matrix gain update rule
- 6:    $\theta_{n+1} = \theta_n - \alpha_{n+1} \hat{A}_{n+1}^{-1} \zeta_n d_{n+1};$  ▷ Zap-Q update rule
- 7:    $n = n + 1$

8: **until**  $n \geq T$

---

#### 4.5.1 Choosing the Eligibility Vectors

Recall that in TD-learning the Galerkin relaxation (4.25) of the Bellman equation is used as a criterion for algorithm design. We might follow the same steps as in TD-learning and Watkins' Q-learning, and choose

$$\zeta_n = \partial_\theta Q^{\theta_n}(X_n, U_n).$$

While adapted to  $(X, U)$  as required, the interpretation as a Galerkin relaxation fails due to dependency on the parameter estimate  $\theta_n$ . We might take a far-sighted view:

$$\zeta_n = \partial_\theta Q^\theta(X_n, U_n) \Big|_{\theta=\theta^*}. \quad (4.131)$$

The solution to (4.128) might be approached through an additional parameter sequence  $\{\bar{\theta}_n\}$ . The following is proposed in [77] in their proposed *least squares Q-learning* for optimal stopping:

$$\bar{\theta}_{n+1} = \bar{\theta}_n + \delta[\theta_{n+1} - \bar{\theta}_n],$$

where  $\delta > 0$  is a small constant. We then take the approximation of (4.131)

$$\zeta_n = \partial_\theta Q^\theta(X_n, U_n) \Big|_{\theta=\bar{\theta}_n}.$$

The sequence  $\zeta$  can be approximated by a stationary process, which may simplify analysis as well as provide a clearer understanding of the limit  $\theta^*$  of the algorithm.

### 4.5.2 Theory and Challenges

The algorithm above is the Zap–SA algorithm of Sect. 4.2.4 applied to solve (4.128). It is intended to approximate the ODE (4.37):

$$\frac{d}{dt} \xi(t) = -[A(\xi(t))]^{-1} \bar{f}(\xi(t)) \quad (4.132)$$

where  $A(\theta)$  is defined in (4.129). Theorem 4.3 can be extended to obtain the stability and convergence of the solution to the above ODE: For each initial condition  $\xi_0$ ,

$$\begin{aligned} \frac{d}{dt} \bar{f}(\xi(t)) &= -\bar{f}(\xi(t)) \\ \lim_{t \rightarrow \infty} \|\bar{f}(\xi(t))\| &= \lim_{t \rightarrow \infty} \|\bar{f}(\xi(0))\| e^{-t} = 0. \end{aligned}$$

The proof is contained [11]. The only non-triviality in the proof (in comparison to the proof of Theorem 4.3) is dealing with the fact that  $\bar{f}(\theta)$  defined in (4.128) is not continuously differentiable.

The biggest challenge is proving that the Zap-Q algorithm above is in fact an approximation of the ODE (4.132), in the sense discussed in Sect. 4.3.6.3. Since  $A(\theta) = \partial_\theta \bar{f}(\theta)$  is discontinuous, and the noise is Markovian, standard tools to analyze the SA recursion for  $\{\hat{A}_n\}$  cannot be applied. In Sect. 4.3.6, the technical assumption (Q3) was used to deal with this issue:

$$\sum_n \gamma_n \mathbb{I}\{\phi_n \neq \phi_{n+1}\} < \infty.$$

A convergence result for a general function-approximation setting that avoids such an assumption is a topic of current research—we expect this will be resolved soon.

### 4.5.3 Regularized Zap-Q

Another algorithmic issue with Zap-Q is the fact that the sequence of matrices  $\{\hat{A}_n\}$  may not be invertible for each  $n$ . A projection obtained using the matrix inversion lemma was suggested in Sect. 4.3.6 to enforce invertibility. We discuss here an alternative that was proposed in [11].

An approximation to the pseudoinverse is proposed: for small  $\varepsilon > 0$ , consider the matrix gain

$$G_n = -[\varepsilon I + \hat{A}_n^\top \hat{A}_n]^{-1} \hat{A}_n^\top.$$

The inverse exists for each  $n$ , since  $[\varepsilon I + \hat{A}_n^\top \hat{A}_n]$  is positive definite. Suppose the algorithm is convergent, so that in particular  $G_n \rightarrow G$ , where

$$G = -[\varepsilon I + A^\top A]^{-1} A^\top, \quad A = A(\theta^*).$$

The sub-optimality gap in the resulting asymptotic covariance (which can be calculated by solving the Lyapunov equation (4.51)) can be bounded as  $O(\varepsilon^2)$  [11]:

$$\Sigma_\theta = \Sigma_\theta^* + O(\varepsilon^2), \quad \text{with} \quad \Sigma_\theta^* = A^{-1} \Sigma_\Delta (A^\top)^{-1},$$

where  $\Sigma_\Delta$  is defined in (4.46).

## 4.6 Conclusions and Future Work

Stochastic approximation is an important class of algorithms that cannot be ignored. In this chapter, we have seen how SA theory provides techniques to establish stability and convergence of RL algorithms, and tools for algorithm design.

The elegant Q-learning algorithm of Watkins is a perfect vehicle to illustrate these points: it is easy to see that the ODE associated with this algorithm is stable, for both synchronous and asynchronous implementations. Moreover, a glance at either ODE approximation, (4.86) or (4.88), explains the slow convergence of Q-learning. These observations were the basis of Theorems 4.6 and 4.7.

There are many avenues for future research.

**Eliminating Assumption (Q3):** Assumption (Q3) in (4.102) was introduced to address a discontinuity issue in Zap-Q-learning. Based on an ODE analysis, it seems likely that the conclusions of Theorem 4.8 hold without this additional assumption.

**Removing discount-factor sensitivity in RL:** Proposition 4.3 reveals that the value function is large when  $\beta \sim 1$  only because its mean is large. Under Assumptions (Q1) and (Q2), the same can be said for the Q-function:

$$Q^*(x, u) = \tilde{Q}^*(x, u) + \frac{1}{1 - \beta} \eta^*,$$

where the scalar  $\eta^*$  denotes the optimal average cost (independent of  $(x, u)$ ) under the assumptions of Sect. 4.3), and  $\tilde{Q}^*(x, u)$  is uniformly bounded in  $\beta$ ,  $x$ , and  $u$ . The above observation can be used in the following more general manner.

As an alternative to solving the original Bellman equation (4.69):

$$Q^*(x, u) = c(x, u) + \beta \sum_{x' \in \mathcal{X}} P_u(x, x') \underline{Q}^*(x')$$

consider the following *relative* Bellman equation:

$$H^*(x, u) = c(x, u) + \beta \sum_{x' \in \mathcal{X}} P_u(x, x') \underline{H}^*(x') - \langle \mu, H^* \rangle, \quad (4.133)$$

where  $\mu : \mathcal{X} \times \mathcal{U} \rightarrow [0, 1]$  is a probability measure (a design choice), and

$$\langle \mu, H^* \rangle = \sum_{x \in \mathcal{X}, u \in \mathcal{U}} \mu(x, u) H^*(x, u).$$

The following relationship between  $H^*$  and  $Q^*$  is immediate:

**Proposition 4.5** *The solution  $H^*$  to (4.133) satisfies*

$$H^*(x, u) = Q^*(x, u) - \frac{\langle \mu, H^* \rangle}{1 - \beta}.$$

In [1], the authors consider the goal of computing the Q-function corresponding to an *average cost problem*, which is formulated as solving for the fixed point equation (4.133), with  $\beta = 1$ . The authors propose an average cost Q-learning algorithm, that corresponds to the SA algorithm to solve for  $\theta^*$  such that

$$0 = \mathbb{E}\left[\{c(X_n, U_n) + \underline{H}^{\theta^*}(X_{n+1}) - H^{\theta^*}(X_n, U_n) - \langle \mu, H^{\theta^*} \rangle\}\psi(X_n, U_n)\right].$$

The basis functions  $\{\psi_i : 1 \leq i \leq d\}$  are the indicator functions defined in (4.80). Under an appropriate choice of  $\mu$ , the authors prove stability and convergence of the SA algorithm, and show that the limit satisfies  $H^{\theta^*} = H^*$ , which is a solution to

$$H^*(x, u) = c(x, u) + \sum_{x' \in \mathcal{X}} P_u(x, x') \underline{H}^*(x') - \eta^*, \quad (4.134)$$

The solution to the fixed point equation (4.134) is *not* unique, since adding a constant to any solution  $H^*$  will still satisfy the equation. Equation (4.133) has a unique solution for any value of  $\beta$ . In the special case  $\beta = 1$ , its solution  $H^*$  is the unique solution to (4.134), subject to the constraint  $\langle \mu, H^* \rangle = \eta^*$ .

The results of [1] combined with Proposition 4.5 suggest that a similar SA algorithm to solve (4.133) will be stable and consistent for any choice of  $\beta \leq 1$ . The only change to their algorithm is to scale a term in the SA recursion by  $\beta$  and choose the step-size according to the guidelines in this chapter. We conclude with two conjectures for this algorithm<sup>1</sup>: using the step-size  $\alpha_n = g/n$  in which  $g$  is independent of  $\beta$  (chosen sufficiently large),

- (i) The asymptotic covariance is finite and uniformly bounded in  $\beta$ .
- (ii) The finite-time error bound (4.7) holds:

$$\mathbb{P}\{\|\theta_n - \theta^*\| > \varepsilon\} \leq \bar{b} \exp(-n\bar{I}(\varepsilon))$$

with  $\bar{I}$  independent of  $\beta$ .

**Optimal Scalar Gain Stochastic Approximation and Q-learning:** We do not know how to choose a scalar gain outside of the narrow setting of Q-learning in the tabular setting. If  $A$  is Hurwitz, we may set  $g = 1/\varrho_{\min}$ ,  $\varrho_{\min} := \min\{-\operatorname{Re}(\lambda(A))\}$ , where

---

<sup>1</sup>These conjectures were recently addressed in our work [23], when this book was in press.

the minimum is overall eigenvalues of  $A$ . If the step-size  $\alpha_n = g/n$  is used, then the eigenvalue test is successful:  $\text{Re}(\lambda(gA)) \leq -1 < -\frac{1}{2}$ . An iterative procedure to estimate such a constant  $g$ , without having to explicitly estimate the matrix  $A$ , would be a good starting point to obtain efficient scalar gain algorithms. It is conjectured that an extension of *Oja's algorithm* can be used for this [7]. Even better would be the development of techniques to optimize a matrix gain within a given class, such as block diagonal matrices.

However, we would like to stress that no scalar gain SA algorithm can guarantee universal stability, as is expected for the Zap–SA algorithm (cf. Theorem 4.3).

**Matrix Momentum Stochastic Approximation:** We introduced in [19] a new class of *matrix momentum* SA algorithms that achieve the optimal MSE convergence rate of  $1/n$ , just like the Zap–SA algorithm. It is even possible to optimize the asymptotic covariance. The fact that these algorithms only involve computing matrix–vector products in each iteration, as opposed to matrix inverses, makes them an attractive option in high-dimensional settings. However, just like scalar gain SA recursions, matrix momentum techniques *do not* currently ensure universal stability as in Zap–SA. Stability theory for nonlinear SA with matrix momentum is an open area for research.

We are currently searching for techniques that combine the best of matrix-momentum and Zap, so that the resulting algorithm is globally stable *and* computationally inexpensive.

**Acknowledgements** Financial support from ARO grant W911NF1810334 is gratefully acknowledged. Additional support from EPCN 1609131 and CPS 1646229, and French National Research Agency grant ANR-16-CE05-0008.

## References

1. Abounadi, J., Bertsekas, D., Borkar, V.S.: Learning algorithms for Markov decision processes with average cost. *SIAM J. Control Optim.* **40**(3), 681–698 (electronic) (2001)
2. Azar, M.G., Munos, R., Ghavamzadeh, M., Kappen, H.: Speedy Q-learning. In: Advances in Neural Information Processing Systems (2011)
3. Benveniste, A., Métivier, M., Priouret, P.: Adaptive Algorithms and Stochastic Approximations. Springer, Berlin (2012)
4. Bertsekas, D.P.: Dynamic Programming and Optimal Control, vol. 2, 4th edn. Athena Scientific (2012)
5. Bhandari, J., Russo, D., Singal, R.: A finite time analysis of temporal difference learning with linear function approximation (2018). arXiv preprint [arXiv:1806.02450](https://arxiv.org/abs/1806.02450)
6. Blum, J.R.: Multidimensional stochastic approximation methods. *Ann. Math. Stat.* **25**, 737–744 (1954)
7. Borkar, V., Meyn, S.P.: Oja's algorithm for graph clustering, Markov spectral decomposition, and risk sensitive control. *Automatica* **48**(10), 2512–2519 (2012)
8. Borkar, V.S.: Stochastic Approximation: A Dynamical Systems Viewpoint. Hindustan Book Agency and Cambridge University Press (jointly), Delhi and Cambridge (2008)
9. Borkar, V.S., Meyn, S.P.: The ODE method for convergence of stochastic approximation and reinforcement learning. *SIAM J. Control Optim.* **38**(2), 447–469 (2000). (Also presented at the IEEE CDC, December, 1998)

10. Chen, S., Devraj, A.M., Bušić, A., Meyn, S.: Zap Q-learning for optimal stopping (2019). In: Proceedings of the American Control Conference (ACC) (2020)
11. Chen, S., Devraj, A.M., Bušić, A., Meyn, S.: Zap Q-learning with nonlinear function approximation. arXiv preprint: [arXiv:1910.05405](https://arxiv.org/abs/1910.05405) (2019)
12. Chen, S., Devraj, A.M., Bušić, A., Meyn, S.P.: Explicit mean-square error bounds for Monte-Carlo and linear stochastic approximation. In: Proceedings of the Twenty Third International Conference on Artificial Intelligence and Statistics (AISTATS) (2020)
13. Choi, D., Van Roy, B.: A generalized Kalman filter for fixed point approximation and efficient temporal-difference learning. *Discret. Event Dyn. Syst.: Theory Appl.* **16**(2), 207–239 (2006)
14. Chung, K.L., et al.: On a stochastic approximation method. *Ann. Math. Stat.* **25**(3), 463–483 (1954)
15. Dai, B., Shaw, A., Li, L., Xiao, L., He, N., Liu, Z., Chen, J., Song, L.: SBEED: Convergent reinforcement learning with nonlinear function approximation. In: International Conference on Machine Learning, pp. 1133–1142 (2018)
16. Dalal, G., Szörényi, B., Thoppe, G., Mannor, S.: Concentration bounds for two timescale stochastic approximation with applications to reinforcement learning. In: Proceedings of the Conference on Computational Learning Theory, and ArXiv e-prints, pp. 1–35 (2017)
17. Dembo, A., Zeitouni, O.: Large Deviations Techniques And Applications, 2nd edn. Springer, New York (1998)
18. Devraj, A.M.: Reinforcement learning design with optimal learning rate. Ph.D. thesis, University of Florida (2019)
19. Devraj, A.M., Bušić, A., Meyn, S.: On matrix momentum stochastic approximation and applications to Q-learning. In: Proceedings of the 57th Annual Allerton Conference on Communication, Control, and Computing (2019)
20. Devraj, A.M., Bušić, A., Meyn, S.: Zap Q-Learning – a user’s guide. In: Proceedings of the Fifth Indian Control Conference (2019)
21. Devraj, A.M., Meyn, S.P.: Fastest convergence for Q-learning. arXiv preprint [arXiv:1707.03770](https://arxiv.org/abs/1707.03770) (2017)
22. Devraj, A.M., Meyn, S.P.: Zap Q-learning. In: Proceedings of the 31st International Conference on Neural Information Processing Systems (2017)
23. Devraj, A.M., Meyn, S.P.: Q-learning with uniformly bounded variance. arXiv preprint [arXiv:2002.10301](https://arxiv.org/abs/2002.10301) (2020)
24. Duffy, K., Meyn, S.: Large deviation asymptotics for busy periods. *Stoch. Syst.* **4**(1), 300–319 (2014)
25. Duffy, K.R., Meyn, S.P.: Most likely paths to error when estimating the mean of a reflected random walk. *Perform. Eval.* **67**(12), 1290–1303 (2010)
26. Even-Dar, E., Mansour, Y.: Learning rates for Q-learning. *J. Mach. Learn. Res.* **5**, 1–25 (2003)
27. Glynn, P.W., Ormoneit, D.: Hoeffding’s inequality for uniformly ergodic Markov chains. *Stat. Prob. Lett.* **56**, 143–146 (2002)
28. Hasselt, H.V.: Double Q-learning. In: Advances in Neural Information Processing Systems, pp. 2613–2621 (2010)
29. Kiefer, J., Wolfowitz, J.: Stochastic estimation of the maximum of a regression function. *Ann. Math. Stat.* **23**(3), 462–466 (1952)
30. Konda, V.R., Tsitsiklis, J.N.: Actor-critic algorithms. In: Advances in Neural Information Processing Systems, pp. 1008–1014 (2000)
31. Konda, V.R., Tsitsiklis, J.N.: On actor-critic algorithms. *SIAM J. Control Optim.* **42**(4), 1143–1166 (electronic) (2003)
32. Konda, V.R., Tsitsiklis, J.N.: Convergence rate of linear two-time-scale stochastic approximation. *Ann. Appl. Prob.* **14**(2), 796–819 (2004)
33. Konda, V.V.G.: Actor-critic algorithms. Ph.D. thesis, Massachusetts Institute of Technology (2002)
34. Kontoyiannis, I., Meyn, S.P.: Spectral theory and limit theorems for geometrically ergodic Markov processes. *Ann. Appl. Prob.* **13**, 304–362 (2003)

35. Kushner, H.J., Yin, G.G.: Stochastic Approximation Algorithms and Applications, Applications of Mathematics (New York), vol. 35. Springer, New York (1997)
36. Lakshminarayanan, C., Szepesvari, C.: Linear stochastic approximation: How far does constant step-size and iterate averaging go? In: International Conference on Artificial Intelligence and Statistics, pp. 1347–1355 (2018)
37. Lattimore, T., Szepesvari, C.: Bandit algorithms (2019). <https://banditalgs.com/about/>
38. Lillicrap, T.P., Hunt, J.J., Pritzel, A., Heess, N., Erez, T., Tassa, Y., Silver, D., Wierstra, D.: Continuous control with deep reinforcement learning (2015). arXiv preprint [arXiv:1509.02971](https://arxiv.org/abs/1509.02971)
39. Lund, R.B., Meyn, S.P., Tweedie, R.L.: Computable exponential convergence rates for stochastically ordered Markov processes. *Ann. Appl. Prob.* **6**(1), 218–237 (1996)
40. Ma, D.J., Makowski, A.M., Shwartz, A.: Stochastic approximations for finite-state Markov chains. *Stoch. Process. Appl.* **35**(1), 27–45 (1990)
41. Mehta, P.G., Meyn, S.P.: Q-learning and Pontryagin’s minimum principle. In: Proceedings of the IEEE Conference on Decision and Control, pp. 3598–3605 (2009)
42. Metivier, M., Priouret, P.: Applications of a Kushner and Clark lemma to general classes of stochastic algorithms. *IEEE Trans. Inf. Theory* **30**(2), 140–151 (1984)
43. Meyn, S.P.: Large deviation asymptotics and control variates for simulating large functions. *Ann. Appl. Prob.* **16**(1), 310–339 (2006)
44. Meyn, S.P.: Control Techniques for Complex Networks. Cambridge University Press, Cambridge (2007). (Pre-Publication Edition Available Online)
45. Meyn, S.P., Surana, A.: TD-learning with exploration. In: 50th IEEE Conference on Decision and Control, and European Control Conference, pp. 148–155 (2011)
46. Meyn, S.P., Tweedie, R.L.: Computable bounds for convergence rates of Markov chains. *Ann. Appl. Prob.* **4**, 981–1011 (1994)
47. Meyn, S.P., Tweedie, R.L.: Markov Chains and Stochastic Stability, 2nd edn. Cambridge University Press, Cambridge (2009). (Published in the Cambridge Mathematical Library. 1993 Edition Online)
48. Mnih, V., Badia, A.P., Mirza, M., Graves, A., Lillicrap, T., Harley, T., Silver, D., Kavukcuoglu, K.: Asynchronous methods for deep reinforcement learning. In: International Conference on Machine Learning, pp. 1928–1937 (2016)
49. Mnih, V., Kavukcuoglu, K., Silver, D., Rusu, A.A., Veness, J., Bellemare, M.G., Graves, A., Riedmiller, M., Fidjeland, A.K., Ostrovski, G., et al.: Human-level control through deep reinforcement learning. *Nature* **518**(7540), 529 (2015)
50. Moulines, E., Bach, F.R.: Non-asymptotic analysis of stochastic approximation algorithms for machine learning. In: Advances in Neural Information Processing Systems 24, pp. 451–459. Curran Associates, Inc. (2011)
51. Osband, I., Van Roy, B., Russo, D., Wen, Z.: Deep exploration via randomized value functions (2017). arXiv preprint [arXiv:1703.07608](https://arxiv.org/abs/1703.07608)
52. Paulin, D.: Concentration inequalities for Markov chains by Marton couplings and spectral methods. *Electron. J. Prob.* **20**, 32 pp. (2015)
53. Polyak, B.T.: A new method of stochastic approximation type. *Avtomatika i telemekhanika* (in Russian). translated in *Automat. Remote Control*, 51 (1991) pp. 98–107 (1990)
54. Polyak, B.T., Juditsky, A.B.: Acceleration of stochastic approximation by averaging. *SIAM J. Control Optim.* **30**(4), 838–855 (1992)
55. Robbins, H., Monro, S.: A stochastic approximation method. *Ann. Math. Stat.* **22**, 400–407 (1951)
56. Rummery, G.A., Niranjan, M.: On-line Q-learning using connectionist systems. Technical report 166, Cambridge University, Department of Engineering, Cambridge, CUED/F-INENG/ (1994)
57. Ruppert, D.: A Newton-Raphson version of the multivariate Robbins-Monro procedure. *Ann. Stat.* **13**(1), 236–245 (1985)
58. Ruppert, D.: Efficient estimators from a slowly convergent Robbins-Monro processes. Technical Report No. 781, Cornell University, School of Operations Research and Industrial Engineering, Ithaca (1988)

59. Russo, D.J., Van Roy, B., Kazerouni, A., Osband, I., Wen, Z.: A Tutorial on Thompson Sampling (2018)
60. Srikanth, R., Ying, L.: Finite-time error bounds for linear stochastic approximation and TD learning (2019). CoRR [arXiv:abs/1902.00923](https://arxiv.org/abs/1902.00923)
61. Sutton, R., Barto, A.: Reinforcement Learning: An Introduction, 2nd edn. MIT Press, Cambridge, MA (2018). On-Line Edition at <http://www.cs.ualberta.ca/~sutton/book/the-book.html>
62. Sutton, R.S.: Learning to predict by the methods of temporal differences. *Mach. Learn.* **3**(1), 9–44 (1988)
63. Sutton, R.S., McAllester, D.A., Singh, S.P., Mansour, Y.: Policy gradient methods for reinforcement learning with function approximation. In: Advances in Neural Information Processing Systems, pp. 1057–1063 (2000)
64. Szepesvári, C.: The asymptotic convergence-rate of Q-learning. In: Proceedings of the 10th International Conference on Neural Information Processing Systems, NIPS'97, pp. 1064–1070. MIT Press, Cambridge (1997)
65. Szepesvári, C.: Algorithms for Reinforcement Learning. Synthesis Lectures on Artificial Intelligence and Machine Learning. Morgan & Claypool Publishers (2010)
66. Thoppe, G., Borkar, V.: A concentration bound for stochastic approximation via Alekseev's formula. *Stoch. Syst.* **9**(1), 1–26 (2019)
67. Tsitsiklis, J.: Asynchronous stochastic approximation and Q-learning. *Mach. Learn.* **16**, 185–202 (1994)
68. Tsitsiklis, J.N., Roy, B.V.: Average cost temporal-difference learning. *Automatica* **35**(11), 1799–1808 (1999)
69. Tsitsiklis, J.N., Van Roy, B.: An analysis of temporal-difference learning with function approximation. *IEEE Trans. Autom. Control* **42**(5), 674–690 (1997)
70. Tsitsiklis, J.N., Van Roy, B.: Optimal stopping of Markov processes: Hilbert space theory, approximation algorithms, and an application to pricing high-dimensional financial derivatives. *IEEE Trans. Autom. Control* **44**(10), 1840–1851 (1999)
71. Venter, J., et al.: An extension of the Robbins-Monro procedure. *Ann. Math. Stat.* **38**(1), 181–190 (1967)
72. Wainwright, M.J.: Stochastic approximation with cone-contractive operators: Sharp  $\ell_\infty$ -bounds for Q-learning (2019). CoRR [arXiv:abs/1905.06265](https://arxiv.org/abs/1905.06265)
73. Wardi, Y., Seatzu, C., Egerstedt, M., Buckley, I.: Performance regulation and tracking via look-ahead simulation: preliminary results and validation. In: Proceedings of the IEEE Conference on Decision and Control, pp. 6462–6468 (2017)
74. Watkins, C.J.C.H.: Learning from delayed rewards. Ph.D. thesis, King's College, Cambridge, Cambridge (1989)
75. Watkins, C.J.C.H., Dayan, P.: Q-learning. *Mach. Learn.* **8**(3–4), 279–292 (1992)
76. Yu, H., Bertsekas, D.P.: Q-learning algorithms for optimal stopping based on least squares. In: Proceedings of the European Control Conference (ECC) (2007)
77. Yu, H., Bertsekas, D.P.: Q-learning and policy iteration algorithms for stochastic shortest path problems. *Ann. Oper. Res.* **208**(1), 95–132 (2013)

# Chapter 5

## Mixed Density Methods for Approximate Dynamic Programming



Max L. Greene, Patryk Deptula, Rushikesh Kamalapurkar,  
and Warren E. Dixon

**Abstract** This chapter discusses mixed density reinforcement learning (RL)-based approximate optimal control methods applied to deterministic systems. Such methods typically require a persistence of excitation (PE) condition for convergence. In this chapter, data-based methods will be discussed to soften the stringent PE condition by learning via simulation-based extrapolation. The development is based on the observation that, given a model of the system, RL can be implemented by evaluating the Bellman error (BE) at any number of desired points in the state space, thus virtually simulating the system. The sections will discuss necessary and sufficient conditions for optimality, regional model-based RL, local (StaF) RL, combining regional and local model-based RL, and RL with sparse BE extrapolation. Notes on stability follow within each method's respective section.

---

M. L. Greene · W. E. Dixon (✉)

Department of Mechanical and Aerospace Engineering, University of Florida,  
Gainesville, FL, USA

e-mail: [wdixon@ufl.edu](mailto:wdixon@ufl.edu)

e-mail: [maxgreene12@ufl.edu](mailto:maxgreene12@ufl.edu)

P. Deptula

The Charles Stark Draper Laboratory, Inc., Cambridge, MA, USA

e-mail: [pdeptula@draper.com](mailto:pdeptula@draper.com)

R. Kamalapurkar

Department of Mechanical and Aerospace Engineering, Mechanical and Aerospace  
Engineering, Stillwater, OK, USA

e-mail: [rushikesh.kamalapurkar@okstate.edu](mailto:rushikesh.kamalapurkar@okstate.edu)

## 5.1 Introduction

Reinforcement learning (RL) enables a cognitive agent to learn desirable behavior from interactions with its environment. In control theory, the desired behavior is typically quantified using a cost function, and the control problem is formulated to find the optimal policy that minimizes a cumulative cost function. Leveraging function approximation architectures, RL-based techniques have been developed to approximately solve optimal control problems for continuous-time and discrete-time deterministic systems by computing the optimal policy based on an estimate of the optimal cost-to-go function, i.e., the value function (cf., [1–12]). In RL-based approximate online optimal control, the Hamilton–Jacobi–Bellman equation (HJB), along with an estimate of the state derivative (cf. [6, 9]), or an integral form of the HJB (cf. [13, 14]), is utilized to approximately measure the quality of the estimate of the value function evaluated at each visited state along the system trajectory. This measurement is called the Bellman error (BE).

In online RL-based techniques, estimates for the uncertain parameters in the value function are updated using the BE as a performance metric. Hence, the unknown value function parameters are updated based on the evaluation of the BE along the system trajectory. In particular, the integral BE is meaningful as a measure of quality of the value function estimate only if evaluated along the system trajectories, and state derivative estimators can only generate numerical estimates of the state derivative along the system trajectories. Online RL-based techniques can be implemented in either model-based or model-free form. Generally speaking, both approaches have their respective advantages and disadvantages. Model-free approaches learn optimal actions without requiring knowledge of the system [15]. Model-based approaches improve data efficiency by observing that if the system dynamics are known, the state derivative, and hence the BE, can be evaluated at any desired point in the state space [15].

Methods that seek online solutions to optimal control problems are comparable to adaptive control (cf., [2, 7, 9, 11, 16, 17] and the references therein), where the estimates for the uncertain parameters in the plant model are updated using the tracking error as a performance metric. Similarly, in approximate dynamic programming (ADP), the BE is used as a performance metric. Parameter convergence has long been a focus of research in adaptive control. To establish regulation or tracking, adaptive control methods do not require the adaptive estimates to converge to the true values. However, convergence of the RL-based controller to a neighborhood of the optimal controller requires convergence of the parameter estimates to a neighborhood of their ideal values.

Least squares and gradient descent-based update laws are used in RL-based techniques to solve optimal control problems online [15, 18]. Such update laws generally require persistence of excitation (PE) in the system state for parameter estimate convergence. Hence, the challenges are that the updated law must be PE and the system trajectory needs to visit enough points in the state space to generate a sufficient approximation of the value function over the entire domain of operation. These chal-

lenges are often addressed in the related literature (cf. [4, 7, 9, 19–25]) by adding an exploration signal to the control input. However, no analytical methods exist to compute the appropriate exploration signal when the system dynamics are nonlinear.

Unlike results requiring PE, this chapter discusses model-based approaches used to mitigate the need to inject probing signals into the system to facilitate learning. In Sect. 5.2, the infinite horizon optimal control problem is introduced along with conditions which establish the optimal control policy. It is shown that the value function is the optimal cost-to-go and satisfies the HJB equation. In Sect. 5.3, the regional model-based RL (R-MBRL) method is presented where unknown weights in the value function are adjusted based on least square minimization of the BE evaluated at any number of user-selected arbitrary trajectories in the state space. Since the BE can be evaluated at any desired point in the state space, sufficient exploration is achieved by selecting points distributed over the system’s operating domain. R-MBRL establishes online approximate learning of the optimal controller while maintaining overall system stability. In Sect. 5.4, the local state following MBRL (StaF-RL) method is presented where the computational complexity of MBRL problems is reduced by estimating the optimal value function within a local domain around the state. A reduction in the computational complexity via StaF-RL is achieved by reducing the number of basis functions required for approximation. While the StaF method is computationally efficient, it lacks memory, i.e., the information about the value function in a region is lost once the system state leaves that region. That is, since StaF-RL estimates the value function in a local domain around the state, the value function approximation is a local solution. In Sect. 5.5, a strategy that uses R-MBRL and StaF-RL together to approximate the value function is described. This technique eliminates the need to perform BE extrapolation over a large region of the state space, as in R-MBRL, and the inability for the StaF method to develop a global estimate of the value function. Specifically, using knowledge about where the system is to converge, a R-MBRL approximation is used in the regional neighborhood to maintain an accurate approximation of the value function of the goal location. Moreover, to ensure stability of the system before entering the regional neighborhood, StaF-RL is leveraged to guide the system to the regional neighborhood. In Sect. 5.6, a strategy is described to overcome the computational cost of R-MBRL by using a set of sparse off-policy trajectories, which are used to calculate extrapolated BEs. Furthermore, the state-space is divided into a user-selected number of segments. Drawing inspiration from the approach in Sect. 5.5, a certain set of trajectories, and, hence, sets of extrapolated BEs, can be marked active when the state enters the corresponding segment, i.e., the only active set of extrapolated BEs are those that belong to the same segment as the current trajectory. Sparse neural networks (SNNs) could then be used within each segment to extrapolate the BE due to their small amount of active neurons, whose activity can be switched on or off based on the active segment, to make BE extrapolation more computationally efficient.

## 5.2 Unconstrained Affine-Quadratic Regulator

Consider a controlled dynamical system described by the initial value problem

$$\dot{x} = f(x, u, t), \quad x(t_0) = x_0, \quad (5.1)$$

where  $t_0$  is the initial time,  $x \in \mathbb{R}^n$  denotes the system state,  $u \in U \subset \mathbb{R}^m$  denotes the control input, and  $U$  denotes the action space. Consider a family (parameterized by  $t$ ) of optimal control problems described by the cost functionals

$$J(t, y, u(\cdot)) = \int_t^\infty L(\phi(\tau; t, y, u(\cdot)), u(\tau), \tau) d\tau, \quad (5.2)$$

where  $L : \mathbb{R}^n \times U \times \mathbb{R}_{\geq 0} \rightarrow \mathbb{R}$  is the Lagrange cost, with  $L(x, u, t) \geq 0$ , for all  $(x, u, t) \in \mathbb{R}^n \times U \times \mathbb{R}_{\geq 0}$ , and the notation  $\phi(\tau; t, y, u(\cdot))$  is used to denote a trajectory of the system in (5.1), evaluated at time  $\tau$ , under the controller  $u : \mathbb{R}_{\geq t_0} \rightarrow U$ , starting at the initial time  $t$ , and with the initial state  $y$ . The short notation  $x(\tau)$  is used to denote  $\phi(\tau; t, y, u(\cdot))$  when the controller, the initial time, and the initial state are clear from the context. Throughout this discussion, it is assumed that the controllers and the dynamical systems are such that the initial value problem in (5.1) admits a unique complete solution starting from any initial condition.

Let the optimal value function  $V^* : \mathbb{R}^n \times \mathbb{R}_{\geq 0} \rightarrow \mathbb{R}$  be defined as

$$V^*(x, t) := \inf_{u_{[t, \infty)} \in \mathcal{U}_{(t, x)}} J(t, x, u(\cdot)), \quad (5.3)$$

where the notation  $u_{[t, \infty)}$  for  $t \geq t_0$  denotes the controller  $u(\cdot)$  restricted to the time interval  $[t, \infty)$  and  $\mathcal{U}_{(t, x)}$  denotes the set of controllers that are admissible for  $x$ . The following theorem is a generalization of [26, Theorem 1.2] to infinite horizon problems.

**Theorem 5.1** *Given  $t_0 \in \mathbb{R}_{\geq 0}$ ,  $x_0 \in \mathbb{R}^n$ , let  $\mathcal{U}_{(t_0, x_0)}$  include all Lebesgue measurable locally bounded controllers so that the initial value problem in (5.1) admits a unique complete solution starting from  $(t_0, x_0)$ . Assume  $V^* \in \mathcal{C}^1(\mathbb{R}^n \times \mathbb{R}_{\geq t_0}, \mathbb{R})$ . If there exists a function  $V : \mathbb{R}^n \times \mathbb{R}_{\geq t_0} \rightarrow \mathbb{R}$  such that*

1.  $V \in \mathcal{C}^1(\mathbb{R}^n \times \mathbb{R}_{\geq t_0}, \mathbb{R})$  and  $V$  satisfies the HJB equation

$$0 = -\nabla_t V(x, t) - \inf_{\mu \in U} \{L(x, \mu, t) + V'^T(x, t) f(x, \mu, t)\}, \quad (5.4)$$

for all  $t \in [t_0, \infty)$  and all  $x \in \mathbb{R}^n$ ,

<sup>1</sup>The notation  $\nabla_x h(x, y, t)$  denotes the partial derivative of generic function  $h(x, y, t)$  with respect to generic variable  $x$ . The notation  $h'(x, y)$  denotes the gradient with respect to the first argument of the generic function,  $h(\cdot, \cdot)$ , e.g.,  $h'(x, y) = \nabla_x h(x, y)$ .

2. for every controller  $v(\cdot) \in \mathcal{U}_{(t_0, x_0)}$  for which there exists  $M_v \geq 0$  so that  $\int_{t_0}^t L(\phi(\tau, t_0, x_0, v(\cdot)), v(\tau), \tau) d\tau \leq M_v$  for all  $t \in \mathbb{R}_{\geq t_0}$ , the function  $V$ , evaluated along the resulting trajectory, satisfies

$$\lim_{t \rightarrow \infty} V(\phi(t; t_0, x_0, v(\cdot))) = 0, \quad (5.5)$$

and

3. there exists  $u(\cdot) \in \mathcal{U}_{(t_0, x_0)}$ , such that the function  $V$ , the controller  $u(\cdot)$ , and the trajectory  $x(\cdot)$  of (5.1) under  $u(\cdot)$  with the initial condition  $x(t_0) = x_0$ , satisfy, the Hamiltonian minimization condition

$$\begin{aligned} & L(x(t), u(t), t) + V'^T(x(t), t) f(x(t), u(t), t) \\ &= \min_{\mu \in U} \{L(x(t), \mu, t) + V'^T(x(t), t) f(x(t), \mu, t)\}, \quad \forall t \in \mathbb{R}_{\geq t_0}, \end{aligned} \quad (5.6)$$

and the bounded cost condition

$$\exists M_u \geq 0 \quad | \quad \int_{t_0}^t L(x(\tau), v(\tau), \tau) d\tau \leq M_u, \quad \forall t \in \mathbb{R}_{\geq t_0}, \quad (5.7)$$

then,  $V(t_0, x_0)$  is the optimal cost (i.e.,  $V(t_0, x_0) = V^*(t_0, x_0)$ ) and  $u(\cdot)$  is an optimal controller.

**Proof** Let  $x(\cdot) \triangleq \phi(\cdot; t_0, x_0, u(\cdot))$ , where  $u(\cdot)$  is an admissible controller that satisfies (5.6) and (5.7), and  $y(\cdot) \triangleq \phi(\cdot; t_0, x_0, v(\cdot))$  where  $v(\cdot)$  is any other admissible controller. The Hamiltonian minimization condition in (5.6) implies that along the trajectory  $x(\cdot)$ , the control  $\mu = u(t)$  achieves the infimum in (5.4) for all  $t \in \mathbb{R}_{\geq t_0}$ . Thus, along the trajectory  $x(\cdot)$ , (5.4) implies that

$$-\nabla_t V(x(t), t) - V'^T(x(t), t) f(x(t), u(t), t) = L(x(t), u(t), t).$$

That is,

$$-\frac{d}{dt} V(x(t), t) = L(x(t), u(t), t). \quad (5.8)$$

Since  $V$  satisfies the HJB equation everywhere, it is clear that along the trajectory  $y(\cdot)$ ,

$$\begin{aligned} & \inf_{\mu \in U} \{L(y(t), \mu, t) + V'^T(y(t), t) f(y(t), \mu, t)\} \\ & \leq L(y(t), v(t), t) + V'^T(y(t), t) f(y(t), v(t), t) \end{aligned}$$

and as a result, the HJB equation, evaluated along  $y(\cdot)$ , yields

$$0 \geq -\nabla_t V(y(t), t) - V'^T(y(t), t) f(y(t), v(t), t) - L(y(t), v(t), t).$$

That is,

$$-\frac{d}{dt}V(y(t), t) \leq L(y(t), v(t), t). \quad (5.9)$$

Integrating (5.8) and (5.9) over a finite interval  $[t_0, T]$ ,

$$\begin{aligned} -\int_{t_0}^T \frac{d}{dt}V(x(t), t) dt &= (V(x(t_0), t_0) - V(x(T), T)) \\ &= \int_{t_0}^T L(x(t), u(t), t) dt \end{aligned}$$

and

$$\begin{aligned} -\int_{t_0}^T \frac{d}{dt}V(y(t), t) dt &= (V(y(t_0), t_0) - V(y(T), T)) \\ &\leq \int_{t_0}^T L(y(t), v(t), t) dt. \end{aligned}$$

Since  $x(t_0) = y(t_0) = x_0$ , it can be concluded that

$$\begin{aligned} V(x_0, t_0) &= \int_{t_0}^T L(x(t), u(t), t) dt + V(x(T), T) \\ &\leq \int_{t_0}^T L(y(t), v(t), t) dt + V(y(T), T), \forall T \in \mathbb{R}_{\geq t_0}, \end{aligned}$$

and as a result,

$$\begin{aligned} V(x_0, t_0) &= \lim_{T \rightarrow \infty} \int_{t_0}^T L(x(t), u(t), t) dt + V(x(T), T) \\ &\leq \lim_{T \rightarrow \infty} \int_{t_0}^T L(y(t), v(t), t) dt + V(y(T), T). \end{aligned}$$

Since  $u(\cdot)$  satisfies (5.7) and  $(x, u, t) \mapsto L(x, u, t)$  is nonnegative,

$$\int_{t_0}^{\infty} L(x(t), u(t), t) dt$$

exists, is bounded, and equal to the total cost  $J(t_0, x_0, u(\cdot))$ . Taking (5.5) into account, it can thus be concluded that

$$\lim_{T \rightarrow \infty} \int_{t_0}^T L(x(t), u(t), t) dt + V(x(T), T) = J(t_0, x_0, u(\cdot)).$$

If  $v(\cdot)$  satisfies (5.7), then a similar analysis yields

$$\lim_{T \rightarrow \infty} \int_{t_0}^T L(y(t), v(t), t) dt + V(y(T), T) = J(t_0, x_0, v(\cdot)),$$

and as a result,

$$V(x_0, t_0) = J(t_0, x_0, u(\cdot)) \leq J(t_0, x_0, v(\cdot)). \quad (5.10)$$

If  $v(\cdot)$  does not satisfy (5.7), then nonnegativity of  $(x, u, t) \mapsto L(x, u, t)$  implies that the total cost resulting from  $v(\cdot)$  is unbounded, and (5.10) holds canonically. In conclusion,  $V(t_0, x_0)$  is the optimal cost (i.e.,  $V(t_0, x_0) = V^*(t_0, x_0)$ ) and  $u(\cdot)$  is an optimal controller.

For the remainder of this section, a controller  $v : \mathbb{R}_{\geq t_0} \rightarrow U$  is said to be admissible for a given initial state  $(t_0, x_0)$  if it is bounded, generates a unique bounded trajectory starting from  $x_0$ , and results in bounded total cost. An admissible controller that results in the smallest cost among all admissible controllers is called an optimal controller. The dynamics and the Lagrange cost are assumed to be time-invariant, and as a result, if  $v : \mathbb{R}_{\geq t_0} \rightarrow U$  is admissible for a given initial state  $(t_0, x_0)$ , then  $v' : \mathbb{R}_{\geq t_1} \rightarrow U$ , defined as  $v'(t) \triangleq v(t + t_0 - t_1)$ , for all  $t \in \mathbb{R}_{\geq t_1}$  is admissible for  $(t_1, x_0)$ , and trajectories of the system starting from  $(t_0, x_0)$  under  $v(\cdot)$  and those starting from  $(t_1, x_0)$  under  $v'(\cdot)$  are identical. As a result, the set of admissible controllers, system trajectories, value functions, and total costs can be considered independent of  $t_0$  without loss of generality. The following two theorems detail conditions under which value functions are characterized by the HJB equation.

**Theorem 5.2** Consider the optimal control problem

$$P : \min_{u(\cdot) \in \mathcal{U}_{x_0}} J(x_0, u(\cdot)) \triangleq \int_{t_0}^{\infty} r(\phi(\tau; t_0, x_0, u(\cdot)), u(\tau)) d\tau \quad (5.11)$$

subject to

$$\dot{x} = f(x) + g(x)u, \quad (5.12)$$

where the local cost  $r : \mathbb{R}^n \times \mathbb{R}^m \rightarrow \mathbb{R}$  is defined as  $r(x, u) \triangleq Q(x) + u^T R u$ , with  $Q : \mathbb{R}^n \rightarrow \mathbb{R}$ , a continuously differentiable positive definite function and  $R \in \mathbb{R}^{m \times m}$ , a symmetric positive definite matrix. Assume further that the optimal value function  $V^* : \mathbb{R}^n \rightarrow \mathbb{R}$  corresponding to  $P$  is continuously differentiable.

If  $x \mapsto V(x)$  is positive definite and satisfies the closed-loop HJB equation

$$r(x, \psi(x)) + V'(x)(f(x) + g(x)\psi(x)) = 0, \quad \forall x \in \mathbb{R}^n, \quad (5.13)$$

with

$$\psi(x) = -\frac{1}{2}R^{-1}g^T(x)(V'(x))^T, \quad (5.14)$$

then  $V(\cdot)$  is the optimal value function and the state feedback law  $u(t) = \psi(x(t))$  is the optimal controller.

**Proof** Note that (5.13) and the positive definiteness of  $Q$ ,  $R$ , and  $V$ , imply that under the state feedback law  $u(t) = \psi(x(t))$ , the closed-loop system  $\dot{x} = f(x) + g(x)\psi(x)$  is globally asymptotically stable. Furthermore, since  $V(0) = 0$ , every trajectory of the closed-loop system converges to the origin and since (5.13) holds for all  $x \in \mathbb{R}^n$ , and in particular, holds along every trajectory of the closed-loop system, it can be concluded that

$$\int_t^\infty r(x(\tau), \psi(x(\tau))) dt = V(x(t)) = J(x(t), \psi(x(\cdot))), \forall t \in \mathbb{R}$$

along every trajectory of the closed-loop system. As a result, all control signals resulting from the state-feedback law  $u(t) = \psi(x(t))$  are admissible for all initial conditions. For each  $x \in \mathbb{R}^n$  it follows that

$$\frac{\partial(r(x, u) + V'(x)(f(x) + g(x)u))}{\partial u} = 2u^T R + V'(x)g(x).$$

Hence,  $u = -\frac{1}{2}R^{-1}g^T(x)(V'(x))^T = \psi(x)$  extremizes

$$r(x, u) + V'(x)(f(x) + g(x)u).$$

Furthermore, the Hessian

$$\frac{\partial^2(r(x, u) + V'(x)(f(x) + g(x)u))}{\partial^2 u} = 2R$$

is positive definite. Hence,  $u = \psi(x)$  minimizes

$$u \mapsto r(x, u) + V'(x)(f(x) + g(x)u)$$

for each  $x \in \mathbb{R}^n$ .

As a result, the closed-loop HJB equation (5.13), along with the control law (5.14) are equivalent to the HJB equation (5.4). Furthermore, all trajectories starting from all initial conditions in response to the controller  $u(t) = \psi(x(t))$  satisfy the Hamiltonian minimization condition (5.6) and the bounded cost condition (5.7). In addition, given any initial condition  $x_0$  and a controller  $v(\cdot)$  that is admissible for  $x_0$ , boundedness of the controller  $v(\cdot)$  and the resulting trajectory  $\phi(\cdot; t_0, x_0, v(\cdot))$ , continuity of  $x \mapsto f(x, u)$  and  $x \mapsto g(x, u)$ , and continuity of  $x \mapsto Q'(x)$  can be used to conclude that  $t \mapsto Q(\phi(t; t_0, x_0, v(\cdot)))$  is uniformly continuous.

Admissibility of  $v(\cdot)$  and positive definiteness of  $R$  imply that

$$\left| \int_{t_0}^T Q(\phi(t; t_0, x_0, v(\cdot))) dt \right| \leq M$$

for all  $T \geq t_0$  and some  $M \geq 0$ . Furthermore, positive definiteness of  $x \mapsto Q(x)$  implies monotonicity of  $T \mapsto \int_{t_0}^T Q(\phi(t; t_0, x_0, v(\cdot))) dt$ . As a result, the limit  $\lim_{T \rightarrow \infty} \int_{t_0}^T Q(\phi(t; t_0, x_0, v(\cdot))) dt$  exists and is bounded.

By Barbalat's lemma,  $\lim_{t \rightarrow \infty} Q(\phi(t; t_0, x_0, v(\cdot))) = 0$ , which, due to positive definiteness and continuity of  $x \mapsto Q(x)$  implies that

$$\lim_{t \rightarrow \infty} \phi(t; t_0, x_0, v(\cdot)) = 0,$$

and finally, from continuity and positive definiteness of  $V$ ,

$$\lim_{t \rightarrow \infty} V(\phi(t; t_0, x_0, v(\cdot))) = 0,$$

which establishes (5.5).

Arguments similar to the proof of Theorem 5.1 can then be invoked to conclude that  $V(x_0)$  is the optimal cost and  $u(t) = \psi(x(t))$  is the unique optimal controller among all admissible controllers. Since the initial condition was arbitrary, the proof of Theorem 5.2 is complete.

To facilitate the following discussion, let  $\mathcal{U}_{x,[t_1,t_2]}$  denote the space of controllers that are restrictions over  $[t_1, t_2]$  of controllers admissible for  $x$ . The task is then to show that value functions satisfy HJB equations.

**Theorem 5.3** Consider the optimal control problem  $P$  stated in Theorem 5.2 and assume that for every initial condition  $x_0$ , an optimal controller that is admissible for  $x_0$  exists. If the optimal value function corresponding to  $P$ , defined as

$$V^*(x) \triangleq \inf_{u(\cdot) \in \mathcal{U}_x} \int_t^\infty r(\phi(\tau; t, x, u(\cdot)), u(\tau)) d\tau, \quad (5.15)$$

is continuously differentiable then it satisfies the HJB equation

$$r(x, \psi(x)) + V^{*\prime}(x) (f(x) + g(x) \psi^*(x)) = 0, \quad \forall x \in \mathbb{R}^n, \quad (5.16)$$

with

$$\psi^*(x) = -\frac{1}{2} R^{-1} g^T(x) (V^{*\prime}(x))^T \quad (5.17)$$

being the optimal feedback policy.

**Proof** First, it is shown that the value function satisfies the principle of optimality. To facilitate the discussion, given  $x \in \mathbb{R}^n$ , let  $v_{(x,t)}^* : \mathbb{R}_{\geq t} \rightarrow U$  denote an optimal controller starting from the initial state  $x$  and initial time  $t$ .

*Claim* (Principle of optimality under admissibility restrictions) For all  $x \in \mathbb{R}^n$ , and for all  $\Delta t > 0$ ,

$$V^*(x) = \inf_{u(\cdot) \in \mathcal{U}_{x,[t,t+\Delta t]}} \left\{ \int_t^{t+\Delta t} r(\phi(\tau; t, x, u(\cdot)), u(\tau)) d\tau + V^*(x(t + \Delta t)) \right\}. \quad (5.18)$$

**Proof** Consider the function  $V : \mathbb{R}^n \rightarrow \mathbb{R}$  defined as

$$V(x) \triangleq \inf_{u(\cdot) \in \mathcal{U}_{x,[t,t+\Delta t]}} \left\{ \int_t^{t+\Delta t} r(\phi(\tau; t, x, u(\cdot)), u(\tau)) d\tau + V^*(x(t + \Delta t)) \right\}.$$

Based on the definition in (5.15)

$$V(x) = \inf_{u(\cdot) \in \mathcal{U}_{x,[t,t+\Delta t]}} \left\{ \int_t^{t+\Delta t} r(\phi(\tau; t, x, u(\cdot)), u(\tau)) d\tau + \inf_{v(\cdot) \in \mathcal{U}_{x(t+\Delta t)}} \int_{t+\Delta t}^{\infty} r(\phi(\tau; t, x(t + \Delta t), v(\cdot)), v(\tau)) d\tau \right\}.$$

Since the first integral is independent of the control over  $\mathbb{R}_{\geq t+\Delta t}$ ,

$$V(x) = \inf_{u(\cdot) \in \mathcal{U}_{x,[t,t+\Delta t]}} \inf_{v(\cdot) \in \mathcal{U}_{x(t+\Delta t)}} \left\{ \int_t^{t+\Delta t} r(\phi(\tau; t, x, u(\cdot)), u(\tau)) d\tau + \int_{t+\Delta t}^{\infty} r(\phi(\tau; t, x(t + \Delta t), v(\cdot)), v(\tau)) d\tau \right\}.$$

Combining the integrals and using the fact that concatenation of admissible restrictions and admissible controllers result in admissible controllers,  $\inf_{u(\cdot) \in \mathcal{U}_{x,[t,t+\Delta t]}} \inf_{v(\cdot) \in \mathcal{U}_{x(t+\Delta t)}}$  is equivalent to  $\inf_{w(\cdot) \in \mathcal{U}_x}$ , where  $w : \mathbb{R}_{\geq t} \rightarrow U$  is defined as  $w(\tau) := \begin{cases} u(\tau) & t \leq \tau \leq t + \Delta t, \\ v(\tau) & \tau > t + \Delta t, \end{cases}$  it can be concluded that

$$V(x) = \inf_{w(\cdot) \in \mathcal{U}_x} \int_t^{\infty} r(\phi(\tau; t, x, w(\cdot)), w(\tau)) d\tau = V^*(x).$$

Thus,

$$V(x) \geq V^*(x). \quad (5.19)$$

On the other hand, by the definition of the infimum, for all  $\epsilon > 0$ , there exists an admissible controller  $u_\epsilon(\cdot)$  such that

$$V^*(x) + \epsilon \geq J(x, u_\epsilon(\cdot)).$$

Let  $x_\epsilon : \mathbb{R}_{\geq t_0} \rightarrow \mathbb{R}^n$  denote the trajectory corresponding to  $u_\epsilon(\cdot)$ . Since the restriction  $u_{\epsilon, \mathbb{R}_{\geq t_1}}(\cdot)$  of  $u_\epsilon(\cdot)$  to  $\mathbb{R}_{\geq t_1}$  is admissible for  $x_\epsilon(t_1)$  for all  $t_1 > t_0$ ,

$$\begin{aligned} J(x, u_\epsilon(\cdot)) &= \int_t^{t+\Delta t} r(x_\epsilon(\tau), u_\epsilon(\tau)) d\tau + J(x_\epsilon(t + \Delta t), u_{\epsilon, \mathbb{R}_{\geq t+\Delta t}}(\cdot)) \\ &\geq \int_t^{t+\Delta t} r(x_\epsilon(\tau), u_\epsilon(\tau)) d\tau + V^*(x_\epsilon(t + \Delta t)) \geq V(x). \end{aligned}$$

Thus,  $V(x) \leq V^*(x)$ , which, along with (5.19), implies  $V(x) = V^*(x)$ .

Since  $V^* \in C^1(\mathbb{R}^n, \mathbb{R})$ , given any admissible  $u(\cdot)$  and corresponding trajectory  $x(\cdot)$ ,

$$V^*(x(t + \Delta t)) = V^*(x) + V^{*\prime}(x)((f(x) + g(x)u(t))\Delta t) + o(\Delta t).$$

Furthermore,

$$\int_t^{t+\Delta t} r(x(\tau), u(\tau)) d\tau = r(x, u(t))\Delta t + o(\Delta t).$$

From the principle of optimality in (5.18),

$$\begin{aligned} V^*(x) &= \inf_{u(\cdot) \in \mathcal{U}_{x, [t, t+\Delta t]}} \{r(x, u(t))\Delta t + V^*(x) \\ &\quad + V^{*\prime}(x)((f(x) + g(x)u(t))\Delta t) + o(\Delta t)\}. \end{aligned}$$

That is,

$$\begin{aligned} 0 &= \inf_{u(\cdot) \in \mathcal{U}_{x, [t, t+\Delta t]}} \{r(x, u(t))\Delta t \\ &\quad + V^{*\prime}(x)((f(x) + g(x)u(t))\Delta t) + o(\Delta t)\}. \end{aligned}$$

Dividing by  $\Delta t$  and taking the limit as  $\Delta t$  goes to zero,

$$0 = \inf_{u \in U} \{r(x, u) + V^{*'}(x)(f(x) + g(x)u)\}, \quad \forall x \in \mathbb{R}^n.$$

In conclusion, under the assumptions made in this section, the optimal value function is continuously differentiable, positive definite, and satisfies the HJB equation. All functions that are continuously differentiable and positive definite and satisfy the HJB equation are optimal value functions, and optimal value functions are, by definition, unique. As a result, if there is a continuously differentiable and positive definite solution of the HJB equation then it is unique and is also the optimal value function.

### 5.3 Regional Model-Based Reinforcement Learning

The following section examines the dynamical system in (5.1) and a controller,  $u$ , is designed to solve the infinite horizon optimal control problem via a R-MBRL approach. The R-MBRL technique, described in detail in [15], uses a data-based approach to improve data efficiency by observing that if the system dynamics are known, the state derivative, and hence, the BE can be evaluated at any desired point in the state space. Unknown parameters in the value function can therefore be adjusted based on least square minimization of the BE evaluated at any number of arbitrary points in the state space. For instance, in an infinite horizon regulation problem, the BE can be computed at points uniformly distributed in a neighborhood around the origin of the state space. Convergence of the unknown parameters in the value function is guaranteed provided the selected points satisfy a rank condition. Since the BE can be evaluated at any desired point in the state space, sufficient exploration can be achieved by appropriately selecting the points to cover the domain of operation.

If each new evaluation of the BE along the system trajectory is interpreted as gaining experience via exploration, the use of a model to evaluate the BE at an unexplored point in the state space can be interpreted as a simulation of the experience. Learning based on simulation of experience has been investigated in results such as [27–32] for stochastic model-based RL; however, these results solve the optimal control problem offline in the sense that repeated learning trials need to be performed before the algorithm learns the controller, and system stability during the learning phase is not analyzed.

The following subsections explore nonlinear, control affine plants and provides an online solution to a deterministic infinite horizon optimal regulation problems by leveraging BE extrapolation.

### 5.3.1 Preliminaries

Consider a control affine nonlinear dynamical system in (5.12).<sup>2</sup>

**Assumption 5.1** The drift dynamic,  $f$ , is a locally Lipschitz function such that  $f(0) = 0$ , and the control effectiveness,  $g$ , is a known bounded locally Lipschitz function. Furthermore,  $f' : \mathbb{R}^n \rightarrow \mathbb{R}^{n \times n}$  is continuous.

The control objective is to solve the infinite horizon optimal regulation problem online, i.e., to design a control signal  $u$  online to minimize the cost function in (5.11) under the dynamic constraint in (5.12) while regulating the system state to the origin.

It is well known that since the functions  $f$ ,  $g$ , and  $Q$  are stationary (time-invariant) and the time horizon is infinite, the optimal control input is a stationary state feedback policy. Furthermore, the value function is also a stationary function [33, Eq. 5.19]. Hence, the optimal value function  $V^* : \mathbb{R}^n \rightarrow \mathbb{R}_{\geq 0}$  can be expressed in (5.15) while regulating the system states to the origin (i.e.,  $x = 0$ ) for  $\tau \in \mathbb{R}_{\geq t}$ , with  $u(\tau) \in \mathbb{U} | \tau \in \mathbb{R}_{\geq t}$ , where  $\mathbb{U} \in \mathbb{R}^m$  denotes the set of admissible inputs, which, by Theorem 5.2, are admissible for all initial conditions. In (5.15),  $r : \mathbb{R}^n \times \mathbb{R}^m \rightarrow \mathbb{R}_{\geq 0}$  denotes the instantaneous cost defined as  $r(x, u) \triangleq x^T Q x + u^T R u$ , where  $Q \in \mathbb{R}^{n \times n}$  and  $R \in \mathbb{R}^{m \times m}$  are constant positive definite (PD) symmetric matrices.

**Property 5.1** The PD matrix  $Q$  satisfies  $\underline{q} I_n \leq Q \leq \bar{q} I_n$  for  $\underline{q}, \bar{q} \in \mathbb{R}_{>0}$ .<sup>3</sup>

Provided the assumptions and conditions in Sect. 5.2 regarding a unique solution hold, the optimal value function,  $V^*$ , is the solution to the corresponding HJB equation in (5.16) with boundary condition  $V^*(0) = 0$  [34, Sect. 3.11]. From Theorems 5.2 and 5.3, provided the HJB in (5.16) admits a continuously differentiable and positive definite solution, it constitutes a necessary and sufficient condition for optimality. The optimal control policy,  $u^* : \mathbb{R}^n \rightarrow \mathbb{R}^m$ , is defined as

$$u^* = -\frac{1}{2} R^{-1} g^T(x) (V'^*(x))^T. \quad (5.20)$$

### 5.3.2 Regional Value Function Approximation

Approximations of the optimal value function,  $V^*$ , and the optimal policy,  $u^*$ , are designed based on neural network (NN) representations. Given any compact set  $\chi \subset \mathbb{R}^n$  and a positive constant  $\bar{\epsilon} \in \mathbb{R}$ , the universal function approximation property of NNs can be exploited to represent the optimal value function as

---

<sup>2</sup>For notational brevity, unless otherwise specified, the domain of all the functions is assumed to be  $\mathbb{R}_{\geq 0}$ , where  $\mathbb{R}_{\geq a}$  denotes the interval  $[a, \infty)$ . The notation  $\|\cdot\|$  denotes the Euclidean norm for vectors and the Frobenius norm for matrices.

<sup>3</sup>The notation  $I_n$  denotes the  $n \times n$  identity matrix.

$$V^*(x) = W^T \sigma(x) + \epsilon(x), \quad (5.21)$$

for all  $x \in \chi$ , where  $W \in \mathbb{R}^L$  is the ideal weight matrix bounded above by a known positive constant  $\bar{W}$  in the sense that  $\|W\| \leq \bar{W}$ ,  $\sigma : \mathbb{R}^n \rightarrow \mathbb{R}^L$  is a continuously differentiable nonlinear activation function such that  $\sigma(0) = 0$  and  $\sigma'(0) = 0$ ,  $L \in \mathbb{N}$  is the number of neurons, and  $\epsilon : \mathbb{R}^n \rightarrow \mathbb{R}$  is the function reconstruction error such that  $\sup_{x \in \chi} |\epsilon(x)| \leq \bar{\epsilon}$  and  $\sup_{x \in \chi} |\epsilon'(x)| \leq \bar{\epsilon}$ .

Using Assumptions 5.1, conditions in Sect. 5.2, and based on the NN representation of the value function, a NN representation of the optimal controller is derived from (5.20), where  $\hat{V} : \mathbb{R}^n \times \mathbb{R}^L \rightarrow \mathbb{R}$  and  $\hat{u} : \mathbb{R}^n \times \mathbb{R}^L \rightarrow \mathbb{R}^m$  denote value function and controller estimates defined as

$$\hat{V}(x, \hat{W}_c) \triangleq \hat{W}_c^T \sigma(x), \quad (5.22)$$

$$\hat{u}(x, \hat{W}_a) \triangleq -\frac{1}{2} R^{-1} g^T(x) (\sigma'(x))^T \hat{W}_a. \quad (5.23)$$

In (5.22) and (5.23),  $\hat{W}_c \in \mathbb{R}^L$  and  $\hat{W}_a \in \mathbb{R}^L$  denote the critic and actor estimates of  $W$ , respectively. The use of two sets of weights to estimate the same set of ideal weights is motivated by the stability analysis and the fact that it enables a formulation of the BE that is linear in the critic weight estimates  $\hat{W}_c$ , enabling a least squares-based adaptive update law.

### 5.3.3 Bellman Error

In traditional RL-based algorithms, the value function and policy estimates are updated based on observed data. The use of observed data to learn the value function naturally leads to a sufficient exploration condition which demands sufficient richness in the observed data. In stochastic systems, this is achieved using a randomized stationary policy (cf., [6, 35, 36]), whereas in deterministic systems, a probing noise is added to the derived control law (cf., [7, 9, 37–39]).

Learning-based techniques often require PE to achieve convergence. The PE condition is relaxed in [24] to a finite excitation condition by using integral RL along with experience replay, where each evaluation of the BE along the system trajectory is interpreted as gained experience. These experiences are stored in a history stack and are repeatedly used in the learning algorithm to improve data efficiency. In this chapter, a different approach is used to circumvent the PE condition. Using (5.22) and (5.23) in (5.13) results in the BE  $\delta : \mathbb{R}^n \times \mathbb{R}^L \times \mathbb{R}^L \rightarrow \mathbb{R}$ , defined as

$$\begin{aligned} \delta(x, \hat{W}_c, \hat{W}_a) &\triangleq \hat{V}'(x, \hat{W}_c) (f(x) + g(x) \hat{u}(x, \hat{W}_a)) \\ &\quad + \hat{u}(x, \hat{W}_a)^T R \hat{u}(x, \hat{W}_a) + x^T Q x. \end{aligned} \quad (5.24)$$

Given a model of the system and the current parameter estimates  $\hat{W}_c(t)$  and  $\hat{W}_a(t)$ , the BE in (5.24) can be evaluated at any point  $x_i \in \chi$ . The critic can gain experience on how well the value function is estimated at any arbitrary point  $x_i$  in the state space without actually visiting  $x_i$ . Given a fixed state  $x_i$  and a corresponding planned action  $\hat{u}(x_i, \hat{W}_a)$ , the critic can use the dynamic model to simulate a visit to  $x_i$  by computing the state derivative at  $x_i$ . This results in simulated experience quantified by the BE. The technique developed in this section implements simulation of experience in a model-based RL scheme by extrapolating the approximate BE to a user-specified set of trajectories  $\{x_i \in \mathbb{R}^n \mid i = 1, \dots, N\}$  in the state space. The BE in (5.24) is evaluated along the trajectories of (5.12) to get the instantaneous BE  $\delta_t : \mathbb{R}_{\geq t_0} \rightarrow \mathbb{R}$  defined as  $\delta_t(t) \triangleq \delta(x(t), \hat{W}_c(t), \hat{W}_a(t))$ . Moreover, extrapolated trajectories  $\{x_i \in \mathbb{R}^n \mid i = 1, \dots, N\}$  are leveraged to generate an extrapolated BE  $\delta_{ti} : \mathbb{R}_{\geq t_0} \rightarrow \mathbb{R}$  defined as  $\delta_{ti}(t) \triangleq \delta(x_i, \hat{W}_c(t), \hat{W}_a(t))$ .

Defining the mismatch between the estimates and the ideal values as  $\tilde{W}_c \triangleq W - \hat{W}_c$  and  $\tilde{W}_a \triangleq W - \hat{W}_a$ , substituting (5.20) and (5.21) in (5.13), and subtracting from (5.24) yields the analytical BE given by

$$\delta = \omega^T \tilde{W}_c + \frac{1}{4} \tilde{W}_a^T G_\sigma \tilde{W}_a + O(\varepsilon), \quad (5.25)$$

where  $\omega : \mathbb{R}^n \times \mathbb{R}^L \rightarrow \mathbb{R}^n$  is defined as

$$\omega(x, \hat{W}_a) \triangleq \sigma'(x) \left( f(x) + g(x) \hat{u}(x, \hat{W}_a) \right),$$

and  $O(\varepsilon) \triangleq \frac{1}{4} G_\varepsilon - \varepsilon' f + \frac{1}{2} W^T \sigma' G \varepsilon'^T$ .<sup>4</sup> Since the HJB in (5.13) is equal to zero for all  $x \in \mathbb{R}^n$ , the aim is to find critic and actor weight estimates,  $\hat{W}_c$  and  $\hat{W}_a$ , respectively, such that  $\hat{\delta} \rightarrow 0$  as  $t \rightarrow \infty$ . Intuitively, the state trajectory,  $x$ , needs to visit as many points in the operating domain as possible to approximate the optimal value function over an operating domain. The simulated experience is then used along with gained experience by the critic to approximate the value function.

### 5.3.3.1 Extension to Unknown Dynamics

If a system model is available, then the approximate optimal control technique can be implemented using the model. However, if an exact model of the system is unavailable, then parametric system identification can be employed to generate an estimate of the system model. A possible approach is to use parameters that are estimated offline in a separate experiment. A more useful approach is to use the offline esti-

---

<sup>4</sup>The notation  $G$ ,  $G_\sigma$ , and  $G_\varepsilon$  is defined as  $G = G(x) \triangleq g(x) R^{-1} g^T(x)$ ,  $G_\sigma = G_\sigma \triangleq \sigma'(x) G(x) \sigma'(x)^T$ , and  $G_\varepsilon = G_\varepsilon(x) \triangleq \varepsilon'(x) G(x) \varepsilon'(x)^T$ , respectively.

mate as the initial guess and to employ a dynamic system identification technique capable of refining the initial guess based on input–output data.

To facilitate online system identification, let  $f(x) = Y(x)\theta$  denote the linear parametrization of the function  $f$ , where  $Y : \mathbb{R}^n \rightarrow \mathbb{R}^{n \times p}$  is the regression matrix and  $\theta \in \mathbb{R}^p$  is the vector of constant unknown parameters. Let  $\hat{\theta} \in \mathbb{R}^p$  be an estimate of the unknown parameter vector  $\theta$ . The following development assumes that an adaptive system identifier that satisfies conditions detailed in Assumption 5.2 is available.

**Assumption 5.2** A compact set  $\Theta \subset \mathbb{R}^p$  such that  $\theta \in \Theta$  is known a priori. The estimates  $\hat{\theta} : \mathbb{R}_{\geq t_0} \rightarrow \mathbb{R}^p$  are updated based on a switched update law of the form

$$\dot{\hat{\theta}}(t) = f_{\theta s}(\hat{\theta}(t), t), \quad (5.26)$$

$\hat{\theta}(t_0) = \hat{\theta}_0 \in \Theta$ , where  $s \in \mathbb{N}$  denotes the switching index and  $\{f_{\theta s} : \mathbb{R}^p \times \mathbb{R}_{\geq 0} \rightarrow \mathbb{R}^p\}_{s \in \mathbb{N}}$  denotes a family of continuously differentiable functions. The dynamics of the parameter estimation error  $\tilde{\theta} : \mathbb{R}_{\geq t_0} \rightarrow \mathbb{R}^p$ , defined as  $\tilde{\theta}(t) \triangleq \theta - \hat{\theta}(t)$  can be expressed as  $\dot{\tilde{\theta}}(t) = f_{\theta s}(\theta - \tilde{\theta}(t), t)$ . Furthermore, there exists a continuously differentiable function  $V_\theta : \mathbb{R}^p \times \mathbb{R}_{\geq 0} \rightarrow \mathbb{R}_{\geq 0}$  that satisfies

$$\underline{v}_\theta(\|\tilde{\theta}\|) \leq V_\theta(\tilde{\theta}, t) \leq \bar{v}_\theta(\|\tilde{\theta}\|), \quad (5.27)$$

$$V'_\theta(\tilde{\theta}, t)(-f_{\theta s}(\theta - \tilde{\theta}, t)) + \frac{\partial V_\theta(\tilde{\theta}, t)}{\partial t} \leq -K \|\tilde{\theta}\|^2 + D \|\tilde{\theta}\|, \quad (5.28)$$

for all  $s \in \mathbb{N}$ ,  $t \in \mathbb{R}_{\geq t_0}$ , and  $\tilde{\theta} \in \mathbb{R}^p$ , where  $\underline{v}_\theta$ ,  $\bar{v}_\theta : \mathbb{R}_{\geq 0} \rightarrow \mathbb{R}_{\geq 0}$  are class  $\mathcal{K}$  functions,  $K \in \mathbb{R}_{>0}$  is an adjustable parameter, and  $D \in \mathbb{R}_{>0}$  is a positive constant.<sup>5</sup>

Using an estimate  $\hat{\theta}$ , the BE in (5.24) can be approximated by  $\hat{\delta} : \mathbb{R}^{n+2L+p} \rightarrow \mathbb{R}$  as

$$\begin{aligned} \hat{\delta}(x, \hat{W}_c, \hat{W}_a, \hat{\theta}) &= x^T Q x + \hat{u}^T(x, \hat{W}_a) R \hat{u}(x, \hat{W}_a) \\ &\quad + \hat{V}'(x, \hat{W}_c)(Y(x)\hat{\theta} + g(x)\hat{u}(x, \hat{W}_a)). \end{aligned} \quad (5.29)$$

In the following, the approximate BE in (5.29) is used to obtain an approximate solution to the HJB equation in (5.13).

---

<sup>5</sup>The subsequent analysis in Sect. 5.3.5 indicates that when a system identifier that satisfies Assumption 5.2 is employed to facilitate online optimal control, the ratio  $\frac{D}{K}$  needs to be sufficiently small to establish set-point regulation and convergence to optimality.

### 5.3.4 Actor and Critic Update Laws

A least squares update law for the critic weights is designed based on the stability analysis in [15] as

$$\dot{\hat{W}}_c(t) = -\eta_{c1}\Gamma \frac{\omega(t)}{\rho(t)} \hat{\delta}_t(t) - \frac{\eta_{c2}}{N} \Gamma \sum_{i=1}^N \frac{\omega_i(t)}{\rho_i(t)} \hat{\delta}_{ti}(t), \quad (5.30)$$

$$\dot{\Gamma}(t) = \left( \beta\Gamma(t) - \eta_{c1} \frac{\Gamma(t)\omega(t)\omega(t)^T\Gamma(t)}{\rho^2(t)} \right) \mathbf{1}_{\{\|\Gamma\| \leq \bar{\Gamma}\}}, \quad (5.31)$$

where  $\Gamma : \mathbb{R}_{\geq t_0} \rightarrow \mathbb{R}^{L \times L}$  is a time-varying least squares gain matrix,  $\|\Gamma(t_0)\| \leq \bar{\Gamma}$ ,  $\omega(t) \triangleq \omega(x_i, \hat{W}_a(t))$ ,  $\omega_i(t) \triangleq \omega(x(t), \hat{W}_a(t))$ ,  $\rho(t) \triangleq 1 + \nu\omega^T(t)\Gamma(t)\omega(t)$ , and  $\rho_i(t) \triangleq 1 + \nu\omega_i^T(t)\Gamma(t)\omega_i(t)$ . In addition,  $\nu \in \mathbb{R}_{>0}$  is a constant normalization gain,  $\bar{\Gamma} \in \mathbb{R}_{>0}$  is a saturation constant,  $\beta \in \mathbb{R}_{>0}$  is a constant forgetting factor, and  $\eta_{c1}, \eta_{c2} \in \mathbb{R}_{>0}$  are constant adaptation gains.

Motivate by the subsequent stability analysis, the actor weights are updated as

$$\begin{aligned} \dot{\hat{W}}_a(t) &= -\eta_{a1} \left( \hat{W}_a(t) - \hat{W}_c(t) \right) - \eta_{a2} \hat{W}_a(t) + \frac{\eta_{c1} G_\sigma^T(t) \hat{W}_a(t) \omega^T(t)}{4\rho(t)} \hat{W}_c(t) \\ &\quad + \sum_{i=1}^N \frac{\eta_{c2} G_{\sigma i}^T \hat{W}_a(t) \omega_i^T(t)}{4N\rho_i(t)} \hat{W}_c(t), \end{aligned} \quad (5.32)$$

where  $\eta_{a1}, \eta_{a2} \in \mathbb{R}_{>0}$  are constant adaptation gains and  $G_{\sigma i} \triangleq G_\sigma(x_i)$ . The update law in (5.31) ensures that the adaptation gain matrix is bounded such that

$$\underline{\Gamma} \leq \|\Gamma(t)\| \leq \bar{\Gamma}, \quad \forall t \in \mathbb{R}_{\geq t_0}. \quad (5.33)$$

Using the weight estimates  $\hat{W}_a$ , the controller for the system in (5.12) is designed as

$$u(t) = \hat{u}(x(t), \hat{W}_a(t)). \quad (5.34)$$

The following rank condition facilitates the subsequent stability analysis.

**Assumption 5.3** There exists a finite set of fixed points  $\{x_i \in \mathbb{R}^n \mid i = 1, \dots, N\}$  such that  $\forall t \in \mathbb{R}_{\geq t_0}$

$$0 < c \triangleq \frac{1}{N} \left( \inf_{t \in \mathbb{R}_{\geq t_0}} \left( \lambda_{\min} \left\{ \sum_{i=1}^N \frac{\omega_i(t)\omega_i^T(t)}{\rho_i(t)} \right\} \right) \right). \quad (5.35)$$

Compared to the typical PE condition, the condition in (5.35) can be verified online at each time  $t$ . Furthermore, the condition in (5.35) can be heuristically met by

collecting redundant data (i.e., by selecting more points than the number of neurons by choosing  $N \gg L$ ).

### 5.3.5 Stability Analysis

To facilitate the subsequent stability analysis, the approximate BE is expressed in terms of the weight estimation errors  $\tilde{W}_c \triangleq W - \hat{W}_c$  and  $\tilde{W}_a \triangleq W - \hat{W}_a$ . Subtracting (5.13) from (5.24), an unmeasurable form of the instantaneous BE can be expressed as

$$\begin{aligned}\hat{\delta}_t = & -\omega^T \tilde{W}_c - W^T \sigma' Y \tilde{\theta} + \frac{1}{4} \tilde{W}_a^T G_\sigma \tilde{W}_a \\ & + \frac{1}{4} G_\epsilon - \epsilon' f + \frac{1}{2} W^T \sigma' G \epsilon'^T.\end{aligned}\quad (5.36)$$

Similarly, the approximate BE evaluated at the sampled states  $\{x_i \mid i = 1, \dots, N\}$  can be expressed as

$$\hat{\delta}_{ti} = -\omega_i^T \tilde{W}_c + \frac{1}{4} \tilde{W}_a^T G_{\sigma_i} \tilde{W}_a - W^T \sigma'_i Y_i \tilde{\theta} + \Delta_i, \quad (5.37)$$

where  $\epsilon'_i = \epsilon'(x_i)$ ,  $f_i = f(x_i)$ ,  $G_i \triangleq g_i R^{-1} g_i^T \in \mathbb{R}^{n \times n}$ ,  $G_{\epsilon i} \triangleq \epsilon'_i G_i \epsilon'^T_i \in \mathbb{R}$ , and  $\Delta_i \triangleq \frac{1}{2} W^T \sigma'_i G_i \epsilon'^T_i + \frac{1}{4} G_{\epsilon i} - \epsilon'_i f_i \in \mathbb{R}$  is a constant.

On any compact set  $\chi \subset \mathbb{R}^n$  the function  $Y$  is Lipschitz continuous, and hence, there exists a positive constant  $L_Y \in \mathbb{R}$  such that<sup>6</sup>

$$\|Y\| \leq L_Y \|x\|, \forall x \in \chi. \quad (5.38)$$

Using (5.33), the normalized regressor  $\frac{\omega}{\rho}$  can be bounded as

$$\sup_{t \in \mathbb{R}_{\geq t_0}} \left\| \frac{\omega}{\rho} \right\| \leq \frac{1}{2\sqrt{v\underline{\Gamma}}}. \quad (5.39)$$

For brevity of notation, the following positive constants are defined:

$$\begin{aligned}\vartheta_1 &\triangleq \frac{\eta_{c1} L_Y \|\theta\| \bar{\epsilon}'}{4\sqrt{v\underline{\Gamma}}}, \quad \vartheta_2 \triangleq \sum_{i=1}^N \left( \frac{\eta_{c2} \|\sigma'_i Y_i\| \bar{W}}{4N\sqrt{v\underline{\Gamma}}} \right), \\ \vartheta_3 &\triangleq \frac{L_Y \eta_{c1} \bar{W} \|\sigma'\|}{4\sqrt{v\underline{\Gamma}}}, \quad \vartheta_4 \triangleq \left\| \frac{1}{4} G_\epsilon \right\|,\end{aligned}$$

---

<sup>6</sup>The Lipschitz property is exploited here for clarity of exposition. The bound in (5.38) can be easily generalized to  $\|Y(x)\| \leq L_Y (\|x\|) \|x\|$ , where  $L_Y : \mathbb{R} \rightarrow \mathbb{R}$  is a positive, non-decreasing function.

$$\begin{aligned}
\vartheta_5 &\triangleq \frac{\eta_{c1} \overline{\|2W^T \sigma' G \epsilon'^T + G_\epsilon\|}}{8\sqrt{v\Gamma}} + \left\| \sum_{i=1}^N \frac{\eta_{c2} \omega_i \Delta_i}{N\rho_i} \right\|, \\
\vartheta_6 &\triangleq \overline{\left\| \frac{1}{2} W^T G_\sigma + \frac{1}{2} \epsilon' G^T \sigma'^T \right\|} + \vartheta_7 \overline{W}^2 + \eta_{a2} \overline{W}, \\
\vartheta_7 &\triangleq \frac{\eta_{c1} \overline{\|G_\sigma\|}}{8\sqrt{v\Gamma}} + \sum_{i=1}^N \left( \frac{\eta_{c2} \|G_{\sigma i}\|}{8N\sqrt{v\Gamma}} \right), \quad \underline{q} \triangleq \lambda_{\min}\{Q\}, \\
v_l &= \frac{1}{2} \min \left( \frac{\underline{q}}{2}, \frac{\eta_{c2} \underline{c}}{3}, \frac{\eta_{a1} + 2\eta_{a2}}{6}, \frac{K}{4} \right), \\
\iota &= \frac{3\vartheta_5^2}{4\eta_{c2} \underline{c}} + \frac{3\vartheta_6^2}{2(\eta_{a1} + 2\eta_{a2})} + \frac{D^2}{2K} + \vartheta_4,
\end{aligned} \tag{5.40}$$

where  $\overline{(\cdot)} \triangleq \sup_{x \in \chi} (\cdot) : \mathbb{R}_{\geq 0} \rightarrow \mathbb{R}_{\geq 0}$ . Let  $Z : \mathbb{R}_{\geq t_0} \rightarrow \mathbb{R}^{n+2L+p}$  be defined as

$$Z(t) \triangleq \left[ x^T(t), \tilde{W}_c^T(t), \tilde{W}_a^T(t), \tilde{\theta}^T(t) \right]^T, \tag{5.41}$$

where  $x(\cdot)$ ,  $\tilde{W}_c(\cdot)$ ,  $\tilde{W}_a(\cdot)$ , and  $\tilde{\theta}(\cdot)$  denote the solutions of the differential equations in (5.12), (5.30), and (5.32), respectively, with appropriate initial conditions. The sufficient conditions for ultimate boundedness of  $Z(\cdot)$  are derived based on the subsequent stability analysis as

$$\begin{aligned}
\frac{\eta_{a1} + 2\eta_{a2}}{6} &> \vartheta_7 \overline{W} \left( \frac{2\zeta_2 + 1}{2\zeta_2} \right), \\
\frac{K}{4} &> \frac{\vartheta_2 + \zeta_1 \zeta_3 \vartheta_3 \overline{Z}}{\zeta_1}, \\
\frac{\eta_{c2}}{3} &> \frac{\zeta_2 \vartheta_7 \overline{W} + \eta_{a1} + 2(\vartheta_1 + \zeta_1 \vartheta_2 + (\vartheta_3/\zeta_3) \overline{Z})}{2\underline{c}}, \\
\frac{\underline{q}}{2} &> \vartheta_1,
\end{aligned} \tag{5.42}$$

where  $\overline{Z} \triangleq \bar{v}^{-1} \left( \bar{v} \left( \max \left( \|Z(t_0)\|, \sqrt{\frac{\iota}{v_l}} \right) \right) \right)$ ,  $\zeta_1, \zeta_2, \zeta_3 \in \mathbb{R}$  are known positive adjustable constants, and  $v$  and  $\bar{v}$  are subsequently defined class  $\mathcal{K}$  functions. The Lipschitz constants in (5.38) and the NN function approximation errors depend on the underlying compact set; hence, given a bound on the initial condition  $Z(t_0)$  for the concatenated state  $Z(\cdot)$ , a compact set that contains the concatenated state trajectory needs to be established before adaptation gains satisfying the conditions in (5.42) can be selected. Based on the subsequent stability analysis, an algorithm to

compute the required compact set, denoted by  $\mathcal{Z} \subset \mathbb{R}^{2n+2L+p}$ , is developed in [15]. Since the constants  $\iota$  and  $v_l$  depend on  $L_Y$  only through the products  $L_Y\bar{\epsilon}$  and  $\frac{L_Y}{\zeta_3}$ , proper gain selection ensures that

$$\sqrt{\frac{\iota}{v_l}} \leq \frac{1}{2} \text{diam}(\mathcal{Z}), \quad (5.43)$$

where  $\text{diam}(\mathcal{Z})$  denotes the diameter of the set  $\mathcal{Z}$  defined as  $\text{diam}(\mathcal{Z}) \triangleq \sup\{\|x - y\| \mid x, y \in \mathcal{Z}\}$ . The main result of this section can now be stated as follows.

**Theorem 5.4** *Provided Assumptions 5.1–5.3 hold and gains  $q$ ,  $\eta_{c2}$ ,  $\eta_{a2}$ , and  $K$  are sufficiently large, the controller in (5.34) along with the adaptive update laws in (5.30) and (5.32) ensure that the  $x(\cdot)$ ,  $\tilde{W}_c(\cdot)$ ,  $\tilde{W}_a(\cdot)$ , and  $\tilde{\theta}(\cdot)$  are uniformly ultimately bounded (UUB).*

**Proof** For brevity, a sketch of the proof is provided, the detailed proof can be seen in [15]. Consider a candidate Lyapunov function candidate  $V_L : \mathbb{R}^{n+2L+p} \times \mathbb{R}_{\geq t_0} \rightarrow \mathbb{R}$  defined as

$$V_L(Z, t) \triangleq V^*(x) + \frac{1}{2} \tilde{W}_c^T \Gamma^{-1}(t) \tilde{W}_c + \frac{1}{2} \tilde{W}_a^T \tilde{W}_a + V_\theta(\tilde{\theta}, t). \quad (5.44)$$

Since the optimal value function is positive definite, (5.33) and [40, Lemma 4.3] can be used to show that the candidate Lyapunov function satisfies the following bounds

$$\underline{v}_l(\|Z\|) \leq V_L(Z, t) \leq \overline{v}_l(\|Z\|), \quad (5.45)$$

for all  $t \in \mathbb{R}_{\geq t_0}$  and for all  $Z \in \mathbb{R}^{n+2L+p}$ . In (5.45),  $\underline{v}_l, \overline{v}_l : \mathbb{R}_{\geq 0} \rightarrow \mathbb{R}_{\geq 0}$  are class  $\mathcal{K}$  functions. Taking the time -derivative of (5.44) along the system trajectory, substituting (5.30)–(5.32) along with (5.36) and (5.37). Bounding and enforcing (5.42) produces the stated result.

### 5.3.6 Summary

In this section, the PE condition is replaced by a set of rank conditions that can be verified online using current and recorded observations. UUB regulation of the system states to a neighborhood of the origin, and convergence of the policy to a neighborhood of the optimal policy are established using a Lyapunov-based analysis. While the result in Sect. 5.3 demonstrates online approximate optimal regulation using BE extrapolation, it tends to be computationally inefficient since it performs value function approximations across the entire state-space. In Sect. 5.4, a computationally efficient method is discussed by performing local value function approximations.

## 5.4 Local (State-Following) Model-Based Reinforcement Learning

Sufficiently accurate approximation of the value function over a sufficiently large neighborhood often requires a large number of basis functions, and hence, introduces a large number of unknown parameters. One way to achieve accurate function approximation with fewer unknown parameters is to use prior knowledge about the system to determine the basis functions. However, generally, prior knowledge of the features of the optimal value function is not available; hence, a large number of generic basis functions is often the only feasible option.

Fast approximation of the value function over a large neighborhood requires sufficiently rich data to be available for learning. In traditional ADP methods such as [7, 9, 38], richness of data manifests itself as the amount of excitation in the system. In experience replay-based techniques such as [24, 41–43], richness of data is quantified by eigenvalues of a recorded history stack. In R-MBRL techniques such as [44–46], richness of data corresponds to the eigenvalues of a learning matrix. As the dimension of the system and the number of basis functions increases, the richer data is required to achieve learning. In experience replay-based ADP methods and in R-MBRL, the demand for richer data causes exponential growth in the required data storage. Hence, implementations of traditional ADP techniques such as [1–11, 38] and data-driven ADP techniques such as [24, 44–48] in high-dimensional systems are scarcely found in the literature.

This section presents a MBRL technique with a lower computational cost than current data-driven ADP techniques. Motivated by the fact that the computational effort required to implement ADP and the data-richness required to achieve convergence both decrease with decreasing number of basis functions, this technique reduces the number of basis functions used for value function approximation.

A key contribution of [18, 49] is the observation that online implementation of an ADP-based approximate optimal controller does not require an estimate of the optimal value function over the entire domain of operation of the system. Instead, only an estimate of the value function gradient at the current state is required. Since it is reasonable to postulate that approximation of the value function over a local domain would require fewer basis functions than approximation over the entire domain of operation, this section focuses on the reduction of the size of approximation domain. Such a reduction is achieved via the selection of basis functions that travel with the system state (referred to as state-following (StaF) kernels).

Unlike traditional value function approximation, where the unknown parameters are constants, the unknown parameters corresponding to the StaF kernels are functions of the system state. To facilitate the proof of continuous differentiability, the StaF kernels are selected from a reproducing kernel Hilbert space (RKHS). Other function approximation methods, such as radial basis functions, sigmoids, higher order NNs, support vector machines, etc., can potentially be utilized in a state-following manner to achieve similar results provided continuous differentiability of the ideal weights can be established.

### 5.4.1 StaF Kernel Functions

In this section, Theorem 5.5 motivates the use of StaF kernels for model-based RL, and Theorem 5.6 facilitates implementation of gradient-based update laws to learn the time-varying ideal weights in real-time.

**Theorem 5.5** *Let  $\epsilon, r > 0$  and let  $p$  denote a polynomial that approximates  $\bar{V}^*$  within an error  $\epsilon$  over  $B_r(x)$ . Let  $N(r, x, \epsilon)$  denote the degree of  $p$ . Let  $k(y, x) = e^{y^T x}$  be the exponential kernel function, which corresponds to a universal RKHS. Then, for each  $x \in \chi$ , there exists a finite number of centers,  $c_1, c_2, \dots, c_{M(r, x, \epsilon)} \in B_r(x)$  and weights  $w_1, w_2, \dots, w_{M(r, x, \epsilon)}$  such that*

$$\left\| \bar{V}^*(y) - \sum_{i=1}^{M(r, x, \epsilon)} w_i e^{y^T c_i} \right\|_{B_r(x), \infty} < \epsilon,$$

where  $M(r, x, \epsilon) < \binom{n+N(r, x, \epsilon)+S(r, x, \epsilon)}{N(r, x, \epsilon)+S(r, x, \epsilon)}$ , asymptotically, for some  $S(r, x, \epsilon) \in \mathbb{N}$ . Moreover,  $r$ ,  $N(r, x, \epsilon)$  and  $S(r, x, \epsilon)$  can be bounded uniformly over  $\chi$  for any fixed  $\epsilon$  [18].<sup>7</sup>

The Weierstrass theorem indicates that as  $r$  decreases, the degree  $N(r, x, \epsilon)$  of the polynomial needed to achieve the same error  $\epsilon$  over  $B_r(x)$  decreases [50]. Hence, by Theorem 5.5, approximation of a function over a smaller domain requires a smaller number of exponential kernels. Furthermore, provided the region of interest is small enough, the number of kernels required to approximate continuous functions with arbitrary accuracy can be reduced to  $\binom{n+2}{2}$ .

In the StaF approach, the centers are selected to follow the current state  $x$ , i.e., the locations of the centers are defined as a function of the system state. Since the system state evolves in time, the ideal weights are not constant. To approximate the ideal weights using gradient-based algorithms, it is essential that the weights change smoothly with respect to the system state. The following theorem allows the use of gradient-based update laws to determine the time-varying ideal weights of the value function.

**Theorem 5.6** *Let the kernel function  $k$  be such that the functions  $k(\cdot, x)$  are  $l$ -times continuously differentiable for all  $x \in \chi$ . Let  $C \triangleq [c_1, c_2, \dots, c_L]^T$  be a set of distinct centers such that  $c_i \in B_r(x)$ ,  $\forall i = 1, \dots, L$ , be an ordered collection of  $L$  distinct centers with associated ideal weights*

$$W_{H_{x,r}}(C) = \arg \min_{a \in R^M} \left\| \sum_{i=1}^M a_i k(\cdot, c_i) - V(\cdot) \right\|_{H_{x,r}}. \quad (5.46)$$

Then, the function  $W_{H_{x,r}}$  is  $l$ -times continuously differentiable with respect to each component of  $C$  [18].

---

<sup>7</sup>The notation  $\binom{a}{b}$  denotes the combinatorial operation “ $a$  choose  $b$ ”.

### 5.4.2 Local Value Function Approximation

Similar to Sect. 5.3, an approximate solution to the HJB equation is sought. The optimal value function  $V^*$  is approximated using a parametric estimate. The expression for the optimal policy in (5.20) indicates that, to compute the optimal action when the system is at any given state  $x$ , one only needs to evaluate the gradient  $V^{*\prime}$  at  $x$ . Hence, to compute the optimal policy at  $x$ , one only needs to approximate the value function over a small neighborhood around the current state,  $x$ . As established in Theorem 5.5, the number of basis functions required to approximate the value function decreases if the approximation space decreases (with respect to the ordering induced by set containment). The aim is to obtain a uniform approximation of the value function over a small neighborhood around the system state.

To facilitate the development, let  $\chi \subset \mathbb{R}^n$  be compact and let  $x$  be in the interior of  $\chi$ . Then, for all  $\epsilon > 0$ , there exists a function  $\bar{V}^* \in H_{x,r}$  such that  $\sup_{y \in B_r(x)} |V^*(y) - \bar{V}^*(y)| < \epsilon$ , where  $H_{x,r}$  is a restriction of a universal RKHS,  $H$ , introduced in Sect. 5.4.1, to  $B_r(x)$ . In the developed StaF-based method, a small compact set  $B_r(x)$  around the current state  $x$  is selected for value function approximation by selecting the centers  $C \in B_r(x)$  such that  $C = c(x)$  for some continuously differentiable function  $c : \chi \rightarrow \chi^L$ . Using StaF kernels centered at a point  $x$ , the value function can be represented as

$$V^*(y) = W(x)^T \sigma(y, c(x)) + \epsilon(x, y).$$

Since the centers of the kernel functions change as the system state changes, the ideal weights also change as the system state changes. The state-dependent nature of the ideal weights differentiates this approach from aforementioned ADP methods in the sense that the stability analysis needs to account for changing ideal weights. Based on Theorem 5.6, the ideal weight function  $W : \chi \rightarrow \mathbb{R}^L$  defined as  $W(x) \triangleq W_{H_{x,r}}(c(x))$ , where  $W_{H_{x,r}}$  was introduced in (5.46), is continuously differentiable, provided the functions  $\sigma$  and  $c$  are continuously differentiable.

The approximate value function  $\hat{V} : \mathbb{R}^n \times \mathbb{R}^n \times \mathbb{R}^L \rightarrow \mathbb{R}$  and the approximate policy  $\hat{u} : \mathbb{R}^n \times \mathbb{R}^n \times \mathbb{R}^L \rightarrow \mathbb{R}^m$ , evaluated at a point  $y \in B_r(x)$ , using StaF kernels centered at  $x$ , can then be expressed as

$$\hat{V}(y, x, \hat{W}_c) \triangleq \hat{W}_c^T \sigma(y, c(x)), \quad (5.47)$$

$$\hat{u}(y, x, \hat{W}_d) \triangleq -\frac{1}{2} R^{-1} g^T(y) \sigma'(y, c(x))^T \hat{W}_d. \quad (5.48)$$

### 5.4.3 Actor and Critic Update Laws

In this section, the BE, weight update laws, and  $\omega$ , are redefined to clarify the distinction that the BE is calculated from time-varying points in the neighborhood of the current trajectory. The critic uses the BEs

$$\delta_t(t) \triangleq \delta(x(t), x(t), \hat{W}_c(t), \hat{W}_a(t)), \quad (5.49)$$

and

$$\delta_{ti}(t) = \delta(x_i(x(t), t), x(t), \hat{W}_c(t), \hat{W}_a(t)). \quad (5.50)$$

to improve the StaF-based estimate  $\hat{W}_c(t)$  using the recursive least squares-based update law

$$\dot{\hat{W}}_c(t) = -k_{c1}\Gamma(t) \frac{\omega(t)}{\rho(t)}\delta_t(t) - \frac{k_{c2}}{N}\Gamma(t) \sum_{i=1}^N \frac{\omega_i(t)}{\rho_i(t)}\delta_{ti}(t), \quad (5.51)$$

where  $\rho_i(t) \triangleq \sqrt{1 + \gamma_1\omega_i^T(t)\omega_i(t)}$ ,  $\rho(t) \triangleq \sqrt{1 + \gamma_1\omega^T(t)\omega(t)}$ ,  $k_{c1}, k_{c2}, \gamma_1 \in \mathbb{R}_{>0}$  are constant learning gains,

$$\begin{aligned} \omega(t) &\triangleq \sigma'(x(t), c(x(t))) f(x(t)) \\ &\quad + \sigma'(x(t), c(x(t))) g(x(t)) \hat{u}(x(t), x(t), \hat{W}_a(t)), \end{aligned}$$

and

$$\begin{aligned} \omega_i(t) &\triangleq \sigma'(x_i(x(t)), c(x(t))) f(x_i(x(t), t)) \\ &\quad + \sigma'(x_i(x(t)), c(x(t))) g(x_i(x(t), t)) \hat{u}(x_i(x(t), t), x(t), \hat{W}_a(t)). \end{aligned}$$

In (5.51),  $\Gamma(t)$  denotes the least-square learning gain matrix updated according to

$$\begin{aligned} \dot{\Gamma}(t) &= \beta\Gamma(t) - k_{c1}\Gamma(t) \frac{\omega(t)\omega^T(t)}{\rho^2(t)}\Gamma(t) \\ &\quad - \frac{k_{c2}}{N}\Gamma(t) \sum_{i=1}^N \frac{\omega_i(t)\omega_i^T(t)}{\rho_i^2(t)}\Gamma(t), \\ \Gamma(t_0) &= \Gamma_0, \end{aligned} \quad (5.52)$$

where  $\beta \in \mathbb{R}_{>0}$  is a constant forgetting factor. Motivated by a Lyapunov-based stability analysis, the update law for the actor is designed as

$$\begin{aligned}\dot{\hat{W}}_a(t) &= -k_{a1} \left( \hat{W}_a(t) - \hat{W}_c(t) \right) - k_{a2} \hat{W}_a(t) \\ &\quad + \frac{k_{c1} G_\sigma^T(t) \hat{W}_a(t) \omega(t)^T}{4\rho(t)} \hat{W}_c(t) \\ &\quad + \sum_{i=1}^N \frac{k_{c2} G_{\sigma i}^T(t) \hat{W}_a(t) \omega_i^T(t)}{4N\rho_i(t)} \hat{W}_c(t),\end{aligned}\tag{5.53}$$

where  $k_{a1}, k_{a2} \in \mathbb{R}_{>0}$  are learning gains,

$$\begin{aligned}G_\sigma(t) &\triangleq \sigma'(x(t), c(x(t))) g(x(t)) R^{-1} g^T(x(t)) \\ &\quad \cdot \sigma'^T(x(t), c(x(t))),\end{aligned}$$

and

$$\begin{aligned}G_{\sigma i}(t) &\triangleq \sigma'(x_i(x(t), t), c(x(t))) g(x_i(x(t), t)) \\ &\quad \cdot R^{-1} g^T(x_i(x(t), t)) \sigma'^T(x_i(x(t), t), c(x(t))).\end{aligned}$$

#### 5.4.4 Analysis

Let  $B_\zeta \subset \mathbb{R}^{n+2L}$  denote a closed ball with radius  $\zeta$  centered at the origin. Let  $\chi \triangleq B_\zeta \cap \mathbb{R}^n$ . Let the notation  $\overline{\|(\cdot)\|}$  be defined as  $\overline{\|h\|} \triangleq \sup_{\xi \in \chi} \|h(\xi)\|$ , for some continuous function  $h : \mathbb{R}^n \rightarrow \mathbb{R}^k$ . To facilitate the subsequent stability analysis, the BEs in are expressed in terms of the weight estimation errors  $\tilde{W}_c$  and  $\tilde{W}_a$ , defined in Sect. 5.3, as

$$\begin{aligned}\delta_t &= -\omega^T \tilde{W}_c + \frac{1}{4} \tilde{W}_a G_\sigma \tilde{W}_a + \Delta(x), \\ \delta_{ti} &= -\omega_i^T \tilde{W}_c + \frac{1}{4} \tilde{W}_a^T G_{\sigma i} \tilde{W}_a + \Delta_i(x),\end{aligned}\tag{5.54}$$

where the functions  $\Delta, \Delta_i : \mathbb{R}^n \rightarrow \mathbb{R}$  are uniformly bounded over  $\chi$  such that the bounds  $\overline{\|\Delta\|}$  and  $\overline{\|\Delta_i\|}$  decrease with decreasing  $\overline{\|\varepsilon^\nabla\|}$  and  $\overline{\|\nabla W\|}$ . To facilitate learning, the system states  $x$  and the selected functions  $x_i$  are assumed to satisfy the following.

**Assumption 5.4** There exist constants  $T \in \mathbb{R}_{>0}$  and  $\underline{c}_1, \underline{c}_2, \underline{c}_3 \in \mathbb{R}_{\geq 0}$ , such that

$$\begin{aligned}\underline{c}_1 I_L &\leq \int_t^{t+T} \left( \frac{\omega(\tau) \omega^T(\tau)}{\rho^2(\tau)} \right) d\tau, \quad \forall t \in \mathbb{R}_{\geq t_0}, \\ \underline{c}_2 I_L &\leq \inf_{t \in \mathbb{R}_{\geq t_0}} \left( \frac{1}{N} \sum_{i=1}^N \frac{\omega_i(t) \omega_i^T(t)}{\rho_i^2(t)} \right), \\ \underline{c}_3 I_L &\leq \frac{1}{N} \int_t^{t+T} \left( \sum_{i=1}^N \frac{\omega_i(\tau) \omega_i^T(\tau)}{\rho_i^2(\tau)} \right) d\tau, \quad \forall t \in \mathbb{R}_{\geq t_0},\end{aligned}$$

where at least one of the constants  $\underline{c}_1$ ,  $\underline{c}_2$ , and  $\underline{c}_3$  is strictly positive.

Assumption 5.4 only requires either the regressor  $\omega$  or the regressor  $\omega_i$  to be PE. The regressor  $\omega$  is completely determined by the system state  $x$ , and the weights  $\hat{W}_a$ . Hence, excitation in  $\omega$  vanishes as the system states and the weights converge. Hence, in general, it is unlikely that  $\underline{c}_1 > 0$ . However, the regressor  $\omega_i$  depends on  $x_i$ , which can be designed independent of the system state  $x$ . Hence,  $\underline{c}_3$  can be made strictly positive if the signal  $x_i$  contains enough frequencies, and  $\underline{c}_2$  can be made strictly positive by selecting a sufficient number of extrapolation functions.

Selection of a single time-varying BE extrapolation function results in virtual excitation. That is, instead of using input–output data from a persistently excited system, the dynamic model is used to simulate PE to facilitate parameter convergence.

**Lemma 5.1** *Provided Assumption 5.4 holds and  $\lambda_{\min}\{\Gamma_0^{-1}\} > 0$ , the update law in (5.52) ensures that the least squares gain matrix satisfies*

$$\underline{\Gamma} I_L \leq \Gamma(t) \leq \overline{\Gamma} I_L, \quad (5.55)$$

where

$$\overline{\Gamma} = \frac{1}{\min\{k_{c1}\underline{c}_1 + k_{c2} \max\{\underline{c}_2 T, \underline{c}_3\}, \lambda_{\min}\{\Gamma_0^{-1}\}\} e^{-\beta T}},$$

and

$$\underline{\Gamma} = \frac{1}{\lambda_{\max}\{\Gamma_0^{-1}\} + \frac{(k_{c1} + k_{c2})}{\beta \gamma_1}}.$$

Furthermore,  $\overline{\Gamma} > 0$  [18, 26].

### 5.4.5 Stability Analysis

To facilitate the analysis, let  $\underline{c} \in \mathbb{R}_{>0}$  be a constant defined as

$$\underline{c} \triangleq \frac{\beta}{2\bar{\Gamma}k_{c2}} + \frac{c_2}{2}, \quad (5.56)$$

and let  $\iota \in \mathbb{R}_{>0}$  be a constant defined as

$$\begin{aligned} \iota \triangleq & \frac{3 \left( \frac{(k_{c1}+k_{c2})\|\Delta\|}{\sqrt{v}} + \frac{\|\nabla W f\|}{\bar{\Gamma}} + \frac{\|\Gamma^{-1}G_{W\sigma}W\|}{2} \right)^2}{4k_{c2}\underline{c}} \\ & + \frac{1}{(k_{a1}+k_{a2})} \left( \frac{\|G_{W\sigma}W\| + \|G_{V\sigma}\|}{2} + k_{a2}\|W\| \right. \\ & \left. + \frac{(k_{c1}+k_{c2})\|G_\sigma\|\|W\|}{4\sqrt{v}} \right)^2 \\ & + \frac{1}{2}\|G_{VW\sigma}\| + \frac{1}{2}\|G_{V\varepsilon}\|, \end{aligned}$$

where  $G_{W\sigma} \triangleq \nabla W G \sigma'^T$ ,  $G_{V\sigma} \triangleq V'^* G \sigma'^T$ ,  $G_{VW} \triangleq V'^* G \nabla W^T$ , and  $G_{V\varepsilon} \triangleq V'^* G \varepsilon'^T$ . Let  $v_l : \mathbb{R}_{\geq 0} \rightarrow \mathbb{R}_{\geq 0}$  be a class  $\mathcal{K}$  function such that

$$v_l(\|Z\|) \leq \frac{Q(x)}{2} + \frac{k_{c2}\underline{c}}{6} \|\tilde{W}_c\|^2 + \frac{(k_{a1}+k_{a2})}{8} \|\tilde{W}_a\|^2.$$

The sufficient conditions for Lyapunov-based stability are given by

$$\frac{k_{c2}\underline{c}}{3} \geq \frac{\left( \frac{\|G_{W\sigma}\|}{2\bar{\Gamma}} + \frac{(k_{c1}+k_{c2})\|W^T G_\sigma\|}{4\sqrt{v}} + k_{a1} \right)^2}{(k_{a1}+k_{a2})}, \quad (5.57)$$

$$\frac{(k_{a1}+k_{a2})}{4} \geq \left( \frac{\|G_{W\sigma}\|}{2} + \frac{(k_{c1}+k_{c2})\|W\|\|G_\sigma\|}{4\sqrt{v}} \right), \quad (5.58)$$

$$v_l^{-1}(\iota) < \overline{v_l}^{-1}(v_l(\zeta)). \quad (5.59)$$

The sufficient condition in (5.57) can be satisfied provided the points for BE extrapolation are selected such that the minimum eigenvalue  $\underline{c}$ , introduced in (5.56) is large enough. The sufficient condition in (5.58) can be satisfied without affecting (5.57) by increasing the gain  $k_{a2}$ . The sufficient condition in (5.59) can be satisfied provided  $\underline{c}$ ,  $k_{a2}$ , and the state penalty  $Q(x)$  are selected to be sufficiently large and the StaF kernels for value function approximation are selected such that  $\|\nabla W\|$ ,  $\|\varepsilon\|$ , and

$\|\nabla \varepsilon\|$  are sufficiently small.<sup>8</sup> To improve computational efficiency, the size of the domain around the current state where the StaF kernels provide good approximation of the value function is desired to be small. Smaller approximation domain results in almost identical extrapolated points, which in turn, results in smaller  $\underline{c}$ . Hence, the approximation domain cannot be selected to be arbitrarily small and needs to be large enough to meet the sufficient conditions in (5.57)–(5.59).

**Theorem 5.7** *Provided Assumption 5.4 holds and the sufficient gain conditions in (5.57)–(5.59) are satisfied, the controller  $u(t)$  and the update laws in (5.51)–(5.53) ensure that the state  $x$  and the weight estimation errors  $\tilde{W}_c$  and  $\tilde{W}_a$  are UUB.*

**Proof** The proof follows from Theorem 5.4, see [18] for a detailed analysis.

#### 5.4.6 Summary

In this section, an infinite horizon optimal control problem is solved using an approximation methodology called the StaF kernel method. Motivated by the fact that a smaller number of basis functions is required to approximate functions on smaller domains, the StaF kernel method aims to maintain a good approximation of the value function over a small neighborhood of the current state. Computational efficiency of model-based RL is improved by allowing selection of fewer time-varying extrapolation trajectories instead of a large number of autonomous extrapolation functions.

Methods to solve infinite horizon optimal control problems online aim to approximate the value function over the entire operating domain. Since the approximate optimal policy is completely determined by the value function estimate, solutions generate policies that are valid over the entire state space but at a high computational cost. Since the StaF kernel method aims at maintaining local approximation of the value function around the current system state, the StaF kernel method lacks memory, in the sense that the information about the ideal weights over a region of interest is lost when the state leaves the region of interest. Thus, unlike aforementioned techniques, the StaF method trades global optimality for computational efficiency to generate a policy that is near-optimal only over a small neighborhood of the origin. A memory-based modification to the StaF technique that retains and reuses past information is a subject for the following section.

The technique developed in this section can be extended to a class of trajectory tracking problems in the presence of uncertainties in the system drift dynamics by using a concurrent learning-based adaptive system identifier (cf., [15, 18, 26, 45]).

---

<sup>8</sup>Similar to NN-based approximation methods such as [1–8], the function approximation error,  $\varepsilon$ , is unknown, and in general, infeasible to compute for a given function, since the ideal NN weights are unknown. Since a bound on  $\varepsilon$  is unavailable, the gain conditions in (5.57)–(5.59) cannot be formally verified. However, they can be met using trial and error by increasing the gain  $k_{a2}$ , the number of StaF basis functions, and  $\underline{c}$ , by selecting more points to extrapolate the BE.

## 5.5 Combining Regional and Local State-Following Approximations

Reduction in the number of unknown parameters motivates the use of StaF basis functions as described in the previous section (c.f., previously, and [51]), which travel with the state to maintain an accurate local approximation. However, the StaF approximation method trades global optimality for computational efficiency since it lacks memory. Since accurate estimation of the value function results in a better closed-loop response and lower operating costs, it is desirable to accurately estimate the value function near the origin in optimal regulation problems.

In [52], a framework is developed to merge local and regional value function approximation methods to yield an online optimal control method that is computationally efficient and simultaneously accurate over a specified critical region of the state-space. The ability of R-MBRL (c.f., previously, and [15]) to approximate the value function over a predefined region and the computational efficiency of the StaF method [18] in approximating the value function locally along the state trajectory motivates the additional development. Instead of generating an approximation of the value function over the entire operating region, which is computationally expensive, the operating domain can be separated into two regions: a closed set  $A$ , containing the origin, where a regional approximation method is used to approximate the value function and the complement of  $A$ , where the StaF method is used to approximate the value function. Using a switching-based approach to combine regional and local approximations injects discontinuities to the system and result in a non-smooth value function which would introduce discontinuities in the control signal. To overcome this challenge, a state-varying convex combination of the two approximation methods can be used to ensure a smooth transition from the StaF to the R-MBRL approximation as the state enters the closed convex set containing the origin. Once the state enters this region, R-MBRL regulates the state to the origin. The developed result can be generalized to allow for the use of any R-MBRL method. This strategy is motivated by the observation that in many applications such as station keeping of marine craft, like in [53], accurate approximation of the value function in a neighborhood of the goal state can improve the performance of the closed-loop system near the goal state. Since the StaF method uses state-dependent centers, the unknown optimal weight are themselves also state-dependent, which makes analyzing stability difficult. To add to the technical challenge, using a convex combination of R-MBRL and StaF results in a complex representation of the value function and resulting BE. To provide insights into how to combine StaF and R-MBRL while also preserving stability, see [52].

## 5.6 Reinforcement Learning with Sparse Bellman Error Extrapolation

Motivated by additional computational efficiency, sparsification techniques are motivated to collectively perform BE extrapolation in segmented parts of the operating domain. Sparsification techniques enable local approximation across the segments, which allows characterization of regions with significantly varying dynamics or unknown uncertainties.

Sparse neural networks (SNNs), like conventional NNs, are a tool to facilitate learning in uncertain systems (cf., [54–62]). SNNs have been used to reduce the computational complexity in NNs by decreasing the number of active neurons; hence, reducing the number of computations overall. Sparse adaptive controllers have been developed to update a small number of neurons at certain points in the state space in works such as [60]. Sparsification encourages local learning through intelligent segmentation [56], and encourages learning without relying on a high adaptive learning rate [62]. In practice, high learning rates can cause oscillations or instability due to unmodeled dynamics in the control bandwidth [62]. SNNs create a framework for switching and segmentation as well as computational benefits due to the small number of active neurons. Sparsification techniques enable local approximation across the segments, which characterizes regions with significantly varying dynamics or unknown uncertainties.

In [61], a method is developed to better estimate the value function across the entire state space by using a set of sparse off-policy trajectories, which are used to calculate extrapolated BEs. The set of off-policy trajectories will be determined by the location in the state space of the system. Hence, sets of input–output data pairs corresponding to each segment of the operating domain are developed and used in the actor and critic update laws. Compared to results such as [15, 63, 64], this technique does not perform BE extrapolation over the entire operating domain at each time instance. Instead, the operating domain is divided into segments where a certain set of trajectories, and, hence, sets of extrapolated BEs, are active when the state enters the corresponding segment. SNNs are used within each segment to extrapolate the BE due to their small amount of active neurons, whose activity can be switched on or off based on the active segment, to make BE extrapolation more computationally efficient. Using the increased computational efficiency of SNNs and segmentation to extrapolate the BE, the BE can be estimated across the entire state space.

## 5.7 Conclusion

This chapter discussed mixed density RL-based approximate optimal control methods applied to deterministic systems. Implementations of model-based RL to solve approximate optimal regulation problems online using different value function approximation and BE extrapolation techniques were discussed. While the mixed

density methods presented in this chapter shed some light on potential solutions, methods must be developed and refined to address future needs.

In Sect. 5.2, the infinite horizon optimal control problem is introduced along with conditions that establish the optimal control policy. It is shown that the value function is the optimal cost-to-go and satisfies the HJB equation.

In Sect. 5.3, the R-MBRL method is presented where unknown weights in the value function are adjusted based on least squares minimization of the BE evaluated at any number of user-selected arbitrary trajectories in the state space. Since the BE can be evaluated at any desired point in the state space, sufficient exploration is achieved by selecting points distributed over the system's operating domain. R-MBRL utilizes BE extrapolation over a large region of the state space but is computationally complex. The strategies in Sects. 5.4–5.6 address the computational constraints of this method. Future work includes extending this result to hybrid systems.

In Sect. 5.4, the StaF-RL method is presented where the computational complexity of R-MBRL problems is reduced by estimating the optimal value function within a local domain around the state. Future work will focus on extending this method to nonaffine systems [18].

In Sect. 5.5, a strategy that uses R-MBRL and StaF-RL together to approximate the value function is described. This technique eliminates the need to perform BE extrapolation over a large region of the state space, as in R-MBRL, and the inability for the StaF method to develop a global estimate of the value function. Future work includes investigating the rate at which the optimal value function is learned and how it changes based on the size of the R-MBRL region [52].

In Sect. 5.6, a strategy is described to overcome the computational cost of R-MBRL by using a set of sparse off-policy trajectories, which are used to calculate extrapolated BEs. Furthermore, the state space is divided into a user-selected number of segments. SNNs could then be used within each segment to extrapolate the BE due to their small amount of active neurons, whose activity can be switched on or off based on the active segment, to make BE extrapolation more computationally efficient. Future work will study the accuracy of using SNNs as opposed to conventional NNs, quantify the computational savings of using SNNs, and generating a Zeno-free switched system (i.e., exclude Zeno behavior with respect to switching).

## References

1. Doya, K.: Reinforcement learning in continuous time and space. *Neural Comput.* **12**(1), 219–245 (2000)
2. Padhi, R., Unnikrishnan, N., Wang, X., Balakrishnan, S.: A single network adaptive critic (SNAC) architecture for optimal control synthesis for a class of nonlinear systems. *Neural Netw.* **19**(10), 1648–1660 (2006)
3. Al-Tamimi, A., Lewis, F.L., Abu-Khalaf, M.: Discrete-time nonlinear HJB solution using approximate dynamic programming: convergence proof. *IEEE Trans. Syst. Man Cybern. Part B Cybern.* **38**, 943–949 (2008)

4. Lewis, F.L., Vrabie, D.: Reinforcement learning and adaptive dynamic programming for feedback control. *IEEE Circuits Syst. Mag.* **9**(3), 32–50 (2009)
5. Dierks, T., Thumati, B., Jagannathan, S.: Optimal control of unknown affine nonlinear discrete-time systems using offline-trained neural networks with proof of convergence. *Neural Netw.* **22**(5–6), 851–860 (2009)
6. Mehta, P., Meyn, S.: Q-learning and pontryagin's minimum principle. In: *Proceedings of the IEEE Conference on Decision and Control*, pp. 3598–3605
7. Vamvoudakis, K.G., Lewis, F.L.: Online actor-critic algorithm to solve the continuous-time infinite horizon optimal control problem. *Automatica* **46**(5), 878–888 (2010)
8. Zhang, H., Cui, L., Zhang, X., Luo, Y.: Data-driven robust approximate optimal tracking control for unknown general nonlinear systems using adaptive dynamic programming method. *IEEE Trans. Neural Netw.* **22**(12), 2226–2236 (2011)
9. Bhasin, S., Kamalapurkar, R., Johnson, M., Vamvoudakis, K.G., Lewis, F.L., Dixon, W.E.: A novel actor-critic-identifier architecture for approximate optimal control of uncertain nonlinear systems. *Automatica* **49**(1), 89–92 (2013)
10. Zhang, H., Cui, L., Luo, Y.: Near-optimal control for nonzero-sum differential games of continuous-time nonlinear systems using single-network adp. *IEEE Trans. Cybern.* **43**(1), 206–216 (2013)
11. Zhang, H., Liu, D., Luo, Y., Wang, D.: Adaptive Dynamic Programming for Control Algorithms and Stability, ser. *Communications and Control Engineering*. Springer, London (2013)
12. Kaelbling, L., Littman, M., Moore, A.: Reinforcement learning: a survey. *J. Artif. Intell. Res.* **4**, 237–285 (1996)
13. Vrabie, D.: Online adaptive optimal control for continuous-time systems, Ph.D. dissertation, University of Texas at Arlington (2010)
14. Vamvoudakis, K.G., Vrabie, D., Lewis, F.L.: Online adaptive algorithm for optimal control with integral reinforcement learning. *Int. J. Robust Nonlinear Control* **24**(17), 2686–2710 (2014)
15. Kamalapurkar, R., Walters, P., Dixon, W.E.: Model-based reinforcement learning for approximate optimal regulation. *Automatica* **64**, 94–104 (2016)
16. He, P., Jagannathan, S.: Reinforcement learning neural-network-based controller for nonlinear discrete-time systems with input constraints. *IEEE Trans. Syst. Man Cybern. Part B Cybern.* **37**(2), 425–436 (2007)
17. Zhang, H., Wei, Q., Luo, Y.: A novel infinite-time optimal tracking control scheme for a class of discrete-time nonlinear systems via the greedy hdp iteration algorithm. *SIEEE Trans. Syst. Man Cybern. Part B Cybern.* **38**(4), 937–942 (2008)
18. Kamalapurkar, R., Rosenfeld, J., Dixon, W.E.: Efficient model-based reinforcement learning for approximate online optimal control. *Automatica* **74**, 247–258 (2016)
19. Al-Tamimi, A., Lewis, F.L., Abu-Khalaf, M.: Model-free q-learning designs for linear discrete-time zero-sum games with application to  $H_\infty$  control. *Automatica* **43**, 473–481 (2007)
20. Vamvoudakis, K.G., Lewis, F.L.: Multi-player non-zero-sum games: Online adaptive learning solution of coupled hamilton-jacobi equations. *Automatica* **47**, 1556–1569 (2011)
21. Vamvoudakis, K.G., Lewis, F.L., Hudas, G.R.: Multi-agent differential graphical games: Online adaptive learning solution for synchronization with optimality. *Automatica* **48**(8), 1598–1611 (2012). <http://www.sciencedirect.com/science/article/pii/S0005109812002476>
22. Modares, H., Lewis, F.L., Naghibi-Sistani, M.-B.: Adaptive optimal control of unknown constrained-input systems using policy iteration and neural networks. *IEEE Trans. Neural Netw. Learn. Syst.* **24**(10), 1513–1525 (2013)
23. Kiumarsi, B., Lewis, F.L., Modares, H., Karimpour, A., Naghibi-Sistani, M.-B.: Reinforcement Q-learning for optimal tracking control of linear discrete-time systems with unknown dynamics. *Automatica* **50**(4), 1167–1175 (2014)
24. Modares, H., Lewis, F.L., Naghibi-Sistani, M.-B.: Integral reinforcement learning and experience replay for adaptive optimal control of partially-unknown constrained-input continuous-time systems. *Automatica* **50**(1), 193–202 (2014)
25. Modares, H., Lewis, F.L.: Optimal tracking control of nonlinear partially-unknown constrained-input systems using integral reinforcement learning. *Automatica* **50**(7), 1780–1792 (2014)

26. Kamalapurkar, R., Walters, P.S., Rosenfeld, J.A., Dixon, W.E.: Reinforcement Learning for Optimal Feedback Control: A Lyapunov-Based Approach. Springer, Berlin (2018)
27. Singh, S.P.: Reinforcement learning with a hierarchy of abstract models. AAAI Natl. Conf. Artif. Intell. **92**, 202–207 (1992)
28. Atkeson, C.G., Schaal, S.: Robot learning from demonstration. Int. Conf. Mach. Learn. **97**, 12–20 (1997)
29. Abbeel, P., Quigley, M., Ng, A.Y.: Using inaccurate models in reinforcement learning. In: International Conference on Machine Learning, pp. 1–8. ACM, New York (2006)
30. Deisenroth, M.P.: Efficient Reinforcement Learning Using Gaussian Processes. KIT Scientific Publishing (2010)
31. Mitrovic, D., Klanke, S., Vijayakumar, S.: Adaptive optimal feedback control with learned internal dynamics models. In: Sigaud, O., Peters, J., (eds.), From Motor Learning to Interaction Learning in Robots. Series Studies in Computational Intelligence, vol. 264, pp. 65–84. Springer Berlin (2010)
32. Deisenroth, M.P., Rasmussen, C.E., Pilco: a model-based and data-efficient approach to policy search. In: International Conference on Machine Learning 2011, pp. 465–472 (2011)
33. Liberzon, D.: Calculus of Variations and Optimal Control Theory: A Concise Introduction. Princeton University Press, Princeton (2012)
34. Kirk, D.: Optimal Control Theory: An Introduction. Dover, Mineola (2004)
35. Sutton, R.S., Barto, A.G.: Reinforcement Learning: An Introduction. MIT Press, Cambridge (1998)
36. Konda, V., Tsitsiklis, J.: On actor-critic algorithms. SIAM J. Control Optim. **42**(4), 1143–1166 (2004)
37. Dierks, T., Jagannathan, S.: Optimal tracking control of affine nonlinear discrete-time systems with unknown internal dynamics. In: Proceedings of the IEEE Conference on Decision and Control, Shanghai, CN, Dec. 2009, pp. 6750–6755 (2009)
38. Vamvoudakis, K.G., Lewis, F.L.: Online synchronous policy iteration method for optimal control. In: Yu, W. (ed.) Recent Advances in Intelligent Control Systems, pp. 357–374. Springer, London (2009)
39. Dierks, T., Jagannathan, S.: Optimal control of affine nonlinear continuous-time systems. In: Proceedings of the American Control Conference, 2010, pp. 1568–1573 (2010)
40. Khalil, H.K.: Nonlinear Systems, 3rd edn. Prentice Hall, Upper Saddle River (2002)
41. Chowdhary, G.: Concurrent learning for convergence in adaptive control without persistency of excitation, Ph.D. dissertation, Georgia Institute of Technology (2010)
42. Chowdhary, G., Johnson, E.: A singular value maximizing data recording algorithm for concurrent learning. In: Proceedings of the American Control Conference, 2011, pp. 3547–3552 (2011)
43. Chowdhary, G., Yucelen, T., Mühlegg, M., Johnson, E.N.: Concurrent learning adaptive control of linear systems with exponentially convergent bounds. Int. J. Adapt. Control Signal Process. **27**(4), 280–301 (2013)
44. Kamalapurkar, R., Walters, P., Dixon, W.E.: Model-based reinforcement learning for approximate optimal regulation. Automatica **64**, 94–104 (2016)
45. Kamalapurkar, R., Andrews, L., Walters, P., Dixon, W.E.: Model-based reinforcement learning for infinite-horizon approximate optimal tracking. IEEE Trans. Neural Netw. Learn. Syst. **28**(3), 753–758 (2017)
46. Kamalapurkar, R., Klotz, J., Dixon, W.E.: Concurrent learning-based online approximate feedback Nash equilibrium solution of N-player nonzero-sum differential games. IEEE/CAA J. Autom. Sin. **1**(3), 239–247 (2014)
47. Luo, B., Wu, H.-N., Huang, T., Liu, D.: Data-based approximate policy iteration for affine nonlinear continuous-time optimal control design. Automatica (2014)
48. Yang, X., Liu, D., Wei, Q.: Online approximate optimal control for affine non-linear systems with unknown internal dynamics using adaptive dynamic programming. IET Control Theory Appl. **8**(16), 1676–1688 (2014)

49. Rosenfeld, J.A., Kamalapurkar, R., Dixon, W.E.: The state following (staf) approximation method. *IEEE Trans. Neural Netw. Learn. Syst.* **30**(6), 1716–1730 (2019)
50. Lorentz, G.G.: Bernstein Polynomials, 2nd edn. Chelsea Publishing Co., New York (1986)
51. Rosenfeld, J.A., Kamalapurkar, R., Dixon, W.E.: State following (StaF) kernel functions for function approximation Part I: theory and motivation. In: Proceedings of the American Control Conference, 2015, pp. 1217–1222 (2015)
52. Deptula, P., Rosenfeld, J., Kamalapurkar, R., Dixon, W.E.: Approximate dynamic programming: combining regional and local state following approximations. *IEEE Trans. Neural Netw. Learn. Syst.* **29**(6), 2154–2166 (2018)
53. Walters, P.S.: Guidance and control of marine craft: an adaptive dynamic programming approach, Ph.D. dissertation, University of Florida (2015)
54. Glorot, X., Bordes, A., Bengio, Y.: Deep sparse rectifier neural networks. In: Proceeding of the International Conference on Artificial Intelligence and Statistics, 2011, pp. 315–323 (2011)
55. Lee, H., Battle, A., Raina, R., Ng, A.Y.: Efficient sparse coding algorithms. In: Proceedings of the Advances in Neural Information Processing Systems, 2007, pp. 801–808 (2007)
56. Nivison, S.A., Khargonekar, P.: Improving long-term learning of model reference adaptive controllers for flight applications: a sparse neural network approach. In: Proceedings of the AIAA Guidance, Navigation and Control Conference, Jan. 2017 (2017)
57. Nivison, S.A., Khargonekar, P.P.: Development of a robust deep recurrent neural network controller for flight applications. In: Proceedings of the American Control Conference, IEEE, 2017, pp. 5336–5342 (2017)
58. Ian Boureau, Y., Cun, Y.L., Ranzato, M.: Sparse feature learning for deep belief networks. In: Proceedings of the Advances in Neural Information Processing Systems, 2008, pp. 1185–1192 (2008)
59. Nivison, S.A., Khargonekar, P.P.: Development of a robust, sparsely-activated, and deep recurrent neural network controller for flight applications. In: Proceedings of the IEEE Conference on Decision and Control, pp. 384–390. IEEE (2018)
60. Nivison, S.A., Khargonekar, P.: A sparse neural network approach to model reference adaptive control with hypersonic flight applications. In: Proceedings of the AIAA Guidance, Navigation and Control Conference, 2018, p. 0842 (2018)
61. Greene, M.L., Deptula, P., Nivison, S., Dixon, W.E.: Sparse learning-based approximate dynamic programming with barrier constraints. *IEEE Control Syst. Lett.* **4**(3), 743–748 (2020)
62. Nivison, S.A.: Sparse and deep learning-based nonlinear control design with hypersonic flight applications, Ph.D. dissertation, University of Florida (2017)
63. Walters, P., Kamalapurkar, R., Voight, F., Schwartz, E., Dixon, W.E.: Online approximate optimal station keeping of a marine craft in the presence of an irrotational current. *IEEE Trans. Robot.* **34**(2), 486–496 (2018)
64. Fan, Q.-Y., Yang, G.-H.: Active complementary control for affine nonlinear control systems with actuator faults. *IEEE Trans. Cybern.* **47**(11), 3542–3553 (2016)

# Chapter 6

## Model-Free Linear Quadratic Regulator



Hesameddin Mohammadi, Mahdi Soltanolkotabi, and Mihailo R. Jovanović

**Abstract** We review recent results on the convergence and sample complexity of the random search method for the infinite-horizon linear quadratic regulator (LQR) problem with unknown model parameters. This method directly searches over the space of stabilizing feedback gain matrices and, in spite of the lack of convexity, it converges to the globally optimal LQR solution at a linear rate. These results demonstrate that for a model-free method that utilizes two-point gradient estimates, the simulation time and the total number of function evaluations required for achieving  $\epsilon$ -accuracy are both  $O(\log(1/\epsilon))$ .

### 6.1 Introduction to a Model-Free LQR Problem

The infinite-horizon LQR problem for continuous-time systems is given by

$$\begin{aligned} & \underset{x,u}{\text{minimize}} \mathbb{E} \left[ \int_0^\infty (x^T(t) Q x(t) + u^T(t) R u(t)) dt \right] \\ & \text{subject to } \dot{x} = Ax + Bu, \quad x(0) \sim \mathcal{D}. \end{aligned} \tag{6.1}$$

Here,  $x(t) \in \mathbb{R}^n$  is the state,  $u(t) \in \mathbb{R}^m$  is the control input,  $A$  and  $B$  are constant matrices of appropriate dimensions that determine parameters of the model,  $Q$  and  $R$  are positive definite matrices, and the expectation is taken over the zero-mean random initial condition  $x(0)$  with distribution  $\mathcal{D}$ . For any controllable pair  $(A, B)$ , the globally optimal solution to (6.1) takes the state feedback form  $u(t) = -K^* x(t)$  and the optimal feedback gain  $K^* \in \mathbb{R}^{m \times n}$  can be obtained by solving the algebraic

---

H. Mohammadi · M. Soltanolkotabi · M. R. Jovanović (✉)

Department of Electrical and Computer Engineering, University of Southern California,  
Los Angeles, CA 90089, USA

e-mail: [mihailo@usc.edu](mailto:mihailo@usc.edu)

H. Mohammadi

e-mail: [hesamedm@usc.edu](mailto:hesamedm@usc.edu)

M. Soltanolkotabi

e-mail: [soltanol@usc.edu](mailto:soltanol@usc.edu)

Riccati equation. However, this approach is not viable for large-scale systems, when prior knowledge of system matrices  $A$  and  $B$  is not available. In this scenario, an alternative approach is to exploit the linearity of the optimal controller and formulate the LQR problem as a direct search over the set of feedback gain matrices  $K$ , namely

$$\underset{K}{\text{minimize}} \quad f(K), \quad (6.2)$$

where  $f(K)$  is the objective function in (6.1) associated with the feedback law  $u = -Kx$ . However, since  $f$  is a nonconvex function of  $K$ , the analysis of local search optimization algorithms is non-trivial. Furthermore, when the model parameters  $A$  and  $B$  are not known, the gradient of the objective function  $f$  is not accessible and only zeroth-order methods that estimate the gradient can be used.

In this chapter, we review recent results on the convergence and sample complexity of the random search method for optimization problem (6.2). This problem was recently examined for both discrete-time [1–3] and continuous-time [4] systems. A common theme is that approximations of the gradient  $\nabla f(K)$  can be obtained via stochastic simulations of system (6.1) [5, 6]. This naturally leads to a zeroth-order optimization approach that attempts to emulate the behavior of gradient descent. To implement this approach, it is essential to have access to approximate function values of the form

$$f_{x_0, \tau}(K) := \int_0^\tau (x^T(t) Q x(t) + u^T(t) R u(t)) dt, \quad (6.3)$$

where  $x(0) = x_0$  is a random initial condition and  $[0, \tau]$  is the finite time horizon. Empirical evidence suggests that the random search method can solve benchmark control problems with state-of-the-art sample efficiency [5]. The fundamental theoretical question is how many function evaluations and what simulation time this method requires to solve problem (6.2) up to the desired level of accuracy  $\epsilon$ .

In [4], the above question was answered for the two-point setting in which, for any pair of points  $K$  and  $K'$ , the simulation engine returns the random values  $f_{x_0, \tau}(K)$  and  $f_{x_0, \tau}(K')$  for some random initial condition  $x_0$ . This is in contrast to the one-point setting in which, at each query, the simulation engine can receive only one specified point  $K$  and return the random value  $f_{x_0, \tau}(K)$ . For convex problems, the gradient estimates obtained in the two-point setting are known to yield faster convergence rates than the one-point setting [7]. However, the two-point setting requires simulations of the system for two different feedback gain matrices under the same initial condition.

For the random search method with one-point gradient estimates to achieve an accuracy level  $\epsilon$  in solving the LQR problem, reference [8] derived an upper bound on the required number of function evaluations that is proportional to  $1/\epsilon^2$ . In this chapter, we review the results in [4] which demonstrated that the random search method with two-point gradient estimates converges at a linear rate with high probability. More specifically, the simulation time and the total number of function evaluations that the random search method requires to achieve an accuracy level  $\epsilon$  are proportional to  $\log(1/\epsilon)$ . These findings suggest that the use of two-point gradi-

ent estimates significantly improves the sample complexity relative to the one-point setting. Finally, while we only focus on continuous-time systems, we note that the proof strategy and results presented here readily extend to discrete-time systems as well [3].

## 6.2 A Gradient-Based Random Search Method

Herein, we present the gradient descent and the random search method for problem (6.2). The LQR objective function in (6.1) associated with the feedback law  $u = -Kx$  is determined by

$$f(K) := \begin{cases} \text{trace}((Q + K^T R K)X(K)), & K \in \mathcal{S} \\ \infty, & \text{otherwise,} \end{cases} \quad (6.4a)$$

where  $\mathcal{S} := \{K \in \mathbb{R}^{m \times n} \mid A - BK \text{ is Hurwitz}\}$  is the set of stabilizing feedback gains  $K$ ,

$$X(K) := \int_0^\infty e^{(A-BK)t} \Omega e^{(A-BK)^T t} dt \succ 0, \quad (6.4b)$$

and  $\Omega := \mathbb{E}[x(0)x^T(0)] \succ 0$  is the covariance matrix of the initial condition  $x(0)$ . Controllability of the pair  $(A, B)$  guarantees  $\mathcal{S} \neq \emptyset$  and, for any  $K \in \mathcal{S}$  [9],

$$\nabla f(K) = 2(RK - B^T P(K))X(K), \quad (6.4c)$$

where

$$P(K) := \int_0^\infty e^{(A-BK)^T t} (Q + K^T R K) e^{(A-BK)t} dt \succ 0. \quad (6.4d)$$

---

### Algorithm 6.1 Gradient estimation

---

**Input:** Feedback gain  $K \in \mathbb{R}^{m \times n}$ , state and control weight matrices  $Q$  and  $R$ , distribution  $\mathcal{D}$ , smoothing constant  $r$ , simulation time  $\tau$ , number of random samples  $N$ .

**for**  $i = 1$  to  $N$  **do**

- Define two perturbed feedback gains  $K_{i,1} := K + rU_i$  and  $K_{i,2} := K - rU_i$ , where  $\text{vec}(U_i)$  is a random vector uniformly distributed on the sphere  $\sqrt{mn} S^{mn-1}$ .

- Sample an initial condition  $x_i$  from the distribution  $\mathcal{D}$ .

- For  $j \in \{1, 2\}$ , simulate system (6.1) up to time  $\tau$  with the feedback gain  $K_{i,j}$  and initial condition  $x_i$  to form  $\hat{f}_{i,j} = \int_0^\tau (x^T(t)Qx(t) + u^T(t)Ru(t)) dt$ .

**end for**

**Output:** The gradient estimate  $\bar{\nabla} f(K) := \frac{1}{2rN} \sum_{i=1}^N (\hat{f}_{i,1} - \hat{f}_{i,2}) U_i$ .

---

With the explicit expression for  $\nabla f(K)$  in (6.4c), the gradient descent method for problem (6.2) with the step-size  $\alpha > 0$  is given by

$$K^{k+1} := K^k - \alpha \nabla f(K^k), \quad K^0 \in \mathcal{S}. \quad (\text{GD})$$

In the model-free setting, the gradient descent method is not directly implementable because computing the gradient  $\nabla f(K)$  requires knowledge of system parameters  $A$  and  $B$ . To address this challenge, we consider the random search method and study its convergence properties. At each iteration, this method calls Algorithm 6.1 to form an empirical approximation  $\bar{\nabla} f(K^k)$  to the gradient  $\nabla f(K^k)$  via simulation of system (6.1) for randomly perturbed feedback gains  $K^k \pm U_i, i = 1, \dots, N$ , (that belong to the sphere of dimension  $mn - 1$  with radius  $\sqrt{mn}$ ) and updates  $K^k$  via

$$K^{k+1} := K^k - \alpha \bar{\nabla} f(K^k), \quad K^0 \in \mathcal{S}. \quad (\text{RS}).$$

Note that the gradient estimation scheme in Algorithm 6.1 does not require knowledge of system matrices  $A$  and  $B$  in (6.1) but only access to a simulation engine.

### 6.3 Main Results

We recently analyzed the sample complexity and convergence of the random search method (RS) for the model-free setting in [4]. Our main convergence result exploits two key properties of the LQR objective function  $f$ , namely smoothness and the Polyak-Łojasiewicz (PL) condition over its sublevel sets  $\mathcal{S}(a) := \{K \in \mathcal{S} \mid f(K) \leq a\}$ , where  $a$  is a positive scalar. In particular, it can be shown that, restricted to any sublevel set  $\mathcal{S}(a)$ , the function  $f$  satisfies

$$\text{Smoothness: } f(K') - f(K) \leq \langle \nabla f(K), K' - K \rangle + \frac{L_f(a)}{2} \|K - K'\|_F^2$$

$$\text{PL condition: } f(K) - f(K^*) \leq \frac{1}{2\mu_f(a)} \|\nabla f(K)\|_F^2$$

for all  $K$  and  $K'$  whose line segment is in  $\mathcal{S}(a)$ , where the smoothness and PL parameters  $L_f(a)$  and  $\mu_f(a)$  are positive rational functions of  $a$  [10]. We also make the following assumption on the statistical properties of the initial condition.

**Assumption 6.1 (Initial distribution)** Let the distribution  $\mathcal{D}$  of the initial condition of system (6.1) have i.i.d. zero-mean unit-variance entries with bounded sub-Gaussian norm. This implies that any random vector  $v \sim \mathcal{D}$  satisfies  $\|v_i\|_{\psi_2} \leq \kappa$ , for some constant  $\kappa$  and  $i = 1, \dots, n$ , where  $\|\cdot\|_{\psi_2}$  is the sub-Gaussian norm.

We note that the zero-mean Gaussian distribution with identity covariance matrix obeys Assumption 6.1. For Gaussian distributions, the above covariance condition is without loss of generality as we can use a change of variables to make the covariance matrix identity. With these definitions and assumptions in place, we are now ready to state the main theoretical result.

**Theorem 6.1** ([4]) Consider the random search method (RS) that uses the gradient estimates of Algorithm 6.1 for finding the optimal solution  $K^*$  of problem (6.2). Let the initial condition  $x_0$  obey Assumption 6.1 and let the simulation time  $\tau$ , the smoothing constant  $r$ , and the number of samples  $N$  satisfy

$$\tau \geq \theta'(a) \log \frac{1}{r\epsilon}, \quad r < \min\{r(a), \theta''(a)\sqrt{\epsilon}\}, \quad N \geq c(1 + \beta^4 \kappa^4 \theta(a) \log^6 n) n,$$

for some  $\beta > 0$  and a desired accuracy  $\epsilon > 0$ . Then, starting from any  $K^0 \in \mathcal{S}(a)$ , the method in (RS) with the constant step-size  $\alpha \leq 1/(32\omega(a)L_f(a))$  achieves  $f(K^k) - f(K^*) \leq \epsilon$  with probability not smaller than  $1 - kp - 2kN\epsilon^{-n}$  in at most

$$k \leq \left( \log \frac{f(K^0) - f(K^*)}{\epsilon} \right) \Big/ \left( \log \frac{1}{1 - \mu_f(a)\alpha/8} \right)$$

iterations. Here,  $\mu_f(a)$  and  $L_f(a)$  are the PL and smoothness parameters of the function  $f$  over the sublevel set  $\mathcal{S}(a)$ ,  $p := c'(n^{-\beta} + N^{-\beta} + Ne^{-\frac{n}{8}} + e^{-c'N})$ ,  $\omega(a) := c''(\sqrt{m} + \beta\kappa^2\theta(a)\sqrt{mn}\log n)^2$ ,  $c$ ,  $c'$ , and  $c''$  are positive absolute constants, and  $\theta(a)$ ,  $\theta'(a)$ ,  $\theta''(a)$  and  $r(a)$  are problem-dependent positive parameters.

For a desired accuracy level  $\epsilon > 0$ , Theorem 6.1 shows that the random search iterates (RS) with constant step-size (that does not depend on  $\epsilon$ ) reach an accuracy level  $\epsilon$  at a linear rate (i.e., in at most  $O(\log(1/\epsilon))$  iterations) with high probability. Furthermore, the total number of function evaluations and the simulation time required to achieve an accuracy level  $\epsilon$  are proportional to  $\log(1/\epsilon)$ . This significantly improves existing results for model-free LQR that require  $O(1/\epsilon)$  function evaluations [8].

## 6.4 Proof Sketch

The smoothness of the objective function along with the PL condition are sufficient for the gradient descent method to achieve linear convergence even for nonconvex problems [11]. These properties were recently used to show convergence of gradient descent for both *discrete-time* [1] and *continuous-time* [10] LQR problems.

**Theorem 6.2** ([10]) Consider the gradient descent method (GD) for finding the optimal solution  $K^*$  of problem (6.2). For any initialization  $K^0 \in \mathcal{S}(a)$ , with the constant step-size  $\alpha = 1/L_f(a)$ , we have

$$\begin{aligned} f(K^k) - f(K^*) &\leq \gamma^k (f(K^0) - f(K^*)) \\ \|K^k - K^*\|_F^2 &\leq b \gamma^k \|K^0 - K^*\|_F^2, \end{aligned}$$

where  $\gamma = 1 - \mu_f(a)/L_f(a)$  is the linear rate of convergence and  $b > 0$  depends only on the scalar  $a$  and the parameters of the LQR objective function  $f$ .

The random search method (RS), however, does not have access to the true value of the gradient  $\nabla f(K)$  as Algorithm 6.1 produces only a biased estimate  $\bar{\nabla}f(K)$  of  $\nabla f(K)$ . Unlike existing techniques that directly work with the gradient estimation error [1, 8], the proof of Theorem 6.1 is based on establishing a high correlation between the gradient estimate and the true gradient. In particular, the proof exploits Proposition 6.1 that establishes a geometric decrease in the objective value by using an update direction  $G$  which is not necessarily close to  $\nabla f(K)$  but is well correlated with it.

**Proposition 6.1** (Approximate GD [4]) *If the matrix  $G \in \mathbb{R}^{m \times n}$  and the feedback gain  $K \in \mathcal{S}(a)$  are such that*

$$\langle G, \nabla f(K) \rangle \geq \mu_1 \|\nabla f(K)\|_F^2, \quad \|G\|_F^2 \leq \mu_2 \|\nabla f(K)\|_F^2, \quad (6.5)$$

for some scalars  $\mu_1, \mu_2 > 0$ , then  $K - \alpha G \in \mathcal{S}(a)$  for all  $\alpha \in [0, \mu_1/(\mu_2 L_f(a))]$ , and

$$f(K - \alpha G) - f(K^*) \leq (1 - \mu_f(a)\mu_1\alpha)(f(K) - f(K^*)),$$

where  $L_f(a)$  and  $\mu_f(a)$  are the smoothness and PL parameters of the LQR objective function  $f$  over  $\mathcal{S}(a)$ .

To better understand the implications of Proposition 6.1, let us consider the trivial example  $G = \nabla f(K)$ . In this case, the inequalities in (6.5) hold with  $\mu_1 = \mu_2 = 1$ . Thus, for the step-size  $\alpha = 1/L_f(a)$ , we recover the linear convergence rate of  $1 - \mu_f(a)/L_f(a)$  that was established for the gradient descent method in Theorem 6.2.

In our convergence analysis, we do not show that  $\bar{\nabla}f(K)$  obeys the approximate GD condition in Proposition 6.2 directly. Instead, we introduce an unbiased estimate  $\hat{\nabla}f(K)$  of the gradient  $\nabla f(K)$  in Eq. (6.7) and establish the approximate GD condition for this estimate. We then show that the approximate gradient  $\bar{\nabla}f(K)$  that is utilized in our algorithm remains close to this unbiased estimate. More specifically, we establish the following two key properties: first, for any  $\epsilon > 0$ , using a simulation time  $\tau = O(\log(1/\epsilon))$  and an appropriate smoothing parameter  $r$  in Algorithm 6.1, the estimation bias  $\|\hat{\nabla}f(K) - \bar{\nabla}f(K)\|_F$  can be made smaller than  $\epsilon$ ; and, second, with  $N = \tilde{O}(n)$  samples, the unbiased estimate  $\hat{\nabla}f(K)$  becomes well correlated with  $\nabla f(K)$  with *high probability*. In particular, the events

$$\begin{aligned} M_1 &:= \left\{ \langle \hat{\nabla}f(K), \nabla f(K) \rangle \geq \mu_1 \|\nabla f(K)\|_F^2 \right\}, \\ M_2 &:= \left\{ \|\hat{\nabla}f(K)\|_F^2 \leq \mu_2 \|\nabla f(K)\|_F^2 \right\} \end{aligned} \quad (6.6)$$

occur with high probability for some positive scalars  $\mu_1$  and  $\mu_2$ .

These two properties combined with Proposition 6.1 are the key ingredients that were used to analyze convergence of the random search method (RS) and prove Theorem 6.1. We next present main ideas that were used to establish these properties.

### 6.4.1 Controlling the Bias

Herein, we define the unbiased estimate  $\widehat{\nabla} f(K)$  of the gradient and quantify an upper bound on its distance to the output  $\overline{\nabla} f(K)$  of Algorithm 6.1. To simplify our presentation, for any  $K \in \mathbb{R}^{m \times n}$ , we define the closed-loop Lyapunov operator  $\mathcal{A}_K: \mathbb{S}^n \rightarrow \mathbb{S}^n$ ,

$$\mathcal{A}_K(X) := (A - BK)X + X(A - BK)^T,$$

where  $\mathbb{S}^n$  is the set of symmetric matrices. For  $K \in \mathcal{S}$ , both  $\mathcal{A}_K$  and its adjoint

$$\mathcal{A}_K^*(P) = (A - BK)^T P + P(A - BK)$$

are invertible. Moreover, we can represent the matrices  $X(K)$  and  $P(K)$  in (6.4) as

$$X(K) = \mathcal{A}_K^{-1}(-\Omega), \quad P(K) = (\mathcal{A}_K^*)^{-1}(-K^T R K - Q).$$

For any  $\tau \geq 0$  and  $x_0 \in \mathbb{R}^n$ , let  $f_{x_0, \tau}(K)$  denote the  $\tau$ -truncated version of the LQR objective function associated with system (6.1) with the initial condition  $x(0) = x_0$  and the feedback law  $u = -Kx$  as defined in (6.3). For any  $K \in \mathcal{S}$  and  $x_0 \in \mathbb{R}^n$ , the infinite-horizon cost  $f_{x_0}(K) := f_{x_0, \infty}(K)$  exists and it satisfies  $f(K) = \mathbb{E}_{x_0}[f_{x_0}(K)]$ . Furthermore, the gradient of  $f_{x_0}(K)$  is given by (cf. (6.4c))

$$\nabla f_{x_0}(K) = 2(RK - B^T P(K))X_{x_0}(K), \quad X_{x_0}(K) := \mathcal{A}_K^{-1}(-x_0 x_0^T).$$

Since the gradients  $\nabla f(K)$  and  $\nabla f_{x_0}(K)$  are linear in  $X(K)$  and  $X_{x_0}(K)$ , respectively, for the random initial condition  $x(0) = x_0$  with  $\mathbb{E}[x_0 x_0^T] = \Omega$ , it follows that

$$\mathbb{E}_{x_0}[X_{x_0}(K)] = X(K), \quad \mathbb{E}_{x_0}[\nabla f_{x_0}(K)] = \nabla f(K).$$

Next, we define the following three estimates of the gradient

$$\begin{aligned} \overline{\nabla} f(K) &:= \frac{1}{2rN} \sum_{i=1}^N (f_{x_i, \tau}(K + rU_i) - f_{x_i, \tau}(K - rU_i)) U_i \\ \widetilde{\nabla} f(K) &:= \frac{1}{2rN} \sum_{i=1}^N (f_{x_i}(K + rU_i) - f_{x_i}(K - rU_i)) U_i \\ \widehat{\nabla} f(K) &:= \frac{1}{N} \sum_{i=1}^N \langle \nabla f_{x_i}(K), U_i \rangle U_i. \end{aligned} \tag{6.7}$$

Here,  $U_i \in \mathbb{R}^{m \times n}$  are i.i.d. random matrices with  $\text{vec}(U_i)$  uniformly distributed on the sphere  $\sqrt{mn} S^{mn-1}$  and  $x_i \in \mathbb{R}^n$  are i.i.d. random initial conditions sampled from distribution  $\mathcal{D}$ . Note that  $\widetilde{\nabla} f(K)$  is the infinite-horizon version of

$\bar{\nabla} f(K)$  produced by Algorithm 6.1 and  $\widehat{\nabla} f(K)$  is an unbiased estimate of  $\nabla f(K)$ . The fact that  $\mathbb{E}[\widehat{\nabla} f(K)] = \nabla f(K)$  follows from  $\mathbb{E}_{U_1}[\text{vec}(U_1)\text{vec}(U_1)^T] = I$  and  $\mathbb{E}_{x_i, U_i}[\text{vec}(\widehat{\nabla} f(K))] = \mathbb{E}_{U_1}[\langle \nabla f(K), U_1 \rangle \text{vec}(U_1)] = \text{vec}(\nabla f(K))$ .

**Local boundedness of the function  $f(K)$ :** An important requirement for the gradient estimation scheme in Algorithm 6.1 is the stability of the perturbed closed-loop systems, i.e.,  $K \pm rU_i \in \mathcal{S}$ ; violating this condition leads to an exponential growth of the state and control signals. Moreover, this condition is necessary and sufficient for  $\widetilde{\nabla} f(K)$  and  $\widehat{\nabla} f(K)$  to be well defined. It can be shown that for any sublevel set  $\mathcal{S}(a)$ , there exists a positive radius  $r$  such that  $K + rU \in \mathcal{S}$  for all  $K \in \mathcal{S}(a)$  and  $U \in \mathbb{R}^{m \times n}$  with  $\|U\|_F \leq \sqrt{mn}$ . Herein, we further require that  $r$  is small enough so that  $K \pm rU_i \in \mathcal{S}(2a)$  for all  $K \in \mathcal{S}(a)$ . Such upper bound on  $r$  is provided in Lemma 6.1.

**Lemma 6.1** ([4]) *For any  $K \in \mathcal{S}(a)$  and  $U \in \mathbb{R}^{m \times n}$  with  $\|U\|_F \leq \sqrt{mn}$ ,  $K + r(a)U \in \mathcal{S}(2a)$ , where  $r(a) := \tilde{c}/a$  for some constant  $\tilde{c} > 0$  that depends on the problem data.*

Note that for any  $K \in \mathcal{S}(a)$  and  $r \leq r(a)$  in Lemma 6.1,  $\widetilde{\nabla} f(K)$  and  $\widehat{\nabla} f(K)$  are well defined since the feedback gains  $K + rU_i$  are all stabilizing. We next present an upper bound on the difference between the output  $\bar{\nabla} f(K)$  of Algorithm 6.1 and the unbiased estimate  $\widehat{\nabla} f(K)$  of the gradient  $\nabla f(K)$ . This can be accomplished by bounding the difference between these two quantities and  $\widetilde{\nabla} f(K)$  through the use of the triangle inequality

$$\|\widehat{\nabla} f(K) - \bar{\nabla} f(K)\|_F \leq \|\widetilde{\nabla} f(K) - \bar{\nabla} f(K)\|_F + \|\widehat{\nabla} f(K) - \widetilde{\nabla} f(K)\|_F. \quad (6.8)$$

Proposition 6.2 provides an upper bound on each term on the right-hand side of (6.8).

**Proposition 6.2** ([4]) *For any  $K \in \mathcal{S}(a)$  and  $r \leq r(a)$ , where  $r(a)$  is given by Lemma 6.1,*

$$\begin{aligned} \|\widetilde{\nabla} f(K) - \bar{\nabla} f(K)\|_F &\leq \frac{\sqrt{mn} \max_i \|x_i\|^2}{r} \kappa_1(2a) e^{-\tau/\kappa_2(2a)} \\ \|\widehat{\nabla} f(K) - \widetilde{\nabla} f(K)\|_F &\leq \frac{(rmn)^2}{2} \ell(2a) \max_i \|x_i\|^2, \end{aligned}$$

where  $\ell(a) > 0$ ,  $\kappa_1(a) > 0$ , and  $\kappa_2(a) > 0$  are polynomials of degree less than 5.

The first term on the right-hand side of (6.8) corresponds to a bias arising from the finite-time simulation. Proposition 6.2 shows that although small values of  $r$  may yield large  $\|\widetilde{\nabla} f(K) - \bar{\nabla} f(K)\|_F$ , because of the exponential dependence of the upper bound on the simulation time  $\tau$ , this error can be controlled by increasing  $\tau$ . In addition, since  $\widehat{\nabla} f(K)$  is independent of the parameter  $r$ , this result provides a quadratic bound on the estimation error in terms of  $r$ . It is also worth mentioning that the third derivative of the function  $f_{x_0}(K)$  is utilized in obtaining the second inequality.

### 6.4.2 Correlation of $\widehat{\nabla} f(K)$ and $\nabla f(K)$

In this section, we show that the events  $M_i$  in (6.6) hold with high probability. The key enabler of the proof is that the random inner product  $\langle \nabla f(K), \widehat{\nabla} f(K) \rangle$  is very well concentrated around its mean  $\|\nabla f(K)\|_F^2$ . We next describe this phenomenon in more detail. The proof exploits the problem structure to confine the dependence of  $\widehat{\nabla} f(K)$  on the random initial conditions  $x_i$  into a zero-mean random vector. In particular, for any given feedback gain  $K \in \mathcal{S}$  and initial condition  $x_0 \in \mathbb{R}^n$ , we have  $\nabla f(K) = EX$ , and  $\nabla f_{x_0}(K) = EX_{x_0}$ , where  $E := 2(RK - B^T P(K)) \in \mathbb{R}^{m \times n}$  is a fixed matrix,  $X = -\mathcal{A}_K^{-1}(\Omega)$ , and  $X_{x_0} = -\mathcal{A}_K^{-1}(x_0 x_0^T)$ . Thus, we can represent the gradient estimate  $\widehat{\nabla} f(K)$  as

$$\widehat{\nabla} f(K) = \frac{1}{N} \sum_{i=1}^N \langle EX_{x_i}, U_i \rangle U_i = \widehat{\nabla}_1 + \widehat{\nabla}_2,$$

where  $\widehat{\nabla}_1 := \frac{1}{N} \sum_{i=1}^N \langle E(X_{x_i} - X), U_i \rangle U_i$  and  $\widehat{\nabla}_2 := \frac{1}{N} \sum_{i=1}^N \langle \nabla f(K), U_i \rangle U_i$ . Note

that  $\widehat{\nabla}_2$  does not depend on the initial conditions  $x_i$ . Moreover, from  $\mathbb{E}[X_{x_i}] = X$  and the independence of  $X_{x_i}$  and  $U_i$ , we have  $\mathbb{E}[\widehat{\nabla}_1] = 0$  and  $\mathbb{E}[\widehat{\nabla}_2] = \nabla f(K)$ .

We next present the key technical results that were used to study the probability of the events  $M_i$  in (6.6) for suitable values of  $\mu_1$  and  $\mu_2$ . These results are obtained using standard tools from non-asymptotic statistical analysis of the concentration of random variables around their average; see a recent book [12] for a comprehensive discussion. Herein, we use  $c, c', c'',$  etc. to denote positive absolute constants.

Proposition 6.3 can be used to show that with enough samples  $N = \tilde{O}(n)$ , the inner product of the zero-mean term  $\widehat{\nabla}_1$  and the gradient  $\nabla f(K)$  can be controlled with high probability. This result is the key for analyzing the probability of the event  $M_1$ .

**Proposition 6.3** ([4]) *Let  $X_1, \dots, X_N \in \mathbb{R}^{n \times n}$  be i.i.d. random matrices distributed according to  $\mathcal{M}(xx^T)$ , where  $x \in \mathbb{R}^n$  is a random vector whose distribution obeys Assumption 6.1 and  $\mathcal{M}$  is a linear operator, and let  $X := \mathbb{E}[X_1] = \mathcal{M}(I)$ . Also, let  $U_1, \dots, U_N \in \mathbb{R}^{m \times n}$  be i.i.d. random matrices with  $\text{vec}(U_i)$  uniformly distributed on the sphere  $\sqrt{mn} S^{mn-1}$ . For any  $E \in \mathbb{R}^{m \times n}$  and positive scalars  $\delta$  and  $\beta$ ,*

$$\mathbb{P} \left\{ \left| \frac{1}{N} \sum_{i=1}^N \langle E(X_i - X), U_i \rangle \langle EX, U_i \rangle \right| \leq \delta \|EX\|_F \|E\|_F \right\} \geq 1 - C' N^{-\beta} - 4N e^{-\frac{n}{8}}$$

if  $N \geq C(\beta^2 \kappa^2 / \delta)^2 (\|\mathcal{M}^*\|_2 + \|\mathcal{M}^*\|_S)^2 n \log^6 n$ , where  $\|\cdot\|_S$  denotes the spectral induced norm.

The proof of Proposition 6.3 exploits the Hanson–Wright inequality along with a well-known upper bound on the norm of random matrices [13, Theorems 1.1 and 3.2]. In Proposition 6.4, we present a technical result that can be used to show that  $\langle \widehat{\nabla}_2, \nabla f(K) \rangle$  concentrates with high probability around its average  $\|\nabla f(K)\|_F^2$ .

**Proposition 6.4** ([4]) Let  $U_1, \dots, U_N \in \mathbb{R}^{m \times n}$  be i.i.d. random matrices with each  $\text{vec}(U_i)$  uniformly distributed on the sphere  $\sqrt{mn} S^{mn-1}$ . Then, for any  $W \in \mathbb{R}^{m \times n}$  and scalar  $t \in (0, 1]$ , we have

$$\mathbb{P} \left\{ \frac{1}{N} \sum_{i=1}^N \langle W, U_i \rangle^2 < (1 - t) \|W\|_F^2 \right\} \leq 2 e^{-cNt^2}.$$

The proof of Proposition 6.3 relies on the Bernstein inequality [12, Corollary 2.8.3]. Using Propositions 6.3 and 6.4, it is straightforward to show that the event  $M_1$  occurs with high probability.

Next, we turn our attention to quantifying the probability of the event  $M_2$  in (6.6). Proposition 6.5 presents a technical result that can be used to quantify a high probability upper bound on  $\|\widehat{\nabla}_1\|_F / \|\nabla f(K)\|$ . This result is analogous to Proposition 6.3 and it can be used to study the event  $M_2$ .

**Proposition 6.5** ([4]) Let  $X_i$  and  $U_i$  with  $i = 1, \dots, N$  be random matrices defined in Lemma 6.3,  $X := \mathbb{E}[X_1]$ , and let  $N \geq c_0 n$ . For any  $E \in \mathbb{R}^{m \times n}$  and  $\beta > 0$ , we have

$$\frac{1}{N} \left\| \sum_{i=1}^N \langle E(X_i - X), U_i \rangle U_i \right\|_F \leq c_1 \beta \kappa^2 (\|\mathcal{M}^*\|_2 + \|\mathcal{M}^*\|_S) \|E\|_F \sqrt{mn} \log n$$

with probability not smaller than  $1 - c_2(n^{-\beta} + Ne^{-\frac{n}{8}})$ .

In Proposition 6.6, we present a technical result that can be used to study  $\|\widehat{\nabla}_2\|_F / \|\nabla f(K)\|$ .

**Proposition 6.6** ([4]) Let  $U_1, \dots, U_N \in \mathbb{R}^{m \times n}$  be i.i.d. random matrices with  $\text{vec}(U_i)$  uniformly distributed on the sphere  $\sqrt{mn} S^{mn-1}$  and let  $N \geq Cn$ . Then, for any  $W \in \mathbb{R}^{m \times n}$ ,

$$\mathbb{P} \left\{ \frac{1}{N} \left\| \sum_{j=1}^N \langle W, U_j \rangle U_j \right\|_F > C' \sqrt{m} \|W\|_F \right\} \leq 2Ne^{-\frac{mn}{8}} + 2e^{-\hat{c}N}.$$

Using Propositions 6.5 and 6.6, it is straightforward to show that the event  $M_2$  occurs with high probability.

## 6.5 An Example

We consider a mass-spring-damper system with  $s = 10$  masses, where we set all mass, spring, and damping constants to unity. In state-space representation (6.1), the state  $x = [p^T v^T]^T$  contains the position and velocity vectors and the dynamic and input matrices are given by

$$A = \begin{bmatrix} 0 & I \\ -T & -T \end{bmatrix}, \quad B = \begin{bmatrix} 0 \\ I \end{bmatrix},$$

where  $0$  and  $I$  are  $s \times s$  zero and identity matrices, and  $T$  is a Toeplitz matrix with  $2$  on the main diagonal,  $-1$  on the first super and sub-diagonals, and zero elsewhere. In this example, the  $A$ -matrix is Hurwitz and the objective of control is to optimize the LQR cost with  $Q$  and  $R$  equal to identity. We also let the initial conditions  $x_i$  in Algorithm 6.1 be standard normal and use  $N = n = 2s$  samples.

For two values of the smoothing parameter  $r = 10^{-4}$  (blue) and  $r = 10^{-5}$  (red), and for  $K = 0$ , we illustrate in Fig. 6.1a the dependence of the relative error  $\|\widehat{\nabla}f(K) - \bar{\nabla}f(K)\|_F / \|\widehat{\nabla}f(K)\|_F$ , and in Fig. 6.1b, that of the total relative error  $\|\nabla f(K) - \bar{\nabla}f(K)\|_F / \|\nabla f(K)\|_F$  on the simulation time  $\tau$ . In Fig. 6.1a, we observe an exponential decrease in error for small values of  $\tau$ . In addition, the error does not pass a saturation level which is determined by  $r$ . We also see that, as  $r$  decreases, this saturation level becomes smaller. These observations are in harmony with the theoretical developments presented in this chapter; in particular, Proposition 6.2 coupled with the triangle inequality yield

$$\|\widehat{\nabla}f(K) - \bar{\nabla}f(K)\|_F \leq \left( \frac{\sqrt{mn}}{r} \kappa_1(2a) e^{-\kappa_2(2a)\tau} + \frac{r^2 m^2 n^2}{2} \ell(2a) \right) \max_i \|x_i\|^2.$$

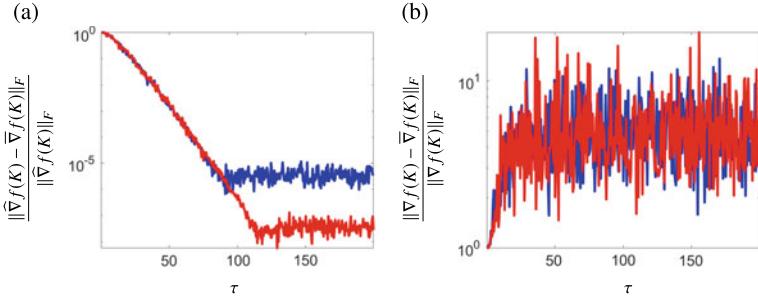
This upper bound clearly captures the exponential dependence of the bias on the simulation time  $\tau$  as well as the saturation level that depends quadratically on the smoothing parameter  $r$ .

On the other hand, in Fig. 6.1b, we observe that the distance between the approximate gradient  $\bar{\nabla}f(K)$  and the true gradient is rather large. In contrast to the existing results which rely on the use of the estimation error shown in Fig. 6.1b, Proposition 6.2 shows that the simulated gradient  $\bar{\nabla}f(K)$  is close to the gradient estimate  $\widehat{\nabla}f(K)$ , which although is not close to the true gradient  $\nabla f(K)$ , is highly correlated with it. This is sufficient for establishing convergence guarantees and reducing both sample complexity and simulation time to  $O(\log(1/\epsilon))$ .

Finally, Fig. 6.2 demonstrates linear convergence of the random search method (RS) with step-size  $\alpha = 10^{-4}$ , and  $(r = 10^{-5}, \tau = 200)$  in Algorithm 6.1, as established in Theorem 6.1.

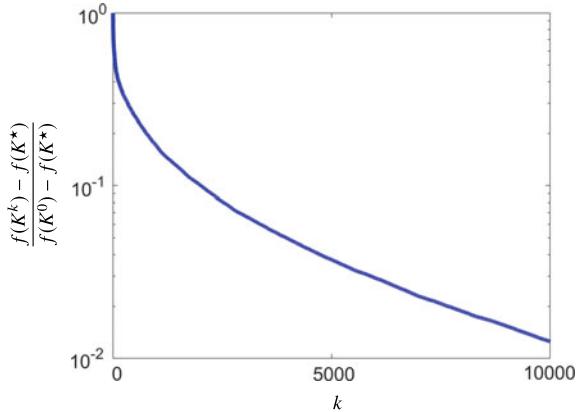
## 6.6 Thoughts and Outlook

For the *discrete-time* LQR problem, global convergence guarantees were provided in [1] for gradient descent and the random search methods with one-point gradient estimates. Furthermore, bound on the sample complexity for reaching the error tolerance  $\epsilon$  that requires a number of function evaluations that is at least proportional to  $(1/\epsilon^4) \log(1/\epsilon)$  was established. If one has access to the infinite-horizon cost values, i.e., if  $\tau = \infty$ , the number of function evaluations for the random search method with



**Fig. 6.1** **a** Bias in gradient estimation and **b** total error in gradient estimation as functions of the simulation time  $\tau$ . The blue and red curves correspond to two values of the smoothing parameter  $r = 10^{-4}$  and  $r = 10^{-5}$ , respectively

**Fig. 6.2** Convergence curve of the random search method (RS)



one-point estimates was improved to  $1/\epsilon^2$  in [8]. Moreover, this work showed that the use of two-point estimates reduces the number of function evaluations to  $1/\epsilon$ .

In this chapter, we focus on the *continuous-time* LQR problem and summarize the results presented in [4, 10, 14, 15]. These recent references demonstrate that the random search method with two-point gradient estimates converges to the optimal solution at a linear rate with high probability. Relative to the existing literature, a significant improvement is offered both in terms of the required function evaluations and simulation time. Specifically, the total number of function evaluations required to achieve an accuracy level  $\epsilon$  is proportional to  $\log(1/\epsilon)$  compared to at least  $(1/\epsilon^4) \log(1/\epsilon)$  in [1] and  $1/\epsilon$  in [8]. Similarly, the simulation time required to achieve an accuracy level  $\epsilon$  is proportional to  $\log(1/\epsilon)$ ; this is in contrast to [1] which requires  $\text{poly}(1/\epsilon)$  simulation time and [8] which assumes an infinite simulation time. We refer the reader to [4] for a comprehensive discussion along with all technical details that are omitted here for brevity.

## References

1. Fazel, M., Ge, R., Kakade, S.M., Mesbahi, M.: Global convergence of policy gradient methods for the linear quadratic regulator. In: Proceedings of the International Conference on Machine Learning, pp. 1467–1476 (2018)
2. Bu, J., Mesbahi, A., Fazel, M., Mesbahi, M.: LQR through the lens of first order methods: discrete-time case (2019). [arXiv:1907.08921](https://arxiv.org/abs/1907.08921)
3. Mohammadi, H., Soltanolkotabi, M., Jovanović, M.R.: On the linear convergence of random search for discrete-time LQR. *IEEE Control Syst. Lett.* **5**(3), 989–994 (2021)
4. Mohammadi, H., Zare, A., Soltanolkotabi, M., Jovanović, M.R.: Convergence and sample complexity of gradient methods for the model-free linear quadratic regulator problem. *IEEE Trans. Autom. Control* (2019), conditionally accepted; also [arXiv:1912.11899](https://arxiv.org/abs/1912.11899)
5. Mania, H., Guy, A., Recht, B.: Simple random search provides a competitive approach to reinforcement learning. In: Proceedings of the Neural Information Processing (NeurIPS) (2018)
6. Recht, B.: A tour of reinforcement learning: the view from continuous control. *Ann. Rev. Control Robot. Auton. Syst.* **2**, 253–279 (2019)
7. Duchi, J.C., Jordan, M.I., Wainwright, M.J., Wibisono, A.: Optimal rates for zero-order convex optimization: the power of two function evaluations. *IEEE Trans. Inf. Theory* **61**(5), 2788–2806 (2015)
8. Malik, D., Panajady, A., Bhatia, K., Khamaru, K., Bartlett, P.L., Wainwright, M.J.: Derivative-free methods for policy optimization: guarantees for linear-quadratic systems. *J. Mach. Learn. Res.* **51**, 1–51 (2020)
9. Levine, W.S., Athans, M.: On the determination of the optimal constant output feedback gains for linear multivariable systems. *IEEE Trans. Autom. Control* **15**(1), 44–48 (1970)
10. Mohammadi, H., Zare, A., Soltanolkotabi, M., Jovanović, M.R.: Global exponential convergence of gradient methods over the nonconvex landscape of the linear quadratic regulator. In: Proceedings of the 58th IEEE Conference on Decision and Control, Nice, France, pp. 7474–7479 (2019)
11. Karimi, H., Nutini, J., Schmidt, M.: Linear convergence of gradient and proximal-gradient methods under the Polyak-Łojasiewicz condition. In: In European Conference on Machine Learning, pp. 795–811 (2016)
12. Vershynin, R.: High-Dimensional Probability: An Introduction with Applications in Data Science, vol. 47. Cambridge University Press, Cambridge (2018)
13. Rudelson, M., Vershynin, R.: Hanson-Wright inequality and sub-Gaussian concentration. *Electron. Commun. Prob.* **18** (2013)
14. Mohammadi, H., Soltanolkotabi, M., Jovanović, M.R.: Random search for learning the linear quadratic regulator. In: Proceedings of the 2020 American Control Conference, Denver, pp. 4798–4803 (2020)
15. Mohammadi, H., Soltanolkotabi, M., Jovanović, M.R.: Learning the model-free linear quadratic regulator via random search. In: Proceedings of Machine Learning Research, 2nd Annual Conference on Learning for Dynamics and Control, Berkeley, vol. 120, pp. 1–9 (2020)

## **Part II**

# **Constraint-Driven and Verified RL**

## Chapter 7

# Adaptive Dynamic Programming in the Hamiltonian-Driven Framework



**Yongliang Yang, Donald C. Wunsch II, and Yixin Yin**

**Abstract** This chapter presents a Hamiltonian-driven framework of adaptive dynamic programming (ADP) for continuous-time nonlinear systems. Three fundamental problems for solving the optimal control problem are presented, i.e., the evaluation of given admissible policy, the comparison between two different admissible policies with respect to the performance, and the performance improvement of given admissible control. It is shown that the Hamiltonian functional can be viewed as the temporal difference for dynamical systems in continuous time. Therefore, the minimization of the Hamiltonian functional is equivalent to the value function approximation. An iterative algorithm starting from an arbitrary admissible control is presented for the optimal control approximation with its convergence proof. The Hamiltonian-driven ADP algorithm can be implemented using a critic only structure,

---

This work was supported in part by the National Natural Science Foundation of China under Grant No. 61903028, in part by the China Post-Doctoral Science Foundation under Grant 2018M641197, in part by the Fundamental Research Funds for the Central Universities under grant No. FRF-TP-18-031A1 and No. FRF-BD-17-002A, in part by the DARPA/Microsystems Technology Office and in part by the Army Research Laboratory under grant No. W911NF-18-2-0260. The views and conclusions contained in this document are those of the authors and should not be interpreted as representing the official policies, either expressed or implied, of the Army Research Laboratory or the U.S. Government. The U.S. Government is authorized to reproduce and distribute reprints for Government purposes notwithstanding any copyright notation herein.

---

Y. Yang (✉)

School of Automation and Electrical Engineering, University of Macau, Avenida da Universidade, Taipa, Macau, China

e-mail: [yangyongliang@ieee.org](mailto:yangyongliang@ieee.org)

Y. Yin

School of Automation and Electrical Engineering, University of Science and Technology Beijing, No. 30 Xueyuan Road, Beijing 100083, China

e-mail: [yx@ies.ustb.edu.cn](mailto:yx@ies.ustb.edu.cn)

D. C. Wunsch II

Department of Electrical and Computer Engineering, Missouri University of Science and Technology, 301 W. 16th St., Rolla, MO 65409, USA

e-mail: [dwunsch@mst.edu](mailto:dwunsch@mst.edu)

which is trained to approximate the optimal value gradient. Simulation example is conducted to verify the effectiveness of Hamiltonian-driven ADP.

**Keywords** Hamiltonian-driven framework · Adaptive dynamic programming · Temporal difference · Value gradient learning

## 7.1 Introduction

The optimal control problems were originally formulated by the calculus of variations. Both the necessary [22] and sufficient [2] conditions for optimality provided the foundation for the optimal control derivations. According to Pontryagin's maximum principle, variational analysis results in a two-point boundary value problem that is challenging to solve and the result is open loop. In contrast, the dynamic programming principle leads to the Hamilton–Jacobi–Bellman (HJB) equation, which does not have an analytical solution and suffers from “the curse of dimensionality”. Therefore, approximation methods for solving the HJB equation have been developed and referred to as adaptive critic designs [23], or alternatively, approximate/adaptive dynamic programming (ADP) [16, 33, 35, 36].

### 7.1.1 Literature Review

After [33], various ADP methods were developed, such as [9, 11–13, 23, 32] among many others. The action-dependent approaches do not use a model NN or any other model to achieve the adaptation of the action and critic networks, which yields the model-free implementation. Heuristic dynamic programming (HDP), dual heuristic dynamic programming (DHP), and their action-dependent versions ADHDP and ADDHP, have been described in [35]. Globalized DHP (GDHP) is described in [16, 37], and is successfully implemented in [35]. Action-dependent GDHP (ADGDHP) is also developed in [23]. Padhi et al. simplifies the actor–critic structure to a single critic network for discrete-time dynamical systems, which is referred to as a single network adaptive critic [21]. In this chapter, critic-only structure is presented for continuous-time dynamical systems.

In addition to the action-dependent variants of ADP algorithms, a data-based model-free extension of ADP algorithms have been developed, such as  $Q$ -learning-based model-free ADP [28, 41]. Also, ADP algorithms evolved from on-policy [29] to off-policy [7, 10, 27]. Off-policy allows for learning without any knowledge of system dynamics. Additionally, in an off-policy learning algorithm, the policy used to generate behavior may in fact be unrelated to the policy that is evaluated and improved [38–40]. Inspired by this idea, the approximate solution to the Hamilton–Jacobi–Isaacs equation was developed [15, 18], resulting from the  $H_\infty$  control problems.

The temporal difference plays an important role in value function approximation. There are a lot of temporal difference-based ADP algorithms developed for discrete-time systems, see [11–13, 16, 23, 32, 35, 37] for reference. Recently, a temporal difference-based method was also presented for continuous-time systems, which is referred to as the integral temporal difference [12, 13, 17, 19, 31]. In this chapter, a novel version of the temporal difference for the continuous-time systems is presented using the Hamiltonian functional.

### 7.1.2 Motivation

Consider the unconstrained optimization problem

$$\min_{x \in X} F(x),$$

where  $X$  is the finite-dimensional decision space. Denote  $x^* = \arg \min_x F(x)$  as the optimal decision and  $x_k$  as the current estimate. In order to achieve a better successive estimation  $x_{k+1}$  with  $F(x_{k+1}) < F(x_k)$ , the condition  $\langle x_{k+1} - x_k, \nabla F(x_k) \rangle < 0$  is required [3]. The optimal solution search problem can be split into the following steps,

1. To evaluate current estimation  $x_k$ ;
2. To build the criterion by which different estimations can be compared;
3. To design an update rule that makes a better estimation  $x_{k+1}$ , based on the criteria in the previous steps.

The optimal control problem can also be viewed as an optimal search problem in the infinite-dimensional policy space. Inspired by the decision search in finite-dimensional space, this chapter develops a novel framework of the optimal control policy search problem that is parallel to the framework above, i.e.,

1. To build a criterion that evaluates an arbitrary admissible control  $u_k(\cdot)$ , i.e., calculate the corresponding cost  $J(u_k)$ ;
2. To establish a rule that compares two admissible controls;
3. To design a successive control  $u_{k+1}(\cdot)$  with a better cost  $J(u_{k+1})$ , depending on the previous steps and current admissible control  $u_k(\cdot)$ .

Steps one and three are identical to the policy evaluation and policy improvement in policy iteration. However, step two is still necessary to generate the successive approximation of optimal control while improving performance. Once the criterion for comparing policies is founded in step two, an improved policy with a better performance can be designed. The work in this paper is motivated by the Hamiltonian, defined in Sect. 7.2, and it develops the three steps of the Hamiltonian-driven framework found in Sect. 7.3.

### 7.1.3 Structure

The remainder of this article is organized as follows: Sect. 7.2 gives the problem formulation. The Hamiltonian-driven framework is discussed in Sect. 7.3. Implementation details are described in Sect. 7.4. Case studies for both linear and nonlinear dynamic systems are presented in Sect. 7.5. Section 7.6 gives the conclusions.

## 7.2 Problem Statement

This paper considers the stabilizable time-invariant input affine nonlinear dynamic system,

$$\dot{x}(t) = f(x) + g(x)u(t), \quad (7.1)$$

where  $x \in \Omega \subseteq \mathbb{R}^n$  is the state vector,  $u \in \mathbb{R}^m$  is the control policy,  $f(\cdot) \in \mathbb{R}^n$  and  $g(\cdot) \in \mathbb{R}^{n \times m}$  are locally Lipschitz functions with  $f(0) = 0$ .  $x(t_0) = x_0$  is the initial state and  $\Omega$  is a compact set in  $\mathbb{R}^n$ .

**Problem 7.1** *Find a control policy  $u^*(\cdot)$  that minimizes the cost*

$$J(u(\cdot); x_0) = \int_{t_0}^{\infty} L(x(t), u(t)) dt, \quad (7.2)$$

where the scalar utility function  $L(x(t), u(t))$  is differentiable and satisfies  $L(x, u) \geq 0$ , for  $\forall x, u$ . In this paper,  $L(x, u)$  is chosen as follows:

$$L(x, u) = Q(x) + \|u\|_R, \quad (7.3)$$

where  $\|u\|_R = u^T R u$ .  $Q(x)$  is a positive semi-definite function and  $R$  is a symmetric positive definite matrix.

The optimal control problem is to  $Q(x)$  and  $\|u\|_R$  represent the trade-off between driving the state to the origin and saving the energy applied to the system.  $J(u(\cdot); x_0)$  is understood as a cost functional of policy  $u(\cdot)$  starting from state  $x_0$ , which is an evaluation of the policy.

**Definition 7.1 (Admissible Policy)** A control policy  $\mu(\cdot)$  is said to be admissible with respect to (7.2) on  $\Omega \subseteq \mathbb{R}^n$ , denoted by  $\mu(\cdot) \in \pi(\Omega)$ , if

1.  $\mu(\cdot)$  is continuous on  $\Omega$ ,
2.  $\mu(0) = 0$ ,
3.  $u(x) := \mu(x)$  stabilizes (7.1) on  $\Omega$ ,
4. The cost  $J(\mu)$  (7.2) is finite  $\forall x \in \Omega$ .

As mentioned in [20], in ADP, stability is required at each iteration rather than simply approaching stability in the limit. Therefore, it is assumed that an admissible control exists.

Note that cost functional (7.2) is a functional of the policy  $u(\cdot)$  evaluated along the state trajectory  $x(t)$  which depends on the initial state  $x_0$ . However, there is no analytical solution of  $x(t)$  for general nonlinear systems (7.1). Therefore, an alternative to the cost functional defined on the whole state space, which is independent of the solution of the nonlinear dynamic system, is needed.  $V(x; u(\cdot))$ , which is referred to as a value function, represents the total cost starting from its argument  $x$  when  $u(\cdot)$  is applied to system (7.1) starting from  $x$ .

**Definition 7.2** The Hamiltonian functional of a given admissible policy  $u(\cdot)$  is defined as,

$$H(u(\cdot); x, V(x; u(\cdot))) = L(x, u) + \left\langle \frac{\partial V(x)}{\partial x}, \dot{x} \right\rangle, \quad (7.4)$$

where  $\langle \cdot, \cdot \rangle$  denotes the inner product between two vectors of the same dimension.

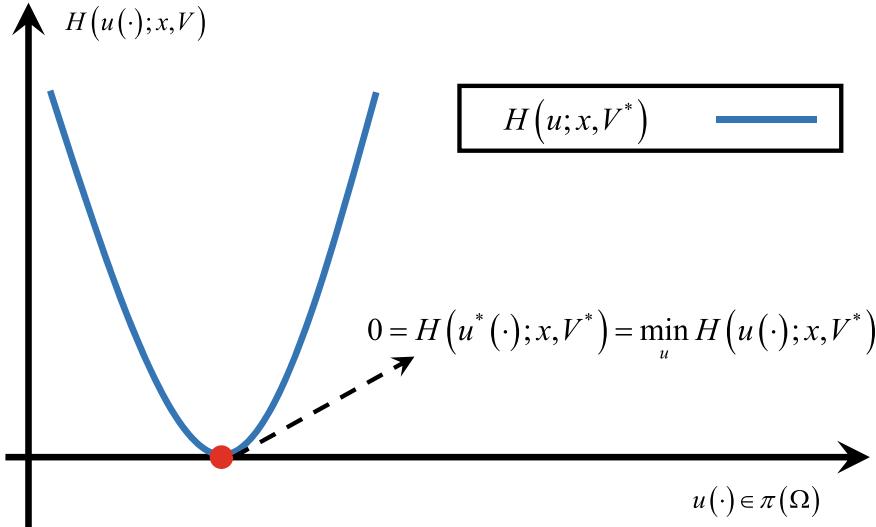
**Remark 7.1** It should be noted that Definition 7.2 for the Hamiltonian is slightly different from those found elsewhere, such as [4, 11–14]. Here,  $x$  stands for an arbitrary element in the state space rather than the solution of (7.1). Both the state  $x$  and the value function  $V(x; u(\cdot))$  can be viewed as parameters of the Hamiltonian. Also, note that the Hamiltonian functional can be parameterized by arbitrary value function, including the optimal value function  $V^*(\cdot)$  as a special case. The selections of proper value function for policy evaluation, policy comparison and policy improvement are investigated in Sect. 7.3.

The sufficient condition of optimality for Problem 7.1, named as the HJB equation, can be expressed in terms of the Hamiltonian functional parameterized by the optimal value function  $V^*(\cdot)$  as

$$0 = \min_{u(t)} H(u(t); x(t), V^*(x(t))), \quad (7.5)$$

with boundary condition  $V^*(x(\infty)) = 0$ . The relationship between the solution of the HJB equation and the optimal control problem is described as the following lemma.

**Lemma 7.1** Suppose [4] a unique and continuously differentiable scalar function  $V^*(x; u^*(\cdot))$  exists that satisfies the HJB equation described in (7.5). Suppose further that  $u^*(\cdot)$  attains the minimum of the right hand-side of (7.5), i.e.,  $u^*(x) = \arg \min_u H(u; x, V^*(x))$ . Let  $x^*(t)$  be the trajectory starting from the given initial state  $x_0$  when the control policy applied to system (7.1) is  $u^*(\cdot)$ . Then under the assumption that  $f(\cdot)$  and  $g(\cdot)$  are locally Lipschitz,  $L(\cdot, \cdot)$  is differentiable, and  $u^*(\cdot)$  is piece-wise continuous, the control policy  $u^*(\cdot)$  is the optimal solution of problem (7.2) subject to (7.1), and  $V^*(x_0; u^*(\cdot))$  is equal to the optimal cost of (7.2), i.e.,



**Fig. 7.1** The Hamiltonian-driven interpretation of the HJB equation (7.5). The figure is an illustration about the Hamiltonian functional of the admissible policy  $u$  parameterized by the optimal value function  $V^*(\cdot)$ . The  $u$ -axis stands for the group of admissible policy space  $\pi(\Omega)$ , and the  $H$ -axis represents the Hamiltonian functional of a given policy. The solid curve represents the Hamiltonian functional parameterized by the optimal value function  $V^*(\cdot)$ . One can observe that (1) the minimum of the Hamiltonian functional is 0; (2) the control policy which attains the minimum of the Hamiltonian is  $u^*(\cdot)$ , which is the solution to the HJB equation (7.5) and solves the continuous-time optimal control problem

$$V^*(x_0; u^*(\cdot)) = J^*(u^*(\cdot); x_0). \quad (7.6)$$

From Definition 7.2, the Hamiltonian functional is quadratic in its argument  $u(\cdot) \in \pi(\Omega)$ , which can be shown as a quadratic curve as in Fig. 7.1. The HJB equation (7.5) can be characterized by the Hamiltonian parameterized by the optimal value function  $V^*(\cdot)$  as  $0 = H(u^*(\cdot); x, V^*)$ . That is, the minimum of the specific Hamiltonian  $H(u; x, V^*)$  is zero. In addition, the policy that minimizes the Hamiltonian functional  $H(u(\cdot); x, V^*)$ , which is parameterized by the optimal value function  $V^*(\cdot)$ , is the optimal control policy  $u^*(\cdot)$ .

Assuming that the minimum on the right-hand side of (7.5) exists and is unique, then the optimal control is

$$u^*(x) = -\frac{1}{2}R^{-1}g^T(x) \frac{\partial V^*(x)}{\partial x}. \quad (7.7)$$

Inserting this result into (7.5), an equivalent formulation of the HJB equation can be found:

$$0 = Q(x) + \left\langle \frac{\partial V^*(x)}{\partial x}, f(x) \right\rangle - \frac{1}{4} \left[ \frac{\partial V^*(x)}{\partial x} \right]^T g(x) R^{-1} g^T(x) \frac{\partial V^*(x)}{\partial x}. \quad (7.8)$$

Note that the optimal control policy (7.7) depends on the solution of the HJB equation (7.8). However, solving the HJB equation (7.8) is challenging since it is a nonlinear PDE quadratic in the optimal value function gradient, and does not have an analytical solution for general nonlinear systems. In the next section, the successive approximation algorithm solving the HJB equation within the Hamiltonian-driven framework is established with convergence proofs and stability analysis.

### 7.3 Hamiltonian-Driven Framework

As in the introduction, the Problem 7.1 can be viewed as the infinite-dimensional optimization. According to [42], three fundamental problems for optimal control theory are presented as

- Policy Evaluation: To build a criterion that evaluates an arbitrary admissible control  $u(\cdot)$ , i.e., calculate the corresponding cost  $J(u(\cdot))$ .
- Policy Comparison: To establish a rule that compares two different admissible policies  $u(\cdot)$  and  $v(\cdot)$ .
- Policy Improvement: Based on the current admissible control  $u_k(\cdot)$ ,  $k \in \mathbb{Z}$ , design a successive control  $u_{k+1}(\cdot)$  with an improved cost  $J(u_{k+1}(\cdot))$ .

#### 7.3.1 Policy Evaluation

First, some terminologies should be clarified to avoid confusion.  $V^*(x_0; u^*(\cdot))$ , the solution to the HJB equation (7.5), is referred to as an optimal value function [25], and value function [14]. In this paper,  $V^*(x_0; u^*(\cdot))$  is referred to as an optimal value function, whereas the value function is defined as follows.

**Definition 7.3 (Value Function)** Suppose  $x(t)$  is the trajectory of system (7.1) starting from initial state  $x_0$  when policy  $u(\cdot)$  is applied. Suppose further that there exists a positive definite continuously differentiable function  $V(x; u(\cdot))$  that satisfies (7.9) below. Then  $V(x(t); u(\cdot))$  is called a value function of state  $x(t)$ .

$$\begin{cases} H(u(\cdot); x(t), V(x(t); u(\cdot))) = 0, \quad \forall x \in \mathbb{R}^n, \\ V(x(\infty); u(\cdot)) = 0. \end{cases} \quad (7.9)$$

**Assumption 7.1** The system dynamics (7.1) and the utility function  $L(x, u) = Q(x) + \|u\|_R$  are such that the solution to the generalized HJB equation (7.9) is continuously differentiable, i.e., belong to Sobolev space  $H^{1,2}(\Omega)$  (for more details of Sobolev space, see reference [1]), and the gradient of the solution to the generalized HJB equation is uniform on  $\Omega$ .

**Remark 7.2** The cost functional (7.2) depends on the solution to the system (7.1). However, the analytical solution to general nonlinear systems is difficult to obtain. In contrast, the value function obtained by solving (7.9) does not require the state trajectory of system (7.1). The Eq. (7.9) is referred to as the generalized HJB (GHJB) equation in [24]. Also, the GHJB equation (7.9) is linear in the value gradient, compared to the HJB equation (7.5) being quadratic in the value gradient. Therefore, theoretically speaking, it should be easier to solve than the nonlinear HJB equation (7.8).

**Remark 7.3** The solution to the generalized HJB equation (7.9) is positive definite, which is guaranteed by the stabilizable system dynamics and the zero state observable performance functional [1, 30].

From Definition 7.3, to obtain the value function, the Hamiltonian is required to be 0 for  $\forall x$ . The relationship between the value function  $V(x(t); u(\cdot))$  and  $J(u(\cdot); x(t))$  is given by the following theorem.

**Theorem 7.1** (Policy Evaluation Theorem) *Assume that the system trajectory  $x(t)$  is generated by applying  $u(\cdot)$  to system (7.1) from initial state  $x_0$ . Assume also that there exists a continuously differentiable and positive definite solution  $V(x(t); u(\cdot))$  to GHJB formulated in (7.9) with respect to system (7.1). Then,  $V(x(t); u(\cdot))$  and  $J(u(\cdot); x(t))$  are equal to each other, i.e.,*

$$V(x(t); u(\cdot)) = J(u(\cdot); x(t)), \quad \forall t \geq t_0. \quad (7.10)$$

**Proof** From (7.9),  $V(x; u(\cdot)) \geq 0$ ,  $\forall x \neq 0$  is continuously differentiable. Then,

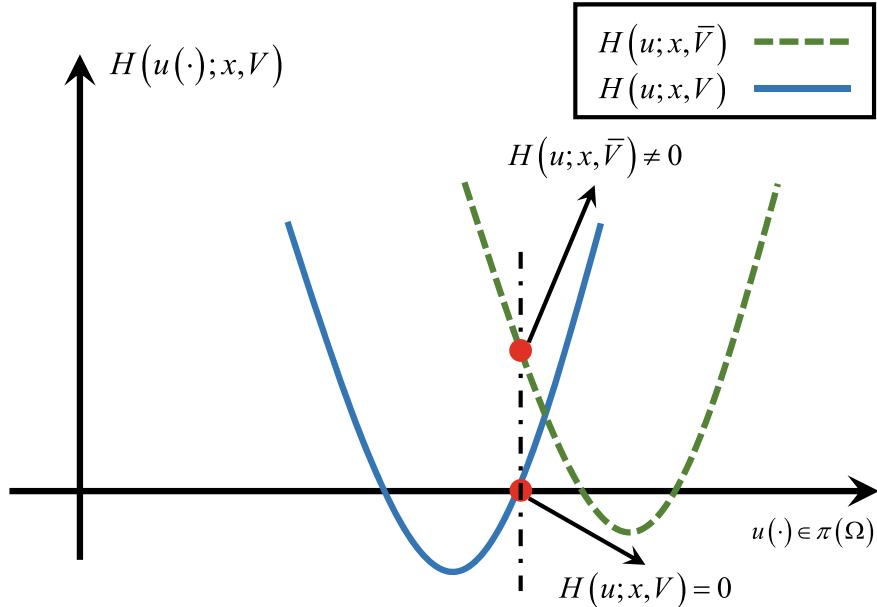
$$\begin{aligned} & \int_t^\infty \left\langle \frac{\partial V}{\partial x}, f(x) + g(x)u(x) \right\rangle d\tau \\ &= V(x(\infty)) - V(x(t)) \\ &= -V(x(t)), \quad \forall t \geq t_0. \end{aligned} \quad (7.11)$$

The second equality results from (7.9) by adding (7.2) to both sides of (7.11):

$$\begin{aligned} & J(u(\cdot); x) - V(x(t)) \\ &= \int_t^\infty \left[ \left\langle \frac{\partial V}{\partial x}, f(x) + g(x)u(x) \right\rangle + L(x, u) \right] d\tau \\ &= 0. \end{aligned} \quad (7.12)$$

The second equality results from (7.9). Hence, (7.10) holds.

Also, based on Lemma 7.1 and Theorem 7.1, it can be concluded that the optimal value function  $V^*(x)$  is a lower bound of the value function  $V(x)$  for admissible policy.



**Fig. 7.2** The Hamiltonian-driven interpretation of the policy evaluation, where the solid and dashed lines correspond to the Hamiltonian parameterized by the value functions  $V(x)$  and  $\bar{V}(x)$ , respectively. The Hamiltonian functionals parameterized by  $V$  and  $\bar{V}$  take different values at the given admissible policy  $u(\cdot)$ . Based on the GHJB equation (7.9), the value function  $V(x)$  is the one makes the Hamiltonian  $H(u; x, V)$  to be zero given the admissible  $u(\cdot)$ . Therefore,  $V(x)$  represents the performance evaluation for admissible policy  $u(\cdot)$

**Corollary 7.1** Denote the optimal control  $u^*(\cdot)$  and optimal value function  $V(\cdot)$  that solves Problem 7.1. Given the admissible policy  $u(\cdot)$  and the corresponding value function  $V(x)$  satisfying the GHJB. Then,

$$0 \prec V^*(x(t); u^*(\cdot)) \leq V(x(t); u(\cdot)). \quad (7.13)$$

Note that Theorem 7.1 bridges a gap between the value function and the cost functional, and it also provides a method to evaluate an arbitrary admissible policy via solving (7.9) for  $V(x; u(\cdot))$ . Therefore, decreasing the cost  $J(u(\cdot); x(t))$  is equivalent to decreasing the  $V(x(t); u(\cdot))$ . It can be seen that the conclusion in Lemma 7.1,  $V^*(x(t); u^*(\cdot)) = J^*(u^*(\cdot); x(t))$ , is a special case of Theorem 7.1. The graphical illustration of the policy evaluation using the Hamiltonian can be depicted in Fig. 7.2.

**Remark 7.4** The HJB equation only characterizes the relation between the optimal value function  $V^*(\cdot)$  and the optimal control  $u^*(\cdot)$ . In contrast, the relation between arbitrary admissible policy  $u(\cdot)$  and the corresponding value function  $V(\cdot)$  is captured by Theorem 7.1. From the discussion above, solving (7.9) for  $V(x; u(\cdot))$ , an equivalent calculation of the cost  $J(u(\cdot); x)$  of any admissible control is obtained.

### 7.3.2 Policy Comparison

From the discussion above, it becomes clear that the Hamiltonian functional is identical to 0 along the trajectory  $x(t)$ , for an arbitrary admissible control  $u(\cdot)$ . This holds even for optimal control  $u^*(\cdot)$ . Thus, the Hamiltonian seems to be a fixed structure for different admissible policies. When comparing two different admissible policies, further information about the Hamiltonian is required, which is the motivation for this subsection.

Assume that an admissible control  $u(x)$  is selected. The corresponding value function  $V(x; u(\cdot))$  is determined by solving (7.9). In order to better analyze the Hamiltonian, as inspired by [24], the property of the minimum of the Hamiltonian is investigated.

Consider system (7.1) with utility function (7.3) and cost functional (7.2), via completion of squares, the Hamiltonian (7.4) is equivalent to

$$\begin{aligned} H(u; x, V) &= Q(x) + \left[ \frac{\partial V(x)}{\partial x} \right]^T f(x) - \frac{1}{4} \left[ \frac{\partial V}{\partial x} \right]^T g(x) R^{-1} g^T(x) \frac{\partial V}{\partial x} \\ &\quad + \left[ u + \frac{1}{2} R^{-1} g^T \frac{\partial V}{\partial x} \right]^T R \left[ u + \frac{1}{2} R^{-1} g^T \frac{\partial V}{\partial x} \right]. \end{aligned} \quad (7.14)$$

Therefore, the minimum of the Hamiltonian parameterized by the value function  $V(\cdot)$  can be expressed as

$$\begin{aligned} h(V) &= \min_u H(u; x, V) \\ &= H(\bar{u}; x, V) \\ &= \left[ \frac{\partial V(x)}{\partial x} \right]^T f(x) - \frac{1}{4} \left[ \frac{\partial V}{\partial x} \right]^T g(x) R^{-1} g^T(x) \frac{\partial V}{\partial x} + Q(x), \end{aligned} \quad (7.15)$$

where

$$\bar{u}(x; V) = \arg \min_u H(u; x, V) = -\frac{1}{2} R^{-1} g^T \frac{\partial V(x)}{\partial x}. \quad (7.16)$$

**Theorem 7.2** (Policy Comparison Theorem) *Let  $u_i(\cdot)$ ,  $i = 1, 2$  be two different admissible policies, with the value function  $V_i(\cdot)$ ,  $i = 1, 2$  determined by solving (7.9), respectively. Denote the minimum of the Hamiltonian parameterized by  $V_1$  and  $V_2$  as*

$$h_i = h(V_i) = \min_{u(\cdot)} H(u(\cdot); x, V_i(x; u(\cdot))), i = 1, 2.$$

Let

$$d_i = \|u_i - \bar{u}_i\|_R, i = 1, 2$$

be the distance between the prescribed policy,  $u_i(\cdot)$ ,  $i = 1, 2$ , and the policy which attains the minimum of the Hamiltonian,

$$\bar{u}_i = \bar{u}(V_i) = \arg \min_{u(\cdot)} H(u(\cdot); x, V_i(\cdot; u)), i = 1, 2.$$

Then, the following results hold:

1.  $h_i \leq 0$ ,  $i = 1, 2$ ;
2.  $h_1 \leq h_2 \Rightarrow V_1(x; u_1(\cdot)) \geq V_2(x; u_2(\cdot))$ ,  $\forall x \in R^n$ ;
3.  $d_1 \geq d_2 \Rightarrow V_1(x; u_1(\cdot)) \geq V_2(x; u_2(\cdot))$ ,  $\forall x \in R^n$ .

**Proof** (1) The first proposition is natural since

$$h_i = \min_u H(u; x, V_i) \leq H(u_i; x, V_i) = 0, i = 1, 2. \quad (7.17)$$

(2) Let  $V_2(x) = V_1(x) + \Lambda(x)$ , from (7.14) and (7.15), resulting in:

$$\begin{aligned} h_2 &= \min_u H(u; x, V_2) \\ &= Q(x) + \left[ \frac{\partial V_2}{\partial x} \right]^T f(x) - \frac{1}{4} \left[ \frac{\partial V_2}{\partial x} \right]^T g(x) R^{-1} g^T(x) \frac{\partial V_2}{\partial x} \\ &= Q(x) + \left[ \frac{\partial V_1}{\partial x} \right]^T f(x) + \left[ \frac{\partial \Lambda}{\partial x} \right]^T f(x) - \frac{1}{4} \left[ \frac{\partial V_1}{\partial x} \right]^T g(x) R^{-1} g^T(x) \frac{\partial V_1}{\partial x} \\ &\quad - \frac{1}{4} \left[ \frac{\partial \Lambda}{\partial x} \right]^T g(x) R^{-1} g^T(x) \frac{\partial \Lambda}{\partial x} - \frac{1}{2} \left[ \frac{\partial \Lambda}{\partial x} \right]^T g(x) R^{-1} g^T(x) \frac{\partial V_1}{\partial x} \\ &= h_1 - \frac{1}{4} \left[ \frac{\partial \Lambda}{\partial x} \right]^T g(x) R^{-1} g^T(x) \frac{\partial \Lambda}{\partial x} \\ &\quad + \left\langle \frac{\partial \Lambda}{\partial x}, f(x) - \frac{1}{2} g(x) R^{-1} g^T(x) \frac{\partial V_1}{\partial x} \right\rangle. \end{aligned} \quad (7.18)$$

Moving the items  $h_1$  and  $-\frac{1}{4} \left[ \frac{\partial \Lambda}{\partial x} \right]^T g(x) R^{-1} g^T(x) \frac{\partial \Lambda}{\partial x}$  to the left hand side of (7.18):

$$\begin{aligned} h_2 - h_1 + \frac{1}{4} \left[ \frac{\partial \Lambda}{\partial x} \right]^T g(x) R^{-1} g^T(x) \frac{\partial \Lambda}{\partial x} \\ = \left\langle \frac{\partial \Lambda}{\partial x}, f(x) - \frac{1}{2} g(x) R^{-1} g^T(x) \frac{\partial V_1}{\partial x} \right\rangle \\ = \frac{d\Lambda}{dt}. \end{aligned} \quad (7.19)$$

Based on the assumption that  $h_1 \leq h_2$  and matrix  $R$  is a positive definite, then

$$\frac{d\Lambda}{dt} \geq 0. \quad (7.20)$$

Provided that  $V_i(\cdot)$  satisfies condition (7.9), thus

$$\Lambda(x(\infty)) = V_2(x(\infty)) - V_1(x(\infty)) = 0. \quad (7.21)$$

(7.20) combined with (7.21) implies that

$$\Lambda(x) \leq 0, \forall x \in R^n. \quad (7.22)$$

Therefore, proposition two is proved.

(3) From the assumption that  $d_1 \geq d_2$ , it is revealed that

$$\left\| u_1 + \frac{1}{2} R^{-1} g^T(x) \frac{\partial V_1}{\partial x} \right\|_R - \left\| u_2 + \frac{1}{2} R^{-1} g^T(x) \frac{\partial V_2}{\partial x} \right\|_R \geq 0. \quad (7.23)$$

First, by extending the items in (7.23), then, by adding and subtracting the appropriate terms to (7.23), one can obtain:

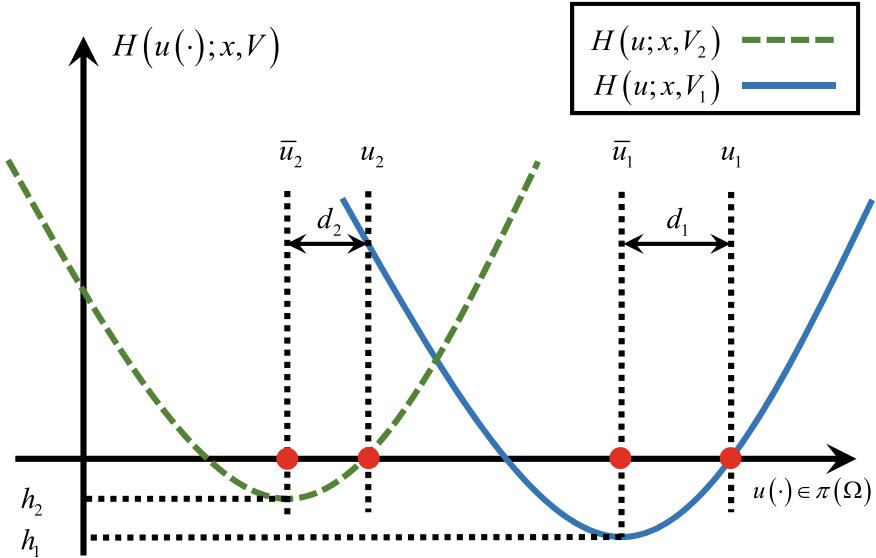
$$\begin{aligned} & -\|u_2\|_R - x^T Q x - \left\langle \frac{\partial V_2}{\partial x}, f(x) + g(x) u_2 \right\rangle + x^T Q x \\ & + \left\langle \frac{\partial V_2}{\partial x}, f(x) \right\rangle - \frac{1}{4} \left( \frac{\partial V_2}{\partial x} \right)^T g(x) R^{-1} g^T(x) \frac{\partial V_2}{\partial x} \\ & \geq -\|u_1\|_R - x^T Q x - \left\langle \frac{\partial V_1}{\partial x}, f(x) + g(x) u_1 \right\rangle + x^T Q x \\ & + \left\langle \frac{\partial V_1}{\partial x}, f(x) \right\rangle - \frac{1}{4} \left( \frac{\partial V_1}{\partial x} \right)^T g(x) R^{-1} g^T(x) \frac{\partial V_1}{\partial x}. \end{aligned} \quad (7.24)$$

Based on (7.15) and (7.24) by the completion of squares in (7.14), the following can be determined:

$$\begin{aligned} h_2 &= \left\langle \frac{\partial V_2}{\partial x}, f(x) \right\rangle - \frac{1}{4} \left( \frac{\partial V_2}{\partial x} \right)^T g(x) R^{-1} g^T(x) \frac{\partial V_2}{\partial x} + x^T Q x \\ &\geq \left\langle \frac{\partial V_1}{\partial x}, f(x) \right\rangle - \frac{1}{4} \left( \frac{\partial V_1}{\partial x} \right)^T g(x) R^{-1} g^T(x) \frac{\partial V_1}{\partial x} + x^T Q x \\ &= h_1. \end{aligned} \quad (7.25)$$

From the results in proposition two, proposition three holds.

**Remark 7.5** Though different policies correspond to different value functions and Hamiltonians, based on (7.9), the Hamiltonians of different admissible policies are always identical to 0, at the intercept of the Hamiltonian curves with the u-axis, which is the invariant property of an admissible policy. However, more knowledge about



**Fig. 7.3** A comparison between two different control policies  $u_1(\cdot)$  and  $u_2(\cdot)$ . Both the state trajectory  $x$  and the value function  $V_i(x; u_i(\cdot))$  are parameters of the Hamiltonian, rather than arguments. The Hamiltonian functionals  $H(u_1(\cdot); x, V_1(x; u_1(\cdot)))$  and  $H(u_2(\cdot); x, V_2(x; u_2(\cdot)))$  are different in both arguments and parameters, as illustrated. For  $u_1(\cdot)$ , the Hamiltonian  $H(u_1; x, V_1)$  is shown by the blue solid curve. For  $u_2(\cdot)$ , the Hamiltonian  $H(u_2; x, V_2)$  is shown by the red-dashed curve. There is a distance between the prescribed control policy,  $u_i(\cdot)$ ,  $i = 1, 2$ , and the policy which attains the minimum of the corresponding Hamiltonian,  $\bar{u}_i = \arg \min_u H(u; x, V_i)$ ,  $i = 1, 2$ . Theorem 7.2 shows that  $d_i$  and  $h_i$  indicate the relationship between  $V_1(x)$  and  $V_2(x)$ : (1)  $d_2 < d_1 \Rightarrow h_2 > h_1$ ; (2)  $h_2 > h_1 \Rightarrow V^*(x) \leq V_2(x) \prec V_1(x)$

the Hamiltonian shows  $d = \|u - \bar{u}\|_R$  and  $h = \min_{u(\cdot)} H(u(\cdot); x, V(\cdot; u))$ , which is important for comparing between different admissible policies. Figure 7.3 illustrates the ideas in Theorem 7.2.

With the Hamiltonian, Theorem 7.2 provides a method for comparing two different admissible policies. Additionally, Theorem 7.2 offers a possible descent direction toward an optimal policy. The descent direction represents the class of variations in the policy that will lead to an improvement in the value function. This is important for the design of successive approximations to an optimal policy, which will be discussed next.

### 7.3.3 Policy Improvement

In Theorem 7.2,  $u_2(\cdot)$  is better than  $u_1(\cdot)$  w.r.t. the cost functional (7.2) or the value function (7.9). However, there is still no explicit expression for the improved policy

$u_2(\cdot)$ . In this subsection, a successive approach, i.e., relating the improved policy  $u_2(\cdot)$  to a prescribed policy  $u_1(\cdot)$  is put forth.

The following is required for the successive improvement of the admissible policy.

**Definition 7.4** (*The Uniform Sequence*) A [8] sequence  $\{h_i(x) | i = 1, 2, \dots\}$  defined on  $\Omega \in \mathbb{R}^n$  is said to be uniform at  $x_0 \in \Omega$  if for every  $\varepsilon > 0$ , there exists a  $\delta > 0$  and  $N > 0$ , such that if  $|x_0 - x| < \delta$ ,  $m > N$  and  $n > N$ , then  $|h_m(x) - h_m(x_0) - h_n(x) + h_n(x_0)| < \varepsilon$ .

**Lemma 7.2** (Convergence of Function Gradient) *Let [8]  $h_i(x)$  be the real-valued function on  $\Omega$ ,  $i = 1, 2, \dots$ . Suppose that the function sequence  $\{h_i(x) | i = 1, 2, \dots\}$  converges to  $h(x)$  on  $\Omega$ , then  $\nabla h(x)$  exists on  $\Omega$  and the function gradient sequence  $\{\nabla h_i(x) | i = 1, 2, \dots\}$  converges uniformly to  $\nabla h$  on  $\Omega$  if and only if  $\nabla h_i(x)$  is uniform on  $\Omega$ ,  $i = 1, 2, \dots$*

**Theorem 7.3** (Policy Improvement Theorem) *Suppose the policy sequence  $\{u_i(\cdot) | i = 1, 2, \dots\}$  begins from an admissible policy  $u_1(\cdot)$ . Every element in the value function sequence  $\{V_i(x; u_i(\cdot)) | i = 1, 2, \dots\}$  satisfies (7.9). The policy sequence is generated by*

$$u_{i+1}(x) = -\frac{1}{2}R^{-1}g^T(x) \frac{\partial V_i(x; u_i(\cdot))}{\partial x}. \quad (7.26)$$

*Then the following statements are true:*

1. *The value function sequence  $\{V_i(x; u_i(\cdot))\}$  is a non-increasing sequence, i.e.,*

$$V_i(x; u_i(\cdot)) \geq V_{i+1}(x; u_{i+1}(\cdot)), \quad i = 1, 2, \dots \quad (7.27)$$

2. *Both  $\{V_i(x; u_i(\cdot))\}$  and  $\{u_i(\cdot)\}$ , generated via the approach mentioned above, converge into  $V^*(x; u^*(\cdot))$  and  $u^*(\cdot)$ , which is the solution to HJB equation (7.5), i.e.,*

$$\lim_{i \rightarrow \infty} V_i(x) = V^*(x), \quad \lim_{i \rightarrow \infty} u_i(x) = u^*(x). \quad (7.28)$$

**Proof** (1) Based on the assumption that  $V_i(x; u_i(\cdot))$  satisfies (7.9), then from (7.26), the Hamiltonian of policy  $u_{i+1}(\cdot)$  is:

$$\begin{aligned} & H(u_{i+1}(\cdot); x, V_{i+1}(x; u_{i+1}(\cdot))) \\ &= \left\langle \frac{\partial V_{i+1}(x)}{\partial x}, f(x) + g(x) \left[ -\frac{1}{2}R^{-1}g^T(x) \frac{\partial V_i(x)}{\partial x} \right] \right\rangle \\ &+ \left[ -\frac{1}{2}R^{-1}g^T(x) \frac{\partial V_i(x)}{\partial x} \right]^T R \left[ -\frac{1}{2}R^{-1}g^T(x) \frac{\partial V_i(x)}{\partial x} \right] + Q(x) \end{aligned}$$

$$\begin{aligned}
&= \left\langle \frac{\partial V_{i+1}(x)}{\partial x}, f(x) \right\rangle + Q(x) - \frac{1}{2} \left[ \frac{\partial V_{i+1}(x)}{\partial x} \right]^T g(x) R^{-1} g^T(x) \frac{\partial V_i(x)}{\partial x} \\
&\quad + \frac{1}{4} \left[ \frac{\partial V_i(x)}{\partial x} \right]^T g(x) R^{-1} g^T(x) \frac{\partial V_i(x)}{\partial x} \\
&= 0.
\end{aligned} \tag{7.29}$$

Let  $V_{i+1}(x) = V_i(x) + \Delta_i(x)$ . Inserting this into (7.29) becomes:

$$\begin{aligned}
&H(u_{i+1}(\cdot); x, V_{i+1}(x; u_{i+1}(\cdot))) \\
&= \left\langle \frac{\partial V_i(x)}{\partial x}, f(x) \right\rangle + \frac{1}{4} \left[ \frac{\partial V_i(x)}{\partial x} \right]^T g(x) R^{-1} g^T(x) \frac{\partial V_i(x)}{\partial x} \\
&\quad + \left\langle \frac{\partial \Delta_i(x)}{\partial x}, f(x) \right\rangle + Q(x) - \frac{1}{2} \left[ \frac{\partial \Delta_i(x)}{\partial x} \right]^T g(x) R^{-1} g^T(x) \frac{\partial V_i(x)}{\partial x} \\
&\quad - \frac{1}{2} \left[ \frac{\partial V_i(x)}{\partial x} \right]^T g(x) R^{-1} g^T(x) \frac{\partial V_i(x)}{\partial x} \\
&= 0.
\end{aligned} \tag{7.30}$$

By rearranging, (7.30) is equivalent to

$$\begin{aligned}
&- \frac{1}{4} \left[ \frac{\partial V_i(x)}{\partial x} \right]^T g(x) R^{-1} g^T(x) \frac{\partial V_i(x)}{\partial x} + \left\langle \frac{\partial V_i(x)}{\partial x}, f(x) \right\rangle + Q(x) \\
&= \frac{1}{2} \left[ \frac{\partial \Delta_i(x)}{\partial x} \right]^T g(x) R^{-1} g^T(x) \frac{\partial V_i(x)}{\partial x} - \left\langle \frac{\partial \Delta_i(x)}{\partial x}, f(x) \right\rangle.
\end{aligned} \tag{7.31}$$

Note that the left-hand side of (7.31) is

$$h_i = \min_{u(\cdot)} H(u(\cdot); x, V_i(x; u(\cdot))), \tag{7.32}$$

and the right hand side is

$$-\left\langle \frac{\partial \Delta_i(x)}{\partial x}, f(x) + g(x) u_{i+1}(x) \right\rangle = -\frac{d \Delta_i(x(t))}{dt}. \tag{7.33}$$

Combining (7.32) and (7.33) lead to:

$$h_i = \min_{u(\cdot)} H(u(\cdot); x, V_i(x; u(\cdot))) = -\frac{d \Delta_i(x(t))}{dt}. \tag{7.34}$$

Proposition (1) in Theorem 7.2,  $h_i \leq 0$ , reveals the following result:

$$\frac{d \Delta_i(x(t))}{dt} \geq 0, \forall x. \tag{7.35}$$

Since the value function sequence  $\{V_i(x; u_i(\cdot))\}$  is generated by solving (7.9), both  $V_i(x)$  and  $V_{i+1}(x)$  satisfy the boundary condition:

$$V_i(x(\infty); u_i(\cdot)) = V_{i+1}(x(\infty); u_{i+1}(\cdot)) = 0. \quad (7.36)$$

Therefore

$$\Delta_i(x(\infty)) = V_{i+1}(x(\infty)) - V_i(x(\infty)) = 0. \quad (7.37)$$

Based on (7.35) and (7.37), it becomes:

$$\Delta_i(x(t)) = V_{i+1}(x(t)) - V_i(x(t)) \leq 0, \quad \forall t \geq t_0. \quad (7.38)$$

which is equivalent to:

$$V_{i+1}(x) \leq V_i(x), \quad \forall x. \quad (7.39)$$

(2) The proof of the second proposition in Theorem 7.3 is given below.

From proposition 1, it is known that the value function sequence  $\{V_i(x; u_i(\cdot))\}$  is non-increasing. Since the value function is positive definite, the value function sequence is lower bounded by  $V(x) = 0, \forall x$ . Thus, the limit of the lower bounded and non-increasing sequence  $\{V_i(x; u_i(\cdot))\}$  exists, as denoted by

$$V_e(x) = \lim_{i \rightarrow \infty} V_i(x). \quad (7.40)$$

Based on Assumption 7.1 and Lemma 7.2, the value gradient sequence  $\left\{\frac{\partial V_i(x)}{\partial x}\right\}$  for  $i = 1, 2, \dots$  converges uniformly to  $\frac{\partial V_e(x)}{\partial x}$  on  $\Omega$ :

$$\frac{\partial V_e(x)}{\partial x} = \lim_{i \rightarrow \infty} \frac{\partial V_i(x)}{\partial x}. \quad (7.41)$$

Consider the variation between these two successive policies:

$$\begin{aligned} u_i - u_{i+1} &= u_i + \frac{1}{2} R^{-1} g^T(x) \frac{\partial V_i(x)}{\partial x} \\ &= \frac{1}{2} R^{-1} g^T(x) \left[ \frac{\partial V_i(x)}{\partial x} - \frac{\partial V_{i-1}(x)}{\partial x} \right]. \end{aligned} \quad (7.42)$$

Based on (7.40), (7.41) and (7.42), for all  $x$ ,

$$\lim_{i \rightarrow \infty} \frac{1}{2} R^{-1} g^T(x) \left[ \frac{\partial V_i(x)}{\partial x} - \frac{\partial V_{i-1}(x)}{\partial x} \right] = 0, \quad (7.43)$$

or equivalently,

$$\lim_{i \rightarrow \infty} \left[ u_i(x) + \frac{1}{2} R^{-1} g^T(x) \frac{\partial V_i(x)}{\partial x} \right] = 0, \quad (7.44)$$

i.e.,

$$\begin{aligned} u_e &= \lim_{i \rightarrow \infty} u_i \\ &= \lim_{i \rightarrow \infty} -\frac{1}{2} R^{-1} g^T(x) \frac{\partial V_i(x)}{\partial x} \\ &= -\frac{1}{2} R^{-1} g^T(x) \frac{\partial V_e}{\partial x}. \end{aligned} \quad (7.45)$$

Note that each pair  $\{u_i(\cdot), V_i(\cdot)\}$  satisfies the definition in (7.9). Hence,  $\{u_e(\cdot), V_e(\cdot)\}$  also satisfies (7.9). Therefore,

$$H(u_e; x, V_e) = 0, \quad (7.46)$$

From (7.14), it can be inferred that the Hamiltonian  $H(u; x, V_e)$  is quadratic in policy  $u$ . Based on (7.16) and (7.45), it is known that  $u_e$  minimizes the Hamiltonian  $H(u; x, V_e)$ , i.e.,

$$0 = H(u_e; x, V_e) = \min_u H(u; x, V_e). \quad (7.47)$$

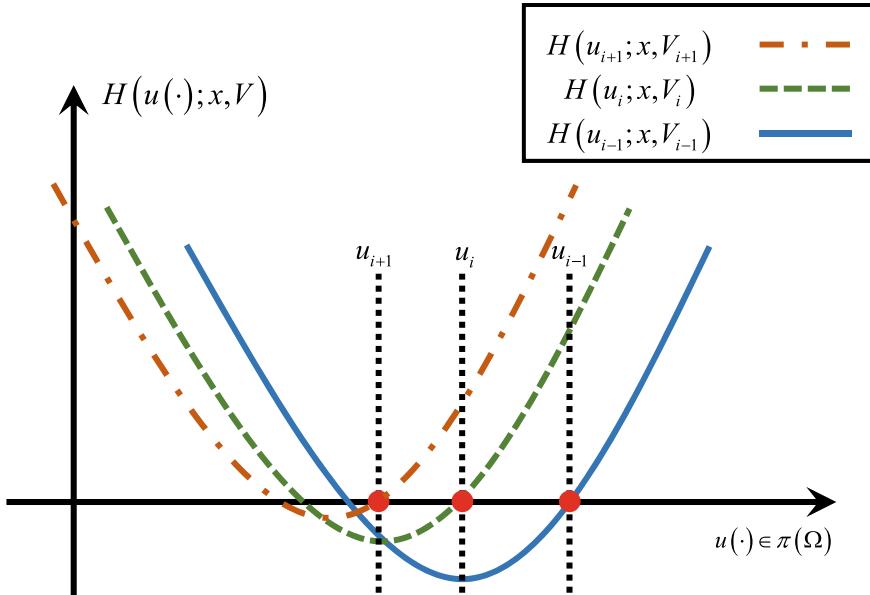
Therefore, the pair  $\{u_e(\cdot), V_e(\cdot)\}$  satisfies the HJB equation (7.5). Based on the assumption that the HJB equation (7.10) or (7.13) has a unique solution, then for all  $x \in \mathbb{R}^n$ :

$$\begin{cases} \lim_{i \rightarrow \infty} V_i(x) = V_e(x) = V^*(x), \\ \lim_{i \rightarrow \infty} u_i(x) = u_e(x) = u^*(x). \end{cases} \quad (7.48)$$

This completes the proof.

Based on Theorem 7.3, the control sequence  $\{u_i(\cdot)\}$  can be generated recursively, resulting in the value function sequence  $\{V_i(x; u_i(\cdot))\}$  converging into an optimal value function  $V^*(x; u^*(\cdot))$ . Another view of Theorem 7.3 could be upper bound minimization. Corollary 7.1 shows that the value function associated with an arbitrary admissible policy is an upper bound of the optimal value function for an optimal policy. Further details will be discussed with graphical illustration given in Fig. 7.4.

**Remark 7.6** Theorem 7.3 is the answer to the final step in the Hamiltonian framework. It gives a recursive scheme to the designed a control policy sequence  $\{u_i\}_{i=1}^{\infty}$  that will converge to an optimal control  $u^*(x)$ . It is a special case of Theorem 7.2. Suppose that we choose  $u_{i+1} = \arg \min_u H_i(u)$ , which leads to  $d_{i+1} \prec d_i$ , as



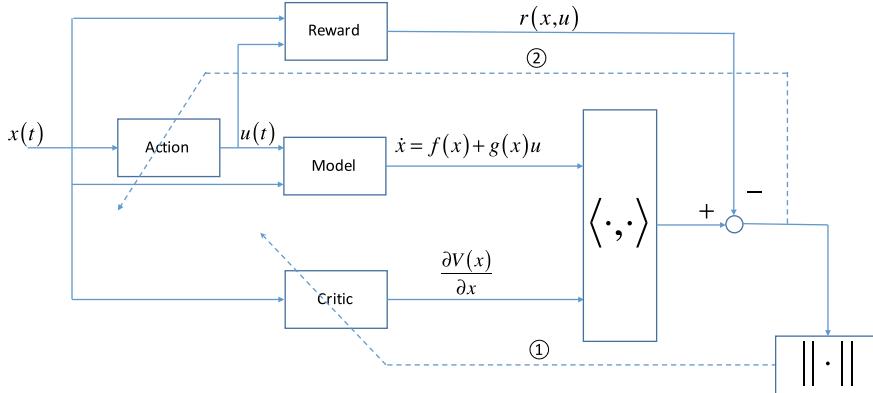
**Fig. 7.4** A recursive approximation design: This case can be seen as a special case of policy comparison. When beginning from an admissible control  $u_i(\cdot)$ , the successive control  $u_{i+1}(\cdot)$  is defined as  $\arg \min_u H(u; x, V_i)$ , which is exactly the minimum of the previous Hamiltonian. Theorem 7.3 reveals the relationship between  $V_i(x)$  and  $V_{i+1}(x)$ : (1)  $d_{i+1} < d_i \Rightarrow h_{i+1} > h_i$ ; (2)  $h_{i+1} > h_i \Rightarrow V_{i+1}(x) < V_i(x)$

shown in Fig. 7.4. Based on Theorem 7.2,  $d_{i+1} < d_i \Rightarrow 0 \geq h_{i+1} > h_i$ , hence,  $V^* \leq V_{i+1}(x) < V_i(x)$ . Therefore, we will have a policy sequence  $\{u_i\}_{i=1}^\infty$  and a value function sequence  $\{V_i(\cdot)\}$  satisfying (7.48).

## 7.4 Discussions on the Hamiltonian-Driven ADP

### 7.4.1 Implementation with Critic-Only Structure

In this section, the implementation of the Hamiltonian driven iterative ADP algorithm is given via a NN. The recursive value function sequence generated by Theorem 7.3 has been proven to converge into the solution of the HJB equation that originates from the nonlinear continuous-time system optimal control. It should be noted that  $H(u_i(x); x, V_i(x)) = 0$  should be satisfied along the trajectory at each epoch, such that  $V_i(x)$  is the value function for the admissible control  $u_i(x)$ . However, solving  $H(u_i(x); x, V_i(x)) = 0$  for  $V_i(x)$  is difficult for general nonlinear systems. For implementation purposes,  $H(u_i(x); x, V_i(x)) = 0$  is approximated by  $\min_u \|H(u(\cdot); x, V_i(x; u_i(\cdot)))\|$ .



**Fig. 7.5** A schematic of the Hamiltonian-driven iterative ADP framework. (1) The tuning of the critic network tries to minimize the norm of the Hamiltonian. Once the norm of the Hamiltonian is close to zero, the Hamiltonian is also close to zero. Then the critic outputs the approximation to the value function; (2) Once the critic network is well trained, the action network training starts. The tuning of the action network aims to decrease the Hamiltonian functional in order to decrease the value function

Inspired by [23], this paper puts forward the Hamiltonian-driven framework for ADP. The diagram of the ADP algorithm is given in Fig. 7.5. The critic network outputs the value function, and the action network outputs the control policy in the following alternative ways: (1) The training of the critic network tries to minimize the norm of the Hamiltonian. Once the Hamiltonian is close to zero, the critic outputs the approximated value function. (2) After the critic network is well trained, the action network training starts. The training of the action network aims to minimize the Hamiltonian in order to improve the value function.

This paper utilizes a three-layer feed-forward NN as the function approximation scheme. The output of the action network can be formulated as

$$\hat{u}_i(x_k, w_u^i) = (w_u^{i,2}) \cdot \sigma((w_u^{i,1}) \cdot x_k + b_u^{i,1}) + b_u^{i,2}, \quad (7.49)$$

where  $w_u^{i,1}$  and  $b_u^{i,1}$  are the weight matrix and bias vector between the input layer and the hidden layer, and  $w_u^{i,2}$  and  $b_u^{i,2}$  are the weight matrix and bias vector between the hidden layer and the output layer.  $\sigma(\cdot)$  is the activation function in the hidden layer. During the iteration, the target of the control input,  $u_i(x_k)$ , is formulated in (7.26). The weights of the action network are updated to minimize the following objective function:

$$E_u^i(x_k) = \hat{H}_i(\hat{u}_i(x_k, w_u^i); x_k, \hat{V}^i). \quad (7.50)$$

The critic network is constructed by a multilayer feed-forward NN. The output of the critic network is

$$\hat{V}_i(x_k, w_V^i) = \left( w_V^{i,2} \right) \cdot \sigma \left( \left( w_V^{i,1} \right) \cdot x_k + b_u^{i,1} \right) + b_u^{i,2}, \quad (7.51)$$

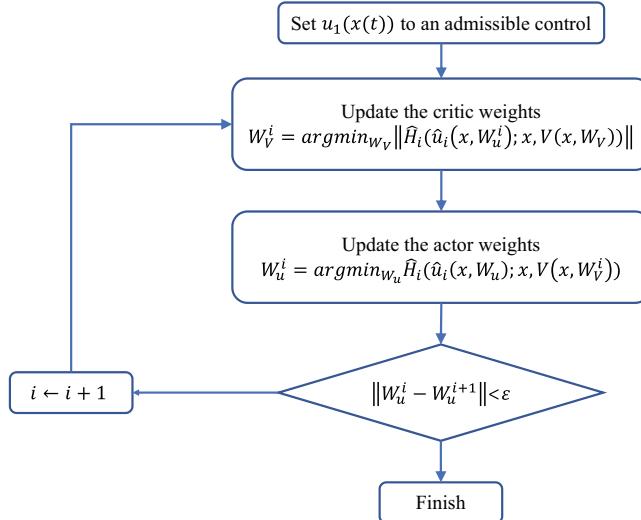
where  $w_V^{i,1}$  and  $b_u^{i,1}$  are the weight matrix and the bias vector between the input layer and hidden layer, and  $w_V^{i,2}$  and  $b_u^{i,2}$  are the weight matrix and bias vector between the input layer and hidden layer. The error function for training the critic network is

$$e_V^i(x_k) = 0 - \hat{H}_i(\hat{u}_i(x_k, w_u^i); x_k, \hat{V}^i), \quad (7.52)$$

and the objective function to be minimized is defined as

$$\begin{aligned} E_V^i(x_k) &= \langle e_V^i(x_k), e_V^i(x_k) \rangle \\ &= \left\| \hat{H}_i(\hat{u}_i(x_k, w_u^i); x_k, \hat{V}^i) \right\|. \end{aligned} \quad (7.53)$$

With these objective functions, many methods like backpropagation and the Levenberg–Marquardt algorithm can be used to tune the weights of the NN. A summary of this algorithm is given in Fig. 7.6.



**Fig. 7.6** The flowchart of a Hamiltonian-driven iterative ADP algorithm. (1) The tuning of the critic network tries to minimize the norm of the Hamiltonian. Based on 7.9, once the Hamiltonian is close to zero, the critic outputs the correct value function. (2) Once the critic is well trained, the action network training starts. The tuning of the action network aims to minimize the Hamiltonian functional, which is equivalent to improving the critic

### 7.4.2 Connection to Temporal Difference Learning

#### 7.4.2.1 Continuous-Time Integral Temporal Difference

Based on the Bellman residual for discrete-time systems [2, 4, 12] (also named as the temporal difference [23, 26, 27, 34]), For continuous-time systems, the integral temporal difference is defined as [31]

$$e(t : t + T) = \int_t^{t+T} L(x(\tau), u(\tau)) \\ + \hat{V}(x(t+T); u(\cdot)) - \hat{V}(x(t); u(\cdot)), \quad (7.54)$$

which is required to be zero when calculating the approximated value function  $\hat{V}(x)$  with respect to the admissible control  $u(\cdot)$ . Essentially, the integral temporal difference in (7.54) can be viewed as the integral of the Hamiltonian on the interval  $[t, t + T]$ .

#### 7.4.2.2 Continuous-Time Least Squares Temporal Difference

In [1], the value function approximation is transformed into the least square solution of

$$\left\langle \frac{d\varepsilon(x)}{dw}, \varepsilon(x) \right\rangle = 0, \quad (7.55)$$

where  $\varepsilon(x) = H(x; u, \hat{V}(x))$ , and the value function  $V(x)$  is approximated by  $\hat{V}(x) = w^T \sigma(x)$ . It should be noted that the least squares solution of (7.55),  $w^*$ , is equivalent to the solution of the minimization problem of

$$\min_w \frac{1}{2} \|H(x; u, \hat{V}(x; w))\|, \quad (7.56)$$

which seeks to drive the Hamiltonian  $H(x; u, \hat{V}(x; w^*))$  toward 0 as close as possible. That is, the method in [1] is equivalent to approximate the solution of (7.9). Therefore, the Hamiltonian in (7.4) can be viewed as the general temporal difference for the continuous-time systems.

**Remark 7.7** From Definition 7.3, the value function requires the Hamiltonian in (7.4) to be zero along the state trajectory. Therefore, the Hamiltonian serves as the functional temporal difference of the policy learning for continuous-time systems. It can be shown that the ADP method in [1, 31] are equivalent to the Hamiltonian minimization.

### 7.4.3 Connection to Value Gradient Learning

The value function approximation algorithm can be categorized into two types: value function-based learning and value gradient-based learning. Typical value function-based learning is the TD( $\lambda$ )-learning algorithms family [27], which aims at approximating the optimal value function using online implementation. On the other hand, the value gradient-based learning tries to approximate the value function gradient instead, including the VGL( $\lambda$ )-learning algorithms family [5]. For discrete-time dynamical systems, both the value function-based learning and value gradient-based learning include two critics to generate the temporal difference. However, as shown in Fig. 7.5, only one critic is required for the critic to output the approximate value function gradient.

## 7.5 Simulation Study

In this section, two examples are provided to illustrate the effectiveness of the above theoretical results.

### Example 7.1 Nonlinear Regulator Problem

Consider the following nonlinear dynamical system [6]:

$$\begin{cases} \dot{x}_1 = -x_1 + x_1 x_2^2 \\ \dot{x}_2 = -x_2 + x_1 u \end{cases}, \quad (7.57)$$

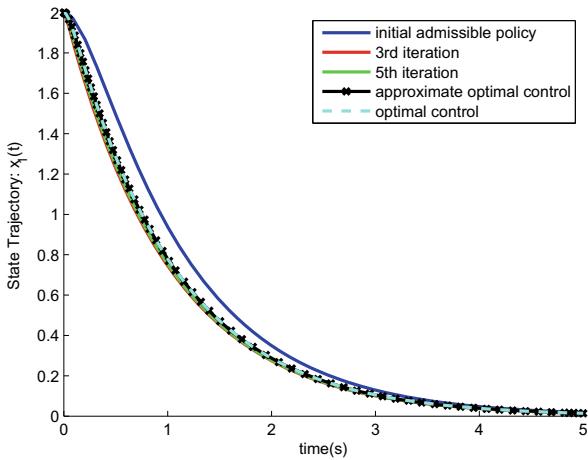
where  $x(t) = [x_1 \ x_2]^T \in \mathbb{R}^2$  is the state and  $u(t) \in \mathbb{R}$  is the control input. According to the inverse optimal nonlinear regulators theory in [6], with the utility function  $r(x, u) = 2(x_1^2 + x_2^2) + 0.5u^2$  and the optimal value function  $V^*(x) = x_1^2 + x_2^2$ , the optimal control is  $u^* = -2x_1 x_2$ .

In this example, the critic and the action network are approximated by the three-layer feed-forward NNs with structure 2-8-2 and 2-8-1, respectively. In the training process, the learning rate for both the networks are selected as 0.05. Since the Hamiltonian-driven ADP requires an admissible policy from the start, an initial admissible policy,  $u_0 = -0.3x_1 x_2$ , was selected and the initial weights of the action network were determined to approximate the admissible initial policy. Then the initial action network weight matrices and bias vectors are obtained as

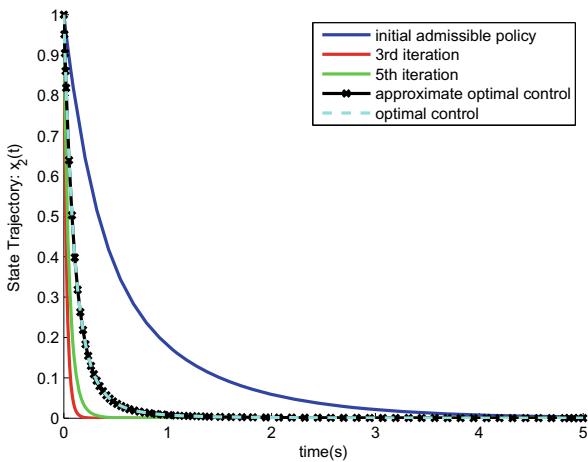
$$W_u^{0,1} = \begin{bmatrix} 0.4673 & -0.6387 & -0.5629 & 0.2118 \\ 1.9965 & -0.3561 & 0.3639 & -1.7266 \\ -0.2468 & -0.7314 & -0.8507 & -1.8226 \\ 1.8733 & -0.3313 & 0.4938 & 0.9494 \end{bmatrix}^T,$$

$$b_u^{0,1} = [3.7346 \ 1.5949 \ 1.5377 \ -0.5633]$$

**Fig. 7.7** The state trajectories  $x_1(t)$  in Example 2 (Nonlinear Regulator Problem)



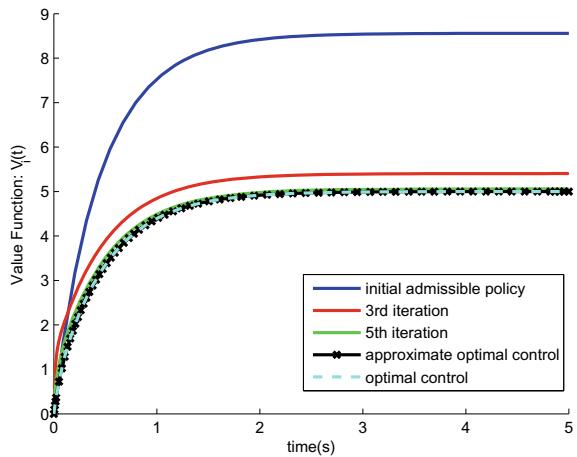
**Fig. 7.8** The state trajectories  $x_2(t)$  in Example 2 (Nonlinear Regulator Problem)



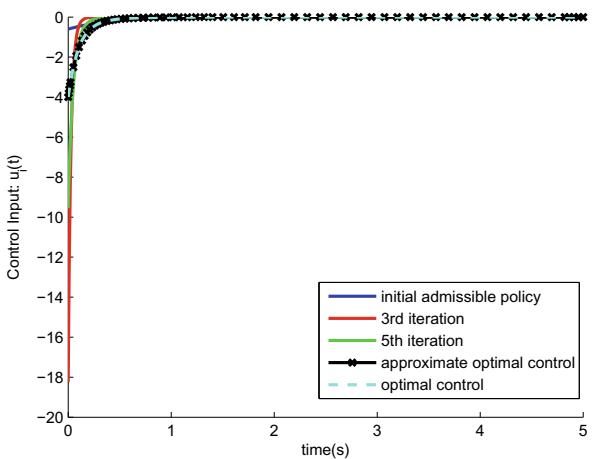
$$\begin{aligned} & 0.5961 \ -1.6931 \ -2.0020 \ -3.6428 \Big]^T, \\ W_u^{0,2} = & \Big[ 0.0736 \ 0.9980 \ -1.1636 \ -0.4963 \\ & -0.4319 \ -0.7699 \ 1.1788 \ -0.2473 \Big]^T, \\ b_u^{0,1} = & 0.2088. \end{aligned}$$

The initial state is chosen as nonzero  $x_0 = [2 \ 1]^T$ . The iteration stop criteria is  $\|w_u^i - w_u^{i+1}\| \leq \varepsilon_u = 10^{-6}$ . The convergence processes of the state trajectories  $x_1(t)$ ,  $x_2(t)$ , the cost function and the control input of the results are shown in Figs. 7.7, 7.8, 7.9, and 7.10. The simulation results show that the improving performance of the controller was derived from the Hamiltonian-driven ADP method. At the end of the training, the action network weight matrices and bias vectors converge to

**Fig. 7.9** The convergence of value function/cost functional in Example 2 (Nonlinear Regulator Problem)



**Fig. 7.10** The control input in Example 2 (Nonlinear Regulator Problem)



$$W_u^{*,1} = \begin{bmatrix} -0.9904 & 0.9000 & -0.5554 & 0.6990 \\ -1.0204 & -0.9142 & 0.4688 & 0.7290 \\ 0.4993 & 0.5131 & -0.8170 & 1.0513 \\ -0.7561 & 0.5464 & 0.9925 & -1.1744 \end{bmatrix}^T,$$

$$b_u^{*,1} = [3.5111 \quad -3.3617 \quad 1.0123 \quad -1.2554 \quad 0.4313 \quad 1.7611 \quad -2.1143 \quad 4.2294]^T,$$

$$W_u^{*,2} = [3.2331 \quad 3.0239 \quad -3.1557 \quad -2.3512 \quad -1.1918 \quad 5.6804 \quad 1.8584 \quad -2.9646]^T,$$

$$b_u^{*,1} = 0.0823.$$

## 7.6 Conclusion

Value function approximation in continuous-time systems is a challenging problem in ADP. Establishing a Hamiltonian-driven framework of ADP is based on the convergence proof of the presented iterative ADP algorithm for the approximate solution of the HJB equation continuous-time nonlinear dynamical systems. A detailed intuitive interpretation of the Hamiltonian-driven framework in function space is also given. The Hamiltonian can serve as the general temporal difference for the continuous-time systems. Within the Hamiltonian framework, both the value function gradient and the policy sequence converge to the optimum.

## References

1. Abu-Khalaf, M., Lewis, F.L.: Nearly optimal control laws for nonlinear systems with saturating actuators using a neural network hjb approach. *Automatica* **41**(5), 779–791 (2005)
2. Bellman, R.: *Dynamic Programming*. Princeton University Press, Princeton, NJ, USA (1957)
3. Dimitri, P.: *Nonlinear programming*, Athena Scientific, Belmont, MA (1995)
4. Bertsekas, D.P.: *Dynamic programming and optimal control*, 3rd edn., vol. 1, Athena Scientific, Belmont, MA (2007)
5. Fairbank, M., Alonso, E.: Value-gradient learning. In: The 2012 International Joint Conference on Neural Networks (IJCNN), pp. 1–8 (2012)
6. Hadda, W., Chellaboina, V.: Nonlinear dynamical systems and control (2008)
7. Jiang, Yu., Jiang, Z.-P.: Computational adaptive optimal control for continuous-time linear systems with completely unknown dynamics. *Automatica* **48**(10), 2699–2704 (2012)
8. Orrin Frink, J.R.: Differentiation of sequences. *Bull. Am. Math. Soc.* **41**, 553–560 (1935)
9. Kiumarsi, B., Vamvoudakis, K.G., Modares, H., Lewis, F.L.: Optimal and autonomous control using reinforcement learning: a survey. *IEEE Trans. Neural Netw. Learn. Syst.* **29**(6), 2042–2062 (2018)
10. Kiumarsi, B., Lewis, F.L., Jiang, Z.-P.:  $H_\infty$  control of linear discrete-time systems: Off-policy reinforcement learning. *Automatica* **78**, 144–152 (2017)
11. Lewis, F.L., Liu, D.: Reinforcement learning and approximate dynamic programming for feedback control, vol. 17. Wiley, Hoboken (2013)
12. Lewis, F.L., Vrabie, D.: Reinforcement learning and adaptive dynamic programming for feedback control. *Circuits Syst. Mag. IEEE* **9**(3), 32–50 (2009)
13. Lewis, F.L., Vrabie, D., Vamvoudakis, K.G.: Reinforcement learning and feedback control: using natural decision methods to design optimal adaptive controllers. *IEEE Control Syst.* **32**(6), 76–105 (2012)
14. Daniel, L.: *Calculus of variations and optimal control theory: a concise introduction*. Princeton University Press, Princeton (2012)
15. Biao, L., Wu, H.-N., Tingwen, H.: Off-policy reinforcement learning for control design. *IEEE Trans. Cybern.* **45**(1), 65–76 (2015)
16. Miller, W.T., Werbos, P.J., Sutton, R.S.: *Neural networks for control*. MIT press, Cambridge (1995)
17. Modares, H., Lewis, F.L.: Optimal tracking control of nonlinear partially-unknown constrained-input systems using integral reinforcement learning. *Automatica* **50**(7), 1780–1792 (2014)
18. Modares, H., Lewis, F.L., Jiang, Z.-P.: Tracking control of completely unknown continuous-time systems via off-policy reinforcement learning. *IEEE Trans. Neural Netw. Learn. Syst.* **26**(10), 2550–2562 (2015)

19. Modares, H., Lewis, F.L., Naghibi-Sistani, M.-B.: Integral reinforcement learning and experience replay for adaptive optimal control of partially-unknown constrained-input continuous-time systems. *Automatica* **50**(1), 193–202 (2014)
20. Murray, J.J., Cox, C.J., Lendaris, G.G., Saeks, R.: Adaptive dynamic programming. *IEEE Trans. Syst. Man Cybern. Part C (Appl. Rev.)* **32**(2), 140–153 (2002)
21. Padhi, R., Unnikrishnan, N., Wang, X., Balakrishnan, S.N.: A single network adaptive critic (snac) architecture for optimal control synthesis for a class of nonlinear systems. *Neural Netw.* **19**(10), 1648–1660 (2006)
22. Lev Semenovich Pontryagin.: Mathematical theory of optimal processes. CRC Press, Boca Raton (1987)
23. Prokhorov, D.V., Wunsch, D.C., et al.: Adaptive critic designs. *Neural Netw. IEEE Trans.* **8**(5), 997–1007 (1997)
24. Saridis, G.N., Lee, C.-S.G.: An approximation theory of optimal control for trainable manipulators. *IEEE Trans. Syst. Man Cybern.* **9**(3), 152–159 (1979)
25. Speyer, J.L., Jacobson, D.H.: Primer on optimal control theory, vol. 20. SIAM, Philadelphia (2010)
26. Sutton, R.S.: Learning to predict by the methods of temporal differences. *Mach. Learn.* **3**(1), 9–44 (1988)
27. Sutton, R.S., Barto, A.G.: Reinforcement learning: an introduction, vol. 1, MIT Press, Cambridge (1998)
28. Vamvoudakis, K.G., Ferraz, H.: Model-free event-triggered control algorithm for continuous-time linear systems with optimal performance. *Automatica* **87**, 412–420 (2018)
29. Vamvoudakis, K.G., Lewis, F.L.: Online actor–critic algorithm to solve the continuous-time infinite horizon optimal control problem. *Automatica* **46**(5), 878–888 (2010)
30. Van der Schaft, A.J.: L<sub>2</sub>-gain analysis of nonlinear systems and nonlinear state-feedback h control. *IEEE Trans. Autom. Control* **37**(6), 770–784 (1992)
31. Vrabie, D., Lewis, F.: Neural network approach to continuous-time direct adaptive optimal control for partially unknown nonlinear systems. *Neural Netw.* **22**(3), 237–246 (2009)
32. Wang, F.-Y., Zhang, H., Liu, D.: Adaptive dynamic programming: an introduction. *IEEE Comput. Intell. Mag.* **4**(2), 39–47 (2009)
33. Werbos, P.J.: Beyond regression: new tools for prediction and analysis in the behavioral sciences. Ph.D. thesis, Harvard (1974)
34. Paul John Werbos: Consistency of hdp applied to a simple reinforcement learning problem. *Neural Netw.* **3**(2), 179–189 (1990)
35. Paul John Werbos: Approximate dynamic programming for real-time control and neural modeling. *Handb. Intell. Control.: Neural Fuzzy Adapt. Approaches* **15**, 493–525 (1992)
36. Werbos, P.J.: The roots of backpropagation: from ordered derivatives to neural networks and political forecasting, vol. 1. Wiley, Hoboken (1994)
37. White, D.A., Sofge, D.A.: Handbook of intelligent control: neural, fuzzy, and adaptative approaches. Van Nostrand Reinhold Company, New York (1992)
38. Yang, Y., Guo, Z.-S., Haoyi Xiong, Z.-S., Ding, D.-W., Yin, Y., Wunsch, D.C.: Data-driven robust control of discrete-time uncertain linear systems via off-policy reinforcement learning. *IEEE Trans. Neural Netw. Learn. Syst.* (2019)
39. Yang, Y., Modares, H., Wunsch, D.C., Yin, Y.: Leader-follower output synchronization of linear heterogeneous systems with active leader using reinforcement learning. *IEEE Trans. Neural Netw. Learn. Syst.* **29**(6), 2139–2153 (2018)
40. Yang, Y.: Optimal containment control of unknown heterogeneous systems with active leaders. *IEEE Trans. Control Syst. Technol.* **27**(3), 1228–1236 (2019)
41. Yang, Y., Vamvoudakis, K.G., Ferraz, H., Modares, H.: Dynamic intermittent Q-learning-based model-free suboptimal co-design of L<sub>2</sub>-stabilization. *Int. J. Robust Nonlinear Control* **29**(9), 2673–2694 (2019)
42. Yang, Y., Wunsch, D.C., Yin, Y.: Hamiltonian-driven adaptive dynamic programming for continuous nonlinear dynamical systems. *IEEE Trans. Neural Netw. Learn. Syst.* **28**(8), 1929–1940 (2017)

## Chapter 8

# Reinforcement Learning for Optimal Adaptive Control of Time Delay Systems



Syed Ali Asad Rizvi, Yusheng Wei, and Zongli Lin

**Abstract** This chapter presents reinforcement learning techniques for solving the optimal stabilization problem for time delay systems with unknown delays and unknown system dynamics. The class of systems that we consider are general discrete-time linear systems that are subject to multiple delays in both the state and the input. Both state and output feedback algorithms are presented. The central idea in the proposed scheme revolves around the state augmentation that brings the system into a delay-free form. In particular, we present an extended state augmentation method that utilizes only the upper bounds of the delays instead of requiring the exact knowledge of the number and lengths of the delays. This extended augmentation scheme accommodates unknown delays. The associated controllability and observability conditions of the extended augmented system are established. The model-free reinforcement learning scheme that we consider is based on iterative Q-learning, which enables the controller to learn the optimal control parameters based on the input-state or input–output data. The proposed scheme has the advantage that it is immune to the exploration bias. As a result, the need of discounted cost functions is circumvented, which in turn guarantees closed-loop stability and optimality. Simulation results are presented to demonstrate the effectiveness of the proposed scheme.

---

S. A. A. Rizvi · Y. Wei · Z. Lin (✉)  
University of Virginia, Charlottesville, VA, USA  
e-mail: [z15y@virginia.edu](mailto:z15y@virginia.edu)

S. A. A. Rizvi  
e-mail: [sr9gs@virginia.edu](mailto:sr9gs@virginia.edu)

Y. Wei  
e-mail: [yw4dp@virginia.edu](mailto:yw4dp@virginia.edu)

## 8.1 Introduction

Time delay is present in many real-world systems in science and engineering. Very often delays arise in control implementation such as those associated with the sensors, actuators, communication, and computation. The presence of time delays makes the control problem quite challenging because of the performance degradation and instabilities that may result from these delays. As a result, stabilization of time delay systems is a practical control issue. Furthermore, stability is only a bare minimum requirement in operating a time delay system. It is often desirable that a system behaves in a prescribed optimal manner to meet certain performance objectives. To this date, significant progress has been demonstrated in the control literature on model-based control approaches to addressing time delay control problems [3]. However, the requirement of perfectly known models is not always realistic in many applications owing to the complexity of the systems themselves and the ever presence of model uncertainties. As a result, model-free optimal control techniques are desirable in these scenarios.

A well-established approach in the time delay control literature revolves around the idea of bringing the delay system into a delay-free form so that the standard control techniques may then become applicable. The classical techniques such as the Smith predictor [14] or the finite spectrum assignment (predictor feedback) [10] employ an internal model of the system to perform future predictions in order to cancel out the effect of delays, thereby making the closed-loop system delay-free. Standard optimal control results such as linear quadratic regulation (LQR) then become readily applicable based on these predictive approaches [11]. Furthermore, these predictive techniques work in both continuous-time and discrete-time settings. However, the standard prediction-based optimal control approaches require knowledge of the system model as well as the time delays.

Recent advances in reinforcement learning (RL) and adaptive (approximate) dynamic programming (ADP) have shown a revived interest in optimal control problems. These techniques have repeatedly demonstrated their potential in designing control algorithms that feature both adaptivity and optimality without requiring the knowledge of the underlying system model [8, 16]. The basic idea in ADP is to learn an optimal controller by estimating the solution of the Hamilton Jacobi Bellman (HJB) equation, a partial differential equation whose solution is often intractable [17, 21]. In the special case of linear dynamics, this problem boils down to estimating the solution of an algebraic Riccati equation (ARE). ADP involves sequential improvements in control policies based on the evaluation of online system trajectory data. The power of ADP is evident from its success in solving a range of control problems such as optimal stabilization [4], optimal tracking [12], robust control [15], and output regulation [2] for a wide range of systems, both linear and nonlinear systems, in both continuous-time and discrete-time settings [18]. While these works reflect significant applicability of ADP, the ADP-based control of time delay systems entails further attention.

Recently, there have been some limited developments in the RL and ADP literature towards addressing control problems for time delay systems. One salient difficulty in applying the predictor feedback approach in ADP is that the computation of the prediction signal is based on an embedded model of the system, which is unfortunately not available in the model-free ADP framework. This difficulty was addressed in [20], where a bicausal change of coordinates was used to bring the discrete-time delay system into a delay-free form. Unlike the predictor feedback approach, the original open-loop system is brought into a delay-free form, and the need of a predictor feedback signal is avoided. This work was also extended to solve the optimal tracking problem [9]. However, the existence of the bicausal transformation could be quite a restrictive assumption and is difficult to verify when the system dynamics are unknown. Furthermore, while these designs are model-free, the exact knowledge of the delay is required.

We make use of the finite dimensionality property of the discrete-time delay systems that enables us to treat the delayed variables as a new state. By state augmentation, the delay system is brought into a delay-free form without requiring any transformation conditions or the need of a predictor feedback signal. To uplift the requirement of the knowledge of the delays, we use an upper bound on the state and input delays to form an extended augmented delay-free system. In other words, the tasks of dealing with unknown dynamics and unknown delays (both the number of delays and the lengths of the delays) are combined into a larger but manageable unknown dynamics problem. The controllability conditions of the extended augmented system are presented and it is shown that the controllability is preserved under the extended augmentation. An iterative Q-learning algorithm is presented that makes use of the augmented states to learn the optimal control parameters. Compared with the previous works, the presented scheme relaxes both the requirements of the bicausal transformation and the exact knowledge of the time delays.

The second main part of this chapter focuses on the design of output feedback Q-learning controller. In general, output feedback design in a model-free setting is more challenging as the lack of a system model hinders the design of a classical state observer. To overcome this difficulty, we present a direct output feedback learning and control approach that circumvents the need of a separate state observer. First, we derive the observability conditions of the augmented and extended augmented systems. In contrast to the controllability conditions, the observability conditions are more involved and, in certain cases, full observability may not be achievable. A parameterization of the state in terms of the delayed input and output measurements is embedded in the Bellman equation to directly learn the optimal output feedback control parameters. An iterative Q-learning scheme to learn the optimal output feedback Q-function is presented using only the measured input and output data.

The remainder of this chapter is organized as follows. Section 8.2 formulates the problem. In Sect. 8.3, we present an extended augmentation procedure to bring the system into a delay-free form without requiring the knowledge of actual delays. Controllability and observability conditions of the extended system are derived. Then, in Sect. 8.4, an iterative Q-learning scheme is presented to learn the optimal control parameters without requiring the knowledge of the system dynamics and the

delays. The design for output feedback Q-learning control is presented in Sect. 8.5. Section 8.6 provides simulation results of the given algorithms. Finally, Sect. 8.7 draws a conclusion to the chapter.

## 8.2 Problem Description

Consider a discrete-time linear system given by the following state space representation,

$$\begin{aligned} x_{k+1} &= \sum_{i=0}^S A_i x_{k-i} + \sum_{i=0}^T B_i u_{k-i}, \\ y_k &= C x_k, \end{aligned} \tag{8.1}$$

where  $x_k \in \mathbb{R}^n$  represents the internal state,  $u_k \in \mathbb{R}^m$  is the control input vector and  $y_k \in \mathbb{R}^p$  is the output vector. The system is subject to multiple delays in both the state and the input with  $S$  and  $T$  being the maximum amount of state and input delays, respectively. We assume that neither the system dynamics ( $A_i$ ,  $B_i$ ,  $C$ ) nor the information of delays (both number and lengths) are known. The only assumptions we make are given as follows.

**Assumption 8.1** The following rank condition holds,

$$\rho \left[ \sum_{i=0}^S A_i \lambda^{S-i} - \lambda^{S+1} I \quad \sum_{i=0}^T B_i \lambda^{T-i} \right] = n,$$

for any  $\lambda \in \mathbb{C} : \det \left( \sum_{i=0}^S A_i \lambda^{S-i} - \lambda^{S+1} I \right) = 0 \right\}$ , where  $\rho(\cdot)$  denotes the rank of a matrix.

**Assumption 8.2** The following rank conditions hold,

$$\rho \left[ \sum_{i=0}^S A_i^\top \lambda^{S-i} - \lambda^{S+1} I \quad C^\top \right] = n,$$

for any  $\lambda \in \mathbb{C} : \det \left( \sum_{i=0}^S A_i \lambda^{S-i} - \lambda^{S+1} I \right) = 0 \text{ and } \lambda \neq 0 \right\}$  and  $\rho(A_S) = n$ .

**Assumption 8.3** The upper bounds  $\bar{S} \geq S$  and  $\bar{T} \geq T$  on the state and the input delays are known.

As will be shown, Assumption 8.1 and 8.2 are the generalization of the controllability and observability conditions of a linear system to systems with multiple state and input delays. Assumption 8.3 is needed for the extended augmentation presented in this paper. Note that this assumption is mild because we can always choose a large enough upper bound on the delay without requiring any knowledge of the maximum delays as long as the conditions  $\bar{S} \geq S$  and  $\bar{T} \geq T$  hold. Under these assumptions,

we are interested in solving the linear quadratic regulation (LQR) problem for the time delay system (8.1) with the following cost function,

$$V(x, u) = \sum_{i=k}^{\infty} r(x_i, u_i), \quad (8.2)$$

where  $r(x_k, u_k)$  takes the following quadratic form,

$$r_k = x_k^T Q x_k + u_k^T R u_k, \quad (8.3)$$

and where  $Q = Q^T \geq 0$  and  $R = R^T > 0$  correspond to the desired parameters for penalizing the states and the delayed control, respectively.

### 8.3 Extended State Augmentation

We first present a state augmentation procedure that brings the system into a delay-free form. The augmentation is carried out by introducing the delayed states and the delayed control inputs as additional states. To this end, let us define the augmented state vector as

$$X_k = [x_k^T \ x_{k-1}^T \ \cdots \ x_{k-S}^T \ u_{k-T}^T \ u_{k-T+1}^T \ \cdots \ u_{k-1}^T]^T. \quad (8.4)$$

The dynamic equation of the augmented system can be obtained from the original dynamics (8.1) as follows:

$$X_{k+1} = \left[ \begin{array}{c|ccccc} A_0 & A_1 & A_2 & \cdots & A_S & B_T & B_{T-1} & B_{T-2} & \cdots & B_1 \\ I_n & 0 & 0 & \cdots & 0 & 0 & \cdots & 0 & \cdots & 0 \\ 0 & I_n & 0 & \cdots & 0 & 0 & \cdots & 0 & 0 & 0 \\ \vdots & \ddots & \ddots & \ddots & \vdots & \vdots & \ddots & \vdots & \vdots & \vdots \\ 0 & 0 & \cdots & I_n & 0 & 0 & \cdots & 0 & 0 & 0 \\ \hline 0 & \cdots & 0 & 0 & 0 & 0 & I_m & 0 & \cdots & 0 \\ 0 & \cdots & 0 & 0 & 0 & 0 & 0 & I_m & \vdots & 0 \\ \vdots & \ddots & \ddots & \vdots \\ 0 & \cdots & 0 & 0 & 0 & 0 & 0 & 0 & \cdots & 0 & I_m \\ 0 & \cdots & 0 & 0 & 0 & 0 & 0 & 0 & \cdots & 0 & 0 \end{array} \right] X_k + \begin{bmatrix} B_0 \\ 0 \\ 0 \\ \vdots \\ \vdots \\ 0 \\ I_m \end{bmatrix} u_k$$

$$\stackrel{\Delta}{=} AX_k + Bu_k. \quad (8.5)$$

Since the maximum state delay  $S$  and input delay  $T$  are not known, we will extend the augmentation further up to their upper bounds  $\bar{S}$  and  $\bar{T}$ , respectively. For this purpose, we introduce the extended augmented state vector as

$$\bar{X}_k = \begin{bmatrix} x_k^T & x_{k-1}^T & \cdots & x_{k-S}^T & x_{k-S-1}^T & \cdots & x_{k-\bar{S}}^T \\ u_{k-\bar{T}}^T & u_{k-\bar{T}+1}^T & \cdots & u_{k-T}^T & \cdots & u_{k-1}^T \end{bmatrix}^T, \quad (8.6)$$

and the corresponding extended augmented dynamics is given by

$$\begin{aligned} \bar{X}_{k+1} = & \begin{bmatrix} \overbrace{A_0 \cdots A_S}^{\bar{S}+1 \text{ blocks}} & 0 & \cdots & 0 & \cdots & B_T & \cdots & B_1 \\ I_n & 0 & 0 & \cdots & 0 & 0 & \cdots & 0 \\ 0 & I_n & 0 & \cdots & 0 & 0 & \cdots & 0 \\ \vdots & \ddots & \ddots & \vdots & \vdots & \vdots & \cdots & \vdots \\ 0 & \cdots & 0 & I_n & 0 & 0 & \cdots & 0 \\ 0 & \cdots & 0 & 0 & 0 & 0 & \cdots & 0 \\ 0 & \cdots & 0 & 0 & 0 & 0 & \cdots & 0 \\ \vdots & \cdots & \vdots & \vdots & \vdots & \vdots & \ddots & \vdots \\ 0 & \cdots & 0 & 0 & 0 & 0 & \cdots & 0 \\ 0 & \cdots & 0 & 0 & 0 & 0 & \cdots & 0 \end{bmatrix} \bar{X}_k + \begin{bmatrix} B_0 \\ 0 \\ 0 \\ \vdots \\ \vdots \\ u_k \\ \vdots \\ 0 \\ I_m \end{bmatrix} \\ & \triangleq \bar{A} \bar{X}_k + \bar{B} u_k. \end{aligned} \quad (8.7)$$

Comparing the extended dynamics (8.7) with the original dynamics (8.1), we see that the problem of finding an optimal controller for a time delay system with both unknown delays and unknown dynamics is equivalent to finding an optimal controller for an extended delay-free system with unknown dynamics.

We will now study the controllability property of the augmented systems (8.5) and (8.7).

**Theorem 8.1** *The delay-free augmented systems (8.5) and (8.7) are controllable if and only if*

$$\rho \left[ \sum_{i=0}^S A_i \lambda^{S-i} - \lambda^{S+1} I \quad \sum_{i=0}^T B_i \lambda^{T-i} \right] = n, \quad (8.8)$$

for any  $\lambda \in \mathbb{C} : \det \left( \sum_{i=0}^S A_i \lambda^{S-i} - \lambda^{S+1} I \right) = 0 \right\}.$

**Proof** Consider the augmented system (8.5). From linear systems theory, the system is controllable if and only if  $[A - \lambda I \ B]$  has a full row rank for all  $\lambda \in \mathbb{C}$ . We evaluate  $\rho [A - \lambda I \ B]$  as

$$\rho [A - \lambda I \ B] = \rho \left[ \begin{array}{cc|ccccc|cc} A_0 - \lambda I & A_1 & A_2 & \cdots & A_S & B_T & B_{T-1} & B_{T-2} & \cdots & B_1 & B_0 \\ I_n & -\lambda I & 0 & \cdots & 0 & 0 & \cdots & 0 & \cdots & 0 & 0 \\ 0 & I_n & -\lambda I & \vdots & \vdots & 0 & \cdots & 0 & \cdots & 0 & 0 \\ \vdots & \ddots & \ddots & \ddots & \vdots & \vdots & \cdots & \vdots & \cdots & \vdots & \vdots \\ 0 & \cdots & 0 & I_n & -\lambda I & 0 & \cdots & 0 & \cdots & 0 & 0 \\ \hline 0 & \cdots & 0 & 0 & 0 & -\lambda I & I_m & 0 & \cdots & 0 & 0 \\ 0 & \cdots & 0 & 0 & 0 & 0 & -\lambda I & I_m & \vdots & 0 & 0 \\ \vdots & \cdots & \vdots & \vdots & \vdots & \vdots & \cdots & \ddots & \ddots & \vdots & \vdots \\ 0 & \cdots & 0 & 0 & 0 & 0 & 0 & \cdots & -\lambda I & I_m & 0 \\ 0 & \cdots & 0 & 0 & 0 & 0 & 0 & \cdots & 0 & -\lambda I & I_m \end{array} \right].$$

Consider the columns associated with the  $B_i$ 's. Adding  $\lambda$  times the last column into the second last column, we have

$$\rho [A - \lambda I \ B] = \rho \left[ \begin{array}{c|ccccc|c} * & B_T & B_{T-1} & B_{T-2} & \cdots & B_1 + B_0\lambda & B_0 \\ * & 0 & 0 & \cdots & \cdots & 0 & 0 \\ \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots \\ * & 0 & 0 & \cdots & \cdots & 0 & 0 \\ \hline 0 & -\lambda I & I_m & 0 & \cdots & 0 & 0 \\ 0 & 0 & -\lambda I & I_m & \cdots & 0 & 0 \\ 0 & 0 & 0 & \ddots & \ddots & I_m & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & I_m \end{array} \right].$$

Repeating the above steps for each of the remaining columns results in

$$\rho [A - \lambda I \ B] = \rho \left[ \begin{array}{c|ccccc|c} * & B_T + B_{T-1}\lambda + \cdots + B_0\lambda^T & \cdots & \cdots & \cdots & B_1 + B_0\lambda & B_0 \\ * & 0 & 0 & \cdots & \cdots & 0 & 0 \\ \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots \\ * & 0 & 0 & \cdots & \cdots & 0 & 0 \\ \hline 0 & 0 & I_m & 0 & \cdots & 0 & 0 \\ 0 & 0 & 0 & I_m & \cdots & 0 & 0 \\ \vdots & \vdots & \cdots & \ddots & \ddots & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & I_m & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & I_m \end{array} \right].$$

Similarly, row operations can be used to cancel the entries in the first row and result in

$$\rho [A - \lambda I \ B] = \rho \left[ \begin{array}{c|cc|ccc} * & B_T + B_{T-1}\lambda + \cdots + B_0\lambda^T & 0 & 0 & \cdots & 0 \\ * & 0 & 0 & 0 & \cdots & 0 \\ \vdots & \vdots & \vdots & \vdots & \ddots & \vdots \\ * & 0 & 0 & 0 & 0 & \cdots & 0 \\ \hline 0 & 0 & I_m & 0 & \cdots & 0 \\ 0 & 0 & 0 & I_m & 0 & 0 \\ \vdots & \vdots & \vdots & \ddots & \ddots & \vdots \\ 0 & 0 & 0 & 0 & 0 & I_m \end{array} \right].$$

We can now consider the columns related to the  $A'_i$ 's in

$$\rho [A - \lambda I \ B] = \rho \left[ \begin{array}{c|cc|ccc} A_0 - \lambda I & A_1 & \cdots & A_S & B_T + B_{T-1}\lambda + \cdots + B_0\lambda^T & 0 & 0 & \cdots & 0 \\ I_n & -\lambda I & 0 & \cdots & 0 & 0 & 0 & \cdots & 0 \\ 0 & \ddots & \ddots & \vdots & \vdots & \vdots & \vdots & \ddots & \vdots \\ 0 & \cdots & I_n & -\lambda I & 0 & 0 & 0 & \cdots & 0 \\ \hline 0 & 0 & 0 & 0 & 0 & I_m & 0 & \cdots & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & I_m & 0 & 0 \\ \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \ddots & \ddots & \vdots \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & \cdots & I_m \end{array} \right].$$

Applying similar row and column operations to cancel out the entries corresponding to the columns of  $A'_i$ 's results in

$$\begin{aligned} \rho [A - \lambda I \ B] &= \rho \left[ \begin{array}{c|cc|ccc} 0 & 0 & \cdots & \sum_{i=0}^S A_i \lambda^{S-i} - \lambda^{S+1} I & \sum_{i=0}^T B_i \lambda^{T-i} & 0 & 0 & \cdots & 0 \\ I_n & 0 & \cdots & 0 & 0 & 0 & 0 & \cdots & 0 \\ 0 & \ddots & \vdots & \vdots & \vdots & \vdots & \vdots & \ddots & \vdots \\ 0 & \cdots & I_n & 0 & 0 & 0 & 0 & \cdots & 0 \\ \hline 0 & 0 & 0 & 0 & 0 & I_m & 0 & \cdots & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & I_m & 0 & 0 \\ \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \ddots & \ddots & \vdots \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & \cdots & I_m \end{array} \right] \\ &= \rho \left[ \sum_{i=0}^S A_i \lambda^{S-i} - \lambda^{S+1} I \ \sum_{i=0}^T B_i \lambda^{T-i} \right] + nS + mT. \end{aligned}$$

At full row rank,  $\rho [A - \lambda I \ B] = n + nS + mT$ , which requires

$$\rho \left[ \sum_{i=0}^S A_i \lambda^{S-i} - \lambda^{S+1} I \ \sum_{i=0}^T B_i \lambda^{T-i} \right] = n, \quad \lambda \in \mathbb{C}.$$

Let  $P(\lambda) = A_S + A_{S-1}\lambda + \cdots + A_0\lambda^S - \lambda^{S+1}I$  be a matrix polynomial of  $\lambda$ . Then,  $P(\lambda)$  loses its rank for any  $\lambda \in \{\lambda \in \mathbb{C} : \det(\sum_{i=0}^S A_i \lambda^{S-i} - \lambda^{S+1} I) = 0\}$ . Thus, only for such  $\lambda$ 's does it need to be ensured that

$$\rho \left[ \sum_{i=0}^S A_i \lambda^{S-i} - \lambda^{S+1} I \sum_{i=0}^T B_i \lambda^{T-i} \right] = n,$$

which is condition (8.8).

For the extended augmented system (8.7), we evaluate  $\rho \left[ \bar{A} - \lambda I \quad \bar{B} \right]$  following similar row and column operations to result in,

$$\begin{aligned} \rho \left[ \bar{A} - \lambda I \quad \bar{B} \right] &= \rho \left[ \begin{array}{cccccc|cccccc} A_0 - \lambda I & \cdots & A_S & 0 & \cdots & 0 & \cdots & B_T & \cdots & B_1 & B_0 \\ I_n & -\lambda I & 0 & \cdots & 0 & 0 & \cdots & 0 & \cdots & 0 & 0 \\ 0 & I_n & -\lambda I & \vdots & \vdots & 0 & \cdots & 0 & \cdots & 0 & 0 \\ 0 & \ddots & \ddots & \ddots & \vdots & \vdots & \cdots & \vdots & \cdots & \vdots & \vdots \\ 0 & \cdots & 0 & I_n & -\lambda I & 0 & \cdots & 0 & \cdots & 0 & 0 \\ \hline 0 & \cdots & 0 & 0 & 0 & -\lambda I & I_m & 0 & \cdots & 0 & 0 \\ 0 & \cdots & 0 & 0 & 0 & 0 & -\lambda I & I_m & \vdots & 0 & 0 \\ \vdots & \cdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \ddots & \ddots & 0 & 0 \\ 0 & \cdots & 0 & 0 & 0 & 0 & 0 & 0 & \cdots & -\lambda I & I_m & 0 \\ 0 & \cdots & 0 & 0 & 0 & 0 & 0 & 0 & \cdots & 0 & -\lambda I & I_m \end{array} \right] \\ &= \rho \left[ \sum_{i=0}^S A_i \lambda^{S-i} - \lambda^{S+1} I \sum_{i=0}^T B_i \lambda^{T-i} \right] + n \bar{S} + m \bar{T}, \end{aligned}$$

where we have used the fact that the padded zero columns do not affect the row rank, whereas the  $\bar{S}$  number of  $I_n$  and  $\bar{T}$  number of  $I_m$  matrices contribute  $n \bar{S} + m \bar{T}$  to the row rank. For the controllability of the extended augmented system (8.7), we need  $\rho \left[ \bar{A} - \lambda I \quad \bar{B} \right] = n + n \bar{S} + m \bar{T}$ , which requires

$$\rho \left[ \sum_{i=0}^S A_i \lambda^{S-i} - \lambda^{S+1} I \sum_{i=0}^T B_i \lambda^{T-i} \right] = n,$$

for any  $\lambda \in \left\{ \lambda \in \mathbb{C} : \det \left( \sum_{i=0}^S A_i \lambda^{S-i} - \lambda^{S+1} I \right) = 0 \right\}$ , which is again condition (8.8). This shows that the augmented and extended augmented systems are controllable if and only if (8.8) holds. This completes the proof.  $\square$

**Remark 8.1** The controllability condition can be relaxed to stabilizability, in which case (8.8) needs to hold only for any

$$\lambda \in \left\{ \lambda \in \mathbb{C} : \det \left( \sum_{i=0}^S A_i \lambda^{S-i} - \lambda^{S+1} I \right) = 0 \text{ and } |\lambda| \geq 1 \right\}.$$

We now bring the focus on the design of the optimal controller. To this end, we rewrite the original utility function for (8.1) in terms of the extended augmented state vector (8.6) as

$$r_k = \bar{X}_k^\top \bar{Q} \bar{X}_k + u_k^\top R u_k, \tag{8.9}$$

where

$$\bar{Q} = \begin{bmatrix} Q & 0 \\ 0 & 0 \end{bmatrix}.$$

Since the system is now in a delay-free form, we can readily compute the optimal controller. From optimal control theory [6], we know that there exists a unique optimal control sequence

$$u_k^* = \bar{K}^* \bar{X}_k = -(\bar{R} + \bar{B}^\top \bar{P}^* \bar{B})^{-1} \bar{B}^\top \bar{P}^* \bar{A} \bar{X}_k \quad (8.10)$$

that minimizes (8.2) under the conditions of the stabilizability of  $(\bar{A}, \bar{B})$  and detectability of  $(\bar{A}, \sqrt{\bar{Q}})$ , where  $\bar{P}^* = (\bar{P}^*)^\top$  is the unique positive semi-definite solution to the following ARE,

$$\bar{A}^\top \bar{P} \bar{A} - \bar{P} + \bar{Q} - \bar{A}^\top \bar{P} \bar{B} (\bar{R} + \bar{B}^\top \bar{P} \bar{B})^{-1} \bar{B}^\top \bar{P} \bar{A} = 0. \quad (8.11)$$

**Remark 8.2** The extended augmented states from  $x_{k-T-1}$  to  $x_{k-\bar{T}}$  and from  $x_{k-S-1}$  to  $x_{k-\bar{S}}$  are fictitious states, and therefore, they are not reflected in the optimal control law, as will be seen in the simulation results. That is, the control coefficients corresponding to these states are zero.

We next work toward deriving the conditions for the observability of the extended augmented system. To this end, we introduce an augmented output vector as defined by

$$Y_k = [y_k^\top \ u_{k-T}^\top]^\top = CX_k, \quad C = \left[ \begin{array}{c|ccccc} \overbrace{C \ 0 \ 0 \ \cdots \ 0}^{S+1} & \overbrace{0 \ 0 \ 0 \ \cdots \ 0}^T \\ \hline 0 \ 0 \ 0 \ \cdots \ 0 & I_m \ 0 \ 0 \ \cdots \ 0 \end{array} \right].$$

In the same spirit, we can obtain an extended augmented output vector by incorporating the fictitious states corresponding to the delayed states from  $x_{k-S-1}$  to  $x_{k-\bar{S}}$  and to the delayed inputs from  $u_{k-T-1}$  to  $u_{k-\bar{T}}$  as follows:

$$\tilde{Y}_k = [y_k^\top \ u_{k-\bar{T}}^\top]^\top = \tilde{C} \tilde{X}_k, \quad \tilde{C} = \left[ \begin{array}{c|ccccc} \overbrace{C \ 0 \ 0 \ \cdots \ 0}^{\bar{S}+1} & \overbrace{0 \ 0 \ 0 \ \cdots \ 0}^{\bar{T}} \\ \hline 0 \ 0 \ 0 \ \cdots \ 0 & I_m \ 0 \ 0 \ \cdots \ 0 \end{array} \right].$$

We will now study the observability property of the augmented systems (8.5) and (8.7).

**Theorem 8.2** *The delay-free augmented systems (8.5) is observable if and only if*

$$\rho \left[ \sum_{i=0}^S A_i^\top \lambda^{S-i} - \lambda^{S+1} I \quad C^\top \right] = n, \quad (8.12)$$

for any  $\lambda \in \mathbb{C} : \det \left( \sum_{i=0}^S A_i \lambda^{S-i} - \lambda^{S+1} I \right) = 0$  and  $\lambda \neq 0$ , and

$$\rho(A_S) = n. \quad (8.13)$$

The extended augmented system (8.7) is detectable if and only if (8.12) holds for any  $\lambda \in \{\lambda \in \mathbb{C} : \det(\sum_{i=0}^S A_i \lambda^{S-i} - \lambda^{S+1} I) = 0 \text{ and } |\lambda| \geq 1\}$ .

**Proof** By duality, the observability of  $(A, C)$  implies the controllability of  $(A^\top, C^\top)$ . Thus,  $(A, C)$  is observable if and only if  $[A^\top - \lambda I \ C^\top]$  has a full row rank of  $(S + 1)n + mT$ . We evaluate  $\rho[A^\top - \lambda I \ C^\top]$  as

$$\rho[A^\top - \lambda I \ C^\top] = \rho \begin{bmatrix} A_0^\top - \lambda I & I_n & 0 & \cdots & 0 & 0 & 0 & 0 & \cdots & 0 & C^\top & 0 \\ A_1^\top & -\lambda I & I_n & \cdots & 0 & 0 & 0 & 0 & \cdots & 0 & 0 & 0 \\ A_2^\top & 0 & -\lambda I & \ddots & 0 & 0 & 0 & 0 & \cdots & 0 & 0 & 0 \\ \vdots & \vdots & \ddots & \ddots & I_n & \vdots & \vdots & 0 & \ddots & \vdots & \vdots & \vdots \\ A_S^\top & \cdots & 0 & 0 & -\lambda I & 0 & 0 & 0 & \cdots & 0 & 0 & 0 \\ B_T^\top & \cdots & 0 & 0 & 0 & -\lambda I & 0 & 0 & \cdots & 0 & 0 & I_m \\ B_{T-1}^\top & \cdots & 0 & 0 & 0 & I_m & -\lambda I & 0 & \ddots & 0 & 0 & 0 \\ \vdots & \vdots & \vdots & \vdots & \vdots & 0 & I_m & -\lambda I & 0 & \cdots & 0 & 0 \\ \vdots & \cdots & 0 & 0 & 0 & 0 & 0 & \ddots & \ddots & 0 & 0 & 0 \\ B_1^\top & \cdots & 0 & 0 & 0 & 0 & 0 & 0 & \cdots & I_m & -\lambda I & 0 \end{bmatrix}.$$

Moving the last column block to the beginning of the right side partition and then adding  $\lambda$  times this column to the next column on its right result in

$$\rho[A^\top - \lambda I \ C^\top] = \rho \begin{bmatrix} A_0^\top - \lambda I & I_n & 0 & \cdots & 0 & 0 & 0 & 0 & \cdots & 0 & 0 & C^\top \\ A_1^\top & -\lambda I & I_n & \cdots & 0 & 0 & \cdots & 0 & \cdots & 0 & 0 & 0 \\ A_2^\top & 0 & -\lambda I & \ddots & 0 & 0 & \cdots & \cdots & \cdots & 0 & 0 & 0 \\ \vdots & \vdots & \ddots & \ddots & I_n & \vdots & \vdots & \vdots & \cdots & \vdots & \vdots & \vdots \\ A_S^\top & \cdots & 0 & 0 & -\lambda I & 0 & 0 & \cdots & \cdots & 0 & 0 & 0 \\ B_T^\top & 0 & \cdots & 0 & 0 & I_m & 0 & 0 & \cdots & 0 & 0 & 0 \\ B_{T-1}^\top & 0 & \cdots & 0 & 0 & 0 & I_m & -\lambda I & 0 & \ddots & 0 & 0 \\ \vdots & 0 & \cdots & \vdots & \vdots & 0 & 0 & I_m & -\lambda I & 0 & 0 & 0 \\ \vdots & 0 & \cdots & 0 & 0 & \vdots & 0 & \ddots & \ddots & \vdots & 0 & \vdots \\ B_1^\top & 0 & \cdots & 0 & 0 & 0 & 0 & 0 & \cdots & I_m & -\lambda I & 0 \end{bmatrix}.$$

Repeating the above step for each of the remaining columns results in

$$\rho [A^T - \lambda I \ C^T] = \rho \left[ \begin{array}{cc|ccccc} A_0^T - \lambda I & I_n & 0 & \cdots & 0 & 0 & 0 & \cdots & 0 & 0 & C^T \\ A_1^T & -\lambda I & I_n & \cdots & 0 & 0 & \cdots & 0 & \cdots & 0 & 0 \\ \vdots & \vdots & \ddots & \ddots & I_n & \vdots & \vdots & \vdots & \cdots & \vdots & \vdots \\ A_S^T & \cdots & 0 & 0 & -\lambda I & 0 & 0 & 0 & \cdots & 0 & 0 \\ \hline B_T^T & 0 & \cdots & 0 & 0 & I_m & 0 & 0 & \cdots & 0 & 0 \\ B_{T-1}^T & 0 & \cdots & 0 & 0 & 0 & I_m & 0 & 0 & \vdots & 0 \\ B_{T-2}^T & 0 & \cdots & 0 & 0 & 0 & 0 & I_m & 0 & 0 & 0 \\ \vdots & 0 & \cdots & 0 & 0 & 0 & 0 & 0 & \ddots & \ddots & 0 \\ B_1^T & 0 & \cdots & 0 & 0 & 0 & 0 & 0 & 0 & \cdots & I_m \\ \end{array} \right].$$

Similarly, we can cancel all the  $B_i$  entries in the left partition using the identity columns from the right partition, which gives

$$\rho [A^T - \lambda I \ C^T] = \rho \left[ \begin{array}{cc|ccccc} A_0^T - \lambda I & I_n & 0 & \cdots & 0 & 0 & 0 & \cdots & 0 & 0 & C^T \\ A_1^T & -\lambda I & I_n & \cdots & 0 & 0 & \cdots & 0 & \cdots & 0 & 0 \\ \vdots & \vdots & \ddots & \ddots & I_n & \vdots & \vdots & \vdots & \cdots & \vdots & \vdots \\ A_S^T & 0 & 0 & \cdots & -\lambda I & 0 & \cdots & 0 & \cdots & 0 & 0 \\ \hline 0 & \cdots & 0 & 0 & 0 & I_m & 0 & 0 & \cdots & 0 & 0 \\ 0 & \cdots & 0 & 0 & 0 & 0 & I_m & 0 & 0 & \vdots & 0 \\ \vdots & \vdots & \vdots & \vdots & \vdots & 0 & 0 & I_m & 0 & 0 & \cdots & 0 \\ 0 & \cdots & 0 & 0 & 0 & 0 & \cdots & \ddots & \ddots & 0 & 0 \\ 0 & \cdots & 0 & 0 & 0 & 0 & 0 & 0 & \cdots & I_m & 0 & 0 \\ \end{array} \right].$$

The bottom half rows involve  $T$  number of  $I_m$  matrices which contribute  $mT$  to the rank. We evaluate the remaining non-zero partition in the top half as

$$\rho \left[ \begin{array}{cc|c} A_0^T - \lambda I & I_n & 0 & \cdots & 0 & C^T \\ A_1^T & -\lambda I & I_n & \cdots & 0 & 0 \\ \vdots & \vdots & \ddots & \ddots & \vdots & 0 \\ A_S^T & 0 & 0 & \cdots & -\lambda I & 0 \\ \end{array} \right].$$

For the case when  $\lambda \neq 0$ , we can perform some elementary operations on the above matrix that gives

$$\rho \left[ \begin{array}{cc|c} A_0^T - \lambda I + \frac{1}{\lambda} A_1^T + \cdots + \frac{1}{\lambda^S} A_S^T & 0 & 0 & \cdots & 0 & C^T \\ 0 & I_n & 0 & \cdots & 0 & 0 \\ 0 & 0 & I_n & \ddots & \vdots & 0 \\ \vdots & \vdots & \ddots & \ddots & 0 & \vdots \\ 0 & 0 & 0 & \cdots & I_n & 0 \end{array} \right].$$

The  $S$  number of  $I_n$  matrices contribute  $Sn$  to the rank. Thus,  $(A, C)$  is observable if and only if

$$\rho [ A_0^T - \lambda I + \frac{1}{\lambda} A_1^T + \cdots + \frac{1}{\lambda^S} A_S^T \quad C^T ] = n,$$

or equivalently,

$$\rho [ \sum_{i=0}^S A_i^T \lambda^{S-i} - \lambda^{S+1} I \quad C^T ] = n,$$

for any  $\lambda \in \mathbb{C} : \det (\sum_{i=0}^S A_i \lambda^{S-i} - \lambda^{S+1} I) = 0$  and  $\lambda \neq 0$ , which is condition (8.12).

If  $\lambda = 0$  then

$$\rho \left[ \begin{array}{cc|c} A_0^T & I_n & 0 & \cdots & 0 & C^T \\ A_1^T & 0 & I_n & \cdots & 0 & 0 \\ A_2^T & 0 & 0 & \ddots & \vdots & 0 \\ \vdots & \vdots & \ddots & \ddots & I_n & \vdots \\ A_S^T & \cdots & 0 & 0 & 0 & 0 \end{array} \right] = (S+1)n$$

if and only if

$$\rho(A_S) = n,$$

which is condition (8.13).

For the case of the extended augmented system, it can be readily verified that for  $\lambda \neq 0$ , the first condition (8.12) remains the same due to the additional identity matrices, as in the proof of the controllability results in Theorem 8.1. The second condition in this case becomes  $\rho(A_{\bar{S}}) = n$ . However, this condition cannot be satisfied if  $\bar{S} > S$  because  $A_{\bar{S}} = 0$  for  $\bar{S} > S$ . As a result, the observability condition (8.13) for the extended augmented system loses rank for  $\lambda = 0$ . Since  $\lambda = 0$  is a stable eigenvalue, we still have the detectability condition satisfied. In the special case, when  $\bar{S} = S$ , the extended augmented system is observable if and only if (8.12) and (8.13) hold. This completes the proof.  $\square$

**Remark 8.3** The observability condition for the augmented system (8.5) can also be relaxed to detectability, in which case we require only the first condition (8.12) to hold for any  $\lambda \in \mathbb{C} : \det (\sum_{i=0}^S A_i \lambda^{S-i} - \lambda^{S+1} I) = 0$  and  $|\lambda| \geq 1$ .

## 8.4 State Feedback Q-Learning Control of Time Delay Systems

In this section, we will present a Q-learning scheme for learning the optimal control parameters that uplift the requirement of the knowledge of the system dynamics and the delays.

Consider a stabilizing control policy  $u_k = \bar{K} \bar{X}_k$ , which is not necessarily optimal with respect to our utility function. Corresponding to this controller, there is a cost given by the following value function that represents the infinite horizon cost of executing the control starting from state  $\bar{X}_k$  from time  $k$  to time  $\infty$  as follows:

$$V_{\bar{K}}(\bar{X}_k) = \bar{X}_k^T \bar{P} \bar{X}_k. \quad (8.14)$$

The above infinite horizon value function can be recursively written as

$$V_{\bar{K}}(\bar{X}_k) = \bar{X}_k^T \bar{Q} \bar{X}_k + \bar{X}_k^T \bar{K}^T R \bar{K} \bar{X}_k + V_{\bar{K}}(\bar{X}_{k+1}).$$

Similar to the value function above, we can define a Quality function (Q-function) which gives the value of executing an arbitrary control  $u_k$  instead of  $u_k = \bar{K} \bar{X}_k$  at time  $k$  and then following policy  $\bar{K}$  from time  $k+1$ ,

$$Q_{\bar{K}}(\bar{X}_k, u_k) = \bar{X}_k^T \bar{Q} \bar{X}_k + u_k^T R u_k + V_{\bar{K}}(\bar{X}_{k+1}). \quad (8.15)$$

Substituting the dynamics (8.7) in the above expression results in

$$\begin{aligned} Q_{\bar{K}}(\bar{X}_k, u_k) &= \bar{X}_k^T \bar{Q} \bar{X}_k + u_k^T R u_k + \bar{X}_{k+1}^T \bar{P} \bar{X}_{k+1} \\ &= \bar{X}_k^T \bar{Q} \bar{X}_k + u_k^T R u_k + (\bar{A} \bar{X}_k + \bar{B} u_k)^T \bar{P} (\bar{A} \bar{X}_k + \bar{B} u_k), \end{aligned}$$

or equivalently,

$$Q_{\bar{K}}(\bar{X}_k, u_k) = \begin{bmatrix} \bar{X}_k \\ u_k \end{bmatrix}^T H \begin{bmatrix} \bar{X}_k \\ u_k \end{bmatrix}, \quad (8.16)$$

where

$$H \triangleq \begin{bmatrix} H_{XX} & H_{Xu} \\ H_{uX} & H_{uu} \end{bmatrix} = \begin{bmatrix} \bar{Q} + \bar{A}^T \bar{P} \bar{A} & \bar{A}^T \bar{P} \bar{B} \\ \bar{B}^T \bar{P} \bar{A} & R + \bar{B}^T \bar{P} \bar{B} \end{bmatrix}.$$

When  $\bar{K} = \bar{K}^*$ , we have  $\bar{P} = \bar{P}^*$ ,  $Q_{\bar{K}} = Q^*$ , and the optimal LQR controller can be obtained by solving  $\left[ \frac{\partial}{\partial u_k} Q^* = 0 \right]$  for  $u_k$ , which corresponds to (8.10).

It can be seen that the problem of finding an optimal controller boils down to finding the optimal matrix  $H^*$  or the optimal Q-function  $Q^*$ .

Q-learning is a model-free learning technique that estimates the optimal Q-function without requiring the knowledge of system dynamics. It does so by means

of the following Bellman Q-learning equation,

$$Q_{\bar{K}}(\bar{X}_k, u_k) = \bar{X}_k^T \bar{Q} \bar{X}_k + u_k^T R u_k + Q_K(\bar{X}_{k+1}, \bar{K} \bar{X}_{k+1}), \quad (8.17)$$

which is obtained by substituting  $V_{\bar{K}}(\bar{X}_k) = Q_{\bar{K}}(\bar{X}_k, \bar{K} \bar{X}_k)$  in (8.15). Let

$$z_k = \begin{bmatrix} \bar{X}_k \\ u_k \end{bmatrix}.$$

Then, by definition (8.16), we can write Eq. (8.17) as

$$z_k^T H z_k = \bar{X}_k^T \bar{Q} \bar{X}_k + u_k^T R u_k + z_{k+1}^T H z_{k+1}, \quad (8.18)$$

which is linear in the unknown matrix  $H$ . We can perform the following parameterization on this equation:

$$\begin{aligned} Q_{\bar{K}}(z_k) &\triangleq Q_{\bar{K}}(\bar{X}_k, u_k) \\ &= \bar{H}^T \bar{z}_k, \end{aligned}$$

where

$$\begin{aligned} \bar{H} &= \text{vec}(H) \\ &\triangleq [h_{11}, 2h_{12}, \dots, 2h_{1l}, h_{22}, 2h_{23}, \dots, 2h_{2l}, \dots, h_{ll}]^T \in \mathbb{R}^{l(l+1)/2}, \end{aligned}$$

$h_{ii}$  are the elements of matrix  $H$  and  $l = n + n\bar{S} + m\bar{T} + m$ . The regressor  $\bar{z}_k \in \mathbb{R}^{l(l+1)/2}$  is defined as the following quadratic basis set:

$$\bar{z} = [z_1^2, z_1 z_2, \dots, z_1 z_l, z_2^2, z_2 z_3, \dots, z_2 z_l, \dots, z_l^2]^T.$$

With this parametrization, we have, from (8.18), the following equation:

$$\bar{H}^T (\bar{z}_k - \bar{z}_{k+1}) = \bar{X}_k^T \bar{Q} \bar{X}_k + u_k^T R u_k. \quad (8.19)$$

Notice that (8.19) is a scalar equation with  $l(l+1)/2$  unknowns. We can solve this equation in the least squares sense by collecting at least  $L \geq l(l+1)/2$  datasets of  $\bar{X}_k$  and  $u_k$ . The least squares solution of (8.19) is given by

$$\bar{H} = (\Phi \Phi^T)^{-1} \Phi \Upsilon, \quad (8.20)$$

where  $\Phi \in \mathbb{R}^{l(l+1)/2 \times L}$  and  $\Upsilon \in \mathbb{R}^{L \times 1}$  are the data matrices defined as

$$\begin{aligned} \Phi &= [\bar{z}_{k-L+1} - \bar{z}_{k-L+2}, \quad \bar{z}_{k-L+2} - \bar{z}_{k-L+3}, \quad \dots, \quad \bar{z}_k - \bar{z}_{k+1}], \\ \Upsilon &= [r(\bar{X}_{k-L+1}, u_{k-L+1}), \quad r(\bar{X}_{k-L+2}, u_{k-L+2}), \quad \dots, \quad r(\bar{X}_k, u_k)]^T. \end{aligned}$$

It is important to note that since  $u_k = \bar{K} \bar{X}_k$  is linearly dependent on  $\bar{X}_k$ , the least squares problem cannot be solved unless we inject an independent exploration signal  $v_k$  in  $u_k$  in order to guarantee the invertibility of  $\Phi \Phi^\top$  in (8.20). In other words, the following rank condition should hold:

$$\text{rank}(\Phi) = l(l+1)/2. \quad (8.21)$$

Typical examples of exploration signals include sinusoids, exponentially decaying signals, and white noise [8]. It has been shown in our previous work [13] that these exploration signals do not incur any bias in the Q-learning algorithm. In what follows, we present an iterative Q-learning algorithm to learn the optimal control parameters.

### Algorithm 8.1 State Feedback Iterative Q-learning for Time Delay Systems

**Initialization.** Start with an arbitrary policy  $u_k^0 = v_k$  with  $v_k$  being the exploration signal. Set  $H^0 \leftarrow I$ .

**Collect Data.** Apply the initial policy  $u^0$  to collect  $L$  datasets of  $(\bar{X}_k, u_k)$  along with their quadratic terms.

**Value Update.** Compute the least squares solution of

$$(\bar{H}^j)^\top \bar{z}_k = \bar{X}_k^\top \bar{Q} \bar{X}_k + u_k^\top R u_k + (\bar{H}^{j-1})^\top \bar{z}_{k+1}. \quad (8.22)$$

**Policy Update.** Determine an improved policy using

$$u_k^{j+1} = -(\bar{H}_{uu}^j)^{-1} \bar{H}_{uX}^j \bar{X}_k. \quad (8.23)$$

**Repeat and Stop.** Then, for  $j = 1, 2, \dots$ , repeat the steps in (8.22) and (8.23) until convergence,

$$\|\bar{H}^j - \bar{H}^{j-1}\| < \varepsilon, \quad (8.24)$$

for some small positive constant  $\varepsilon$ .

Algorithm 8.1 presents an iterative algorithm based on Q-learning. We apply an arbitrary initial control with an exploration signal  $v_k$  such that the rank condition (8.21) is satisfied. Based on this data, we solve the recursive Q-learning Bellman equation (8.22) in the value update step. In the next step, we perform a minimization of the Q-function with respect to  $u_k$ , which gives us a new policy. This policy will be evaluated in the next iteration. These iterations are carried out until we see no further updates in the estimate of the matrix  $H$  within a sufficiently small range specified by the positive constant  $\varepsilon$ . Once this criterion is met, the algorithm is considered to have converged to the (near) optimal solution. The data matrices in the recursive iterations are given as

$$\Phi = [\bar{z}_{k-L+1}, \bar{z}_{k-L+2}, \dots, \bar{z}_k],$$

$$\Upsilon = \left[ r(\bar{X}_{k-L+1}, u_{k-L+1}) + (\bar{H}^{j-1})^T \bar{z}_{k-L+2}(\bar{X}_{k-L+2}, u_{k-L+2}) + (\bar{H}^{j-1})^T \bar{z}_{k-L+3}, \dots, \right.$$

$$\left. r(\bar{X}_k, u_k) + (\bar{H}^{j-1})^T \bar{z}_{k+1} \right]^T.$$

We next show the convergence of Algorithm 8.1.

**Theorem 8.3** *Under the stabilizability and detectability conditions on  $(\bar{A}, \bar{B})$  and  $(\bar{A}, \sqrt{\bar{Q}})$ , respectively, the iterative Q-learning Algorithm 8.1 generates a sequence of controls  $\{u_k^j, j = 1, 2, 3, \dots\}$  that converges to the optimal feedback controller given in (8.10) as  $j \rightarrow \infty$  if the rank condition (8.21) is satisfied.*

**Proof** The proposed iterative algorithm is based on the following Q-learning equation:

$$\begin{bmatrix} \bar{X}_k \\ u_k \end{bmatrix}^T H \begin{bmatrix} \bar{X}_k \\ u_k \end{bmatrix} = \bar{X}_k^T \bar{Q} \bar{X}_k + u_k^T R u_k + \begin{bmatrix} \bar{X}_{k+1} \\ u_{k+1} \end{bmatrix}^T H \begin{bmatrix} \bar{X}_{k+1} \\ u_{k+1} \end{bmatrix}.$$

Using the feedback policy  $\bar{K}$  and the definition (8.16), we have

$$Q_{\bar{K}}(\bar{X}_k, \bar{K}\bar{X}_k) = \bar{X}_k^T \bar{Q} \bar{X}_k + \bar{X}_k^T \bar{K}^T R \bar{K} \bar{X}_k + Q_{\bar{K}}(\bar{X}_{k+1}, \bar{K}\bar{X}_{k+1}).$$

Noticing that  $Q_{\bar{K}}(\bar{X}_k, \bar{K}\bar{X}_k) = V_{\bar{K}}(\bar{X}_k)$ , we have the following Bellman equation,

$$V_{\bar{K}}(\bar{X}_k) = \bar{X}_k^T \bar{Q} \bar{X}_k + \bar{X}_k^T \bar{K}^T R \bar{K} \bar{X}_k + V_{\bar{K}}(\bar{X}_{k+1}).$$

As the cost of following policy  $\bar{K}$  is quadratic in the state as given by (8.14), we have

$$\bar{X}_k^T \bar{P} \bar{X}_k = \bar{X}_k^T \bar{Q} \bar{X}_k + \bar{X}_k^T \bar{K}^T R \bar{K} \bar{X}_k + \bar{X}_{k+1}^T \bar{P} \bar{X}_{k+1},$$

or

$$\bar{X}_k^T \bar{P} \bar{X}_k = \bar{X}_k^T \bar{Q} \bar{X}_k + \bar{X}_k^T \bar{K}^T R \bar{K} \bar{X}_k + \bar{X}_k^T (\bar{A} + \bar{B}\bar{K})^T \bar{P} (\bar{A} + \bar{B}\bar{K}) \bar{X}_k,$$

which holds for all  $\bar{X}_k$ . We thus have the following Lyapunov equation:

$$(\bar{A} + \bar{B}\bar{K})^T \bar{P} (\bar{A} + \bar{B}\bar{K}) - \bar{P} + \bar{Q} + \bar{K}^T R \bar{K} = 0. \quad (8.25)$$

On the other hand, by the definition of matrix  $H$  in (8.16), the policy update step computes the policy  $\bar{K}$  as

$$\bar{K} = -(\bar{R} + \bar{B}^T \bar{P} \bar{B})^{-1} \bar{B}^T \bar{P} \bar{A}. \quad (8.26)$$

Substituting (8.26) in (8.25) results in the ARE (8.11). If the rank condition (8.21) holds then the value update Eq. (8.22) is uniquely solvable. Then, the iterations on the value

update and policy update steps corresponding to (8.22) and (8.23) are equivalent to the recursion on the following Riccati Difference Equation (RDE),

$$\bar{P}^{j+1} = \bar{A}^T \bar{P}^j \bar{A} + \bar{Q} - \bar{A}^T \bar{P}^j \bar{B} (\bar{R} + \bar{B}^T \bar{P}^j \bar{B})^{-1} \bar{B}^T \bar{P}^j \bar{A}.$$

The recursions on the RDE converge to the solution of the ARE under the stabilizability and detectability conditions of  $(\bar{A}, \bar{B})$  and  $(\bar{A}, \sqrt{\bar{Q}})$ , respectively [1, 5]. Thus, the iterative Q-learning algorithm converges to the optimal stabilizing solution  $\bar{K}^*$ . As a result, we have  $\bar{X}_k$  and  $x_k$  converging to zero as  $k \rightarrow \infty$ . This completes the proof.  $\square$

**Remark 8.4** Compared to the previous works [19, 20], the proposed scheme relaxes the assumption of the existence of a bicausal change of coordinates. Furthermore, unlike in [19, 20], the information of the state and input delays (both the number and lengths of the delays) is not needed in our proposed scheme.

## 8.5 Output Feedback Q-Learning Control of Time Delay Systems

The control design technique presented in the previous section was based on full state feedback. That is, access to the internal state was needed in the data-driven learning and the control law. However, in many applications, it is often the case that the measurement of the full internal state is not available but only a subset of the state space is measurable via system output. Output feedback techniques enable the design of control algorithms without involving the information of the full state. This section will present a Q-learning-based control algorithm to stabilize the system using only the measurements of the system output instead of the full state. We recall from [7] the following lemma, which allows the reconstruction of the system state by means of the delayed measurements of the system inputs and outputs.

**Lemma 8.1** *Under the assumptions of the observability of the pair  $(A, C)$ , the system state can be represented in terms of the measured input and output sequence as*

$$X_k = M_y Y_{k-1,k-N} + M_u u_{k-1,k-N}, \quad (8.27)$$

where  $N \leq n(S + 1) + mT$  is an upper bound on the system's observability index,  $u_{k-1,k-N} \in \mathbb{R}^{mN}$  and  $Y_{k-1,k-N} \in \mathbb{R}^{(p+m)N}$  are the input and augmented output data vectors defined as

$$u_{k-1,k-N} = [u_{k-1}^T \ u_{k-2}^T \ \cdots \ u_{k-N}^T]^T,$$

$$Y_{k-1,k-N} = [y_{k-1}^T \ u_{k-T-1}^T \ \cdots \ y_{k-N}^T \ u_{k-T-N}^T]^T,$$

$$M_y = A^N (V_N^T V_N)^{-1} V_N^T, \quad M_u = U_N - A^N (V_N^T V_N)^{-1} V_N^T T_N,$$

with

$$V_N = [(\mathcal{C}A^{N-1})^T \cdots (\mathcal{C}A)^T \mathcal{C}^T]^T,$$

$$U_N = [B \ AB \ \cdots \ A^{N-1}B],$$

$$T_N = \begin{bmatrix} 0 & \mathcal{C}B & CAB & \cdots & CA^{N-2}B \\ 0 & 0 & \mathcal{C}B & \cdots & CA^{N-3}B \\ \vdots & \vdots & \ddots & \ddots & \vdots \\ 0 & 0 & 0 & \cdots & \mathcal{C}B \\ 0 & 0 & 0 & \cdots & 0 \end{bmatrix}.$$

In Lemma 8.1,  $V_N$  and  $U_N$  are the observability and controllability matrices, respectively, and  $T_N$  is referred to as the Toeplitz matrix.

**Remark 8.5** The state parameterization (8.27) involves the invertability of the observability matrix. To ensure the observability of the extended augmented pair  $(\bar{\mathcal{A}}, \bar{\mathcal{C}})$ , we will assume in this section that  $\bar{S} = S$ .

Next, we aim to use the state parameterization (8.27) to describe the Q-function in (8.16). Notice that (8.27) can be written as

$$X_k = [M_u \ M_y] \begin{bmatrix} u_{k-1,k-N} \\ Y_{k-1,k-N} \end{bmatrix}. \quad (8.28)$$

Similarly, for the extended augmented system (8.7), we have the following parameterization of the extended augmented state,

$$\bar{X}_k = [\bar{M}_u \ \bar{M}_y] \begin{bmatrix} u_{k-1,k-N} \\ \bar{Y}_{k-1,k-N} \end{bmatrix}, \quad (8.29)$$

where  $\bar{Y}_{k-1,k-N}$ ,  $\bar{M}_u$ , and  $\bar{M}_y$  are formed using the extended augmented system matrices  $(\bar{A}, \bar{B}, \bar{\mathcal{C}})$ . It can be easily verified that substitution of (8.29) in (8.16) results in,

$$\begin{aligned} Q_{\bar{K}} &= \begin{bmatrix} u_{k-1,k-N} \\ \bar{Y}_{k-1,k-N} \\ u_k \end{bmatrix}^T \begin{bmatrix} \mathcal{H}_{\bar{u}\bar{u}} & \mathcal{H}_{\bar{u}\bar{y}} & \mathcal{H}_{\bar{u}u} \\ \mathcal{H}_{\bar{y}\bar{u}} & \mathcal{H}_{\bar{y}\bar{y}} & \mathcal{H}_{\bar{y}u} \\ \mathcal{H}_{uu} & \mathcal{H}_{uy} & \mathcal{H}_{uu} \end{bmatrix} \begin{bmatrix} u_{k-1,k-N} \\ \bar{Y}_{k-1,k-N} \\ u_k \end{bmatrix} \\ &\triangleq \xi_k^T \mathcal{H} \xi_k, \end{aligned} \quad (8.30)$$

where

$$\begin{aligned}\zeta_k &= [u_{k-1,k-N}^\top \bar{Y}_{k-1,k-N}^\top u_k^\top]^\top, \\ \mathcal{H} &= \mathcal{H}^\top \in \mathbb{R}^{(mN+(p+m)N+m) \times (mN+(p+m)N+m)},\end{aligned}$$

and the partitioned matrices are defined as

$$\begin{aligned}\mathcal{H}_{\bar{u}\bar{u}} &= \bar{M}_u^\top (\bar{Q} + \bar{A}^\top \bar{P} \bar{A}) \bar{M}_u \in \mathbb{R}^{mN \times mN}, \\ \mathcal{H}_{\bar{u}\bar{y}} &= \bar{M}_u^\top (\bar{Q} + \bar{A}^\top \bar{P} \bar{A}) \bar{M}_y \in \mathbb{R}^{mN \times (p+m)N}, \\ \mathcal{H}_{\bar{u}u} &= \bar{M}_u^\top \bar{A}^\top \bar{P} \bar{B} \in \mathbb{R}^{mN \times m}, \\ \mathcal{H}_{\bar{y}\bar{y}} &= \bar{M}_y^\top (\bar{Q} + \bar{A}^\top \bar{P} \bar{A}) \bar{M}_y \in \mathbb{R}^{(p+m)N \times (p+m)N}, \\ \mathcal{H}_{\bar{y}u} &= \bar{M}_y^\top \bar{A}^\top \bar{P} \bar{B} \in \mathbb{R}^{(p+m)N \times m}, \\ \mathcal{H}_{uu} &= R + \bar{B}^\top \bar{P} \bar{B} \in \mathbb{R}^{m \times m}.\end{aligned}\tag{8.31}$$

By (8.30) we have obtained a new description of the Q-function of the LQR in terms of the inputs and outputs of the system. We are now in a position to derive our output feedback LQR controller based on this Q-function. We seek to derive an optimal controller that minimizes the cost as expressed by the Q-function (8.30). To obtain the optimal controller, we perform the minimization of the optimal Q-function  $\mathcal{Q}_K^*$  with  $\mathcal{H}^*$  being the optimal  $\mathcal{H}$ . Setting

$$\frac{\partial}{\partial u_k} \mathcal{Q}_K^* = 0$$

and solving it for  $u_k$  result in our output feedback LQR control law,

$$u_k^* = -(\mathcal{H}_{uu}^*)^{-1} (\mathcal{H}_{u\bar{u}}^* u_{k-1,k-N} + \mathcal{H}_{u\bar{y}}^* \bar{Y}_{k-1,k-N}).\tag{8.32}$$

Now that we have an output feedback form of the Q-function for the extended augmented time delay system, the next step is to learn the optimal Q-function  $\mathcal{Q}_K^*$  and the corresponding output feedback optimal controller (8.32).

Consider the state feedback Q-learning equation (8.18). We employ the equivalent output feedback Q-function to write this equation as

$$\zeta_k^\top \mathcal{H} \zeta_k = \bar{Y}_k^\top \bar{Q}_y \bar{Y}_k + u_k^\top R u_k + \zeta_{k+1}^\top \mathcal{H} \zeta_{k+1}.\tag{8.33}$$

It should be noted that in the output feedback learning, we apply the user-defined weighting matrix  $\bar{Q}_y$  to the outputs. The term  $\bar{X}_k^\top \bar{Q} \bar{X}_k$  can be replaced with  $\bar{Y}_k^\top \bar{Q}_y \bar{Y}_k$  without requiring the knowledge of  $\bar{C}$  when  $\bar{Q} = \bar{C}^\top \bar{Q}_y \bar{C}$  and  $\bar{Y}_k = \bar{C} \bar{X}_k$ , where  $\bar{Y}_k$  is measurable. Here,  $u_{k+1}$  is computed as

$$u_{k+1} = -(\mathcal{H}_{uu})^{-1} (\mathcal{H}_{u\bar{u}} \bar{u}_{k,k-N+1} + \mathcal{H}_{u\bar{y}} \bar{Y}_{k,k-N+1}).\tag{8.34}$$

So, (8.33) is the Bellman equation for the Q-function in the output feedback form for which we will develop a reinforcement learning algorithm. Next, we will parameterize the Q-function in (8.30) so that we can separate the unknown matrix  $\mathcal{H}$ .

Consider the output feedback Q-function in (8.30) which can be linearly parametrized as

$$Q_{\bar{K}} = \bar{\mathcal{H}}^T \bar{z}_k, \quad (8.35)$$

where  $\bar{\mathcal{H}} = \text{vec}(\mathcal{H}) \in \mathbb{R}^{l(l+1)/2} \triangleq [\mathcal{H}_{11}, 2\mathcal{H}_{12}, \dots, 2\mathcal{H}_{1l}, \mathcal{H}_{22}, 2\mathcal{H}_{23}, \dots, 2\mathcal{H}_{2l}, \dots, \mathcal{H}_{ll}]^T$  is the vector that contains the upper triangular portion of matrix  $\mathcal{H}$ , where  $l = mN + (p+m)N + m$ . Since  $\mathcal{H}$  is symmetric, the off-diagonal entries are included as  $2\mathcal{H}_{ij}$ . The regression vector  $\bar{\zeta}_k \in \mathbb{R}^{l(l+1)/2}$  is defined by

$$\bar{\zeta}_k = \zeta_k \otimes \zeta_k,$$

which is the quadratic basis set formed as

$$\bar{\zeta} = [\zeta_1^2, \zeta_1 \zeta_2, \dots, \zeta_1 \zeta_l, \zeta_2^2, \zeta_2 \zeta_3, \dots, \zeta_2 \zeta_l, \dots, \zeta_l^2]^T.$$

With the parametrization (8.35), we have the following equation:

$$\bar{\mathcal{H}}^T \bar{\zeta}_k = \bar{Y}_k^T \bar{Q}_y \bar{Y}_k + u_k^T R u_k + \bar{\mathcal{H}}^T \bar{\zeta}_{k+1}. \quad (8.36)$$

Notice that (8.36) is a scalar equation with  $l(l+1)/2$  unknowns. We can solve this equation in the least squares sense by collecting at least  $L \geq l(l+1)/2$  datasets of  $\bar{Y}_k$  and  $u_k$ . The least squares solution of (8.19) is given by

$$\bar{\mathcal{H}} = (\Phi \Phi^T)^{-1} \Phi \Upsilon, \quad (8.37)$$

where  $\Phi \in \mathbb{R}^{l(l+1)/2 \times L}$  and  $\Upsilon \in \mathbb{R}^{L \times 1}$  are the data matrices defined as

$$\begin{aligned} \Phi &= [\bar{\zeta}_{k-L+1} - \bar{\zeta}_{k-L+2}, \quad \bar{\zeta}_{k-L+2} - \bar{\zeta}_{k-L+3}, \quad \dots, \quad \bar{z}_k - \bar{z}_{k+1}], \\ \Upsilon &= [r(\bar{Y}_{k-L+1}, u_{k-L+1}), \quad r(\bar{Y}_{k-L+2}, u_{k-L+2}), \quad \dots, \quad r(\bar{Y}_k, u_k)]^T. \end{aligned}$$

It is important to note that since  $u_k$  is linearly dependent on  $u_{k-1,k-N}$  and  $\bar{Y}_{k-1,k-N}$ , the least squares problem cannot be solved unless we inject an independent exploration signal  $v_k$  in  $u_k$  in order guarantee the invertibility of  $\Phi \Phi^T$  in (8.37). In other words, the following rank condition should hold

$$\text{rank}(\Phi) = l(l+1)/2. \quad (8.38)$$

**Remark 8.6** When  $N < \bar{T}$ ,  $\bar{Y}_{k-1,k-N}$  will contain entries from  $u_{k-1,k-N}$  that will prevent the rank condition (8.38) from being satisfied even in the presence of an exploration signal  $v_k$ . However, in the proof of Theorem 8.2, we see that increasing  $T$  to an arbitrary large  $\bar{T}$  does not affect the observability. Furthermore, the rank condition (8.12) corresponding to the original output  $y_k$  remains unchanged. This implies that the sum of the observability indices from the original output  $y_k$  also remains unchanged. Therefore, the augmented output vector  $u_{k-\bar{T}}$  must be contributing to the observability of the extended states. This causes the sum of the observability indices corresponding to the augmented output vector  $u_{k-\bar{T}}$  to increase to  $m\bar{T}$ , with each component contributing  $\bar{T}$  equally. Thus, for an arbitrarily large  $\bar{T}$ , the observability index becomes  $N = \bar{T}$  and the rank condition (8.38) can be satisfied.

We can now utilize iterative Q-learning to learn our output feedback Q-function matrix. In what follows, we present an iterative Q-learning algorithm to learn the optimal control parameters.

### Algorithm 8.2 Output Feedback Iterative Q-learning for Time Delay Systems

**Initialization.** Start with an arbitrary policy  $u_k^0 = v_k$  with  $v_k$  being the exploration signal. Set  $\mathcal{H}^0 \leftarrow I$ .

**Collect Data.** Apply the initial policy  $u^0$  to collect  $L$  datasets of  $(\bar{Y}_k, u_k)$  along with their quadratic terms.

**Value Update.** Compute the least squares solution of

$$(\bar{\mathcal{H}}^j)^T \bar{z}_k = \bar{Y}_k^T \bar{Q}_y \bar{Y}_k + u_k^T R u_k + (\bar{\mathcal{H}}^{j-1})^T \bar{\xi}_{k+1}. \quad (8.39)$$

**Policy Update.** Determine an improved policy using

$$u_k^{j+1} = -(\mathcal{H}_{uu}^j)^{-1} \left( \mathcal{H}_{u\bar{u}}^j u_{k-1,k-N} + \mathcal{H}_{u\bar{y}}^j \bar{Y}_{k-1,k-N} \right). \quad (8.40)$$

**Repeat and Stop.** Then, for  $j = 1, 2, \dots$ , repeat the steps in (8.22) and (8.23) until convergence,

$$\|\bar{\mathcal{H}}^j - \bar{\mathcal{H}}^{j-1}\| < \varepsilon, \quad (8.41)$$

for some small positive constant  $\varepsilon$ .

Algorithm 8.2 presents an output feedback iterative algorithm based on Q-learning. We apply an arbitrary initial control with an exploration signal  $v_k$  such that the rank condition (8.38) is satisfied. Based on this data, we solve the recursive Q-learning Bellman equation (8.39) in the value update step. In the next step, we perform a minimization of the Q-function with respect to  $u_k$ , which gives us a new policy. This policy will be evaluated in the next iteration. These iterations are carried out until we see no further updates in the estimates of the matrix  $\mathcal{H}$  for a sufficiently small positive constant  $\varepsilon$ . Once this criterion is met, the algorithm has converged to the (near) optimal solution. The data matrices in recursive iterations are given as

$$\Phi = [\bar{\xi}_{k-L+1}, \bar{\xi}_{k-L+2}, \dots, \bar{\xi}_k],$$

$$\Upsilon = \left[ r(\bar{Y}_{k-L+1}, u_{k-L+1}) + (\bar{\mathcal{H}}^{j-1})^\top \bar{\xi}_{k-L+2}, r(\bar{Y}_{k-L+2}, u_{k-L+2}) + (\bar{\mathcal{H}}^{j-1})^\top \bar{\xi}_{k-L+3}, \dots \right.$$

$$\left. r(\bar{Y}_k, u_k) + (\bar{\mathcal{H}}^{j-1})^\top \bar{\xi}_{k+1} \right]^\top.$$

We next show the convergence of Algorithm 8.2.

**Theorem 8.4** *Under the stabilizability and observability conditions on  $(\bar{A}, \bar{B})$  and  $(\bar{A}, \bar{C})$ , respectively, the iterative Q-learning Algorithm 8.2 generates a sequence of controls  $\{u_k^j, j = 1, 2, 3, \dots\}$  that converges to the optimal feedback controller given in (8.10) as  $j \rightarrow \infty$  if  $M = [\bar{M}_u \bar{M}_y]$  is of full row rank and the rank condition (8.38) is satisfied.*

**Proof** Consider the output feedback Q-learning equation

$$\zeta_k^\top \mathcal{H} \zeta_k = \bar{Y}_k^\top \bar{Q}_y \bar{Y}_k + u_k^\top R u_k + \zeta_{k+1}^\top \mathcal{H} \zeta_{k+1}.$$

Based on the definitions of  $\zeta_k$  and  $\mathcal{H}$  in (8.31), we have

$$\begin{aligned} \begin{bmatrix} \bar{u}_{k-1,k-N} \\ \bar{Y}_{k-1,k-N} \\ u_k \end{bmatrix}^\top [M \ I]^\top H [M \ I] \begin{bmatrix} \bar{u}_{k-1,k-N} \\ \bar{Y}_{k-1,k-N} \\ u_k \end{bmatrix} &= \bar{X}_k^\top \bar{Q} \bar{X}_k + u_k^\top R u_k \\ &+ \begin{bmatrix} \bar{u}_{k,k-N+1} \\ \bar{Y}_{k,k-N+1} \\ u_{k+1} \end{bmatrix}^\top [M \ I]^\top H [M \ I] \begin{bmatrix} \bar{u}_{k,k-N+1} \\ \bar{Y}_{k,k-N+1} \\ u_{k+1} \end{bmatrix}, \end{aligned}$$

where  $M = [\bar{M}_u \bar{M}_y]$  and  $\bar{Q} = \bar{C}^\top \bar{Q}_y \bar{C}$ .

Since  $(A, C)$  is observable, by Lemma 8.1, we have

$$\bar{X}_k = M [u_{k-1,k-N}^\top \bar{Y}_{k-1,k-N}^\top]^\top,$$

which results in the state feedback equation,

$$\begin{bmatrix} \bar{X}_k \\ u_k \end{bmatrix}^\top H \begin{bmatrix} \bar{X}_k \\ u_k \end{bmatrix} = \bar{X}_k^\top \bar{Q} \bar{X}_k + u_k^\top R u_k + \begin{bmatrix} \bar{X}_{k+1} \\ u_{k+1} \end{bmatrix}^\top H \begin{bmatrix} \bar{X}_{k+1} \\ u_{k+1} \end{bmatrix}.$$

Let

$$\bar{\mathcal{K}} = (\mathcal{H}_{uu}^j)^{-1} [\mathcal{H}_{uu}^j \ \mathcal{H}_{u\bar{y}}^j].$$

Then, by the definition of  $\mathcal{H}$ , we have  $\bar{\mathcal{K}} = M \bar{K}$  and the policy

$$u_k = \bar{\mathcal{K}} [u_{k-1,k-N}^\top \bar{Y}_{k-1,k-N}^\top]^\top$$

is equivalent to the state feedback policy  $u_k = \bar{K} \bar{X}_k$  based on  $\bar{K} = \bar{\mathcal{K}} M^\top (M M^\top)^{-1}$ . Using the definition (8.16), we have

$$\bar{Q}_{\bar{K}}(\bar{X}_k, \bar{K}\bar{X}_k) = \bar{X}_k^\top \bar{Q} \bar{X}_k + \bar{X}_k^\top \bar{K}^\top R \bar{K} \bar{X}_k + Q_{\bar{K}}(\bar{X}_{k+1}, \bar{K}\bar{X}_{k+1}),$$

Noticing that  $Q_{\bar{K}}(\bar{X}_k, \bar{K}\bar{X}_k) = V_{\bar{K}}(\bar{X}_k)$ , we have the following Bellman equation:

$$V_{\bar{K}}(\bar{X}_k) = \bar{X}_k^\top \bar{Q} \bar{X}_k + \bar{X}_k^\top \bar{K}^\top R \bar{K} \bar{X}_k + V_{\bar{K}}(\bar{X}_{k+1}).$$

As the cost of following policy  $\bar{K}$  is quadratic in the state as given by (8.14), we have

$$\bar{X}_k^\top \bar{P} \bar{X}_k = \bar{X}_k^\top \bar{Q} \bar{X}_k + \bar{X}_k^\top \bar{K}^\top R \bar{K} \bar{X}_k + \bar{X}_{k+1}^\top \bar{P} \bar{X}_{k+1},$$

or,

$$\bar{X}_k^\top \bar{P} \bar{X}_k = \bar{X}_k^\top \bar{Q} \bar{X}_k + \bar{X}_k^\top \bar{K}^\top R \bar{K} \bar{X}_k + \bar{X}_k^\top (\bar{A} + \bar{B} \bar{K})^\top \bar{P} (\bar{A} + \bar{B} \bar{K}) \bar{X}_k,$$

which holds for all  $\bar{X}_k$ . We thus have the following Lyapunov equation:

$$(\bar{A} + \bar{B} \bar{K})^\top \bar{P} (\bar{A} + \bar{B} \bar{K}) - \bar{P} + \bar{Q} + \bar{K}^\top R \bar{K} = 0. \quad (8.42)$$

By definition of  $\bar{K}$  and under the assumption of the full row rank of  $M$ , the policy update step in Algorithm 8.2 is equivalent to finding the policy

$$\bar{K} = (R + \bar{B}^\top \bar{P} \bar{B})^{-1} \bar{B}^\top \bar{P} \bar{A},$$

substitution of which in (8.42) results in the ARE (8.11). Under the rank condition (8.38), the least squares solution of the value update Eq. (8.39) can be obtained. Then, the iterations on the value update and policy update steps corresponding to (8.39) and (8.40) are equivalent to the recursions on the following Riccati Difference Equation (RDE),

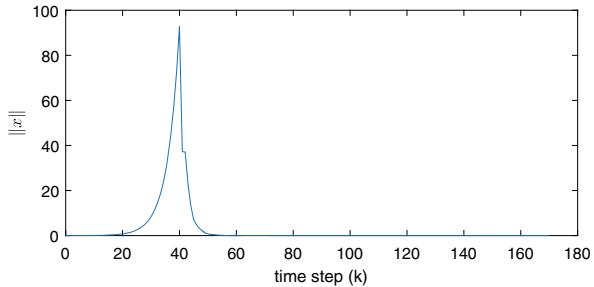
$$\bar{P}^{j+1} = \bar{A}^\top \bar{P}^j \bar{A} + \bar{Q} - \bar{A}^\top \bar{P}^j \bar{B} (R + \bar{B}^\top \bar{P}^j \bar{B})^{-1} \bar{B}^\top \bar{P}^j \bar{A}.$$

The recursions on the RDE converge to the solution of the ARE under the stabilizability and detectability of  $(\bar{A}, \bar{B})$  and  $(\bar{A}, \sqrt{\bar{Q}})$ , respectively [1, 5]. Thus, the output feedback iterative Q-learning algorithm converges to the optimal stabilizing solution. As a result, we have  $\bar{X}_k$  and  $x_k$  converging to zero as  $k \rightarrow \infty$ . This completes the proof.  $\square$

## 8.6 Simulation Results

In this section, we test the proposed scheme using numerical simulation. Consider the discrete-time system (8.1) with

**Fig. 8.1** Algorithm 8.1:  
State trajectory of the  
closed-loop system under  
state feedback



$$A_0 = \begin{bmatrix} 0.6 & 0.3 \\ 0.2 & 0.5 \end{bmatrix}, A_1 = \begin{bmatrix} 0.2 & 0.5 \\ 0.4 & 0.1 \end{bmatrix}, B_1 = \begin{bmatrix} 0.6 \\ 0.4 \end{bmatrix}, B_2 = \begin{bmatrix} 0.1 \\ -0.1 \end{bmatrix}, C = [1 \ 0.8].$$

There are two input delays and one state delay present in the system. Notice that, although the matrices  $A_0$  and  $A_1$  are both Schur stable, the combined system is unstable due to delays, which can be checked by finding the roots of the polynomial matrix  $P(\lambda) = A_0\lambda + A_1 - \lambda^2 I$  or by evaluating the eigenvalues of the augmented matrix  $A$  as defined in (8.5). It can be verified that the controllability condition of the combined delayed system  $\rho \left[ \sum_{i=0}^S A_i \lambda^{S-i} - \lambda^{S+1} I \ \sum_0^T B_i \lambda^{T-i} \right] = n$  holds. Hence, the extended augmented system (8.6) is controllable. The maximum input and state delays present in the system are  $T = 2$  and  $S = 1$ , respectively. We first validate the state feedback Algorithm 8.1. Let  $\bar{T} = 3$  and  $\bar{S} = 1$  be the upper bounds on the input and state delays, respectively. We specify the user-defined performance index as  $Q = I$  and  $R = 1$ . The nominal optimal feedback control matrix as obtained by solving the ARE with known delays is

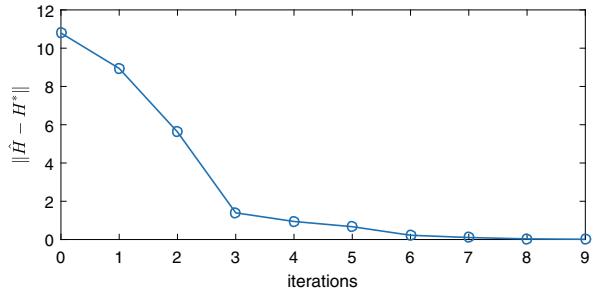
$$\hat{K}^* = [0.7991 \ 0.8376 \ 0.3622 \ 0.3643 \ 0.0007 \ 0.6191].$$

The convergence criterion of  $\varepsilon = 0.001$  was selected on the controller parameters. Since  $l = n(\bar{S} + 1) + m(\bar{T} + 1) = 8$ , we need at least  $l(l + 1)/2 = 36$  data samples to satisfy the rank condition (8.21) to solve (8.19). These data samples are collected by applying some sinusoidal signals of different frequencies and magnitudes in the control. The state response is shown in Fig. 8.1. It takes 9 iterations to converge to the optimal controller as shown in Fig. 8.2. It can be seen in Fig. 8.1 that controller is able to stabilize the unstable system. The final estimate of the control matrix is,

$$\hat{K} = [0.7996 \ 0.8380 \ 0.3625 \ 0.3645 \ 0 \ 0.0007 \ 0.6194].$$

It can be seen that the estimated control parameters correspond only to the actual delays present in the system while the one term corresponding to the extra state is 0. In other words, the final control is equal to the one obtained using the exact knowledge of the delays and system dynamics. Moreover, the rank condition (8.21) is no longer needed once the convergence criterion is met.

**Fig. 8.2** Algorithm 8.1:  
Convergence of the  
parameter estimates under  
state feedback



We next validate the output feedback Algorithm 8.2. It can be verified that the two observability conditions in Theorem 8.2 hold. We specify the user-defined performance index as  $\bar{Q}_y = I$  and  $R = 1$ . The bound on the state delay is the same but the bound on the input delay has been increased to  $\bar{T} = 4$  to make the observability index  $N = \bar{T}$  in order to satisfy the output feedback rank condition (8.38). The nominal output feedback optimal control parameters as obtained by solving the ARE with known delays are

$$\begin{aligned}\mathcal{H}_{u\bar{y}}^* &= [12.1348 \ -1.0313 \ 2.1086 \ 0 \ 1.9331 \ 0 \ 1.7188 \ 0], \\ \mathcal{H}_{u\bar{u}}^* &= [1.7316 \ 2.1151 \ -0.7301 \ -2.0765], \mathcal{H}_{uu}^* = 3.3954.\end{aligned}$$

The convergence criterion of  $\varepsilon = 0.001$  was selected on the controller parameters. Since  $l = mN + (p + m)N + m = 13$ , we need at least  $l(l + 1)/2 = 91$  data samples to satisfy the rank condition (8.21) to solve (8.36). These data samples are collected by applying some sinusoidal signals of different frequencies and magnitudes in the control. The state response is shown in Fig. 8.3. It takes around 18 iterations to converge to the optimal controller as shown in Fig. 8.4. It can be seen in Fig. 8.3 that controller is able to stabilize the unstable system. The final estimates of the control parameters are

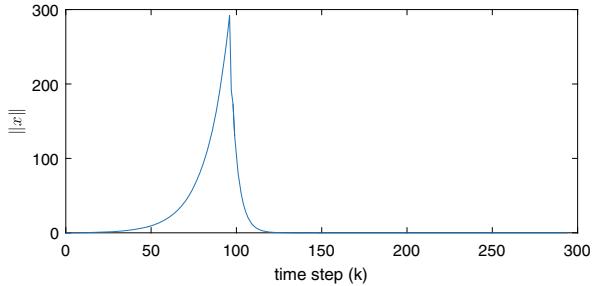
$$\begin{aligned}\hat{\mathcal{H}}_{u\bar{y}} &= [12.1077 \ -1.0289 \ 2.1040 \ 0 \ 1.9287 \ 0 \ 1.7149 \ 0], \\ \hat{\mathcal{H}}_{u\bar{u}} &= [1.7277 \ 2.1102 \ -0.7286 \ -2.0719], \hat{\mathcal{H}}_{uu} = 3.3927.\end{aligned}$$

As can be seen in the state feedback and output feedback results, while the transient performance of the state feedback algorithm is superior due to fewer unknown parameters to be learnt, the output feedback algorithm has the advantage that it does not require the measurement of the full state feedback.

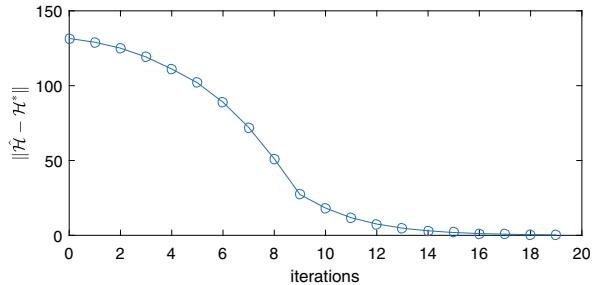
## 8.7 Conclusions

In this chapter, we presented an application of reinforcement learning to solve the model-free optimal stabilization problem for linear time delay systems with mul-

**Fig. 8.3** Algorithm 8.2:  
State trajectory of the  
closed-loop system under  
output feedback



**Fig. 8.4** Algorithm 8.1:  
Convergence of the  
parameter estimates under  
output feedback



tiple unknown state and input delays and unknown system dynamics. To handle the unknown delays, an extended state augmentation approach was presented that transforms the system into an extended delay-free system. The upper bounds on the state and input delays were used in the augmentation process, in which the delayed state and control inputs were treated as additional virtual states of the systems. The transformation of the system to the extended delay-free form does not require any bicausal change of coordinates, as needed in the previous works. Furthermore, it was shown that the controllability property of the system is also preserved in the extended augmentation process. State feedback Q-learning scheme was employed to learn the optimal control parameters. In the second part of this chapter, we focused on the design of the output feedback Q-learning controller in order to circumvent the requirement of full state for feedback. Observability conditions of the augmented and extended augmented systems were presented. Unlike the controllability result, it was found that while the augmented delay-free system (with known maximum state delay and input delay) can recover observability, the extended augmented systems (with unknown maximum state delay and input delay) can achieve at most detectability. Under the special case when the maximum state delay is known, the extended augmented system becomes observable even when the maximum input delay is unknown. An output feedback Q-function was presented based on the parameterization of the state vector using delayed input and output measurements. An iterative Q-learning scheme was presented to learn the optimal output feedback Q-function using only the measured input and output data. Simulation results were used to demonstrate the effectiveness of both the state feedback and the output feedback algorithms.

## References

1. Caines, P.E., Mayne, D.Q.: On the discrete time matrix riccati equation of optimal control. *Int. J. Control* **12**(5), 785–794 (1970)
2. Gao, W., Jiang, Z.P.: Adaptive dynamic programming and adaptive optimal output regulation of linear systems. *IEEE Trans. Autom. Control* **61**(12), 4164–4169 (2016)
3. Gu, K., Chen, J., Kharitonov, V.L.: Stability of time-delay systems. Springer Science & Business Media, Berlin (2003)
4. Jiang, Y., Jiang, Z.P.: Computational adaptive optimal control for continuous-time linear systems with completely unknown dynamics. *Automatica* **48**(10), 2699–2704 (2012)
5. Lancaster, P., Rodman, L.: Algebraic Riccati Equations. Clarendon press, Oxford (1995)
6. Lewis, F.L., Syrmos, V.L.: Optimal Control. Wiley, Hoboken (1995)
7. Lewis, F.L., Vamvoudakis, K.G.: Reinforcement learning for partially observable dynamic processes: adaptive dynamic programming using measured output data. *IEEE Trans. Syst. Man Cybern. Part B (Cybern.)* **41**(1), 14–25 (2011)
8. Lewis, F.L., Vrabie, D., Vamvoudakis, K.G.: Reinforcement learning and feedback control: using natural decision methods to design optimal adaptive controllers. *IEEE Control Syst. Mag.* **32**(6), 76–105 (2012)
9. Liu, Y., Zhang, H., Luo, Y., Han, J.: ADP based optimal tracking control for a class of linear discrete-time system with multiple delays. *J. Frankl. Inst.* **353**(9), 2117–2136 (2016)
10. Manitius, A., Olbrot, A.: Finite spectrum assignment problem for systems with delays. *IEEE Trans. Autom. Control* **24**(4), 541–552 (1979)
11. Mee, D.: An extension of predictor control for systems with control time-delays. *Int. J. Control* **18**(6), 1151–1168 (1973)
12. Mu, C., Ni, Z., Sun, C., He, H.: Data-driven tracking control with adaptive dynamic programming for a class of continuous-time nonlinear systems. *IEEE Trans. Cybern.* **47**(6), 1460–1470 (2017)
13. Rizvi, S.A.A., Lin, Z.: Output feedback Q-learning control for the discrete-time linear quadratic regulator problem. *IEEE Trans. Neural Netw. Learn. Syst.* **30**(5), 1523–1536 (2018)
14. Smith, O.J.: Closed control of loop with dead time. *Chem. Eng. Prog.* **53**, 217–219 (1957)
15. Wang, D., He, H., Liu, D.: Adaptive critic nonlinear robust control: a survey. *IEEE Trans. Cybern.* **47**(10), 3429–3451 (2017)
16. Wang, F.Y., Zhang, H., Liu, D.: Adaptive dynamic programming: an introduction. *IEEE Comput. Intell. Mag.* **4**(2), 39–47 (2009)
17. Wang, Z., Liu, X., Liu, K., Li, S., Wang, H.: Backstepping-based lyapunov function construction using approximate dynamic programming and sum of square techniques. *IEEE Trans. Cybern.* **47**(10), 3393–3403 (2017)
18. Zhang, H., Liang, H., Wang, Z., Feng, T.: Optimal output regulation for heterogeneous multiagent systems via adaptive dynamic programming. *IEEE Trans. Neural Netw. Learn. Syst.* **28**(1), 18–29 (2017)
19. Zhang, H., Liu, Y., Xiao, G., Jiang, H.: Data-based adaptive dynamic programming for a class of discrete-time systems with multiple delays. In: *IEEE Transactions on Systems, Man, and Cybernetics: Systems*, pp. 1–10 (2017)
20. Zhang, J., Zhang, H., Luo, Y., Feng, T.: Model-free optimal control design for a class of linear discrete-time systems with multiple delays using adaptive dynamic programming. *Neurocomputing* **135**, 163–170 (2014)
21. Zhong, X., Ni, Z., He, H.: Gr-GDHP: a new architecture for globalized dual heuristic dynamic programming. *IEEE Trans. Cybern.* **47**(10), 3318–3330 (2017)

## Chapter 9

# Optimal Adaptive Control of Partially Uncertain Linear Continuous-Time Systems with State Delay



Rohollah Moghadam, S. Jagannathan, Vignesh Narayanan,  
and Krishnan Raghavan

**Abstract** In this chapter, the optimal adaptive control (OAC) of partially unknown linear continuous-time systems with state delay is introduced by using integral reinforcement learning (IRL). A quadratic cost function over infinite time horizon is considered and a value function is defined by considering the delayed state. A delay algebraic Riccati equation (DARE) is obtained using the value functional and utilized along with stationarity condition to obtain optimal control policy. Subsequently, it has been shown that the optimal control policy renders the closed-loop system asymptotically stable under mild conditions when the dynamics are known. Then, to overcome the need for drift dynamics, an actor–critic framework using the IRL approach is introduced for OAC. A novel update law for tuning the parameters of the critic function is derived. Lyapunov theory is employed to demonstrate the boundedness of the closed-loop system when the OAC uses periodic and event sampled feedback. Finally, future work involving an image sensor as part of the OAC loop using deep neural network based RL is reported. A simulation example is included to illustrate the effectiveness of the proposed approach.

---

R. Moghadam · S. Jagannathan (✉)  
Arkansas Tech University, 255 Corley Hall,  
1811 N Boulder Ave, Russellville, AR, 72801, USA  
e-mail: [sarangap@mst.edu](mailto:sarangap@mst.edu)

R. Moghadam  
e-mail: [rmoghadam@atu.edu](mailto:rmoghadam@atu.edu)

V. Narayanan  
Washington University in St. Louis, 1 Brookings Dr, St., Louis, MO, USA  
e-mail: [vignesh.narayanan@wustl.edu](mailto:vignesh.narayanan@wustl.edu)

K. Raghavan  
Argonne National Laboratory, 9700 S Cass Ave, Lemont, IL, USA  
e-mail: [krm9c@mst.edu](mailto:krm9c@mst.edu)

## 9.1 Introduction

There has been a significant interest in the development of network control systems (NCS) [1–4] to integrate a physical system and a communication network in order to improve the system performance while reducing the cost. In an NCS, a communication network is placed within the feedback loop to enclose the sensor, actuator, and controller. Due to the communication network, network imperfections such as packet dropouts [5] and the network-induced delays [6] severely degrade the performance of the physical system. Since the delay arising through the communication network affects the system state vector, the study of the delay differential equations becomes an important area of interest to the control scientists [7–17]. Other notable application where delay effects are observed include human–machine integration [18]. Existing optimal control schemes for linear systems without delays are unsuitable for such time-delayed linear systems. Therefore, design of optimal control schemes for such time-delayed systems are necessary.

The optimal control of dynamical systems with state delay has been considered extensively in the literature for both tracking and regulation [19–22] given the system dynamics. In [20, 21], an optimal regulator is studied for linear time-varying systems with state delay and with multiple delays in control input by minimizing a quadratic cost function. On the other hand, in [22], an optimal linear quadratic regulator is proposed for NCS with network delay. The optimal price-based bi-variable decision-making problem is considered for a single-loop stochastic linear NCS. However, in many practical industrial systems, complete and accurate knowledge of the system dynamics is not possible and dynamic uncertainties can affect the closed-loop performance.

To overcome the need for system dynamics, a reinforcement learning (RL) scheme [26, 27] can be utilized for obtaining optimal control policy for such time-delayed systems. In particular, the integral reinforcement learning (IRL) approach [23] has been successfully employed to learn optimal control of both continuous and discrete time systems in the absence of any delays and without requiring the knowledge of the system dynamics [24, 25]. The IRL uses the Bellman equation to find the optimal solution of a cost function online with/without an actor–critic structure. Despite relaxing the need for system dynamics, the traditional IRL requires continuous availability of state vector for control, thus increasing the communication cost in an NCS due to the transmission of feedback and control policy through the communication network.

The event-triggered control techniques [28, 30–34], on the other hand, have been introduced to reduce the communication cost in an NCS. The main aim of the event-triggered controllers is to use an aperiodic mechanism for the transmission of feedback and control policies in an NCS without causing the overall system to become unstable. In the case of zero-order hold (ZOH)-based event-triggered control [28], the controller and the actuators are equipped with a ZOH device to hold the control input and the state vector, received from the controller and the sensor, respectively, between two successive feedback instants or events. An event-triggering condition

is derived and if violated, an event is initiated. In these event sampled approaches, a controller is first designed and the threshold required for the even triggering condition is found based on the closed-loop stability. The event sampled approach is similar to the RL agent interacting with the environment in an aperiodic manner.

This chapter presents optimal adaptive control (OAC) of partially unknown linear continuous-time systems with state delay by using IRL. First a quadratic cost function over infinite time horizon is considered and a value function is defined by considering the delayed state of the system. Then the Hamiltonian is formulated and, using the stationarity condition, the optimal control is determined when the system matrices are known. A delay algebraic Riccati equation (DARE) is derived to guarantee the stability of the closed-loop system under the assumption that the system is controllable and observable.

Subsequently, an actor–critic framework is utilized using the IRL approach in order to relax system dynamics for control policy generation. Bellman error, derived from the difference between the actual and approximated value function, is utilized to find the update law for the parameters of the critic function. Lyapunov theory is employed to demonstrate boundedness of the state vector and the parameter estimation errors. Both periodic and event sampled OAC of time-delayed system are included. In order to enhance the optimality, a novel hybrid scheme, where time-varying number of iterations can be incorporated within the event sampled interval, is introduced.

Note that existing optimal control approaches for time-delayed systems require complete and accurate information about the drift dynamics. Moreover, one has to solve a complex partial differential equation to find the optimal solution. In contrast, in this chapter, the optimal control policy is derived without needing to solve such partial differential equations analytically.

In general, neural networks (NN) are utilized for approximating the critic in the IRL approach for nonlinear systems. It is important to note that the NN utilized for generating the optimal control policy are normally shallow with one hidden layer. Finally, perspectives on controller design with image feedback as part of the OAC loop using deep NN-based RL is covered. Throughout the chapter,  $\|\cdot\|$  denotes the Euclidean vector norm,  $I_n$  is the  $n \times n$  identity matrix. The Kronecker product of matrices  $A$  and  $B$  is defined by  $A \otimes B$ . The transpose of the matrix  $A$  is denoted as  $A^\top$ . Minimum and maximum singular values of the matrix  $A$  are represented by  $\delta_{min}(A)$  and  $\delta_{max}(A)$ , respectively. The minimum and maximum eigenvalue of the matrix  $A$  is defined as  $\lambda_{min}(A)$  and  $\lambda_{max}(A)$ , respectively.

## 9.2 Problem Statement

In this section, the problem of finding the optimal control of linear continuous-time systems with state delay is formulated under the assumption that the system is controllable. A quadratic cost function with respect to state and control input is

defined. In this chapter, the optimal control policy is obtained by assuming that the system dynamics are available first and then relaxed by using an RL approach.

Consider a linear continuous time-invariant system with state delay given by

$$\begin{cases} \dot{x}(t) = A_0 x(t) + Ax(t - d_x) + Bu(t) \\ x(\theta) = \varphi(\theta) \quad \theta \in [-d_x, 0] \\ x(0) = \varphi(0) \end{cases}, \quad (9.1)$$

where  $x(t) \in \mathbb{R}^n$ ,  $u(t) \in \mathbb{R}^m$  denote the state and control input,  $A_0, A \in \mathbb{R}^{n \times n}$  and  $B \in \mathbb{R}^{n \times m}$  represent drift dynamics and input matrix, respectively, with  $\varphi(\cdot)$  being a continuously differentiable initial function and  $d_x$  is a known constant time delay.

**Assumption 9.1** The linear continuous-time system (9.1) with state delay is controllable and observable. In addition, the state vector is available for feedback control [3].

Next, the problem statement is introduced. Consider the linear continuous-time system with state delay defined in (9.1). Design a control policy  $u(t) = -Kx(t)$ , where  $K \in \mathbb{R}^{m \times n}$  is a user-defined gain matrix which will be defined later, such that the closed-loop system is asymptotically stable, i.e.,  $\lim_{t \rightarrow \infty} x(t) \rightarrow 0$ , while minimizing the quadratic cost function over infinite horizon given by

$$V(x(t)) = \int_t^\infty (x(\tau)^\top \tilde{Q}x(\tau) + u(\tau)^\top Ru(\tau)) d\tau, \quad (9.2)$$

where  $\tilde{Q}$  and  $R \in \mathbb{R}^{m \times m}$  represent user-defined positive definite matrices. It has been shown in [19] that the optimal control can be obtained using the following Bellman type equation (BTE) given by

$$\min_u \underbrace{\left\{ \frac{\partial V(x)}{\partial t} + x^\top \tilde{Q}x + u^\top Ru \right\}}_{BTE} = 0, \quad (9.3)$$

where  $\partial V(x)/\partial t$  denotes the time derivative of the value function along the system dynamics (9.1). In the next section, the BTE (9.3) is utilized to find the optimal control policy for the linear continuous-time system (9.1) with state delay, when the dynamics of the system are known.

### 9.3 Linear Quadratic Regulator Design

In this section, an optimal control approach is presented for the linear continuous-time system (9.1) with state delay, using the state feedback. First, a Lyapunov–Krasovskii

function is introduced. Then, using the BTE, the optimal control policy is derived using the stationarity condition. A DARE is defined and, using the Lyapunov analysis, it is shown that closed-loop system using the optimal control policy is asymptotically stable. Two different controllers are proposed. First, it is assumed that the state vector of the system is sampled periodically and, then, an event sampled controller is introduced. In the second case, the triggering threshold is derived using the Lyapunov analysis.

### 9.3.1 Periodic Sampled Feedback

In this subsection, the optimal controller is developed by considering that the state vector of the system is measurable periodically. This assumption is relaxed in the next subsection. The following Lemma is required in the rest of the paper.

**Lemma 9.1** ([35]) *Let  $M$  be a positive semi-definite square matrix and  $N$  be a positive definite square matrix, i.e.,  $v^\top M v \geq 0$  and  $v^\top N v > 0$  for any vector  $v \neq 0$ . Then,  $M + N$  is a positive definite square matrix.*

To proceed, let the following Lyapunov–Krasovskii function be considered as the value function  $V(x(t))$  defined in (9.2)

$$V(x(t)) = x(t)^\top P x(t) + \alpha \int_{t-d_x}^t x(s)^\top x(s) ds, \quad (9.4)$$

where  $P \in \mathbb{R}^{n \times n}$  and  $\alpha > 0$  are a symmetric positive definite matrix and a positive scalar, respectively. Now, taking the first derivative of the value function (9.4) by using Leibniz integral formula and substituting into the BTE (9.3), we get

$$\begin{aligned} \min(A_0 x + Ax(t - d_x) + Bu)^\top Px + x^\top P(A_0 x \\ + Ax(t - d_x) + Bu) + \alpha x^\top x \\ - \alpha x^\top (t - d_x)x(t - d_x) + x^\top \tilde{Q}x + u^\top Ru \end{aligned} \quad . \quad (9.5)$$

To find the optimal control policy, the stationarity condition [36] is utilized for the above equation as  $\partial(BTE)/\partial u = 0$  which results in  $2Ru + 2B^\top Px = 0$ . Therefore, the optimal control policy becomes

$$u^*(t) = -Kx(t) = -R^{-1}B^\top Px(t), \quad (9.6)$$

where  $K$  is the controller gain matrix and  $P$  is the solution to the DARE given by

$$A_0^\top P + PA_0 - P(2BR^{-1}B^\top - \frac{1}{\alpha}AA^\top)P + \tilde{Q} = 0, \quad (9.7)$$

with  $R > 0$  and  $\tilde{Q} = \alpha I_n + Q$ ,  $Q \in \mathbb{R}^{n \times n}$  being a user-defined positive definite matrix. By using Lemma 9.2, one can show that  $\tilde{Q}$  is a positive definite matrix. The Riccati type Eq. (9.7) is termed as DARE because of the presence of the delayed state dynamics, i.e.,  $A$ , in the equation. The discussion of existence and uniqueness of the solution to the DARE (9.7) is presented in [37] when the linear time-delayed system is controllable and observable. From (9.7), one can see that the derived Riccati equation for the time delay system (9.1) is similar to game algebraic Riccati equation (GARE) of the linear continuous-time systems without delay and in the presence of a disturbance.

In the following theorem, it is shown that the optimal control input (9.6) makes the system asymptotically stable in the presence of the state delay while minimizing the cost function (9.2). Before we proceed, the following lemma is required.

**Lemma 9.2** ([38]) Consider the following matrix

$$\mathcal{L} = \begin{bmatrix} \mathcal{L}_{11} & \mathcal{L}_{12} \\ \mathcal{L}_{12}^\top & \mathcal{L}_{22} \end{bmatrix},$$

where  $\mathcal{L}_{12}^\top$  denotes the transpose of  $\mathcal{L}_{12}$ . Then,  $\mathcal{L}$  is a positive definite matrix, if and only if, one of the following conditions is satisfied

$$\begin{cases} \mathcal{L}_{22} > 0 \text{ and } \mathcal{L}_{11} - \mathcal{L}_{12}\mathcal{L}_{22}^{-1}\mathcal{L}_{12}^\top > 0 \\ \mathcal{L}_{11} > 0 \text{ and } \mathcal{L}_{22} - \mathcal{L}_{12}^\top\mathcal{L}_{11}^{-1}\mathcal{L}_{12} > 0 \end{cases}.$$

**Theorem 9.1** Consider the linear continuous-time system (9.1) with the constant state delay. Let the cost function (9.2) along with the value function (9.4) be considered and Assumptions 1 and 2 be satisfied. Assume there exists a positive definite matrix  $P$  as the solution to the DARE (9.7). Then, the optimal control input (9.6) renders the closed-loop system asymptotically stable while minimizing the cost function (9.2).

**Proof** Consider the value function (9.4) as a Lyapunov–Krasovskii candidate function. Taking the derivative of  $V$  along system dynamics (9.1) gives

$$\dot{V}(x(t)) = \dot{x}(t)^\top Px(t) + x(t)^\top P\dot{x}(t) + \alpha \frac{d}{dt} \int_{t-d_x}^t x^\top(s)x(s)ds, \quad (9.8)$$

which by using the Leibniz integral formula gives

$$\begin{aligned} \dot{V}(x(t)) &= (A_0x(t) + Ax(t - d_x) + Bu(t))^\top Px(t) \\ &\quad + x^\top P(A_0x(t) + Ax(t - d_x) + Bu(t)) \\ &\quad + \alpha x^\top(t)x(t) - \alpha x^\top(t - d_x)x(t - d_x) \end{aligned} \quad . \quad (9.9)$$

Substituting the optimal control policy (9.6) into (9.9) and after some manipulations, one has

$$\begin{aligned}\dot{V}(x(t)) &= x^\top(t) A_0^\top P x + x^\top(t - d_x) A^\top P x(t) - x^\top P B R^{-1} B^\top P x \\ &\quad + x^\top(t) P A_0 x(t) + x^\top(t) P A x(t - d_x) - x^\top P B R^{-1} B^\top P x \\ &\quad + \alpha x^\top(t) x(t) - \alpha x^\top(t - d_x) x(t - d_x).\end{aligned}\quad (9.10)$$

Stacking up the vector of the non-delayed and delayed state of the system to get

$$\begin{aligned}\dot{V}(x(t)) &= -\left[ \begin{matrix} x(t) & x(t - d_x) \end{matrix} \right]^\top \\ &\quad \left[ \begin{matrix} -A_0^\top P - P A_0 - \alpha I_n + 2PBR^{-1}B^\top P - PA \\ -A^\top P \end{matrix} \right] \left[ \begin{matrix} x(t) \\ x(t - d_x) \end{matrix} \right].\end{aligned}\quad (9.11)$$

Now, by using Lemma 9.2 and the DARE (9.7), one can show that the derivative of the Lyapunov–Krasovskii function (9.4) is negative definite, i.e.,  $\dot{V} < 0$ , which results in the asymptotic stability of the linear continuous-time system (9.1) in the presence of the state delay. Moreover, it has been shown in [36] that the necessary condition for finding the optimal policy is to use the stationarity condition  $\partial(BTE)/\partial u = 2B^\top Px + 2Ru = 0$ , which shows that the control input (9.6) is optimal.  $\square$

Next, we will present the control policy in the presence of event sampled feedback when the dynamics of the system are known. We will derive an event-trigger condition to derive sampling instants so as to keep the closed-loop system stable.

### 9.3.2 Event Sampled Feedback

In this subsection, the optimal event-trigger control approach is presented for the linear continuous-time system (9.1) with state delay. The event-triggered approach [28] eliminates the need for continuous availability of the system states which definitely reduces the use of the communication resources and decreases the communication cost.

Let the aperiodic sampling instants be defined as  $\{t_k\}_{k=0}^\infty$  with  $t_0 = 0$ . Then, the control input applies to the system at each sampling instant becomes  $u(t_k)$  which is held by the ZOH to make a continuous-time control input. The structure of the event-triggered controller is shown in Fig. 9.1.

Since the control input applied to the system is updated at the triggering instants, the dynamics of the linear continuous-time system (9.1) with state delay and considering the control applies at the sampling instants can be written as

$$\dot{x}(t) = A_0 x(t) + A x(t - d_x) + B u(t_k). \quad (9.12)$$

Next, using the optimal control policy (9.6), the piece-wise continuous control policy using the sample state  $x(t_k)$  at  $t = t_k$  becomes

$$u^*(t_k) = -Kx(t_k) = -R^{-1}B^\top Px(t_k). \quad (9.13)$$

Define the state measurement error as

$$e(t) = x(t_k) - x(t). \quad (9.14)$$

Then, the system dynamic (9.12) with the optimal control policy (9.13), in terms of the measurement error and the system state  $x(t)$ , can be represented by

$$\dot{x}(t) = A_0x(t) + Ax(t - d_x) - BR^{-1}B^\top Px(t) - BR^{-1}B^\top Pe(t). \quad (9.15)$$

In the following theorem, the Lyapunov analysis is employed to derive the triggering condition for the linear continuous-time system (9.1) with state delay while ensuring the asymptotic stability of the closed-loop system.

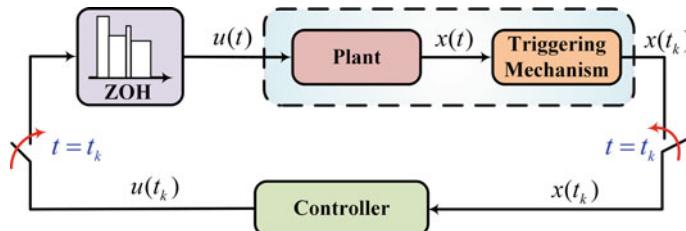
**Theorem 9.2** Consider the linear continuous-time system (9.1) with state delay. Let Assumptions 1 and 2 be satisfied and the positive definite matrix  $P$  be the solution to the DARE (9.7). Then, the control input (9.13) renders the closed-loop system asymptotically stable under the assumption of input-to-state stability (ISS) provided the sampled instants are defined using the following condition given by

$$\|e(t)\| < \eta \frac{\lambda_{\min}(Q)}{\|2PBR^{-1}B^\top P\|} \|x(t)\| \quad (9.16)$$

where  $0 < \eta < 1$  and  $\eta \frac{\lambda_{\min}(Q)}{\|2PBR^{-1}B^\top P\|}$  is the triggering threshold.

**Proof** Consider the value function (9.4) as a Lyapunov–Krasovskii candidate. Taking the derivative of  $V$  along system trajectory gives

$$\dot{V}(x(t)) = \dot{x}^\top Px + x^\top P\dot{x} + \alpha \frac{d}{dt} \int_{t-d_x}^t x^\top(s)x(s)ds. \quad (9.17)$$



**Fig. 9.1** The event-triggered controller scheme

Substituting the system dynamics (9.15) into (9.17) and using the Leibniz integral formula for the last term, one has

$$\begin{aligned}\dot{V} = & (A_0x(t) + Ax(t - d_x) - BR^{-1}B^\top Px(t) - BR^{-1}B^\top Pe(t))^\top Px \\ & + x^\top P(A_0x(t) + Ax(t - d_x) - BR^{-1}B^\top Px(t) - BR^{-1}B^\top Pe(t)) , \quad (9.18) \\ & + \alpha x^\top(t)x(t) - \alpha x^\top(t - d_x)x(t - d_x)\end{aligned}$$

which yields

$$\begin{aligned}\dot{V} = & x^\top (A_0^\top P + PA_0 - 2PBR^{-1}B^\top P + \alpha I) x \\ & + 2x(t - d_x)^\top A^\top Px - 2e^\top PBR^{-1}B^\top Px - \alpha x^\top(t - d_x)x(t - d_x) . \quad (9.19)\end{aligned}$$

Simplifying the cross product terms by adding and subtracting  $\frac{1}{\alpha}PAA^\top P$ , Eq. (9.19) gives

$$\begin{aligned}\dot{V} = & x^\top \left( A_0^\top P + PA_0 - 2PBR^{-1}B^\top P + \alpha I + \frac{1}{\alpha}PAA^\top P \right) x \\ & - 2e^\top PBR^{-1}B^\top Px - \frac{1}{\alpha} [A^\top Px - \alpha x(t - r)]^\top [A^\top Px - \alpha x(t - r)] , \quad (9.20)\end{aligned}$$

which by substituting the DARE (9.7) results in

$$\begin{aligned}\dot{V} = & -x^\top Qx - 2e^\top PBR^{-1}B^\top Px \\ & - \frac{1}{\alpha} [A^\top Px - \alpha x(t - r)]^\top [A^\top Px - \alpha x(t - r)] . \quad (9.21)\end{aligned}$$

Next, using the norm properties, we get

$$\begin{aligned}\dot{V} \leq & -\lambda_{\min}(Q)\|x\|^2 + \|2PBR^{-1}B^\top P\| \|e\| \|x\| \\ & \leq -\|x\| (\lambda_{\min}(Q)\|x\| - \|2PBR^{-1}B^\top P\| \|e\|) , \quad (9.22)\end{aligned}$$

which by using the triggering condition (9.16) becomes

$$\dot{V} < -(1 - \eta)\lambda_{\min}(Q)\|x\|^2 . \quad (9.23)$$

From (9.23) and since  $0 < \eta < 1$ , one can conclude that the derivative of the Lyapunov–Krasovskii function is negative definite, i.e.,  $\dot{V} < 0$ , which results in asymptotic stability of the system, if and only if (9.16) and ISS holds.

It is shown in Theorem 9.1 that when the state vector of the system is continuously available, the control input (9.6) makes the system asymptotically stable. Then, an event-triggered controller, which is proposed, is proven in Theorem 9.2 that the control policy (9.13) along with the triggering condition (9.16), guarantees asymptotic stability of the system without requiring the system states to be available continu-

ously. However, to compute both control inputs, one need to solve the DARE (9.7) which requires a complete and accurate knowledge of the internal system dynamics as well as input dynamics. Due to system uncertainties, having a complete knowledge of the system dynamics is not feasible in many practical applications. To overcome this drawback, in the next section, a learning-based control is proposed to eliminate the need of the internal system dynamics in computing the control input.

## 9.4 Optimal Adaptive Control

In this section, the IRL technique is implemented online based on the actor–critic framework [23]. The technique has two steps—critic and actor update. In IRL, the state vector is measured at specific sample times  $t + i \Delta t$  with  $i \in \mathbb{N}$ . The critic function is updated by computing the IRL Bellman equation corresponding to the current stabilizing controller. Then, actor parameters are updated using the critic function. The process of updating the critic parameters does not require any knowledge of the drift dynamics since the measured information from the system is utilized to obtain the critic structure. A novel Bellman error is introduced which contains the history of the state delay. The critic update law is obtained using the Bellman error. It has been demonstrated that to confirm the convergence of the estimated critic to the actual parameters, a persistency of excitation condition (PE) is required.

### 9.4.1 Periodic Sampled Feedback

In this subsection, to find the optimal control for the linear continuous-time system (9.1) with state delay without requiring the internal dynamics, the actor–critic framework is implemented under the assumption that the state of the system is available continuously. Then, in subsequent sections, this assumption is relaxed by using event-triggered approach.

To proceed, the Bellman equation corresponding to the value function (9.4) is given by [23]

$$V(x(t)) = \int_t^{t+\Delta t} \left( x^\top \tilde{Q} x + u^\top R u \right) d\tau + V(x, t + \Delta t), \quad (9.24)$$

where  $\Delta t$  is a fixed and small time interval. The value function (9.4) can be written as

$$V(x(t)) = \theta^\top \sigma(x) + \alpha \int_{t-d_x}^t x^\top(s) x(s) ds \quad (9.25)$$

with  $\theta = vech(P)^\top \in \mathbb{R}^{\frac{1}{2}n(n+1)}$ , where  $vech(P)$  means making a vector of the symmetric matrix  $P$ . Moreover,  $\sigma(x) = x \otimes x$  denotes the regression function [35].

The ideal parameters for the critic function  $V^*$  are unknown, therefore, one must approximate the parameters as

$$\hat{V}(x(t)) = \hat{\theta}^\top \sigma(x) + \alpha \int_{t-d_x}^t x^\top(s)x(s)ds, \quad (9.26)$$

where  $\hat{\theta} \in \mathbb{R}^{\frac{1}{2}n \times (n+1)}$  are the estimated parameters. It is important to note that the value function estimation has an integral term due to the delayed state.

Now, using the critic estimation (9.26) in the Bellman equation (9.24) results in a Bellman error due to the difference between the actual and estimated parameters which can be described as

$$e_{BE} = \hat{V}(x, t + \Delta t) - \hat{V}(x, t) + \int_t^{t+\Delta t} \left( x^\top \tilde{Q} x + u^\top R u \right) d\tau, \quad (9.27)$$

which is

$$\begin{aligned} e_{BE} &= \int_t^{t+\Delta t} \left( x^\top \tilde{Q} x + u^\top R u \right) d\tau + \hat{\theta}^\top \Delta \sigma(x) \\ &\quad + \alpha \left( \int_{t+\Delta t-d_x}^{t+\Delta t} x^\top(s)x(s)ds - \int_{t-d_x}^t x^\top(s)x(s)ds \right), \end{aligned} \quad (9.28)$$

where  $\Delta \sigma(x) = \sigma(x(t + \Delta t)) - \sigma(x(t))$ . Now, the goal is to find the appropriate update law for the estimated parameters  $\hat{\theta}$  such that the Bellman error (9.28) eventually converges to zero.

To find the update law for the estimated parameters, consider the squared-norm of the Bellman error (9.28) given by

$$E = \frac{1}{2} \|e_{BE}\|^2. \quad (9.29)$$

By using gradient descend method, the update law for the estimated parameters can be constructed as

$$\dot{\hat{\theta}} = -\beta \frac{\partial e_{BE}}{\partial \hat{\theta}} = -\beta \frac{\Delta \sigma(x)}{(1 + \Delta \sigma(x)^\top \Delta \sigma(x))^2} e_{BE}^\top, \quad (9.30)$$

where  $\beta$  is a constant gain that affects the convergence rate.

Now, substituting the estimated value of the critic into the actor, the control policy (9.6) becomes

$$u(t) = -R^{-1} B^\top \nabla^\top \sigma(x) \hat{\theta}. \quad (9.31)$$

Define the parameter estimation error as  $\tilde{\theta} = \theta - \hat{\theta}$ . Then, from the Bellman equation (9.24) and the value function (9.25), one has

$$\begin{aligned} \int_t^{t+\Delta t} \left( x^\top \tilde{Q} x + u^\top R u \right) d\tau &= -(V(x, t + \Delta t) - V(x, t)) \\ &= -\theta^\top \Delta \sigma(x, t) - \alpha \left( \int_{t+\Delta t-d_x}^{t+\Delta t} x^\top(s) x(s) ds - \int_{t-d_x}^t x^\top(s) x(s) ds \right). \end{aligned}$$

Then, the Bellman error (9.28) in terms of the parameter estimation error, i.e.,  $\tilde{\theta}$ , can be represented by

$$e_{BE} = -\tilde{\theta} \Delta \sigma(x). \quad (9.32)$$

Exploiting (9.30) and (9.32), the dynamics of the parameter estimation error can be expressed as

$$\dot{\tilde{\theta}} = -\beta \frac{\Delta \sigma(x) \Delta \sigma(x)^\top}{(1 + \Delta \sigma(x)^\top \Delta \sigma(x))^2} \tilde{\theta}. \quad (9.33)$$

Theorem 9.3 illustrates that the parameter estimation error, i.e.,  $\tilde{\theta}$ , asymptotically converges to zero if the PE criterion is met. A Lyapunov-based proof is defined and it is shown that the derivative of the Lyapunov is negative definite which complies with the asymptotic convergence of the parameter estimation error dynamics in the ideal case.

**Theorem 9.3** Consider the tuning law for the critic approximator (9.30). Then, if  $\Sigma = \frac{\Delta \sigma(x)}{1 + \Delta \sigma(x)^\top \Delta \sigma(x)}$  is PE in the interval  $[t, t + T]$ , in which  $\int_t^{t+T} \Sigma \Sigma^\top \geq \gamma I$ , where  $\gamma$  is a positive scalar and  $I$  is the identity matrix, the origin of the parameter estimation error dynamics (9.33) converges asymptotically.

**Proof** Consider the Lyapunov function candidate as

$$\mathcal{L} = \frac{1}{2} \|\tilde{\theta}\|^2. \quad (9.34)$$

Taking derivative of the Lyapunov function (9.34) yields

$$\dot{\mathcal{L}} = \tilde{\theta}^\top \dot{\tilde{\theta}} = -\beta \tilde{\theta}^\top \frac{\Delta \sigma(x) \Delta \sigma(x)^\top}{(1 + \Delta \sigma(x)^\top \Delta \sigma(x))^2} \tilde{\theta} = -\tilde{\theta}^\top \beta \Sigma \Sigma^\top \tilde{\theta}. \quad (9.35)$$

The Lyapunov derivative function (9.35) is negative definite, i.e.,  $\dot{\mathcal{L}} < 0$ , if and only if  $\Sigma \Sigma^\top > 0$ . In this case, one can conclude that the parameter estimation error, i.e.,  $\tilde{\theta}$ , converges to zero asymptotically. However, if  $\Sigma \Sigma^\top \geq 0$ , it shows that Lyapunov derivative function (9.35) is negative semi-definite, i.e.,  $\dot{\mathcal{L}} \leq 0$ . Therefore, the convergence of the parameter estimation error to zero is not ensured. This implies that more conditions are required to confirm asymptotic stability of the parameter estimation error. To this end, from the linear control theory, the solution to the parameter

error dynamics (9.33) converges to zero if  $\beta \frac{\Delta\sigma(x)\Delta\sigma(x)^\top}{(1 + \Delta\sigma(x)^\top \Delta\sigma(x))^2} > 0$ , i.e.,  $\beta \Sigma \Sigma^\top > 0$ .

The PE condition presented in the statement of the theorem is needed. This deduces that in the time interval  $[t, t + T]$  the matrix  $\Sigma \Sigma^\top$  is positive, therefore, the derivative of the Lyapunov function (9.35) becomes negative definite and results in asymptotic convergence of the parameter estimation error.  $\square$

**Remark 9.1** It has been shown in [19] that the optimal control of a linear continuous-time system with state delay (9.1) contains an additional integral term to include the effect of delay. However, finding the optimal solution requires solving complicated partial differential equations. Although, in this chapter, the integral term has not been considered in the optimal control input, the effect of the delay is incorporated in the Bellman error (9.28).

Now, in the following theorem, the closed-loop stability is proven to show boundedness of the closed-loop system in terms of parameter estimation error and system state.

**Theorem 9.4** Consider the linear continuous-time system with state delay (9.1) and the performance index defined in (9.2). Given initial value of the estimator parameter  $\hat{\theta}_0$ , let  $u_0$  be an initial admissible control policy for the system. Let the estimated optimal control input be given by (9.31), using the estimator parameter update law (9.30). Then, if  $\beta > \frac{\rho\delta_{\max}(PBR^{-1}B^T)}{2\delta_{\min}(\Sigma\Sigma^\top)}$ , the state vector is locally ultimately bounded.

In the above theorem, it is shown that the state vector is locally ultimately bounded provided the learning rate is selected as shown above. In other words, for all initial states  $x(0) < v$ , then,  $\dot{\mathcal{L}} < 0$  which implies that the states of the system are bounded. The parameter estimation error is shown to be bounded in Theorem 9.2.

Now, consider the following modified update law for the estimated parameters as

$$\dot{\hat{\theta}} = -\beta \frac{\Delta\sigma(x)}{(1 + \Delta\sigma(x)^\top \Delta\sigma(x))^2} e_{BE}^\top - \gamma \hat{\theta}. \quad (9.36)$$

The modified tuning law relaxes the PE condition due to parameter drift. In the next theorem, the closed-loop system is shown to be ultimately bounded.

**Theorem 9.5** Consider the linear continuous-time system with state delay (9.1) and the performance index defined in (9.2). Given initial value of the estimator parameter  $\hat{\theta}_0$ , let  $u_0$  be an initial admissible control policy for the system. Let the estimated optimal control input be given by (9.31), using the modified estimator parameter update law (9.36). Then, if  $\beta\delta_{\min}(\Sigma\Sigma^\top) + \gamma > \frac{\rho\delta_{\max}(PBR^{-1}B^T)}{2}$ , the state vector (9.1) and the parameter estimation error (9.33) are ultimately bounded.

It is shown in Theorem 9.5 that the state vector and the parameter estimation errors are locally ultimately bounded. In other words, for all initial states  $x(0) < \gamma_0$  and the initial value for the estimated parameter such that  $\tilde{\theta}(0) > \gamma_2$ ,  $\dot{\mathcal{L}} < 0$  which results in the locally boundedness of the state vector and the parameter estimation error.

**Remark 9.2** The procedure of finding OAC of linear continuous-time system (9.1) with delayed state is presented in Algorithm 9.1. First, the estimated parameters are initialized and an admissible policy is applied to the system. Then, using the current stabilizing controller, the Bellman error (9.28) and the estimated parameters are tuned. Finally, the control policy is updated and, if the stopping criterion is met, the Algorithm 9.1 is stopped. Otherwise, the process of updating the parameters and controller, which is called actor and critic tuning, is continued till the stopping criterion is met.

---

**Algorithm 9.1** Approximate Optimal Adaptive control of linear continuous-time system with state delay using integral reinforcement learning

---

- 1: Initialize system state vector and estimated parameters  $\hat{\theta}_0$  to make an initial admissible control  $u_0$ .
- 2: Evaluate the Bellman error

$$e_{BE} = \int_t^{t+\Delta t} (x^\top \tilde{Q}x + u^\top Ru) d\tau + \hat{\theta}^\top \Delta\varphi(x, t) \\ + \alpha \left( \int_{t+\Delta t-\tau}^{t+\Delta t} x^\top(s)x(s)ds - \int_{t-\tau}^T x^\top(s)x(s)ds \right).$$

- 3: Update the estimated parameters

$$\dot{\hat{\theta}} = -\beta \frac{\Delta\varphi(x, t)}{(1 + \Delta\varphi(x, t)^\top \Delta\varphi(x, t))} e_{BE}^\top$$

- 4: Compute the control policy

$$u(t) = -R^{-1} B^\top \nabla^\top \sigma(x) \hat{\theta}$$

- 5: Apply the exploration noise to guarantee the PE condition

$$u(t) = \begin{cases} u(t) + P_{noise} & t \leq T \\ u(t) & t > T \end{cases}$$


---

**Remark 9.3** Note that to find the optimal control (9.31), only input matrix  $B$  is required. This implies that the proposed approach does not need the drift dynamics  $A_0$  and  $A$  due to the knowledge of the system dynamics being embedded in the state vector, i.e.,  $x(t)$ ,  $x(t - d_x)$ ,  $x(t + \Delta t)$  and  $x(t + \Delta t - d_x)$ . To eliminate the need for the input matrix  $B$  in the control policy calculation, one can estimate the actor as well.

**Remark 9.4** It can be seen from (9.30) that the parameter estimation error, i.e.,  $\tilde{\theta}$ , becomes zero when  $\|x\| = 0$ . This implies that when the states of the system tend to zero, the parameter estimation error and, therefore, the critic approximator is not updated and, thus, does not converge to the actual parameters. This can be considered as the PE requirement for the system states. To do this, to ensure the convergence of the estimation parameter to the actual value, the signal  $\Sigma$  must be PE. In this chapter,

to ensure the PE condition, an exploration noise is added to the control input [41] and has been removed when the parameter estimation error converges to a small bound.

**Remark 9.5** Note that the time interval  $[t, t + T]$ , defined in the statement Theorem 9.3, is the most important part of the design. The value of  $T$  must be designed to confirm the convergence of the estimated parameters to the actual values. To this end, one can consider a large value for  $T$  and then try to make it small enough until the convergence criterion is met.

#### 9.4.2 Event Sampled Feedback

In this subsection, it is assumed that not only the state vector is not available periodically, but also the system dynamics are unknown. Event-triggered controller is employed to find the estimated OAC policy. Consider the control policy defined in the sampling instant  $t_k$  as  $\bar{u} = u(t_k)$ . The value function (9.2) by considering the sampling input at sampling instants can be written as

$$V(x(t)) = \int_t^\infty \left( x(\tau)^\top \tilde{Q} x(\tau) + \bar{u}(\tau)^\top R \bar{u}(\tau) \right) d\tau. \quad (9.37)$$

Then, using the IRL as shown in [23], the following Bellman equation can be represented by

$$V(x(t)) = \int_t^{t+\Delta t} \left( x^\top \tilde{Q} x + \bar{u}^\top R \bar{u} \right) d\tau + V(x, t + \Delta t), \quad (9.38)$$

where  $\Delta t$  is a fixed and small time interval. Next, using the approximation of the value function and its estimation defined in (9.25) and (9.26), the following Bellman error due to the difference between the actual and estimated parameters can be described as

$$\begin{aligned} \bar{e}_{BE} &= \hat{V}(x, t + \Delta t) - \hat{V}(x, t) \\ &\quad + \int_t^{t+\Delta t} \left( x^\top \tilde{Q} x + \bar{u}^\top R \bar{u} \right) d\tau, \end{aligned} \quad (9.39)$$

which is

$$\begin{aligned} \bar{e}_{BE} &= \int_t^{t+\Delta t} \left( x^\top \tilde{Q} x + \bar{u}^\top R \bar{u} \right) d\tau + \hat{\theta}^\top \Delta \sigma(x) \\ &\quad + \alpha \left( \int_{t+\Delta t-d_x}^{t+\Delta t} x^\top(s) x(s) ds - \int_{t-d_x}^t x^\top(s) x(s) ds \right), \end{aligned} \quad (9.40)$$

where  $\Delta \sigma(x(t)) = \sigma(x(t + \Delta t)) - \sigma(x(t))$ . Now, the goal is to find the appropriate update law for the estimated parameters  $\hat{\theta}$  such that the Bellman error (9.40) eventually converges to zero. To this end, considering  $E = \frac{1}{2} \|\bar{e}_{BE}\|^2$  and using the

gradient descend method, the following update law can be constructed for the estimated parameters in the case of sampled states as

$$\begin{cases} \hat{\theta}^+ = \hat{\theta} - \beta \frac{\Delta\sigma(x)}{(1 + \Delta\sigma(x)^\top \Delta\sigma(x))^2} \bar{e}_{BE}^\top(t_k) & t = t_k \\ \dot{\hat{\theta}} = 0 & t \in [t_k, t_{k+1}) \end{cases}, \quad (9.41)$$

where  $\beta$  is a constant gain that affects the convergence rate. The right derivative of the estimated parameter  $\hat{\theta}$  at the event-triggering instant  $t_k$  is denoted as  $\theta^+$ . Note that to ensure the convergence of the estimated parameters, the regression vector should be persistently exciting.

Now, substituting the estimated value of the critic into the actor, the continuous control policy (9.6) becomes

$$u(t) = -R^{-1}B^\top \nabla^\top \sigma(x)\hat{\theta}. \quad (9.42)$$

Define the parameter estimation error as  $\tilde{\theta} = \theta - \hat{\theta}$ . Then, from the Bellman equation (9.24) and the value function (9.37), one has

$$\begin{aligned} \int_t^{t+\Delta t} \left( x^\top \tilde{Q}x + \bar{u}^\top R\bar{u} \right) d\tau &= -(V(x, t + \Delta t) - V(x, t)) \\ &= -\theta^\top \Delta\sigma(x(t)) - \alpha \left( \int_{t+\Delta t-d_x}^{t+\Delta t} x^\top(s)x(s)ds - \int_{t-d_x}^t x^\top(s)x(s)ds \right). \end{aligned}$$

Then, the Bellman error (9.28) in terms of the parameter estimation error, i.e.,  $\tilde{\theta}$ , becomes  $\bar{e}_{BE} = -\tilde{\theta}\Delta\sigma(x)$ . Now, using (9.41), the dynamics of the parameter estimation error can be taken in the form of

$$\begin{cases} \tilde{\theta}^+ = \tilde{\theta} + \beta \frac{\Delta\sigma(x)}{(1 + \Delta\sigma(x)^\top \Delta\sigma(x))^2} \bar{e}_{BE}^\top(t_k), & t = t_k \\ \dot{\tilde{\theta}} = 0, & t \in [t_k, t_{k+1}) . \end{cases} \quad (9.43)$$

By using the above parameter tuning law at the event sampled instants and holding the parameters during the inter-event time, one can show that all the signals of the closed-loop system are bounded [42]. The event-trigger condition is similar to the condition presented in [34] and it is given by

$$\|e_u(t)\| < \eta \left\| \nabla^\top \sigma(x(t))\hat{\theta} \right\|, \quad (9.44)$$

where  $\eta = \|\gamma R^{-1}B^\top\|$  with  $\gamma > 0$  as a design scalar and  $e_u(t) = u(t_k) - u(t)$  denotes the input measurement error that is defined as the error between the sampled control input and actual continuous control input. Note that  $u(t_k) = \psi(x(t_k))$  and  $u(t) = \psi(x(t))$ , where  $x(t_k)$  is a sampled state at  $t_k$ . Therefore, one can find a

relationship between the input measurement error,  $e_u(t)$ , and the state measurement error,  $e$ , defined in (9.14). By using the trigger condition, it can be shown that the control policy is bounded close to the optimal input after a long time due to event sampling and parameter adaptation being at these event sampled instants. To attain optimality faster, tuning the parameters of the critic within the inter-sampled interval is of interest which is presented next.

### 9.4.3 Hybrid Reinforcement Learning Scheme

In this section, a novel hybrid scheme is presented to find the solution of the OAC of linear continuous-time system (9.1) with state delay when the dynamics of the system are unknown. In the hybrid scheme, the unknown parameter vector of the critic function is tuned in an iterative manner within the inter-sample interval. In other words, due to the inter-sample interval varying in the event sampled system, the number of iterations performed on the critic function parameters will change with time. As a consequence, the stability of the closed-loop system is a challenge unless suitable parameter update law is used.

The approach is similar to the event-trigger-based controller presented in Sect. 9.4.2; however, in addition to updating the unknown parameters at sampled instants, the critic parameter vector is also updated during the inter-sampled interval. The parameter tuning law is given by

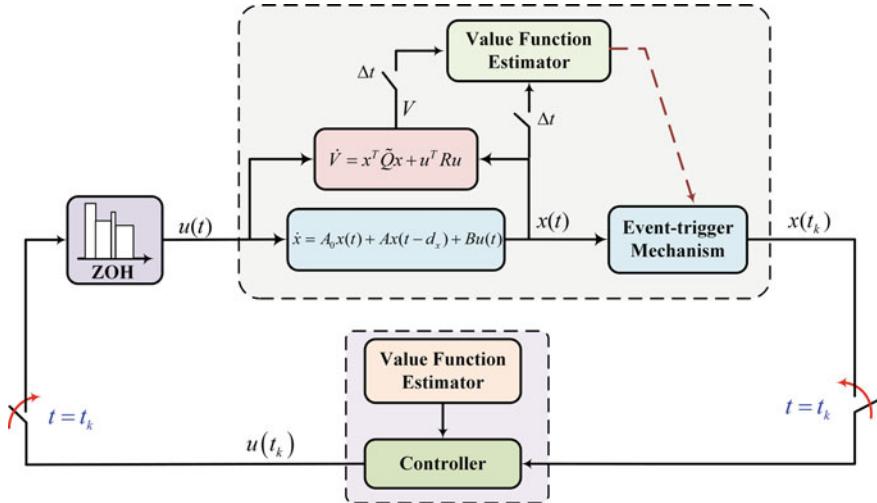
$$\begin{cases} \hat{\theta}^+ = \hat{\theta} - \beta \frac{\Delta\sigma(x)}{(1 + \Delta\sigma(x)^\top \Delta\sigma(x))^2} \bar{e}_{BE}^\top(t_k), & t = t_k \\ \dot{\hat{\theta}} = -\beta \frac{\Delta\sigma(x(t_k))}{(1 + \Delta\sigma(x)^\top \Delta\sigma(x))^2} \bar{e}_{BE}^\top(t_k), & t \in (t_k, t_{k+1}) \end{cases}, \quad (9.45)$$

where  $\beta$  is a constant gain that affects the convergence rate. Now, substituting the estimated value of the critic into the actor, the control policy (9.6) becomes

$$u(t) = -R^{-1}B^\top \nabla^\top \sigma(x)\hat{\theta}. \quad (9.46)$$

Define the parameter estimation error as  $\tilde{\theta} = \theta - \hat{\theta}$  and, using the same procedure presented in Sect. 9.4.2, the dynamics of the parameter estimation error can be taken in the form of

$$\begin{cases} \tilde{\theta}^+ = \tilde{\theta} + \beta \frac{\Delta\sigma(x)}{(1 + \Delta\sigma(x)^\top \Delta\sigma(x))^2} \bar{e}_{BE}^\top(t_k) & t = t_k \\ \dot{\tilde{\theta}} = \beta \frac{\Delta\sigma(x)}{(1 + \Delta\sigma(x)^\top \Delta\sigma(x))^2} \bar{e}_{BE}^\top(t_k) & t \in (t_k, t_{k+1}) \end{cases}. \quad (9.47)$$



**Fig. 9.2** The hybrid event-triggered optimal adaptive controller scheme when internal dynamics are unknown

The above parameter tuning is an extension of the effort in [34] to the time delay systems. The stability of the closed-loop system can be demonstrated using an approach similar to the proof shown in [34]. The overall scheme of the hybrid event-triggered optimal adaptive controller is shown in Fig. 9.2. The value function estimator is located at the event-trigger mechanism as well as the controller in order to prevent transmission of parameters from the controller to the event-triggering mechanism. These two value function estimators are synchronized using same initial conditions. Moreover, in the parameter tuning law for the hybrid scheme (9.45), the value of the regression vector  $\Delta\sigma(x)$  and the Bellman error  $\bar{e}_{BE}$  at the event sampled time instants  $t_k$  is utilized in the update law for the inter-sampled time interval  $t \in (t_k, t_{k+1})$ .

A common theme in most of the abovementioned approaches has been the use of sensors to measure physical quantities such as pressure, temperature, and others. So far we have discussed OAC for linear time delay systems with state delay. However, in order to extend the OAC for nonlinear systems, NN as function approximators are typically utilized. However, for control applications, a single NN with a basis function is normally employed. For complex applications, a deep NN is needed as introduced next.

## 9.5 Perspectives on Controller Design with Image Feedback

For simple control task executed by a human, the sensory input is not numerical values. For instance, one of the canonical tasks a child learns is to catch a ball. In this context, the sensory input is a series of images, the controller is the human brain

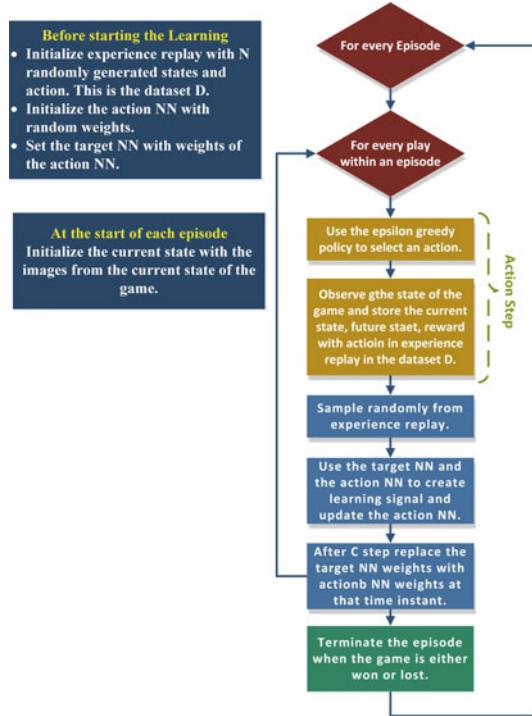
whereas the actuator is the hand. Overall, the child learns to map the sensory input into an action which is executed by the actuator. This map can be understood as a function which is normally considered unknown.

In the past five decades, significant effort is carried out by the researchers to understand and reconstruct this unknown function that the human brain generates. The first known mathematical model of this function was provided by McCulloch and Pitts in 1941 [43]. In late 80s, it was shown by using a simple representation of a neuron, the NN can be function approximators, that is, they can approximate any unknown smooth function [44] over a compact set. With this development, many controllers were developed in early 90s for identification and control [45, 46]. Over the years, supervised learning approaches such as backpropagation through time [47], RL approaches such as policy gradients, [48] and unsupervised learning [49] approaches have been introduced to efficiently design these controllers.

In these controllers, the main idea was to derive a mathematical representation of physical system and then use the NN to approximate the certain aspects of the nonlinear system [45, 46] by using sensors that provide numerical values. In contrast, the sensor inputs for a human are images and audio signals which require deep NN to process the information. In summary, in spite of decades of research, the available controllers may not ensure performance when the sensory input is an image or audio signal.

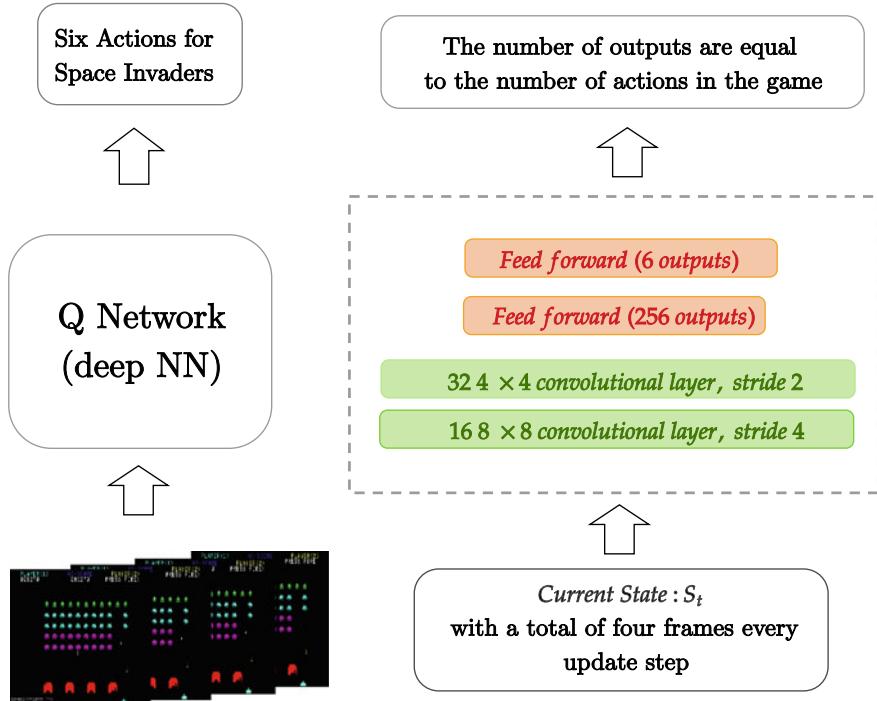
It has been shown in [50, 51] that when the input data is images and audio, the functional map is much complicated and NN with two or three layers are not sufficient. This fact led to the resurgence of the field of deep learning where a NN with more than three layers is used to approximate the map between input and output. In fact, it has been shown that deep NN are more efficient in approximating complex functions [52]. The advent of deep learning has had a significant impact on many areas such as object detection, speech recognition, and language translation [50–53]. Deep learning has similarly accelerated progress in RL, with the use of deep learning schemes within RL defining the field of “deep reinforcement learning” (DRL).

Deep learning enables RL to scale to decision-making problems that were previously intractable, i.e., high-dimensional state (such as images and audio inputs) and action spaces. This property enables the possibility of designing controllers that are efficient and can imitate the human brain. Among recent work in the field of DRL, there have been two outstanding success stories. The first, was the development of an algorithm that could learn to play a range of Atari 2600 video games at a superhuman level, directly from image pixels [54]. The main contribution of this work was to convincingly demonstrate that RL agents could be trained on raw, high-dimensional observations, solely based on a reward signal. The second standout success was the development of a hybrid DRL system to play AlphaGo, that defeated a human world champion in Go [55]. Based on these successes, DRL methods have already been applied to a wide range of problems, such as robotics, where control policies for robots can be learned directly from camera inputs in the real world [56, 57], succeeding controllers that used to be hand-engineered or learned from low-dimensional features of the robot’s state (Fig. 9.3).

**Fig. 9.3** Learning scheme

Consider the application of playing games such as breakout and let the structure of NN utilized to approximate the action value function is depicted in Fig. 9.4. Note that, two convolutional layers are utilized and two feedforward layers are chosen in this setup. A corresponding learning algorithm that is utilized to train the deep NN is provided in Fig. 9.3. The deep NN is trained for a total of 100,000 episodes and each episode is composed of 1000 plays where each play provides a total of four images. Therefore, the NN has to process a total of 400 million frames. The score at the end of each episode is recorded as results. Two hundred episodes are randomly sampled (ordered according to the update steps) from the results and the scores are plotted. For these results, the size of experience replay buffer is fixed at one million. One may observe that the larger the experience, the better the agent performs. In other words, the scores when the buffer size is 4 million are better relative to the scenario when the buffer size is 1 million.

For these testing purposes [50–57], the main learning scheme is backpropagation/stochastic gradient descent unlike the Lyapunov-based nonstandard weight tuning introduced in the previous sections. In this RL-based design, the only feedback from the environment is the reward signals but backpropagation requires targets to be defined for learning. In most DRL schemes available today with stochastic gradient descent, synthetic targets are defined using the reward signal. Due to the processing required to create these targets, the little information provided by the reward signals



**Fig. 9.4** Neural network structure [54]

**Table 9.1** Mean score (standard deviation) for Breakout and Space Invaders over randomly sampled episodes

Method	Game: Breakout	Game: Space Invaders
DQL	287 (78)	1139 (158)
EDQL	315 (155)	980 (163)
Hybrid	310 (120)	1144 (121)

is occluded. As a consequence, the learning scheme takes a long time to converge. For instance, two hundred million image frames must be processed to achieve median performance in [54] as observed from the results demonstrated here.

Since, the learning process is slow, the agent must learn the state of the environment through an extensive trial and error interaction. Table 9.1 depicts the outcome of three methods, one using DRL via gradient descent method, the second using exploratory direct error-driven learning referred to as EDQL [58] and the third using EQDL with hybrid learning scheme of parameter tuning referred to as hybrid for playing breakout and space invader games. From the table, it is clear that hybrid EQDL using hybrid parameter tuning [58] performs slightly better when compared to traditional deep NN-based RL scheme using backpropagation.

Existing RL methods using deep NNs achieve this feat through an episodic process where the reward signals are obtained at the end of an episode and the agent plays each episode for either pass or fail. In many real-world applications, there is no possibility of repeated failure and any failure results in significant loss of performance. In other words, the number of failures is utilized for learning in applications such as playing breakout, but in control applications, the trial and error cannot be tolerated. Therefore, episodic learning process is not preferred in real-time control applications. Moreover, in [51, 54–57] using deep NN RL, processing image information incurs delays which has to be mitigated as discussed in the previous sections. Demonstrating stability of the closed-loop system with deep NN is a challenge. Therefore, development of learning schemes using deep NN with image and audio feedback is of interest for control applications such as self-driving cars, space, and others.

## 9.6 Simulation Results

In this section, several examples are provided to show the effectiveness of the proposed approach.

**Example 9.1** Consider a linear continuous-time system with state delay given by

$$\begin{cases} \dot{x}_1(t) = x_1(t) + 1.5x_2(t) - x_2(t-1) \\ \dot{x}_2(t) = 0.3x_1(t) - 2x_2(t) + 2x_2(t-1) + u(t) \end{cases}, \quad (9.48)$$

where

$$A_0 = \begin{bmatrix} 1 & 1.5 \\ 0.3 & -2 \end{bmatrix}, A = \begin{bmatrix} 0 & -1 \\ 0 & 2 \end{bmatrix}, B = \begin{bmatrix} 0 \\ 1 \end{bmatrix}.$$

The following profile is considered as the initial history of the state as

$$x_1(\theta) = x_2(\theta) = 1 \quad \theta \in [-1, 0]. \quad (9.49)$$

Let the performance index be given by

$$J = \int_0^{\infty} (2x_1^2 + 2x_2^2 + u^2) dt, \quad (9.50)$$

which gives the design variables as  $\tilde{Q} = \begin{bmatrix} 2 & 0 \\ 0 & 2 \end{bmatrix}$ ,  $R = 1$ ,  $\alpha = 1.9$ .

### 9.6.1 Linear Quadratic Regulator with Known Internal Dynamics

In this subsection, the optimal control is simulated assuming the internal dynamics of the system are known. Using the design parameters, the solution of the DARE (9.7) and corresponding control input becomes

$$P = \begin{bmatrix} 21.94 & 6.67 \\ 6.67 & 2.55 \end{bmatrix}, \quad K = -R^{-1}B^T P = -[6.67 \ 2.55].$$

The performance of the proposed controller given by (9.6) is shown in Fig. 9.5. The state vector of the system is shown in Fig. 9.5a and it can be seen that the state vector converges to zero which substantiates Theorem 9.1. Moreover, it is shown in Fig. 9.5c that the BTE (9.3) tends to zero eventually.

Now, to show the effectiveness of the proposed linear quadratic regulator, assume that the optimal control input is derived through traditional algebraic Riccati equation in the form of  $A_0^\top P_T + P_T A_0 + P_T B R^{-1} B^\top P_T + \tilde{Q} = 0$  without considering the effect of the state delayed dynamics to get

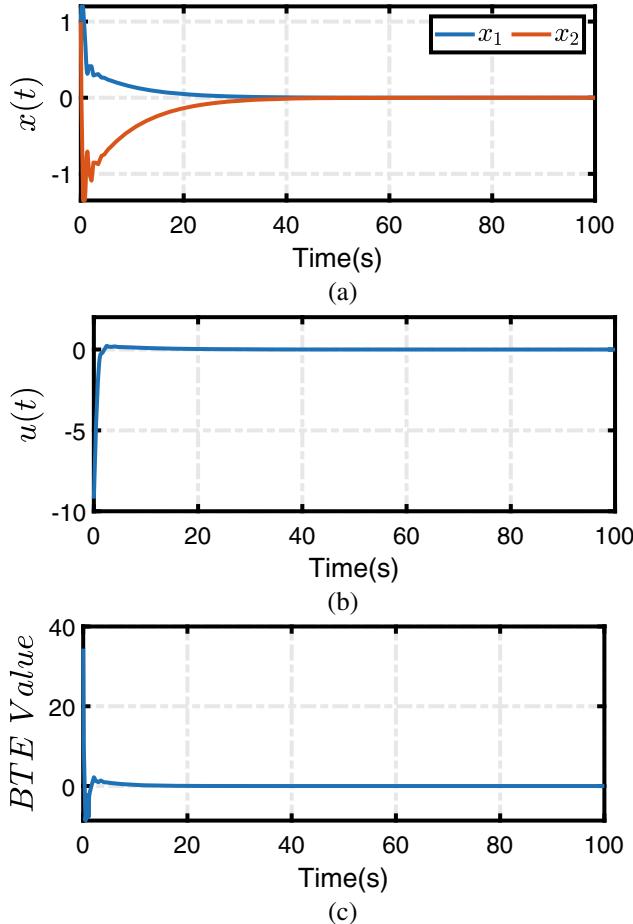
$$P_T = \begin{bmatrix} 14.65 & 5.90 \\ 5.90 & 2.87 \end{bmatrix}, \quad K_T = -\frac{1}{2}R^{-1}B^T P_T = -[2.95 \ 1.43],$$

where  $P_T$  and  $K_T$  represent the solution to the traditional algebraic Riccati equation and the optimal traditional control gain for the non-delayed system, respectively. The value of  $R$  and  $Q$  are similar to the case of the delayed approach. The results of applying the traditional control input to the system is shown in Fig. 9.6. It can be seen from Fig. 9.6a that the state vector diverges indicating the instability of the system. This also verifies the selection of the Lyapunov–Krasovskii function as the value function to ensure that the overall closed-loop system becomes asymptotically stable.

### 9.6.2 Optimal Adaptive Control with Unknown Drift Dynamics

In this subsection, it is assumed that the internal dynamics of the system (9.48), i.e.,  $A_0, A$ , are considered unknown. First, it is considered that the state of the system is available continuously. To this end, the critic parameters such as regression function and the convergence rate are selected as follows

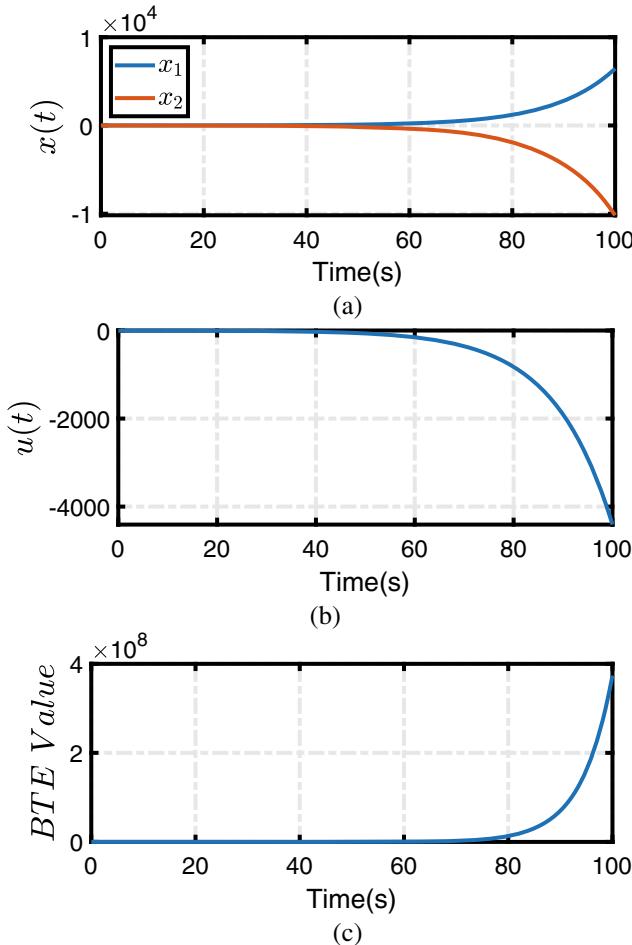
$$\sigma(x) = \begin{bmatrix} x_1^2 \\ x_1 x_2 \\ x_2^2 \end{bmatrix}, \quad \beta = 0.5.$$



**Fig. 9.5** Performance of the proposed optimal controller when DARE (9.7) is employed. **a** State vector, **b** Control input (9.6), and **c** BTE (9.3)

The overall trajectories of the system is depicted in Fig. 9.7, when the OAC presented in Sect. 9.3.1 is applied to the system. An exploration random noise is added to the control input when  $t < 20$  to excite all the modes of the system dynamics and to ensure the PE condition is fulfilled. This guarantees the convergence of the estimated parameters while the system states have not converged. It is shown in Fig. 9.7a that the state vector converges to a bounded value. Figure 9.7d shows the convergence of the estimated parameters. The effect of PE condition can be seen by comparing the state convergence and parameter convergence. This is because the parameter estimation errors attain near zero earlier than the system state.

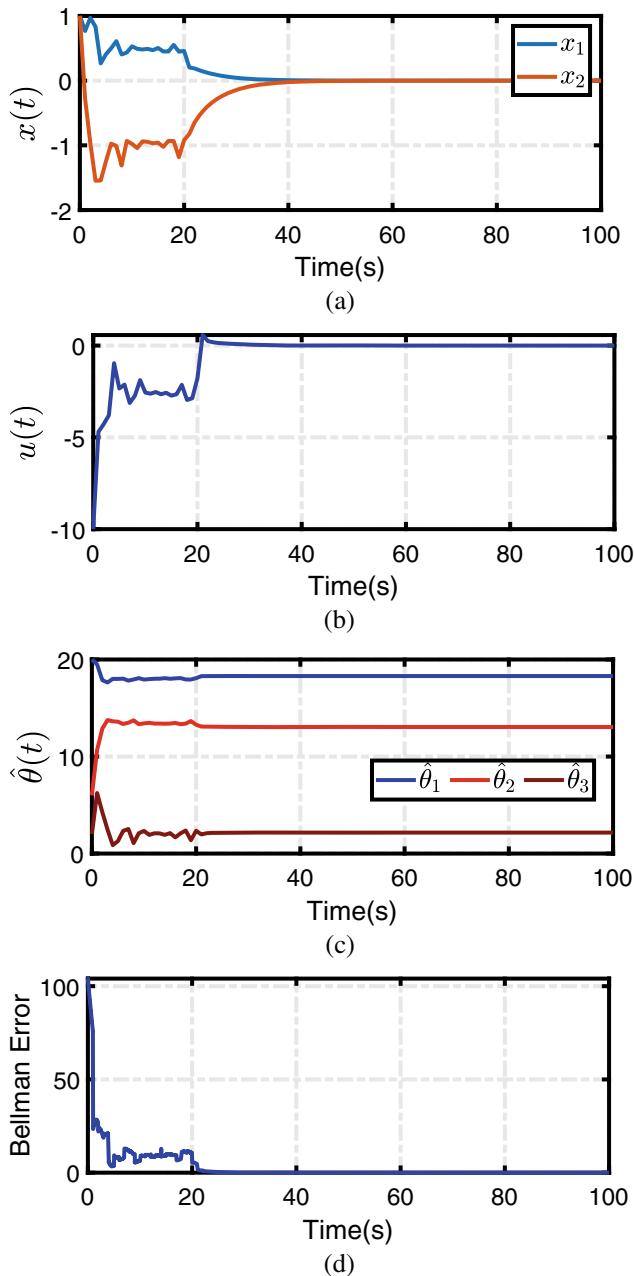
The performance of the hybrid RL scheme is shown in Fig. 9.8 when the drift dynamics are unknown. For the triggering condition, the value of  $\gamma$  is selected as 4.



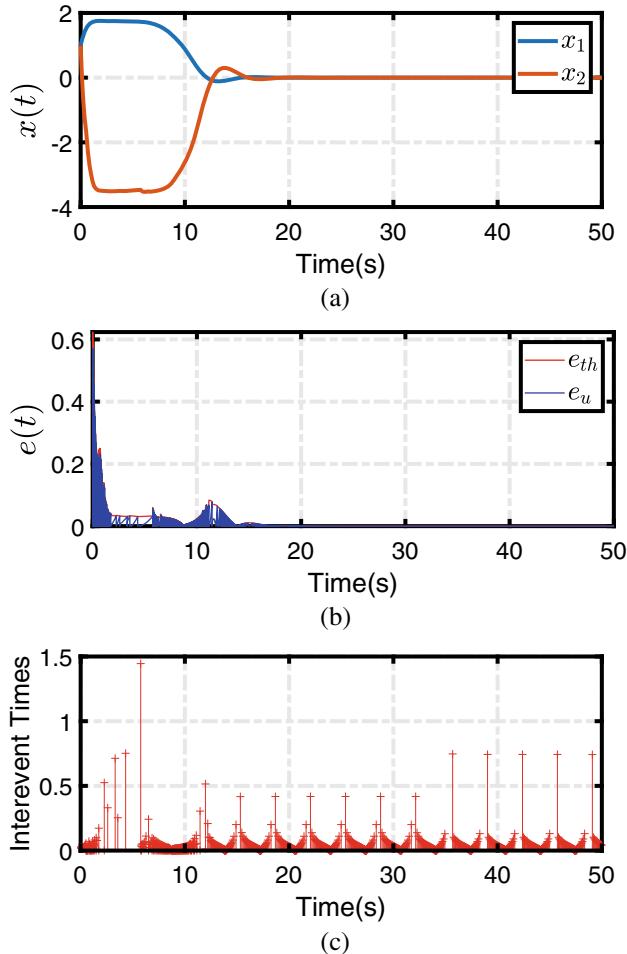
**Fig. 9.6** Performance of the optimal controller using traditional algebraic Riccati equation. **a** State vector, **b** Control input, and **c** BTE (9.3)

## 9.7 Conclusion

In this paper, the optimal adaptive control of partially unknown linear continuous-time systems with state delay by using IRL is addressed. A novel delay algebraic Riccati equation is proposed to confirm the asymptotic stability of the closed-loop system given the system dynamics. A novel value function with the integral term is utilized for critic to ensure performance. It was demonstrated that for the case of known dynamics, the optimal control policy renders asymptotic stability of the closed-loop system provided the linear time-delayed system is controllable and observable.



**Fig. 9.7** Performance of the optimal adaptive control with periodic state feedback. **a** State vector, **b** Control input, **c** Estimated parameters of the value function, and **d** Bellman error



**Fig. 9.8** Performance of the hybrid RL control scheme. **a** State vector, **b** Event threshold and measurement error, and **c** Inter-event times

Next, the IRL approach using actor–critic structure estimates the optimal control input without requiring any information about drift dynamics. However, input matrix is still needed to find the control policy. Lyapunov theory is employed to show boundedness of the closed-loop system. It has been shown that a PE condition is required to ensure the boundedness of the critic parameters. Event-trigger condition is derived in order to show the overall stability when the feedback information is aperiodic. Future perspectives of using deep NN for control applications is also included. A simulation example verifies the effectiveness of the proposed methods.

**Acknowledgements** This work was supported in part by Intelligent Systems Center, National Science Foundation under grants ECCS 1406533 and CMMI 1547042.

## References

1. Hespanha, J.P., Naghshtabrizi, P., Xu, Y.: A survey of recent results in networked control systems. *Proc. IEEE* **95**, 138–162 (2007)
2. Antsaklis, P., Baillieul, J.: Special issue on technology of networked control systems. *Proc. IEEE* **95**(1), 5–8 (2007)
3. Montestruque, L.A., Antsaklis, P.: Stability of model-based networked control systems with time-varying transmission times. *IEEE Trans. Autom. Control* **49**(9), 1562–1572 (2004)
4. Sahoo, A., Jagannathan, S.: Stochastic optimal regulation of nonlinear networked control systems by using event-driven adaptive dynamic programming. *IEEE Trans. Cybern.* **47**(2), 425–438 (2017)
5. Wu, J., Chen, T.: Design of networked control systems with packet dropouts. *IEEE Trans. Autom. control* **52**(7), 1314–1319 (2007)
6. Xu, H., Jagannathan, S., Lewis, F.: Stochastic optimal control of unknown linear networked control system in the presence of random delays and packet losses. *Automatica* **48**(6), 1017–1030 (2012)
7. Gu, K., Chen, J., Kharitonov, V.L.: Stability of time-delay systems. Springer Science & Business Media, Berlin (2003)
8. Emilia, F.: Introduction to Time-delay Systems: Analysis and Control. Springer, Berlin (2014)
9. Richard, Jean-Pierre: Time-delay systems: an overview of some recent advances and open problems. *Automatica* **39**(10), 1667–1694 (2003)
10. Karafyllis, I., Malisoff, M., Mazenc, F., Pepe, P.: Recent results on nonlinear delay control systems. Springer International Publishing Switzerland, Cham (2016)
11. Zhu, Y., Krstic, M., Su, H.: Adaptive output feedback control for uncertain linear time-delay systems. *IEEE Trans. Autom. Control* **62**(2), 545–560 (2017)
12. Ponomarev, A.: Reduction-based robustness analysis of linear predictor feedback for distributed input delays. *IEEE Trans. Autom. Control* **61**(2), 468–472 (2016)
13. Artstein, Z.: Linear systems with delayed controls: a reduction. *IEEE Trans. Autom. Control* **27**(4), 869–879 (1982)
14. Cai, X., Bekiaris-Liberis, N., Krstic, M.: Input-to-state stability and inverse optimality of linear time-varying-delay predictor feedbacks. *IEEE Trans. Autom. Control* **63**(1), 233–240 (2018)
15. Kharitonov, V.L., Niculescu, S., Moreno, J., Michiels, W.: Static output feedback stabilization: necessary conditions for multiple delay controllers. *IEEE Trans. Autom. Control* **50**(1), 82–86 (2005)
16. Bresch-Pietri, D., Krstic, M.: Delay-adaptive predictor feedback for systems with unknown long actuator delay. *IEEE Trans. Autom. Control* **55**(9), 2106–2112 (2010)
17. Miroslav, K.: Delay compensation for nonlinear, adaptive, and PDE systems. Springer, Berlin (2009)
18. McRuer, D.: Human dynamics in man-machine systems. *Automatica* **16**(3), 237–253 (1980)
19. Ross, D., Flügge-Lotz, I.: An optimal control problem for systems with differential-difference equation dynamics. *SIAM J. Control* **7**(4), 609–623 (1969)
20. Basin, M., Rodriguez-Gonzalez, J., Fridman, L.: Optimal and robust control for linear state-delay systems. *J. Frankl. Inst.* **344**(6), 830–845 (2007)
21. Basin, M., Rodriguez-Gonzalez, J.: Optimal control for linear systems with multiple time delays in control input. *IEEE Trans. Autom. Control* **51**(1), 91–97 (2006)
22. Maity, D., Mamduhi, M.H., Hirche, S., Johansson, K.H., Baras, J.S.: Optimal lqg control under delay-dependent costly information. *IEEE Control Syst. Letters* **3**(1), 102–107 (2019)

23. Vrabie, D., Pastravanu, O., Abu-Khalaf, M., Lewis, F.: Adaptive optimal control for continuous-time linear systems based on policy iteration. *Automatica* **45**(2), 477–484 (2009)
24. Kiumarsi, B., Vamvoudakis, K.G., Modares, H., Lewis, F.L.: Optimal and autonomous control using reinforcement learning: a survey. *IEEE Trans. Neural Netw. Learn. Syst.* **29**(6), 2042–2062 (2018)
25. Narayanan, V., Jagannathan, S.: Event-triggered distributed control of nonlinear interconnected systems using online reinforcement learning with exploration. *IEEE Trans. Cybern.* **48**(9), 2510–2519 (2018)
26. Sutton, R.S., Barto, A.G.: Reinforcement learning: an introduction. MIT press, Cambridge (2018)
27. Littman, M.L.: Reinforcement learning improves behaviour from evaluative feedback. *Nature* **521**(7553), 445 (2015)
28. Tabuada, P.: Event-triggered real-time scheduling of stabilizing control tasks. *IEEE Trans. Autom. Control* **52**(9), 1680–1685 (2007)
29. Vamvoudakis, K.G., Ferraz, H.: Model-free event-triggered control algorithm for continuous-time linear systems with optimal performance. *Automatica* **87**, 412–420 (2017)
30. Borgers, D.P., Dolk, V.S., Heemels, W.P.M.H.: Riccati-based design of event-triggered controllers for linear systems with delays. *IEEE Trans. Autom. Control* **63**(1), 174–188 (2018)
31. Garcia, E., Antsaklis, P.J.: Model-based event-triggered control for systems with quantization and time-varying network delays. *IEEE Trans. Autom. Control* **58**(2), 422–434 (2013)
32. Narayanan, V., Jagannathan, S.: Event-triggered distributed approximate optimal state and output control of affine nonlinear interconnected systems. *IEEE Trans. Neural Netw. Learn. Syst.* **29**(7), 2846–2856 (2018)
33. Karafyllis, I., Krstic, M.: Adaptive certainty-equivalence control with regulation-triggered finite-time least-squares identification. *IEEE Trans. Autom. Control* **63**(10), 3261–3275 (2018)
34. Narayanan, V., Jagannathan, S., Moghadam, R.: Optimality in event-triggered adaptive control of uncertain linear dynamical systems. In: AIAA Scitech 2019 Forum (2019)
35. Strang, G.: Introduction to linear algebra. Cambridge Press, Cambridge (2016)
36. Lewis, F.L., Vrabie, D., Syrmos, V.L.: Optimal control. Wiley, Hoboken (2012)
37. Moghadam, R., Jagannathan, S.: Approximate optimal adaptive control of partially unknown linear continuous-time systems with state delay. In: 58th IEEE Conference on Decision and Control (2019)
38. Kreindler, E., Jameson, A.: Conditions for nonnegativeness of partitioned matrices. *IEEE Trans. Autom. Control* **17**(1), 147–148 (1972)
39. Khalil, H.K.: Nonlinear control. Pearson, New York (2015)
40. Kučera, V.: On nonnegative definite solutions to matrix quadratic equations. *Automatica* **8**(4), 413–423 (1972)
41. Ioannou, P., Fidan, B.: Adaptive control tutorial, 11. Siam (2006)
42. Narayanan, V., Jagannathan, S.: Distributed adaptive optimal regulation of uncertain large-scale interconnected systems using hybrid Q-learning approach. *IET Control Theory Appl.* **10**(12), 1448–1457 (2016)
43. Zhang, L., Zhang, B.: A geometrical representation of McCulloch-Pitts neural model and its applications. *IEEE Trans. Neural Netw.* **10**(4), 925–929 (1999)
44. Hornik, K.: Approximation capabilities of multilayer feedforward networks. *Neural Netw.* **4**(2), 251–257 (1991)
45. Narendra, S.K., Parthasarathy, K.: Multilayer discrete-time neural-net controller with guaranteed performance. *IEEE Trans. Neural Netw.* **7**(1), 107–130 (1996)
46. Jagannathan, S., Lewis, F.L.: Identification and control of dynamical systems using neural networks. *IEEE Trans. Neural Netw.* **1**(1), 4–27 (1990)
47. Werbos, P.J.: Backpropagation through time: what it does and how to do it. *Proc. IEEE* **78**(10), 1550–1560 (1990)
48. Liu, D., Wei, Q.: Policy iteration adaptive dynamic programming algorithm for discrete-time nonlinear systems. *IEEE Trans. Neural Netw. Learn. Syst.* **25**(3), 621–634 (2014)

49. Lewis, F.L., Jagannathan, S., Yesildirak, A.: Neural network control of robot manipulators and non-linear systems. CRC Press, Boca Raton (1998)
50. Brahma, P.P., Wu, D., She, Y.: Why deep learning works: a manifold disentanglement perspective. *IEEE Trans. Neural Netw. Learn. Syst.* **27**(10), 1997–2008 (2016)
51. Lin, H.W., Tegmark, M., Rolnick, D.: Why does deep and cheap learning work so well? *J. Stat. Phys.* **168**(6), 1223–1247 (2017)
52. Mhaskar, H.N., Poggio, T.: Deep versus shallow networks: an approximation theory perspective. *Anal. Appl.* **14**(6), 829–848 (2016)
53. Rux, L., Bengio, Y.: Deep belief networks are compact universal approximators. *Neural Comput.* **22**(8), 2192–2207 (2010)
54. Mnih, V., Kavukcuoglu, K., Silver, D., Rusu, A.A., Veness, J., Bellemare, M.G., Graves, A., Riedmiller, M., Fidjeland, A.K., Ostrovski, G., Petersen, S., Beattie, C., Sadik, A., Antonoglou, I., King, H., Kumaran, D., Wierstra, D., Legg, S., Hassabis, D.: Human-level control through deep reinforcement learning. *Nature* **518**, 529–533 (2015)
55. Silver, D., Huang, A., Maddison, C.J., Guez, A., Sifre, L., van den Driessche, G., Schrittwieser, J., Antonoglou, I., Panneershelvam, V., Lanctot, M., Dieleman, S., Grewe, D., Nham, J., Kalchbrenner, N., Sutskever, I., Lillicrap, T., Leach, M., Kavukcuoglu, K., Graepel, T., Hassabis, D.: Mastering the game of Go with deep neural networks and tree search. *Nature* **529**, 484–489 (2016)
56. Levine, S., Finn, C., Darrell, T., Abbeel, P.: End-to-end training of deep visuomotor policies. *J. Mach. Learn. Res.* **17**(1), 1334–1373 (2016)
57. Levine, S., Pastor, P., Krizhevsky, A., Ibarz, J., Quillen, D.: Learning hand-eye coordination for robotic grasping with deep learning and large-scale data collection. *Int. J. Robot. Res.* **37**(4–5), 421–436 (2018)
58. Krishnan, R., Jagannathan, S., Samaranayake, V.A.: Direct error-driven learning for deep neural networks with applications to big data. *IEEE Trans. Neural Netw. Learn. Syst.* **144**, 1–8 (2019)

# Chapter 10

## Dissipativity-Based Verification for Autonomous Systems in Adversarial Environments



Aris Kanellopoulos, Kyriakos G. Vamvoudakis, Vijay Gupta,  
and Panos Antsaklis

**Abstract** In this chapter, we investigate the problem of verifying desired properties of large-scale cyber-physical systems that operate in uncertain, adversarial environments. We focus on properties relating to dissipativity and we propose approaches to achieve verification with minimum knowledge of the system dynamics. In order to achieve compositionality in large-scale models, we distribute the verification process among individual subsystems, which utilize limited information received locally from their immediate neighbors. The need for knowledge of the subsystem parameters is avoided via a novel reinforcement learning-inspired approach that enables decentralized evaluation of appropriate storage functions that can be used to verify dissipativity. Our method allows the addition of learning-enabled subsystems to the cyber-physical network while dissipativity is ensured. We show how different learning rules can be used to guarantee different properties, such as  $L_2$ -gain stability or losslessness. Finally, the need for verification of complex properties is addressed and we propose new directions for this problem.

### 10.1 Introduction

The increasingly tight interconnection of the digital and the physical domains, owing to the advent of large-scale complex cyber-physical systems (CPS) [1], has lead to a resurgence in the field of autonomy [2]. Autonomy in engineering is the ability of

---

A. Kanellopoulos (✉) · K. G. Vamvoudakis  
Georgia Institute of Technology, Atlanta, GA 30332, USA  
e-mail: [ariskan@gatech.edu](mailto:ariskan@gatech.edu)

K. G. Vamvoudakis  
e-mail: [kyriakos@gatech.edu](mailto:kyriakos@gatech.edu)

V. Gupta · P. Antsaklis  
University of Notre Dame, Notre Dame, IN 46556, USA  
e-mail: [vgupta2@nd.edu](mailto:vgupta2@nd.edu)

P. Antsaklis  
e-mail: [antsaklis.1@nd.edu](mailto:antsaklis.1@nd.edu)

machine-operated systems to perceive the physical world, make informed decisions, and act. Future applications of CPS will see them operating alongside human agents, which in turn will increase the need for safety guarantees, especially in critical environments, a phenomenon that has become obvious in fields that CPS are already utilized, both in civilian [3–5] and military [6] life.

One major issue that the CPS community will have to address before further integration of such systems into society will be their resilience in adversarial environments, i.e., their ability to operate under the threat of attack by malicious agents [7]. The first step toward this will be the ability to *a priori* verify whether certain properties of the system are satisfied, a problem whose difficulty is twofold; autonomous systems lie both in the continuous and the discrete domains, which makes them hybrid in nature, and their complexity often prohibits the design of model-based algorithms that require perfect knowledge of the system parameters. To solve this problem, model-free solutions offer a useful alternative to model-based design approaches since they are able to circumvent these issues.

Considering the continuous-time and state space components of CPS, we are able to derive autonomous decision-making rules as feedback policies via optimal control approaches [8]. Those policies are connected to solutions of nonlinear partial differential Hamilton–Jacobi–Bellman (HJB) equation; solutions that are difficult, if not impossible, to acquire [9]. Coupled with our lack of knowledge of the underlying dynamics of the system, the problem of optimal decision-making and control raises important technical difficulties that have been recently addressed via learning techniques. Specifically, reinforcement learning (RL) [10] is a method that enables the solution of the HJB, and thus of the optimal decision and control problem, by leveraging data gathered by the system during its operation. The underlying frameworks combined in our approach to RL are optimal and adaptive control theories [8, 11]. The first provably correct model-free RL method for continuous-time and state systems is Q-learning [12], in which the system learns an optimal action-dependent value function by utilizing “suboptimal policies.”

While learning techniques have been employed successfully in CPS applications, there are few guarantees for the behavior of systems with model-free decision-making mechanisms, especially during the learning process itself. The problem of “assured autonomy”, as it is often referred to, will investigate the methods by which we can design, implement, and maintain high-confidence, high-performance, and dynamically configured secure autonomous CPS for avoiding catastrophic failures. The complexity of an assured autonomy framework is varying since one can consider time-critical objectives that require the system to complete a series of tasks connected by logical arguments, or even take into account the existence of unpredictable human elements in the environment. Before those complex problems are investigated, we address a more basic, albeit important, issue related to safety verification; the guaranteed avoidance of unstable behaviors when a multitude of heterogeneous, learning-based components are connected to form an autonomous CPS. More than that, it is imperative that we design a mechanism that verifies whether the new component, that itself is learning-driven, can operate as part of the autonomous system.

While learning can assist in obviating the need for knowledge of the system dynamics, dissipativity, and passivity can be utilized in decentralizing the verification processes of safety-critical properties of large-scale systems, such as stability and  $L_2$ -gain stability, and, as a result, render these processes more tractable. It has long been known [13] that simple interconnections of systems, under mild conditions, maintain the dissipativity properties of their individual subsystems. Extending and merging these ideas by introducing rigorous distributed guarantees on dissipativity will enable realistic systems to acquire compositionality capabilities; where simpler, local tests will inform the operator whether a system can be integrated into a large-scale operating network.

### 10.1.1 Related Work

Q-learning has been investigated in the context of dynamical systems in [14] for discrete systems and in [15] for continuous. Therein, the underlying action-dependent value function is formulated by utilizing the Hamiltonian of the system as an advantage function. This framework has been extended to both zero-sum [16] and non-zero-sum [17] differential games. Furthermore, the introduction of integral reinforcement learning (IRL) [18] has enabled the development of online algorithms for continuous-time systems. By extension,  $H_\infty$  problems, which are known to correspond to zero-sum games have been solved in a model-free fashion via policy iteration RL techniques [19]. In this case, the derived value function of the game has been shown to be a storage function for specific  $L_2$ -gain stability conditions [13]. Similarly, the authors in [20] have employed RL to solve the Hamilton–Jacobi–Isaacs (HJI) equation for stochastic systems with Markovian jump discontinuities.

Dissipativity as an extension of stability theory for open systems, in which energy and information are transmitted between the system itself and the environment in terms of input and output signals, has been investigated for networked systems of various topologies. The authors in [21] analyze passivity—as a specific type of dissipativity—for large-scale systems in star and cyclic interconnections. Utilizing the parameters of the individual subsystems, the authors of [22] propose centralized validation tests to ensure passivity of the interconnected system. The work of [23] introduces conditions for passivity on systems in cascade interconnection, that can be validated based only on local information. This approach is further expanded upon in [24] for systems with arbitrary interconnections as well as for nonlinear systems in [25]. The interested reader is referred to [26] for a detailed analysis of the subject.

In our research direction, we leverage the work of [23] in order to develop conditions that do not require full knowledge of the individual subsystem dynamics, such that, when such subsystems are connected in a cascade configuration, dissipativity is maintained for the entire system. Initial results related to this chapter have been presented in [27].

### 10.1.2 Contributions

In this chapter, the following issues are investigated. The autonomous systems we consider are modeled as cascade interconnections of linear time-invariant subsystems. We present results via which we derive conditions to guarantee catastrophic failures due to instabilities are avoided, based on  $L_2$ -gain stability theorems. Those conditions are manipulated such that decentralized versions can be stated, which require properties of each subsystem and limited local information exchange. Afterward, by leveraging connections between Q-learning and the dissipativity conditions, we restate the  $L_2$ -gain guarantees in a model-free fashion. As such, learning-based function approximation is employed to show under which conditions on the individual subsystem behavior and the coupling matrices, the overall CPS will have the desired properties. We further show the potency of our approach by extending the framework to different passive behaviors and we show under which model-free, decentralized conditions, the system will achieve losslessness. We leverage those results to derive verification conditions that require knowledge only of the coupling parameters while guaranteeing that the overall system has the desired behavior. Finally, we discuss future directions on the verification problems that arise in autonomous CPS in adversarial environments and the ways that our optimal control-theoretic approach to learning can address them.

### 10.1.3 Structure

The remainder of the chapter is structure as follows. In Sect. 10.2, we introduce the model describing multiple heterogeneous linear time-invariant subsystems connected on a network with cascade topology. Moreover, we describe the dissipativity properties of a general system. In Sect. 10.3, we state guarantees for the  $L_2$ -gain stability of a cascade system which can be verified in a distributed manner by each subsystem. We describe the procedure with which  $L_2$ -gain stability can be verified without knowledge of the system parameters via Q-learning. In Sect. 10.5, we further explore the problem of model-free verification for lossless systems. Section 10.6, we discuss complex time-critical verification problems and the ways that a control-theoretic learning approach would address them. Finally, Sect. 10.7 concludes and discusses directions for future work.

### 10.1.4 Notation

The notation used here is standard.  $\mathbb{R}$  is the set of real numbers. The set of square integrable signals  $v(s)$ ,  $s \in K \subseteq \mathbb{R}$  is denoted by  $\mathcal{L}^2(K)$ . The Euclidean norm of a vector is denoted by  $\|\cdot\|$ . The matrix transpose is  $(\cdot)^T$  while the largest and smallest eigenvalues of a square matrix are denoted by  $\bar{\lambda}(\cdot)$  and  $\underline{\lambda}(\cdot)$ , respectively.  $\nabla_x$  and  $\frac{\partial}{\partial x}$  are used interchangeably and denote the partial derivative with respect to a vector

$x$ . The Kronecker product between two vectors is denoted by  $x \otimes y$  and the half-vectorization of a matrix  $A$ , is denoted by  $\text{vech}(A)$ .

## 10.2 Problem Formulation

Consider a network of  $N \in \mathbb{N}$  interconnected subsystems or agents. Each agent is modeled as a linear time-invariant system, evolving  $\forall t \geq 0$  according to,

$$\dot{x}_i(t) = A_i x_i(t) + B_i v_i + D_i d_i, \quad x_i(0) = x_0, \quad \forall i \in \{1, \dots, N\}, \quad (10.1)$$

where  $x_i \in \mathbb{R}^{n_i}$  denotes the  $i$ th subsystem's state,  $v_i \in \mathbb{R}^{m_i}$  the control input,  $d_i \in \mathbb{R}^{d_i}$  the external disturbance, while  $A_i \in \mathbb{R}^{n_i \times n_i}$ ,  $B_i \in \mathbb{R}^{n_i \times m_i}$  and  $D_i \in \mathbb{R}^{n_i \times d_i}$  denote the drift, input, and disturbance matrices, respectively.

Due to the feedback interconnection between the subsystems, the input  $u_i, \forall i \in \{1, \dots, N\}$  contains a coupling, such that

$$v_i = u_i + \sum_{j \in \mathcal{N}_i} H_{i,j} x_j,$$

where  $H_{i,j} \in \mathbb{R}^{n_i \times n_j}$  denotes the coupling matrix and  $u_i \in \mathbb{R}^{m_i}$  the external control input. Furthermore,  $\mathcal{N}_i$  denotes the neighborhood set of the  $i$ th agent, i.e., the agents that exchange information with the  $i$ th subsystem. Since the topology investigated in this work is the cascade interconnection, then the neighborhood of each agent  $i$  will be  $\mathcal{N}_i = \{i - 1, i, i + 1\}$ .

Now, we can rewrite the dynamics of the interconnected system in terms of a global state  $\forall t \geq 0$  as

$$\dot{x}(t) = Ax(t) + BHx(t) + Bu(t) + Dd(t), \quad x(0) = x_0, \quad (10.2)$$

where  $x = [x_1^T \ x_2^T \ \dots \ x_N^T]^T \in \mathbb{R}^{\sum_{i=1}^N n_i}$  is the global state vector,  $u = [u_1^T \ u_2^T \ \dots \ u_N^T]^T \in \mathbb{R}^{\sum_{i=1}^N m_i}$  the global control vector and, similarly,  $d = [d_1^T \ d_2^T \ \dots \ d_N^T]^T$  is the global disturbance signal. For compactness, we denote  $n = \sum_{i=1}^N n_i$ ,  $m = \sum_{i=1}^N m_i$  and  $d = \sum_{i=1}^N d_i$ .

The overall system's matrices have the following structure:

$$\begin{aligned} A &= \text{diag}(A_1, A_2, \dots, A_N) \in \mathbb{R}^{n \times n}, \\ B &= \text{diag}(B_1, B_2, \dots, B_N) \in \mathbb{R}^{n \times m}, \\ D &= \text{diag}(D_1, D_2, \dots, D_N) \in \mathbb{R}^{n \times d}. \end{aligned}$$

Finally, for the cascade interconnection system, the coupling matrix has the following block tridiagonal structure:

$$H = \begin{bmatrix} H_{1,1} & H_{1,2} & 0 & \dots & 0 & 0 \\ 0 & H_{2,1} & H_{2,2} & H_{2,3} & \dots & 0 \\ \vdots & \dots & H_{i,i-1} & H_{i,i} & H_{i,i+1} & \vdots \\ 0 & \vdots & \vdots & \vdots & \ddots & \ddots \\ 0 & 0 & 0 & \dots & H_{N,N-1} & H_{N,N} \end{bmatrix}.$$

Our objective is to develop model-free learning-based guarantees that ensure dissipativity properties of the cascaded system, with each agent utilizing information solely from their immediate neighbor. Thusly, it will be possible to synthesize large-scale autonomous systems comprised of model-free learning blocks that have been trained locally, which will satisfy certain required stability criteria.

### 10.2.1 $(Q, S, R)$ -Dissipative and $L_2$ -Gain Stable Systems

Let a system evolve according to affine in the control dynamics, and produce a measurable output signal  $\forall t \geq 0$ , as

$$\begin{aligned}\dot{z}(t) &= f(z(t)) + g(z(t))\mu, \quad z(0) = z_0, \\ y(t) &= h(z(t)),\end{aligned}$$

where  $z \in \mathbb{R}^{\tilde{n}}$  is the state of the system,  $\mu \in \mathbb{R}^{\tilde{m}}$  the control input,  $y \in \mathbb{R}^{\tilde{p}}$  the output and  $f(z) : \mathbb{R}^{\tilde{n}} \rightarrow \mathbb{R}^{\tilde{n}}$ ,  $g(z) : \mathbb{R}^{\tilde{n}} \rightarrow \mathbb{R}^{\tilde{m}}$  the drift and input dynamics, respectively. This system is  $(Q, S, R)$ -dissipative if, given a *supply rate*  $\omega : \mathbb{R}^{\tilde{p}} \times \mathbb{R}^{\tilde{m}} \rightarrow \mathbb{R}$ , with quadratic form given by

$$\omega(y, \mu) = [y^T \mu^T] \begin{bmatrix} Q & S \\ S^T & R \end{bmatrix} \begin{bmatrix} y \\ \mu \end{bmatrix}, \quad (10.3)$$

with  $Q, R \succ 0$ ,  $S \succeq 0$ , matrices of appropriate dimensions, then  $\exists V : \mathbb{R}^{\tilde{n}} \rightarrow \mathbb{R}$ ,  $V(z) \succ 0$ ,  $\forall z \in \mathbb{R}^{\tilde{n}} \setminus \{0\}$ ,  $V(0) = 0$ , named *storage function*, such that,  $\forall t_1 \geq t_0 \geq 0$ ,

$$\int_{t_0}^{t_1} \omega(y(\tau), \mu(\tau)) d\tau \geq V(z(t_1)) - V(z(t_0)). \quad (10.4)$$

In the special case when (10.4) is satisfied as an equality, the system is *conservative*. Two important supply functions are given for  $Q = 0$ ,  $R = 0$  and  $S = \frac{1}{2}I$ . If a system satisfies (10.4) for those parameters, it is called *passive*. Further addition of the term  $-\epsilon \|x\|^2$ ,  $\epsilon > 0$  on the passivity supply rate renders the system *state-strictly passive*. In this work, the terms passivity and state-strictly passivity are used interchangeably. We note that conservative systems with respect to the passivity supply rate are called

*lossless*. Similarly, a system satisfying (10.4) with  $Q = I$ ,  $R = -\frac{1}{2}\gamma^2 I$ ,  $\gamma > 0$  and  $S = 0$  is called  *$L_2$ -gain stable*, with  $L_2$  gain of  $\gamma$ .

Let us assume a system that is  $L_2$ -gain stable over the time interval  $t \in [0, t_f]$ , with  $t_f \rightarrow \infty$ . Furthermore, let the storage function that satisfies (10.4) for the  $L_2$ -gain supply rate be smooth. Then, we can take the differential form of (10.4), which yields

$$\dot{V}(t) - \|y(t)\|^2 - \gamma^2 \|\mu(t)\|^2 \leq 0, \quad t \geq 0.$$

It is known [13], that  $L_2$ -gain stability with  $L_2$  gain equal to  $\gamma$  corresponds to the following condition on the input–output properties of the system;

$$\int_t^\infty \|y(\tau)\|^2 d\tau \leq \int_t^\infty \gamma^2 \|\mu(\tau)\|^2 d\tau, \quad t \geq 0.$$

Contrary to passivity and general  $(Q, S, R)$ -dissipativity conditions, which rely on inequality verification, or, in the case of linear systems, on verification of linear matrix inequalities (LMIs),  $L_2$ -gain stability is known to correspond, via Bellman's principle of optimality, to the solution of a Hamilton–Jacobi equation [13]. Accordingly, for a system to be  $L_2$ -gain stable, then,  $\exists V_a : \mathbb{R}^{\tilde{n}} \rightarrow \mathbb{R}$ , smooth function of the state, the satisfies the equation  $\forall z \in \mathbb{R}^{\tilde{n}}$ ,

$$\left( \frac{\partial V_a}{\partial z} \right)^T (f(z) + g(z)\mu^*(z)) + h^T(z)h(z) - \gamma^2 \mu^{(\star T)}(z)\mu^*(z) = 0, \quad (10.5)$$

where  $\mu^*(z)$  corresponds to the worst-case input for which the system remains  $L_2$ -gain stable, expressed as

$$\mu^*(z) = \frac{1}{\gamma^2} g^T(z) \left( \frac{\partial V_a}{\partial z} \right).$$

For linear systems, such as the one described in (10.2), the storage function is known to be quadratic in the states, i.e.,  $V_a(z) = z^T P z$ . Consequently, Eq.(10.5) can be rewritten as a matrix algebraic Riccati equation,

$$A^T P + P A + C^T C + \frac{1}{\gamma^2} P D D^T P = 0. \quad (10.6)$$

### 10.3 Learning-Based Distributed Cascade Interconnection

Given the system (10.2), we first show that under limited communication with only the preceding subsystem in the cascade interconnection, it is possible to enable each agent to verify whether her addition to the network will retain the required  $L_2$  gain of the overall system. Toward that, initially, we state the following proposition,

**Proposition 10.1** A symmetric hollow matrix  $N \in \mathbb{R}^{k \times k}$  can either be indefinite or the zero matrix.  $\square$

**Proof** Let  $\sigma(N)$  be the point spectrum of  $N$ . Then, it is known that  $\text{tr}(N) = \sum_{i=1}^k \lambda_i$ ,  $\forall \lambda_i \in \sigma(N)$ ,  $i \in \{1, \dots, k\}$ . Thus, since  $\text{tr}(N) = 0$ , then  $\exists \lambda_1, \lambda_2 \in \sigma(N)$  such that  $\lambda_1 \lambda_2 < 0$ , or  $\lambda_i = 0$ ,  $\forall i \in \{1, \dots, k\}$ , which completes the proof.  $\blacksquare$

Now, we can state the following theorem for conditions of  $L_2$ -gain stability based on local information:

**Theorem 10.1** The cascade interconnection described by (10.2) is  $L_2$ -gain stable with  $L_2$  gain  $\gamma$ , if each subsystem of the network is  $L_2$ -gain stable with the same gain and storage function given by  $V_i(x) = x_i^T P_i x_i$ , and it holds that

$$H_{i,i+1}^T B_i^T P_i + P_{i+1} B_{i+1} H_{i+1,i} = 0, \quad \forall i \in \{1, \dots, N-1\}. \quad (10.7)$$

**Proof** Consider the interconnected system as given in (10.2) without external control input, i.e.,  $u(t) = 0$ . Assume the existence of a smooth storage function which is quadratic in the global state,  $V(x) = \frac{1}{2}x^T P x$ , such that  $P = \text{diag}(P_1, P_2, \dots, P_N)$ , which renders the system  $L_2$ -gain stable and take the infinitesimal version of (10.4), such that

$$\begin{aligned} & -\|y\|^2 + \gamma^2 \|d\|^2 - \dot{V} \geq 0, \quad t \geq 0 \\ \Rightarrow & -x^T C^T C x + \gamma^2 d^T d - \frac{1}{2} (Ax + BHx + Dd)^T P x \\ & - \frac{1}{2} x^T P (Ax + BHx + Dd) \geq 0 \\ \Rightarrow & -\frac{1}{2} [x^T \quad d^T] \Gamma \begin{bmatrix} x \\ d \end{bmatrix}, \end{aligned}$$

where the kernel  $\Gamma$  is given by

$$\Gamma = \begin{bmatrix} (A + BH)^T P + P(A + BH) + 2C^T C & PD \\ D^T P & -\gamma^2 I \end{bmatrix}.$$

The system is  $L_2$ -gain stable if  $\Gamma \preceq 0$ . Taking the Schur complement of  $\Gamma$ , yields the following conditions for negative semidefiniteness on the blocks of the matrix:

$$-\gamma^2 I \preceq 0, \quad (10.8a)$$

$$2C^T C + (A + BH)^T P + P(A + BH) + \frac{1}{\gamma^2} P D D^T P \preceq 0. \quad (10.8b)$$

It can be seen that (10.8a) is trivially satisfied. Focusing our attention on (10.8b), we will leverage the structure of the coupling matrix  $H$ , to restate the conditions of the inequality in terms of the subsystem dynamics given by (10.1). The resulting expansion of the matrix  $\Gamma$  is given by

$$\Gamma = \begin{bmatrix} M_1 & K_{1,2} & 0 & \dots & \dots & 0 \\ K_{2,1} & M_2 & K_{1,3} & 0 & \dots & \vdots \\ 0 & \ddots & \ddots & \ddots & \dots & \vdots \\ \vdots & \dots & K_{i,i-1} & M_i & K_{i+1,i} & \vdots \\ \vdots & \dots & \vdots & \ddots & \ddots & \vdots \\ 0 & \dots & \dots & 0 & K_{N,N-1} & M_N \end{bmatrix}, \quad (10.9)$$

where the diagonal blocks are given as

$$M_i = (A_i + B_i H_{i,i})^T P_i + P_i (A_i + B_i H_{i,i}) + 2C_i^T C_i + \frac{1}{\gamma^2} P_i D_i^T D_i P_i, \quad (10.10)$$

and the off-diagonal coupling elements as

$$K_{j,k} = H_{k,j}^T B_j^T P_j + P_k B_k H_{k,j}, \quad \forall k, j \in \{2, \dots, N\}, |k - j| = 1.$$

We can notice that the blocks  $M_i, i \in \{1, \dots, N\}$ , correspond to Eq. (10.6) for the linear  $i$ th subsystem. Thus, if each subsystem is  $L_2$ -gain stable, with  $L_2$ -gain of  $\gamma$ , then  $M_i = 0, \forall i \in \{1, \dots, N\}$ .

Consequently, (10.9) will be a symmetric hollow matrix, and, due to Proposition 10.1, we can ensure that the cascade interconnection is dissipative only if  $\Gamma = 0$ . This holds if (10.7) is satisfied. ■

## 10.4 Learning-Based $L_2$ -Gain Composition

### 10.4.1 Q-Learning for $L_2$ -Gain Verification

Following the model-free methods developed in [15], we shall utilize continuous-time Q-learning techniques to validate the conditions presented in Theorem 10.1, without knowledge of the subsystem dynamics. However, since each system is trained a priori, it will be shown that the verification guarantees will require knowledge of the coupling matrices between the components.

Consider the problem of finding a proper storage function solved by an individual subsystem. For ease of exposition, we shall introduce the notation  $V_a(\cdot)$  and  $H(x, d, \nabla V_a)$  to mean the storage function and Hamiltonian, i.e., the expression given by the right-hand side of (10.5) as a function of the state, disturbance, and the storage function, respectively.

The Q-function is defined as

$$Q_i(x_i, d_i) = V_i(x_i) + H(x_i, d_i, \nabla V_i). \quad (10.11)$$

Substituting the Hamiltonian with the one given by for the linear system (10.10), we can reformulate the Q-function (10.11) compactly, as a quadratic in the state and disturbance, function,

$$Q_i(x_i, d_i) = U_i^T \begin{bmatrix} Q_{i,xx} & Q_{i,xd} \\ Q_{i,dx} & Q_{i,dd} \end{bmatrix} U_i = U_i^T \tilde{Q}_i U_i, \quad (10.12)$$

where we have that  $U_i = [x_i^T \ d_i^T]^T$ , and the block elements of  $\tilde{Q}_i$  are given as

$$\begin{aligned} Q_{i,xx} &= P_i + (A_i + B_i H_{i,i})^T P_i + P_i (A_i + B_i H_{i,i}) + 2C_i^T C_i, \\ Q_{i,xd} &= P_i D_i, \quad Q_{i,dx} = D_i^T P_i, \quad Q_{i,dd}^k = \frac{1}{\gamma^2} I. \end{aligned}$$

The worst-case disturbance can be found by taking into account the optimality conditions of the Q-function as

$$\frac{\partial Q_i(x_i, d_i)}{\partial d_i} = 0,$$

which yields the following expression:

$$d_i(x) = -(Q_{i,dd})^{-1} Q_{i,dx} x. \quad (10.13)$$

An IRL scheme that employs two approximators, an actor and a critic, can be used to enable synchronous online learning [28]. The storage function critic approximator approximates the Q-function (10.12), while the actor approximates the worst-case disturbance for which the system remains  $L_2$ -gain stable, given by (10.13).

In order to do that, we write the Q-function as linear with respect to a function basis constructed by the Kronecker product of the joint state/disturbance vector,

$$Q_i(x_i, d_i) = \text{vech}(\tilde{Q}_i)^T (U_i \otimes U_i), \quad (10.14)$$

where the Kronecker product-based polynomial quadratic polynomial function ( $U_i \otimes U_i$ ) is reduced to guarantee linear independence of the elements.

The vectorized Q-function (10.14) can be described in terms of the ideal weights  $W_i = \text{vech}(\tilde{Q}_i)$ , leading to the compact form  $Q_i(x_i, d_i) = W_i^T (U_i \otimes U_i)$ .

**Remark 10.1** Since for the linear subsystem (10.1), it is known that we can find storage functions that are quadratic in the states, then for the ideal weights  $W_i$ , we will have an exact basis representation on the joint state and disturbance space. Consequently, there are no approximation errors for the actor or the critic.  $\square$

Since the ideal weights are unknown, we will consider estimates of the Q-function, according to the estimated critic weights,  $\hat{W}_i = \text{vech}(\hat{\tilde{Q}}_i)$ ,

$$\hat{Q}_i(x_i, d_i) = \hat{W}_i^T (U_i \otimes U_i), \quad (10.15)$$

as well as the estimated actor approximator,

$$\hat{d}_i(x_i) = \hat{W}_{i,d}^T x_i, \quad (10.16)$$

where the state  $x$  serves as the basis for the actor approximator and  $\hat{W}_{i,d}$  denotes the weight estimate of the worst-case disturbance.

The Q-function with the ideal weights, (10.12), has been shown [15] to satisfy the integral form of the following Hamilton–Jacobi equation:

$$\begin{aligned} Q_i(x_i(t), d_i(t)) &= Q_i(x_i(t - T_{\text{IRL}}), d_i(t - T_{\text{IRL}})) \\ &\quad - \int_{t-T_{\text{IRL}}}^t (2x_i^T C_i^T C_i x_i + d_i^T d_i) d\tau, \end{aligned}$$

over every time interval  $[t, t + T_{\text{IRL}}]$ , where  $T_{\text{IRL}} > 0$ .

Subsequently, the Bellman error based on the current estimate of the Q-function which we wish to drive to zero is defined as

$$\begin{aligned} e_i &= \hat{Q}_i(x_i(t), d_i(t)) - \hat{Q}_i(x_i(t - T_{\text{IRL}}), d_i(t - T_{\text{IRL}})) \\ &\quad + \int_{t-T_{\text{IRL}}}^t (2x_i^T C_i^T C_i x_i + \gamma^2 d_i^T d_i) d\tau \\ &= \hat{W}_i^T (U_i(t) \otimes U_i(t)) - \hat{W}_i^T (U_i(t - T_{\text{IRL}}) \otimes U_i(t - T_{\text{IRL}})) \\ &\quad + \int_{t-T_{\text{IRL}}}^t (2x_i^T C_i^T C_i x_i + \gamma^2 d_i^T d_i) d\tau, \end{aligned} \quad (10.17)$$

as well as the policy error,

$$e_{i,d} = W_{i,d}^T x_i + \hat{Q}_{i,dd}^{-1} \hat{Q}_{i,dx} x_i,$$

where the appropriate elements of the  $\hat{Q}_i$  matrix will be extracted from the critic estimate  $\hat{W}_i$ .

Defining the squared error functions for the critic weights  $K_1 = \frac{1}{2} \|e_i\|^2$  and the actor weights  $K_2 = \frac{1}{2} \|e_{i,d}\|^2$ , we construct the tuning rules by applying a modified Levenberg–Marquardt [11] as

$$\dot{\hat{W}}_i = -\alpha \frac{\sigma_i}{(1+\sigma_i^T \sigma_i)^2} e_i^T, \quad (10.18)$$

$$\dot{\hat{W}}_{i,d} = -\alpha_a x_i e_{i,d}^T, \quad (10.19)$$

where  $\sigma_i = (U_i(t) \otimes U_i(t)) - (U_i(t - T_{\text{IRL}}) \otimes U_i(t - T_{\text{IRL}}))$ , and  $\alpha, \alpha_a \in \mathbb{R}^+$  are the tuning gains.

For the purposes of stating convergence guarantees of the learning algorithm, the following definition on persistence of excitation is needed:

**Definition 10.1** A signal  $p(s) \in \mathcal{L}^2(\mathbb{R})$  is persistently excited over a time interval  $[s, s + T^{\text{exc}}]$ ,  $T^{\text{exc}} > 0$  if  $\exists \beta_1^{\text{exc}}, \beta_2^{\text{exc}} > 0$  such that  $\beta_1^{\text{exc}} I \leq \int_t^{t+T^{\text{exc}}} p(\tau) p^T(\tau) d\tau \leq \beta_2^{\text{exc}} I$ , where  $I$  is of appropriate dimensions.  $\square$

**Lemma 10.1** Consider the problem of determining  $L_2$ -gain stability of the subsystem described by (10.1). The approximation of the storage function is achieved by the  $Q$ -function approximator given by (10.15) and the worst-cased disturbance by (10.16). The tuning law for the critic is given by (10.18) and for the actor by (10.19). Then the closed-loop system of the augmented state  $\psi_i = [x_i^T (\hat{W}_i - W_i)^T (\hat{W}_{i,d} - W_{i,d})^T]^T$  is asymptotically stable at the origin if the critic gain is picked sufficiently larger than the actor gain, the signal  $\sigma_i$  satisfies persistence of excitation conditions, and,

$$1 < \alpha_a < \frac{\gamma^2}{\delta} \left( 2\underline{\lambda}(2C_i^T C_i + \frac{1}{\gamma^2} Q_{i,xd} Q_{i,xd}^T) - \bar{\lambda}(Q_{i,xd} Q_{i,xd}^T) \right),$$

where  $\delta \in \mathbb{R}^+$ .

**Proof** The proof follows from [15].  $\blacksquare$

**Remark 10.2** Ensuring persistence of excitation of a signal apriori is a difficult task. To circumvent this, we add a small probing noise as an input to the system that enforces adequate exploration of the state space.  $\square$

Now, given the learned values of the  $\hat{Q}_i(x, d)$  matrix, it is possible to restate the conditions of Theorem 10.1 in terms of block elements of the  $Q$ -function.

**Theorem 10.2** Given the  $i$ th subsystem whose evolution is described by (10.1), the supply rate given by (10.3), with  $Q = -I$ ,  $S = 0$  and  $R = \frac{1}{\gamma^2} I$  and the approximate  $Q$ -function given in (10.15), then the cascade system is  $L_2$ -gain stable if the conditions for Lemma 10.1 hold and,

$$H_{i,i+1}^T \hat{Q}_{i,xd} + \hat{Q}_{i+1,dx} H_{i+1,i}^T = 0, \quad \forall i \in \{1, \dots, N-1\}.$$

**Proof** The proof follows by direct substitution of the provably convergent  $Q$ -function kernel (10.15), into the HJI equation (10.10) and the condition given by (10.7).  $\blacksquare$

**Remark 10.3** An underlying assumption in this section, was that each subsystem achieved a specific  $L_2$  gain equal to  $\gamma$ , which was common to all subsystems and their cascade interconnection. While it is beyond the scope of our verification objectives, it is straightforward to design, via Q-learning, the external input  $u_i$  to each subsystem such that the predefined global  $L_2$  gain is achieved by every subsystem. In this case, the verification equation (10.5) will become a zero-sum game Riccati equation [13].  $\square$

**Remark 10.4**  $L_2$ -gain stability is known to render the overall system stable under mild zero-state observability conditions on the output function  $h(x)$ —or, on the output matrix  $C$  for a linear system. However, decentralized  $L_2$ -gain stability requires the existence of a storage function that satisfies Eq. (10.7), which can be restrictive.  $\square$

### 10.4.2 $L_2$ -Gain Model-Free Composition

Given the results of the model-free  $L_2$ -gain verification, in Algorithm 10.1, we describe the procedure by which subsystems can be appended to the cascade system.

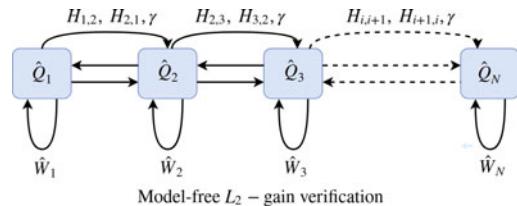
**Algorithm 10.1** Composition of Learning-based Systems with Guaranteed  $L_2$ -gain stability

```

1: procedure
2:   Given a required level of  $L_2$ -gain,  $\gamma$ , and initial estimates of  $\hat{W}_1, \hat{W}_{1,d}$ .
3:   Gather local data of the trajectories of  $x_1(t), d_1(t)$ .
4:   Construct the integral reinforcement error  $e_1(t)$  given by (10.17).
5:   Update the critic and actor weights using Eqs. (10.18) and (10.19), respectively.
6:   Upon convergence to the ideal weights, extract  $\bar{Q}_i$ .
7:   for  $i = 1, \dots$ 
8:     Receive required  $L_2$  gain,  $\gamma$  from the  $i$ th subsystem.
9:     Repeat steps 1 – 6 for the subsystem  $i + 1$ .
10:    Extract  $\bar{Q}_{i+1}$ 
11:    Exchange local information with  $i$ th subsystem by acquiring its  $Q_{i,\text{xd}}, H_{i,i+1}$ 
      matrices.
12:    Verify conditions of Theorem 10.2.
13:    if True
14:      Append  $i + 1$  system to the cascade interconnection.
15:    end if
16:   end for
17: end procedure
```

Figure 10.1 presents the flow of information between the model-free subsystems of the cascade interconnection.

**Fig. 10.1** Cascade interconnection of linear subsystems equipped with learning capabilities



## 10.5 Learning-Based Lossless Composition

In this section, we leverage the results of [15] to develop model-free verification for lossless systems. We note that by following procedures similar to the ones presented in Sect. 10.4.1, the following Lemma can be proven.

**Lemma 10.2** *Consider the system given by (10.2), consisting of the subsystems (10.1). For the supply rate (10.3) with  $Q = 0$ ,  $R = 0$ , and  $S = \frac{1}{2}I$ , the system is lossless if  $\exists \epsilon_i > 0$ , such that  $\exists \tilde{P}_i \in \mathbb{R}^{n_i \times n_i}$ ,  $\forall i \in \{1, \dots, N\}$ , with,  $P_i \succ 0$  and with the following conditions being satisfied:*

$$\begin{aligned} 0 &= (A_i + BH_{i,i})^T \tilde{P}_i + \tilde{P}_i (A_i + BH_{i,i}) + \epsilon_i I, \quad i \in \{1, \dots, N\}, \\ 0 &= H_{j,j+1}^T B_j^T \tilde{P}_j + \tilde{P}_{j+1} B_{j+1} H_{j+1,j}, \quad j \in \{1, \dots, N-1\}. \end{aligned} \quad (10.20)$$

**Proof** The proof follows directly from [23] for a lossless system as defined in Sect. 10.2.1. ■

Similar to Sect. 10.4.1, we shall utilize learning structures to validate the conditions of Lemma 10.2. However, in the case of lossless systems, it is sufficient to employ a single critic network, to approximate the solution of (10.20) and verify its sign definiteness. Toward that, we note that (10.20) can be written as

$$\frac{d}{dt} x_i^T \tilde{P}_i x_i = -\epsilon_i x_i^T x_i, \quad (10.21)$$

where  $U_i(x_i) := x_i^T \tilde{P}_i x_i$ ,  $\forall i \in \{1, \dots, N\}$  is a storage function for the  $i$ th lossless subsystem.

Now, we introduce a critic approximator that will be used to learn the storage function of the lossless system. Since the quadratic in the states structure of the function is known, we can expand it over a function basis constructed by the independent elements of the Kronecker product of the state

$$\mathcal{V}_i(x_i) = \mathcal{W}_i^T(x_i \otimes x_i), \quad \forall i \in \{1, \dots, N\},$$

where  $\mathcal{W}_i = \text{vech}(\tilde{P}_i)$ , are the ideal weights for the storage function  $\mathcal{V}_i(x_i)$ , which are unknown. Furthermore, define the estimation of the storage function

$$\hat{\mathcal{V}}_i(x_i) = \hat{\mathcal{W}}_i^T(t)(x_i \otimes x_i), \quad \forall i \in \{1, \dots, N\}, \quad (10.22)$$

where  $\hat{\mathcal{W}}_i(t)$  is the current estimate of the critic weights.

In order to tune the critic approximator, we leverage data gathered along the subsystem trajectory. Toward that, we define an integral reinforcement error, by considering the evolution of (10.21) over a predefined time period  $[t - T_{\text{IRL}}, t]$ . Consequently, we have

$$\begin{aligned} e_i^p(t) = & \mathcal{W}_i^T \left( (x_i(t) \otimes x_i(t)) - (x_i(t - T_{\text{IRL}}) \otimes x_i(t - T_{\text{IRL}})) \right), \\ & + \int_{t-T_{\text{IRL}}}^t \epsilon_i x_i^T(\tau) x_i(\tau) d\tau. \end{aligned} \quad (10.23)$$

Now, as in the case of the  $L_2$ -gain stability storage function, we define an error function  $K^p(t) = \frac{1}{2} \|e_i^p(t)\|^2$ , which we minimize. Thus, we employ the modified Levenberg–Marquardt algorithm for normalized gradient descent, that yields the tuning law for the estimated critic weight,

$$\dot{\hat{\mathcal{W}}}(t) = -\alpha_i^p \frac{\tilde{\sigma}_i}{(1 + \tilde{\sigma}_i^T \tilde{\sigma}_i)^2} (e_i^p)^T. \quad (10.24)$$

The following theorem describes the conditions for the cascade system to be lossless, given information on the coupling matrices and local behavior.

**Theorem 10.3** *Given the  $i$ th subsystem whose evolution is described by (10.1), the supply rate given by (10.3), with  $Q = 0$ ,  $S = \frac{1}{2}I$  and  $R = 0$  and the approximate storage function given in (10.22), then the cascade system is lossless if*

$$H_{j,j+1}^T B_j^T \hat{P}_j + \hat{P}_{j+1} B_{j+1} H_{j+1,j} = 0, \quad \forall i \in \{1, \dots, N-1\},$$

where the estimate of the storage function kernel  $\hat{P}_i$ ,  $\forall i \in \{1, \dots, N\}$  is constructed from the elements of the weights  $\hat{\mathcal{W}}$  after convergence.

**Proof** The proof follows directly by taking (10.22) into account on the conditions of 10.2. ■

**Remark 10.5** We notice that in order to guarantee that the system is lossless, we require knowledge of the input matrices of each subsystem, since we do not explicitly gather data from any sort of input signals. However, this compromise allows us to consider more general types of dissipativity which might offer verification guarantees for general safety critical properties. As an example, state-strictly passivity, such as the one considered here, can be used to provide guarantees for nonlinear systems based on the properties of its linearization. □

In Algorithm 10.2, we describe the validation procedure which guarantees that the cascade interconnection will be lossless.

### Algorithm 10.2 Composition of Learning-based Lossless Systems

- 
- 1: **procedure**
  - 2:    Given initial estimates of  $\hat{\mathcal{W}}$ .
  - 3:    Gather local data of the trajectories of  $x_1(t)$ .
  - 4:    Construct the integral reinforcement error  $e_1^p(t)$  given by (10.23).
  - 5:    Update the critic weights using Eq. (10.24).
  - 6:    Upon convergence to the ideal weights, extract  $\tilde{P}_i$ .

```

7:   for  $i = 1, \dots$ 
8:     Repeat steps 1 – 6 for the subsystem  $i + 1$ .
9:     Extract  $\tilde{P}_{i+1}$ 
10:    Receive required coupling matrix from an immediate neighbor.
11:    Verify conditions of Theorem 10.2.
12:    if True
13:      Append  $i + 1$  system to the cascade interconnection.
14:    end if
15:  end for
16: end procedure

```

---

## 10.6 Discussion

In this section, we showcase the connections between our model-free RL methodology and complex verification problems. Specifically, we consider high-level specifications expressed in a formal language. The storage functions learned in the previous sections have allowed us to derive conditions of dissipativity; similarly, we encode the specifications into cost functions that will be solved via learning methods. One important challenge will be to choose appropriate logic frameworks [29, 30] that will be able to express probabilistic predicates in continuous-space and actions.

One such framework is the formal language of probabilistic signal temporal logic (PSTL) [31], which will allow specifying properties of real-valued signals. We will be, thus, able to define quantitative semantics for PSTL which will be used to define robustness measures of specification violation. The syntax for PSTL is given by

$$\phi := \lambda(x) < d \mid \neg\phi \mid \phi \wedge \phi \mid \phi \vee \phi \mid F_{[a,b]}\phi \mid G_{[a,b]}\phi \mid \mathcal{P}_\mu(\phi),$$

where  $x : \mathbb{R}^+ \rightarrow \mathbb{R}^n$  is a continuous-time signal,  $\lambda : \mathbb{R}^n \rightarrow \mathbb{R}$ ,  $d \in \mathbb{R}$  a constant and  $a, b \in \mathbb{R}^+ \cup \{\infty\}$  are positive time indices. The Boolean operators  $\neg$ ,  $\wedge$ ,  $\vee$  are the negation, disjunction, and conjunction logical connectives. The operators  $F$  and  $G$  are the temporal logic operators “Eventually” and “Always”, respectively. Finally, the operator  $\mathcal{P}_\mu(\cdot)$  is used to define the probability of satisfying a formula with value  $\mu$ . The use of those tools will allow us to deal with specifications of the CPS state  $x(t)$  that capture mission objectives that might be time-critical or safety-related. An example of such a predicate would be to enforce the system state  $x(t)$  to always avoid certain areas that might be populated by humans, i.e.,  $G_{[0,\infty)}(Ax < d)$ , where the area to be avoided is a polytope described by  $Ax = d$ ,  $A \in \mathbb{R}^{n \times n}$ .

The first step in bringing our Q-learning tools to the problem of verifying complex specifications encoded via the PSTL language, is to define appropriate reward functions in a continuous space that measure the level of satisfaction of those specifications. As such, the following robustness degree functions can be defined:

$$\begin{aligned}
r(x, \lambda(x) < d, t) &= d - \lambda(x(t)), \\
r(x, \phi_1 \wedge \phi_2, t) &= \min(r(x, \phi_1, t), r(x, \phi_2, t)), \\
r(x, G_{[a,b]}\phi, t) &= \min_{\tau \in [t+a, t+b]} r(x, \phi, \tau), \\
r(x, F_{[a,b]}\phi, t) &= \max_{\tau \in [t+a, t+b]} r(x, \phi, \tau).
\end{aligned}$$

The problem of verifying the required specifications can now be quantified via the following value function:

$$V(x) = \min_u \int_{t_0}^T r(x, \cdot, t) dt.$$

Consequently, noting the similarities in the structure of the value function and the storage functions investigated above, we conclude that the model-free methodologies proposed for the dissipativity verification problems can be extended to complex mission objectives.

## 10.7 Conclusion and Future Work

In this work, we presented preliminary results on learning-based verification for autonomous large-scale cascade systems. In order to tackle the practical complexity of validating certain stability-related properties of these systems, such as dissipativity with respect to external disturbances, we derived decentralized conditions that leverage the specific structure of the coupling of the subsystems. Specifically, conditions on the storage function of each individual subsystem were proposed, such that the overall system is  $L_2$ -gain stable, given a predetermined required  $L_2$  gain for the interconnection. Subsequently, we introduced an actor/critic structure that is able to derive those conditions given explicit information only on the evolution of the individual subsystems and the corresponding coupling matrices. Utilizing this, we showed how the cascade network can be extended indefinitely. Finally, we developed guarantees that render the cascade interconnection lossless given only information on the coupling matrices.

Future work will propose relaxed conditions for dissipativity verification based on RL. More general network topologies will also be considered. Finally, we will focus on the verification of more general safety-related properties and propose solutions to the problem presented in Sect. 10.6.

**Acknowledgements** This work was supported in part, by ONR Minerva under grant No. N00014-18-1-2160, by NATO under grant No. SPS G5176, by an NSF CAREER under grant No. CPS-1851588, and by NSF S&AS under grant No. 1849198.

## References

1. Rajkumar, R.R., Lee, I., Sha, L., Stankovic, J.: Cyber-physical systems: the next computing revolution. In: Proceedings of the 47th Design Automation Conference, pp. 731–736. ACM (2010)
2. Vamvoudakis, K.G., Antsaklis, P.J., Dixon, W.E., Hespanha, J.P., Lewis, F.L., Modares, H., Kiumarsi, B.: Autonomy and machine intelligence in complex systems: a tutorial. In: 2015 American Control Conference (ACC), pp. 5062–5079. IEEE (2015)
3. Lee, I., Sokolsky, O.: Medical cyber physical systems. In: Design Automation Conference (DAC), 2010 47th ACM/IEEE, pp. 743–748. IEEE (2010)
4. Mo, Y., Kim, T.H.-J., Brancik, K., Dickinson, D., Lee, H., Perrig, A., Sinopoli, B.: Cyber-physical security of a smart grid infrastructure. Proc. IEEE **100**(1), 195–209 (2012)
5. Kim, J., Kim, H., Lakshmanan, K., Rajkumar, R.R.: Parallel scheduling for cyber-physical systems: analysis and case study on a self-driving car. In: Proceedings of the ACM/IEEE 4th International Conference on Cyber-Physical Systems, pp. 31–40. ACM (2013)
6. Liu, Z., Yang, D.-S., Wen, D., Zhang, W.-M., Mao, W.: Cyber-physical-social systems for command and control. IEEE Intell. Syst. **26**(4), 92–96 (2011)
7. Cardenas, A.A., Amin, S., Sastry, S.: Secure control: towards survivable cyber-physical systems. In: 2008 The 28th International Conference on Distributed Computing Systems Workshops, pp. 495–500. IEEE (2008)
8. Lewis, F.L., Vrabie, D., Syrmos, V.L.: Optimal Control. Wiley, Hoboken (2012)
9. Lewis, F.L., Vrabie, D., Vamvoudakis, K.G.: Reinforcement learning and feedback control: using natural decision methods to design optimal adaptive controllers. IEEE Control. Syst. Mag. **32**(6), 76–105 (2012)
10. Sutton, R.S., Barto, A.G.: Reinforcement Learning: An Introduction. MIT Press, Cambridge (2018)
11. Ioannou, P.A., Sun, J.: Robust Adaptive Control. Courier Corporation, North Chelmsford (2012)
12. Watkins, C.J., Dayan, P.: Q-learning. Mach. Learn. **8**(3–4), 279–292 (1992)
13. van der Schaft, A.J., Van Der Schaft, A.: L2-Gain and Passivity Techniques in Nonlinear Control, vol. 2. Springer, Berlin (2000)
14. Kiumarsi, B., Lewis, F.L., Modares, H., Karimpour, A., Naghibi-Sistani, M.-B.: Reinforcement q-learning for optimal tracking control of linear discrete-time systems with unknown dynamics. Automatica **50**(4), 1167–1175 (2014)
15. Vamvoudakis, K.G.: Q-learning for continuous-time linear systems: a model-free infinite horizon optimal control approach. Syst. Control Lett. **100**, 14–20 (2017)
16. Al-Tamimi, A., Lewis, F.L., Abu-Khalaf, M.: Model-free q-learning designs for linear discrete-time zero-sum games with application to h-infinity control. Automatica **43**(3), 473–481 (2007)
17. Vamvoudakis, K.G.: Non-zero sum Nash q-learning for unknown deterministic continuous-time linear systems. Automatica **61**, 274–281 (2015)
18. Modares, H., Lewis, F.L., Naghibi-Sistani, M.-B.: Integral reinforcement learning and experience replay for adaptive optimal control of partially-unknown constrained-input continuous-time systems. Automatica **50**(1), 193–202 (2014)
19. Abu-Khalaf, M., Huang, J., Lewis, F.L.: Nonlinear H2/H-Infinity Constrained Feedback Control: A Practical Design Approach Using Neural Networks. Springer Science & Business Media, Berlin (2006)
20. Vamvoudakis, K.G., Safaei, F.R.P.: Stochastic zero-sum Nash games for uncertain nonlinear Markovian jump systems. In: 2017 IEEE 56th Annual Conference on Decision and Control (CDC), pp. 5582–5589. IEEE (2017)
21. Ghanbari, V., Wu, P., Antsaklis, P.J.: Large-scale dissipative and passive control systems and the role of star and cyclic symmetries. IEEE Trans. Autom. Control **61**(11), 3676–3680 (2016)
22. Vidyasagar, M.: New passivity-type criteria for large-scale interconnected systems. IEEE Trans. Autom. Control **24**(4), 575–579 (1979)

23. Agarwal, E., Sivarajanji, S., Gupta, V., Antsaklis, P.J.: Sequential synthesis of distributed controllers for cascade interconnected systems. In: 2019 American Control Conference (ACC), pp. 5816–5821. IEEE (2019)
24. Agarwal, E., Sivarajanji, S., Gupta, V., Antsaklis, P.J.: Distributed synthesis of local controllers for networked systems with arbitrary interconnection topologies. *IEEE Trans. Autom. Control* (2020)
25. Agarwal, E., Sivarajanji, S., Gupta, V., Antsaklis, P.J.: Compositional verification of passivity for cascade interconnected nonlinear systems. In: 2020 28th Mediterranean Conference on Control & Automation (MED). IEEE (2019)
26. Agarwal, E.: Compositional Control of Large-Scale Cyber-Physical Systems Using Hybrid Models and Dissipativity Theory. University of Notre Dame (2019)
27. Kanellopoulos, A., Vamvoudakis, K.G., Gupta, V.: Decentralized verification for dissipativity of cascade interconnected systems. In: 2019 IEEE 58th Conference on Decision and Control (CDC), pp. 3629–3634. IEEE (2019)
28. Vamvoudakis, K.G., Lewis, F.L.: Online actor-critic algorithm to solve the continuous-time infinite horizon optimal control problem. *Automatica* **46**(5), 878–888 (2010)
29. Yoo, C., Belta, C.: Control with probabilistic signal temporal logic (2015). arXiv preprint [arXiv:1510.08474](https://arxiv.org/abs/1510.08474)
30. Sadigh, D., Kapoor, A.: Safe control under uncertainty (2015). arXiv preprint [arXiv:1510.07313](https://arxiv.org/abs/1510.07313)
31. Abbas, H., Fainekos, G., Sankaranarayanan, S., Ivaničić, F., Gupta, A.: Probabilistic temporal logic falsification of cyber-physical systems. *ACM Trans. Embed. Comput. Syst. (TECS)* **12**(2s), 1–30 (2013)

# Chapter 11

## Reinforcement Learning-Based Model Reduction for Partial Differential Equations: Application to the Burgers Equation



Mouhacine Benosman, Ankush Chakrabarty, and Jeff Borggaard

### 11.1 Introduction

Partial differential equations (PDEs) are important mathematical models, which are used to model complex dynamic systems in applied sciences. For example, PDEs are used to model flexible beams and ropes [1, 2], crowd dynamics [3, 4], or fluid dynamics [5, 6]. However, PDEs are infinite-dimensional systems, making them hard to solve in closed form, and computationally demanding to solve numerically. For instance, when using finite element methods (FEM), one may end up with a very large discretization space, which incurs large computation times. Because of this complexity, it is often hard to use PDEs to analyze, predict, or control these systems in real time. Instead, one approach that is often used in real applications is to first reduce the PDE model to an ordinary differential equation (ODE) model, which has a finite dimension, and then use this ODE for system identification, estimation, and control. This step of converting a PDE to a reduced-order model (ROM) ODE, while maintaining a small error between the solutions of both models, is known as stable model reduction.

In this work, we focus on ROMs where modes are obtained using proper orthogonal decomposition (POD) [7], which has widely been used to obtain surrogate models of tractable size in fluid flow applications. However, it has been observed, for example, in [8–12], that POD-ROMs do not necessarily preserve the stability properties of the original system; this loss of stability is due to truncation of higher order modes. These modes are responsible for the dissipation of energy, and therefore have a stabi-

---

M. Benosman (✉) · A. Chakrabarty  
Mitsubishi Electric Research Laboratories (MERL),  
Cambridge, MA 02139, USA  
e-mail: [m\\_benosman@ieee.org](mailto:m_benosman@ieee.org)

J. Borggaard  
Interdisciplinary Center for Applied Mathematics, Virginia Tech,  
Blacksburg, VA 24061, USA

lizing effect; maintaining stability is crucial for any ROM that is used for predictions over long time intervals.

We address the stable model reduction problem by following the classical closure modeling approach described in [13]. Closure models are added to the ROM equations to ensure the stability and accuracy of solutions. Since closure models have classically been introduced based on physical intuition, their applicability has been limited to applications where the physics is well understood, and a library of closure models have been previously identified. In this work, we propose the use of reinforcement learning (RL) and control theory to constructively design a new stabilizing closure model that is robust to model uncertainties or unmodeled dynamics.

There are several closure models motivated from physical modeling of fluids, for example: constant eddy viscosity model, or time, and space varying terms, such as Smagorinsky or dynamic subgrid-scale models [14, 15]. We propose a conceptually different closure model in this chapter, which explicitly accounts for model uncertainties in the system. Concretely, we formulate the problem of ROM stabilization at the design step, by considering uncertainties in the ROM model, then using reinforcement learning to design a closure model which stabilizes the ROM. To our knowledge, *this is the first class of closure model that is designed based on reinforcement learning methods*.

Additionally, in this work, we propose to learn some coefficients of the closure model using a data-driven optimization algorithm. This learning can be used in simulations to find the best closure model by tracking the true behavior of the system. However, an important observation is that *this learning algorithm can be incorporated in real-time simulations*, by feeding real-time measurements from the system into the closure model and adapting its coefficients. In this way, we always ensure that the ROM is functioning at its optimal performance, regardless of changes or drifts that the system may experience over time. In other words, most closure models typically use static parameters, either chosen by intuition and experience, or are optimally tuned off-line. However, they are unable to auto-tune themselves online while the model is being evolved. In this work, the obtained closure model has free parameters that are auto-tuned with a data-driven extremum seeking (ES) algorithm to optimally match the predictions (or measurements) from the PDE model. The idea of using extremum seeking to auto-tune closure models has been introduced by the authors in [16–18]. However, the difference with this work lies in the new RL-based stabilizing closure model design, which is then tuned using ES to optimize tracking performance.

Our work extends existing results in the field. Indeed, stable model reduction of Navier-Stokes flow models by adding a nonlinear viscosity term to the reduced-order model is considered in [5]. The coefficients of this term are identified using a variational data-assimilation approach, based on solving a deterministic optimization problem. In [6, 19], incompressible flows are stabilized by an iterative search of the projection modes that satisfy a local Lyapunov stability condition. An optimization-based approach for the POD modes of linear models is derived in [10, 12]. Kalb and Deane [9] added error correction terms to the reduced-order model for improved accuracy and stabilization. Moreover, the authors in [8] calibrated the POD model

by solving a quadratic optimization problem based on three different weighted error norms. Stable model reduction for the Burgers equation using closure models was proposed in [14, 15]. Stable model reduction for the Navier–Stokes and Boussinesq equations using turbulence closure models was presented in [13–15], respectively. These closure models modify some stability-enhancing coefficients of the reduced-order ODE model using either constant additive terms, such as the constant eddy viscosity model, or time and space varying terms, such as Smagorinsky or dynamic subgrid-scale models. The amplitudes of the additional terms are tuned in such a way to accurately stabilize the reduced-order models to the data at hand.

However, such closure models do not take into account parametric uncertainties in the model, and tuning them is not always straightforward. Our work addresses these issues and proposes, using reinforcement learning and control theory, a new closure model that addresses model uncertainties, and using ES to auto-tune its coefficients to improve its tracking performance.

The rest of this chapter is organized as follows. Some basic notation and definitions are recalled in Sect. 11.2, the main idea of this work, namely, the RL-based closure model estimation is then introduced in Sect. 11.3, and its auto-tuning using extremum seeking algorithms is explained in Sect. 11.4. Finally, the performance of the proposed concept of RL-based closure models is demonstrated using the one-dimensional coupled Burgers equation in Sect. 11.5.

## 11.2 Basic Notation and Definitions

For a vector  $q \in \mathbb{R}^n$ , its transpose is denoted by  $q^*$ . The Euclidean vector norm for  $q \in \mathbb{R}^n$  is denoted by  $\|\cdot\|$  so that  $\|q\| = \sqrt{q^* q}$ . The Kronecker delta function is defined as  $\delta_{ij} = 0$ , for  $i \neq j$  and  $\delta_{ii} = 1$ . We shall abbreviate the time derivative by  $\dot{f}(t, x) = \frac{\partial}{\partial t} f(t, x)$ , and consider the following Hilbert spaces:  $\mathcal{H} = L^2(\Omega)$ ,  $\Omega = (0, 1)$ , which is the space of Lebesgue square integrable functions, i.e.,  $f \in \mathcal{H}$ , iff  $\int_{\Omega} |f(x)|^2 dx < \infty$ ,  $\mathcal{V} = H^1(\Omega) \subset (\mathcal{H})$  for velocity and  $\mathcal{T} = H^1(\Omega) \subset \mathcal{H}$  for temperature. We define the inner product  $\langle \cdot, \cdot \rangle_{\mathcal{H}}$  and the associated norm  $\|\cdot\|_{\mathcal{H}}$  on  $\mathcal{H}$  as  $\langle f, g \rangle_{\mathcal{H}} = \int_{\Omega} f(x)g(x)dx$ , for  $f, g \in \mathcal{H}$ , and  $\|f\|_{\mathcal{H}}^2 = \int_{\Omega} |f(x)|^2 dx$ . A function  $f(t, x)$  is in  $L^2([0, t_f]; \mathcal{H})$  if for each  $0 \leq t \leq t_f$ ,  $f(t, \cdot) \in \mathcal{H}$ , and  $\int_0^{t_f} \|f(t, \cdot)\|_{\mathcal{H}}^2 dt < \infty$  with analogous definitions for the vector valued functions in  $(\mathcal{H})$ . To generalize the discussion below, we consider the abstract Hilbert space  $\mathcal{Z}$ , and later specialize to  $\mathcal{Z} = \mathcal{V} \oplus \mathcal{T}$  when considering our coupled Burgers equation example.

## 11.3 RL-Based Model Reduction of PDEs

### 11.3.1 Reduced-Order PDE Approximation

We consider a stable dynamical system modeled by a nonlinear partial differential equation of the form

$$\dot{z}(t) = \mathcal{F}(z(t)), \quad z(0) \in \mathcal{Z}, \quad (11.1)$$

where  $\mathcal{Z}$  is an infinite-dimensional Hilbert space. Solutions to this PDE can be approximated in a finite-dimensional subspace  $\mathcal{Z}^n \subset \mathcal{Z}$  through expensive numerical discretization, which can be impractical for multi-query settings such as analysis and design, and even more so for real-time applications such as prediction and control. In many systems, including fluid flows, solutions of the PDE may be well approximated using only a few suitable (optimal) basis functions [7].

This gives rise to reduced-order modeling through Galerkin projection, which can be broken down into three main steps: One first discretizes the PDE using a finite, but large, number of basis functions, such as piece-wise quadratic (for finite element methods), higher order polynomials (spectral methods), or splines. In this paper, we use the well-established finite element method (FEM), and refer the reader to the large literature for details, such as [20]. We denote the approximation of the PDE solution by  $z_n(t, \cdot) \in \mathcal{Z}^n$ , where  $\mathcal{Z}^n$  is an  $n$ -dimensional finite element subspace of  $\mathcal{Z}$ .

Secondly, one determines a small set of spatial basis vectors  $\phi_i(\cdot) \in \mathcal{Z}^n$ ,  $i = 1, \dots, r$ ,  $r \ll n$ , which approximates the discretized PDE solution with respect to a pre-specified criterion, i.e.,

$$P_n z(t, x) \approx \Phi q(t) = \sum_{i=1}^r q_i(t) \phi_i(x). \quad (11.2)$$

Here,  $P_n$  is the projection of  $z(t, \cdot)$  onto  $\mathcal{Z}^n$ , and  $\Phi$  is a matrix containing the basis vectors  $\phi_i(\cdot)$  as column vectors. Note that the dimension  $n$ , coming from the high-fidelity discretization of the PDE described above, is generally very large, in contrast to the dimension  $r$  of the optimal basis set. Thirdly, a Galerkin projection yields a ROM for the coefficient functions  $q(\cdot)$  of the form

$$\dot{q}(t) = F(q(t)), \quad q(0) \in \mathbb{R}^r. \quad (11.3)$$

The function  $F : \mathbb{R}^r \rightarrow \mathbb{R}^r$  is obtained using the weak form of the original PDE and Galerkin projection.

One significant challenge in this approach lies in the selection of the “optimal” basis matrix  $\Phi$ , the criterion of optimality used, and selecting the dimension of this basis. There are many approaches to compute those basis functions for nonlinear systems. For example, some of the most common methods are proper orthogonal

decomposition (POD) [21], dynamic mode decomposition (DMD) [22], and reduced-basis methods (RBM) [23]. Another challenge for nonlinear problems, in particular, is to model the influence of the discarded basis functions on the retained basis functions  $\Phi$  in the dynamical system for  $q(t)$ .

### 11.3.2 Proper Orthogonal Decomposition for ROMs

POD-based models are most known for retaining a maximal amount of energy in the reduced-order model [7, 21]. The POD basis is computed from a collection of  $s$  time snapshots

$$\mathcal{S} = \{z_n(t_1, \cdot), \dots, z_n(t_s, \cdot)\} \subset \mathcal{Z}^n \quad (11.4)$$

of the dynamical system, usually obtained from a discretized approximation of the PDE model in  $n$  dimensions. The  $\{t_i\}_{i=1}^s$  are time instances at which snapshots are recorded, and do not have to be equally spaced. The correlation matrix  $K$  is then defined as

$$K_{ij} = \frac{1}{s} \langle z_n(t_i, \cdot), z_n(t_j, \cdot) \rangle_{\mathcal{H}}, \quad i, j = 1, \dots, s. \quad (11.5)$$

The normalized eigenvalues and eigenvectors of  $K$  are denoted by  $\lambda_i$  and  $v_i$ , respectively. Note that the  $\lambda_i$  are also referred to as the POD eigenvalues. The  $i$ th POD basis function is computed as

$$\phi_i(x) = \frac{1}{\sqrt{s\lambda_i}} \sum_{j=1}^s [v_i]_j z_n(t_j, x), \quad \forall i = 1, \dots, r, \quad (11.6)$$

where  $r \leq \min\{s, n\}$  is the number of retained POD basis functions and depends upon the application. The POD basis functions are orthonormal; that is, they satisfy

$$\langle \phi_i, \phi_j \rangle_{\mathcal{H}} = \int_{\Omega} \phi_i(x)^* \phi_j(x) dx = \delta_{ij}, \quad (11.7)$$

where  $\delta_{ij}$  denotes the Kronecker delta function.

In this new basis, the solution of the PDE (11.1) can be approximated by

$$z_n^{\text{POD}}(t, \cdot) = \sum_{i=1}^r q_i(t) \phi_i(\cdot) \in \mathcal{Z}^n, \quad (11.8)$$

where  $q_i$ ,  $i = 1, \dots, r$  are the POD projection coefficients.

To find the coefficients  $q_i(t)$ , the (weak form of the) model (11.1) is projected onto the  $r$ th-order POD subspace  $\mathcal{Z}^r \subseteq \mathcal{Z}^n \subset \mathcal{Z}$  using a Galerkin projection in  $\mathcal{H}$ . In particular, both sides of Eq. (11.1) are multiplied by the POD basis functions, where

$z(t)$  is replaced by  $z_n^{\text{POD}}(t) \in \mathcal{Z}^n$ , and then both sides are integrated over  $\Omega$ . Using the orthonormality of the POD basis (11.7) leads to an ODE of the form (11.3). A projection of the initial condition for  $z(0)$  can be used to determine  $q(0)$ . The Galerkin projection preserves the structure of the nonlinearities of the original PDE.

### 11.3.3 Closure Models for ROM Stabilization

We continue to present the problem of stable model reduction in its general form, without specifying a particular type of PDE. However, we now assume an explicit dependence of the general PDE (11.1) on a single physical parameter  $\mu$ ,

$$\dot{z}(t) = \mathcal{F}(z(t), \mu), \quad z(0) = z_0 \in \mathcal{Z}, \quad \mu \in \mathbb{R}, \quad (11.9)$$

as well as

**Assumption 11.1** The solutions of the original PDE model (11.9) are assumed to be in  $L^2([0, \infty); \mathcal{Z})$ ,  $\forall \mu \in \mathbb{R}$ .

We further assume that the parameter  $\mu$  is critical for the stability and accuracy of the model, i.e., changing the parameter can either make the model unstable, or lead to inaccurate predictions. Since we are interested in fluid dynamics problems, we can consider  $\mu$  as a viscosity coefficient. The corresponding reduced-order POD model takes the form (11.3) and (11.8):

$$\dot{q}(t) = F(q(t), \mu). \quad (11.10)$$

The issue with this Galerkin POD-ROM (denoted POD-ROM-G) is that the norm of  $q$ , and hence  $z_n^{\text{pod}}$ , might become unbounded at a finite time, even if the solution of (11.9) is bounded (Lagrange stable).

The main idea behind the closure modeling approach is to replace the viscosity coefficient  $\mu$  in (11.10) by a virtual viscosity coefficient  $\mu_{cl}$ , whose form is chosen to stabilize the solutions of the POD-ROM (11.10). Furthermore, a penalty term  $H(\cdot)$  is added to the original POD-ROM-G, as follows

$$\dot{q}(t) = F(q(t), \mu) + H(q(t)). \quad (11.11)$$

The term  $H(\cdot)$  is chosen depending on the structure of  $F(\cdot, \cdot)$  to stabilize the solutions of (11.11).

### 11.3.4 Main Result: RL-Based Closure Model

Here we introduce the main result of this work, namely, RL-based closure model that is robust to model uncertainties. We first rewrite the right-hand side of the ROM model (11.10) to isolate the linear viscous term as follows:

$$F(q(t), \mu) = \tilde{F}(q(t)) + \mu D q(t), \quad (11.12)$$

where  $D \in \mathbb{R}^{r \times r}$  represents a constant, symmetric negative definite matrix, and the function  $\tilde{F}(\cdot)$  represents the remainder of the ROM model, i.e., the part without damping.

To follow the framework of [24], we discretize our model (11.11), (11.12), for example, by using a simple first-order Euler approximation, as follows:

$$q(k+1) = (I_{r \times r} + h_t \mu D)q(k) + h_t \tilde{F}(q(k)) + h_t H(q(k)), \quad (11.13)$$

where  $k = 0, 1, \dots$ , and  $h_t > 0$  represent the integration time-step. We make the following assumption on the behavior of  $\tilde{F}(\cdot)$ .

**Assumption 11.2** (*Lipschitz continuity of  $\tilde{F}$* ) The nonlinearity  $\tilde{F}$  is Lipschitz continuous in the domain  $\mathbb{D}_q \in \mathbb{R}^r$ . That is,

$$\|\tilde{F}(q_1) - \tilde{F}(q_2)\| \leq \mathcal{L}^* \|q_1 - q_2\| \quad (11.14)$$

for any  $q_1, q_2 \in \mathbb{D}_q$ . Also,  $\tilde{F}(0) = 0$ .

**Remark 11.1** We underline here that we do not need the exact knowledge of the nonlinear term  $\tilde{F}$  to design our RL-based closure model. Indeed, we only need to know an estimate of the vector of component-wise Lipschitz constants  $\mathcal{L}^*$ . This estimate can be obtained, for instance, by using the data-driven algorithm proposed in [24, 25]. In that sense, the proposed RL-based closure model stabilization is robust with respect to the uncertainties of the nonlinear term  $\tilde{F}$ .

The main idea that we are proposing here is to consider the closure model function  $H(q(t))$  as a virtual controller. This controller can then be learned using RL methods such as approximate/adaptive dynamic programming (ADP) for stabilizing the ROM.

Let us first recall the basic formulation of ADP. Given a control policy  $u(q)$ , we define an infinite horizon cost functional given an initial state  $q_0 \in \mathbb{R}^r$  as

$$\mathcal{J}(q(0), u) = \sum_{t=0}^{\infty} \gamma^t \mathcal{U}(q(k), u(q(k))), \quad (11.15)$$

where  $\mathcal{U}$  is a function with non-negative range,  $\mathcal{U}(0, 0) = 0$ , and  $\{q(k)\}$  denotes the sequence of states generated by the closed-loop system

$$q(k+1) = Aq(k) + Bu(q(k)) + \phi(C_q q(k)), \quad (11.16)$$

where, in our case, we define the terms to be

$$\begin{aligned} A &= I_{r \times r} + h_t \mu D, \\ B &= h_t . I_{r \times r}, \quad C_q = I_{r \times r}, \\ \phi &= h_t \tilde{F}(q(t)), \\ u(q) &= H(q). \end{aligned} \tag{11.17}$$

The scalar  $\gamma \in (0, 1]$  in (11.15) is a forgetting/discount factor intended to enable the cost to be emphasized more by current state and control actions and lend less credence to the past.

Before formally stating our objective, we need to introduce the following definition.

**Definition 11.1** A continuous control policy  $u(\cdot) : \mathbb{R}^r \rightarrow \mathbb{R}^r$  is *admissible* on  $X \subset \mathbb{R}^r$  if it stabilizes the closed-loop system (11.16) on  $X$  and  $\mathcal{J}(q(0), u)$  is finite for any  $q(0) \in X$ ; furthermore,  $u(0) = 0$ .

We want to design an optimal control policy that achieves the optimal cost

$$\mathcal{J}_\infty(q(0)) = \inf_{u \in \mathfrak{U}} \mathcal{J}(q(0), u), \tag{11.18}$$

for any  $q(0) \in \mathbb{R}^r$ . Here,  $\mathfrak{U}$  denotes the set of all admissible control policies. In other words, we wish to compute an optimal control policy

$$u_\infty = \arg \inf_{u \in \mathfrak{U}} \mathcal{J}(q(0), u). \tag{11.19}$$

Directly constructing such an optimal controller is very challenging for general nonlinear systems with high state dimension. Therefore, we shall use adaptive/approximate dynamic programming (ADP): a class of iterative, data-driven algorithms that generate a convergent sequence of control policies whose limit is provably the optimal control policy  $u_\infty(q)$ . Recall the optimal value function given by (11.18) and the optimal control policy (11.19). From the Bellman optimality principle, we know that the discrete-time Hamilton–Jacobi–Bellman equations are given by

$$J_\infty(q(k)) = \inf_{u \in \mathfrak{U}} (\mathcal{U}(q(k), u(q(k))) + \gamma J_\infty(q(k+1))), \tag{11.20}$$

$$u_\infty(q(k)) = \arg \inf_{u \in \mathfrak{U}} (\mathcal{U}(q(k), u(q(k))) + \gamma J_\infty(q(k+1))), \tag{11.21}$$

where  $J_\infty(q(k))$  is the optimal value function and  $u_\infty(q(k))$  is the optimal control policy.

The key operations in ADP methods, e.g., [26] involve setting an admissible control policy  $u_0(x)$  and then iterating the policy evaluation step

$$\mathcal{J}_{I+1}(q(k)) = \mathcal{U}(q(k), u_I(q(k))) + \gamma \mathcal{J}_I(q(k+1)) \tag{11.22a}$$

and the policy improvement step

$$u_{I+1}(q(k)) = \arg \min_{u(\cdot)} (\mathcal{U}(q(k), u(q(k))) + \gamma \mathcal{J}_{I+1}(q(k+1))), \quad (11.22b)$$

$I = 0, 1, \dots$ , until convergence.

Next, we recall the following definition.

**Definition 11.2** The equilibrium point  $q = 0$  of the closed-loop system (11.16) is globally exponentially stable with a decay rate  $\alpha$  if there exist scalars  $C_0 > 0$  and  $\alpha \in (0, 1)$  such that  $\|q(k)\| \leq C_0 \alpha^{(k-k_0)} \|q(0)\|$  for any  $q(0) \in \mathbb{R}^r$ .

The following design theorem provides a method to construct an initial linear stabilizing policy  $u_0(x) = K_0 x$  such that the origin is a GES equilibrium state of the closed-loop system (11.16).

**Theorem 11.1** Suppose that Assumptions 11.1–11.2 hold, and that there exist matrices  $P = P^\top \succ 0 \in \mathbb{R}^{n_x \times n_x}$ ,  $K_0 \in \mathbb{R}^{n_u \times n_x}$ , and scalars  $\alpha \in (0, 1)$ ,  $\nu > 0$  such that

$$\Psi + \Gamma^\top \mathcal{M} \Gamma \preceq 0, \quad (11.23)$$

where

$$\begin{aligned} \Psi &= \begin{bmatrix} (A + BK_0)^\top P(A + BK_0) - \alpha^2 P & \star \\ G^\top P(A + BK_0) & G^\top PG \end{bmatrix}, \\ \Gamma &= \begin{bmatrix} C_q & 0 \\ 0 & I \end{bmatrix}, \text{ and } \mathcal{M} = \begin{bmatrix} \nu^{-1}(\mathcal{L}^*)^2 I & 0 \\ 0 & -\nu^{-1} I \end{bmatrix}. \end{aligned}$$

Then the equilibrium  $x = 0$  of the closed-loop system (11.16) is GES with decay rate  $\alpha$ .

**Proof** Refer to [24].

Note that we do not need to know  $\phi(\cdot)$  to satisfy conditions (11.23). If the Lipschitz constant is not known, a numerical algorithm for estimating it is described in [24, 25].

We shall now provide LMI-based conditions for computing the initial control policy  $K_0$ , the initial domain of attraction  $P$  and  $\nu$  via convex programming.

**Theorem 11.2** Fix  $\alpha \in (0, 1)$  and  $\mathcal{L}^*$  as defined in Assumption 11.2. If there exist matrices  $S = S^\top \succ 0$ ,  $Y$ , and a scalar  $\nu > 0$  such that the LMI conditions

$$\begin{bmatrix} -\alpha^2 S & \star & \star & \star \\ 0 & -\nu I & \star & \star \\ AS + BY & GS & -S & \star \\ \mathcal{L}^* C_q S & 0 & 0 & -\nu I \end{bmatrix} \preceq 0 \quad (11.24)$$

are satisfied, then the matrices  $K_0 = Y S^{-1}$ ,  $P = S^{-1}$  and scalar  $\nu$  satisfy the conditions (11.23) with the same  $\alpha$  and  $\mathcal{L}^*$ .

**Proof** Refer to [24].

We can now state the following admissibility Corollary.

**Corollary 11.1** *Let*

$$\mathcal{U}(q(k), u(k)) = q(k)^\top Q q(k) + u(k)^\top R u(k) \quad (11.25)$$

for some matrices  $Q = Q^\top \succeq 0$  and  $R = R^\top \succ 0$ . Then the initial control policy  $u(0) = K_0 q$  obtained by solving (11.24) is an admissible control policy on  $\mathbb{R}^r$ .

Now that we know  $u(0) = K_0 q$  is an admissible control policy, we are ready to proceed with the policy iteration steps (11.22). Typically, an analytical form of  $\mathcal{J}_I$  is not known a priori, so we resort to a neural approximator/truncated basis expansion for fitting this function, assuming  $\mathcal{J}_I$  is smooth for every  $I \in \mathbb{N} \cup \{\infty\}$ . Concretely, we represent the value function and cost functions as

$$\mathcal{J}_I(q) := \omega_I^\top \psi(q), \quad (11.26)$$

where  $\psi_0(\cdot) : \mathbb{R}^r \rightarrow \mathbb{R}^{n_0}$  denotes the set of differentiable basis functions (equivalently, hidden layer neuron activations) and  $\omega : \mathbb{R}^{n_0}$  is the corresponding column vector of basis coefficients (equivalently, neural weights).

It is not always clear how to initialize the weights of the neural approximators (11.26). We propose initializing the weights as follows. Since our initial Lyapunov function is quadratic, we include the quadratic terms of the components of  $x$  to be in the basis  $\psi(q)$ . Then we can express the initial Lyapunov function  $q^\top P q$  obtained by solving (11.24) with appropriate weights in the  $\psi(q)$ , respectively, setting all other weights to be zero. With the approximator initialized as above, the policy evaluation step (11.22a) is replaced by

$$\omega_{I+1}^\top (\psi(q(k)) - \gamma \psi(q(k+1))) = \mathcal{U}(q(k), u_I(q(k))), \quad (11.27a)$$

from which one can solve for  $\omega_{I+1}$  recursively via

$$\omega_{I+1} = \omega_I - \eta_I \varphi_I (\omega_I^\top \varphi_I - \mathcal{U}(q(k), u_I(q(k)))),$$

where  $\eta_I$  is a learning rate parameter that is usually selected to be an element from the sequence  $\{\eta_I\} \rightarrow 0$  as  $I \rightarrow \infty$ , and  $\varphi_I = \psi(q(k)) - \gamma \psi(q(k+1))$ . Subsequently, the policy improvement step (11.22b) is replaced by

$$u_{I+1} = \arg \min_{u(\cdot)} (\mathcal{U}(q(k), u(q(k))) + \gamma \omega_{I+1}^\top \psi(q(k+1))).$$

This minimization problem is typically non-convex, and therefore, challenging to solve to optimality. In some specific cases, one of which is that the cost function is quadratic as described in (11.25), the policy improvement step becomes considerably simpler to execute, with an update rule

$$u_{I+1}(q) = -\frac{\gamma}{2} R^{-1} B^\top \nabla \psi(q)^\top \omega_{I+1}. \quad (11.27b)$$

This can be evaluated as  $R$  and  $B$  are known, and  $\psi$  is differentiable and chosen by the user, so  $\nabla \psi$  is computable.

Since we prove that  $u_0$  is an admissible control policy, we can use arguments identical to [27, Theorems 3.2 and 4.1] to claim that if the optimal value function and the optimal control policy are dense in the space of functions induced by the basis function expansions (11.26), then the weights of the neural approximator employed in the PI steps (11.27) converges to the optimal weights; that is, the optimal value function  $\mathcal{J}_\infty$  and the optimal control policy  $u_\infty$  are achieved asymptotically. A pseudo-code for implementation is provided in Algorithm 11.1.

---

**Algorithm 11.1** Safely Initialized RL for closure-model estimation

---

**Require** Termination condition constant  $\epsilon_{ac}$   
**Require** Compute stabilizing control gain  $K_0$  via SDP (11.24)  
**State** Fix admissible control policy  $u_0(q) = K_0 q$   
**While**  $\|\mathcal{J}_I - \mathcal{J}_{I-1}\| \geq \epsilon_{ac}$   
**State** Solve for the value  $\mathcal{J}_I(q)$  using

$$\mathcal{J}_{I+1}(q(k)) = \mathcal{U}(q(k), u_I(q(k))) + \gamma \mathcal{J}_{I+1}(q(k+1)).$$

**State** Update the control policy  $u_{(I+1)}(q)$  using

$$u_{I+1}(q(k)) = \arg \min_{u(\cdot)} (\mathcal{U}(q(k), u_I(x_t)) + \gamma \mathcal{J}_{I+1}(q(k+1))).$$

**State**  $I := I + 1$

---

We now present our main result.

**Theorem 11.3** (RL-based stabilizing closure model) *Consider the PDE (11.9) under Assumption 11.1, together with its ROM model*

$$\dot{q}(t) = \tilde{F}(q(t)) + \mu Dq(t) + H(q(t)), \quad (11.28)$$

where  $\tilde{F}(\cdot)$  satisfies Assumption 11.2,  $D \in \mathbb{R}^{r \times r}$  is symmetric negative definite, and  $\mu > 0$  is the nominal value of the viscosity coefficient in (11.9). Then, the nonlinear closure model  $H(q)$  computed using the RL controller (11.27a), (11.27b), where  $u_0(q) = K_0 q$ ,  $K_0$  obtained by the SDP (11.24), practically stabilizes the solutions of the ROM (11.28) to an  $\epsilon$ -neighborhood of the origin.

**Proof** The proofs follows naturally from the admissibility results of Theorem 11.2 and Corollary 11.1, which put together with the optimality results of [27] [Theorems 3.2 and 4.1], lead to admissibility and optimality of the closure model  $H$ , given by

(11.27a), (11.27b), where  $u_0(q) = K_0 q$ ,  $K_0$  computed by the SDP (11.24). Next, we conclude about the practical stability of the model (11.28) to an  $\epsilon$ -neighborhood of the origin, by using the arguments in (Theorem 5.3, [28], and Theorem 5.12 in [29]).  $\square$

## 11.4 Extremum Seeking Based Closure Model Auto-Tuning

ES-based closure model auto-tuning has many advantages. First of all, the closure models can be valid for longer time intervals when compared to standard closure models with constant coefficients that are identified off-line over a (fixed) finite time interval. Secondly, the optimality of the closure model ensures that the ROM obtains the most accuracy for a given low-dimensional basis, leading to the smallest possible ROM for a given application.

We begin by defining a suitable learning cost function for the ES algorithm. The goals of the learning is to ensure that the solutions of the ROM (11.10) are close to those of the approximation  $z_n(t, \cdot)$  to the original PDE (11.9).

We first introduce some tuning coefficients in the ROM model (11.28), as follows

$$\dot{q}(t) = \tilde{F}(q(t)) + (\mu + \mu_e) Dq(t) + \mu_{nl} H(q(t)), \quad (11.29)$$

where  $\mu_e > 0$ , and  $\mu_{nl} > 0$  are two positive tuning parameters. We then define the learning cost as a positive definite function of the norm of the error between the numerical solutions of (11.9) and the ROM (11.29),

$$\begin{aligned} Q(\hat{\mu}) &= \tilde{H}(e_z(t, \hat{\mu})), \\ e_z(t, \hat{\mu}) &= z_n^{pod}(t, x; \hat{\mu}) - z_n(t, x; \mu), \end{aligned} \quad (11.30)$$

where  $\hat{\mu} = [\hat{\mu}_e, \hat{\mu}_{nl}]^* \in \mathbb{R}^2$  denotes the learned parameters, and  $\tilde{H}(\cdot)$  is a positive definite function of  $e_z$ . Note that the error  $e_z$  could be computed off-line using solutions of the ROM (11.8), (11.11) and exact (e.g., FEM-based) solutions of the PDE (11.9). The error could be also computed online where the  $z_n^{pod}(t, x; \hat{\mu})$  is obtained from solving the ROM model (11.8), (11.11) online, and the  $z_n(t, x; \mu)$  are real measurements of the system at selected spatial locations  $\{x_i\}$ . The latter approach would circumvent the FEM model, and directly operate on the system, making the reduced-order model more consistent with respect to the operating plant.

A practical way to implement the ES-based tuning of  $\hat{\mu}$ , is to begin with an off-line tuning of the closure model. One then uses the obtained ROM (with the optimal values of  $\hat{\mu}$ , namely,  $\mu^{\text{opt}}$ ) in the online operation of the system, e.g., control and estimation. We can then fine-tune the ROM online by continuously learning the best value of  $\hat{\mu}$  at any given time during the operation of the system.

To derive formal convergence results, we introduce some classical assumptions on the learning cost function.

**Assumption 11.3** The cost function  $Q(\cdot)$  in (11.30) has a local minimum at  $\hat{\mu} = \mu^{\text{opt}}$ .

**Assumption 11.4** The cost function  $Q(\cdot)$  in (11.30) is analytic and its variation with respect to  $\mu$  is bounded in the neighborhood of  $\mu^{\text{opt}}$ , i.e.,  $\|\nabla_\mu Q(\tilde{\mu})\| \leq \xi_2$ ,  $\xi_2 > 0$ , for all  $\tilde{\mu} \in \mathcal{N}(\mu^{\text{opt}})$ , where  $\mathcal{N}(\mu^{\text{opt}})$  denotes a compact neighborhood of  $\mu^{\text{opt}}$ .

Under these assumptions the following lemma holds.

**Lemma 11.1** Consider the PDE (11.9) under Assumption 11.1, together with its ROM model (11.29). Furthermore, suppose the closure model coefficients  $\hat{\mu} = [\mu_e, \mu_{nl}]^*$  are tuned using the ES algorithm

$$\begin{aligned}\dot{y}_1(t) &= a_1 \sin\left(\omega_1 t + \frac{\pi}{2}\right) Q(\hat{\mu}), \\ \hat{\mu}_e(t) &= y_1 + a_1 \sin\left(\omega_1 t - \frac{\pi}{2}\right), \\ \dot{y}_2(t) &= a_2 \sin\left(\omega_2 t + \frac{\pi}{2}\right) Q(\hat{\mu}), \\ \hat{\mu}_{nl}(t) &= y_2 + a_2 \sin\left(\omega_2 t - \frac{\pi}{2}\right),\end{aligned}\tag{11.31}$$

where  $\omega_{\max} = \max(\omega_1, \omega_2) > \omega^{\text{opt}}$ ,  $\omega^{\text{opt}}$  large enough, and  $Q(\cdot)$  is given by (11.30). Let  $e_\mu(t) := [\mu_e^{\text{opt}} - \hat{\mu}_e(t), \mu_{nl}^{\text{opt}} - \hat{\mu}_{nl}(t)]^*$  be the error between the current tuned values, and the optimal values  $\mu_e^{\text{opt}}, \mu_{nl}^{\text{opt}}$ . Then, under Assumptions 11.3, and 11.4, the norm of the distance to the optimal values admits the following bound

$$\|e_\mu(t)\| \leq \frac{\xi_1}{\omega_{\max}} + \sqrt{a_1^2 + a_2^2}, \quad t \rightarrow \infty,\tag{11.32}$$

where  $a_1, a_2 > 0$ ,  $\xi_1 > 0$ , and the learning cost function approaches its optimal value within the following upper bound

$$\|Q(\hat{\mu}) - Q(\mu^{\text{opt}})\| \leq \xi_2 \left( \frac{\xi_1}{\omega} + \sqrt{a_1^2 + a_2^2} \right),\tag{11.33}$$

as  $t \rightarrow \infty$ , where  $\xi_2 = \max_{\mu \in \mathcal{N}(\mu^{\text{opt}})} \|\nabla_\mu Q(\mu)\|$ .

**Proof** Refer to [17].

## 11.5 The Case of the Burgers Equation

As an example application of our approach, we consider the coupled Burgers equation of the form

$$\begin{cases} \frac{\partial w(t,x)}{\partial t} + w(t,x) \frac{\partial w(t,x)}{\partial x} = \mu \frac{\partial^2 w(t,x)}{\partial x^2} - \kappa T(t,x) \\ \frac{\partial T(t,x)}{\partial t} + w(t,x) \frac{\partial T(t,x)}{\partial x} = c \frac{\partial^2 T(t,x)}{\partial x^2} + f(t,x), \end{cases}\tag{11.34}$$

where  $T(\cdot, \cdot)$  represents the temperature,  $w(\cdot, \cdot)$  represents the velocity field,  $\kappa$  is the coefficient of the thermal expansion,  $c$  the heat diffusion coefficient,  $\mu$  the viscosity

(inverse of the Reynolds number  $Re$ ),  $x \in [0, 1]$  is the one-dimensional space variable,  $t > 0$ , and  $f(\cdot, \cdot)$  is the external forcing term such that  $f \in L^2((0, \infty), X)$ ,  $X = L^2([0, 1])$ . The boundary conditions are imposed as

$$\begin{aligned} w(t, 0) &= w_l, \quad \frac{\partial w(t, 1)}{\partial x} = w_r, \\ T(t, 0) &= T_l, \quad T(t, 1) = T_r, \end{aligned} \quad (11.35)$$

where  $w_l, w_r, T_l, T_r$  are positive constants, and  $l$  and  $r$  denote left and right boundary, respectively. The initial conditions are imposed as

$$\begin{aligned} w(0, x) &= w_0(x) \in L^2([0, 1]), \\ T(0, x) &= T_0(x) \in L^2([0, 1]), \end{aligned} \quad (11.36)$$

and are specified below. Following a Galerkin projection onto the subspace spanned by the POD basis functions, the coupled Burgers equation is reduced to a POD ROM with the following structure (e.g., see [14])

$$\begin{aligned} \begin{pmatrix} \dot{q}_w \\ \dot{q}_T \end{pmatrix} &= B_1 + \mu B_2 + \mu D q + \tilde{D} q + C q q^T, \\ w_n^{pod}(x, t) &= w_{av}(x) + \sum_{i=1}^{r_w} \phi_{wi}(x) q_{wi}(t), \\ T_n^{pod}(x, t) &= T_{av}(x) + \sum_{i=1}^{r_T} \phi_{Ti}(x) q_{Ti}(t), \end{aligned} \quad (11.37)$$

where matrix  $B_1$  is due to the projection of the forcing term  $f$ , matrix  $B_2$  is due to the projection of the boundary conditions, matrix  $D$  is due to the projection of the viscosity damping term  $\mu \frac{\partial^2 w(t, x)}{\partial x^2}$ , matrix  $\tilde{D}$  is due to the projection of the thermal coupling and the heat diffusion terms  $-\kappa T(t, x)$ ,  $c \frac{\partial^2 T(t, x)}{\partial x^2}$ , and the matrix  $C$  is due to the projection of the gradient-based terms  $w \frac{\partial w(t, x)}{\partial x}$ , and  $w \frac{\partial T(t, x)}{\partial x}$ . The notations  $\phi_{wi}(x)$ ,  $q_{wi}(t)$  ( $i = 1, \dots, r_w$ ),  $\phi_{Ti}(x)$ ,  $q_{Ti}(t)$  ( $i = 1, \dots, r_T$ ), stand for the space basis functions and the time projection coordinates, for the velocity and the temperature, respectively. The terms  $w_{av}(x)$ ,  $T_{av}(x)$  represent the mean values (over time) of  $w$  and  $T$ , respectively.

We test the stabilization performance of our RL-based closure model, by considering the coupled Burgers equation (11.34), with the parameters  $Re = 1000$ ,  $\kappa = 5 \times 10^{-4}$ ,  $c = 1 \times 10^{-2}$ , the trivial boundary conditions  $w_l = w_r = 0$ ,  $T_l = T_r = 0$ , a simulation time-length  $t_f = 1s$  and zero forcing,  $f = 0$ . We use 10 POD modes for both variables (temperature and velocity). For the choice of the initial conditions, we follow [14], where the simplified Burgers' equation has been used in the context of POD ROM stabilization. Indeed, in [14], the authors propose two types of initial conditions for the velocity variable, which led to instability of the nominal POD ROM, i.e., the basic Galerkin POD ROM (POD ROM-G) without any closure model. Accordingly, we choose the following initial conditions:

$$w(x, 0) = \begin{cases} 1, & \text{if } x \in [0, 0.5] \\ 0, & \text{if } x \in ]0.5, 1], \end{cases} \quad (11.38)$$

$$T(x, 0) = \begin{cases} 1, & \text{if } x \in [0, 0.5] \\ 0, & \text{if } x \in ]0.5, 1]. \end{cases} \quad (11.39)$$

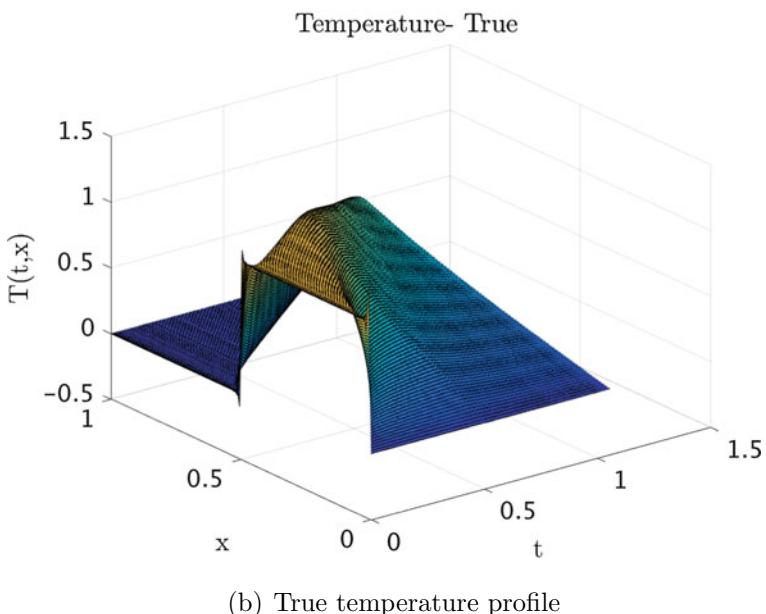
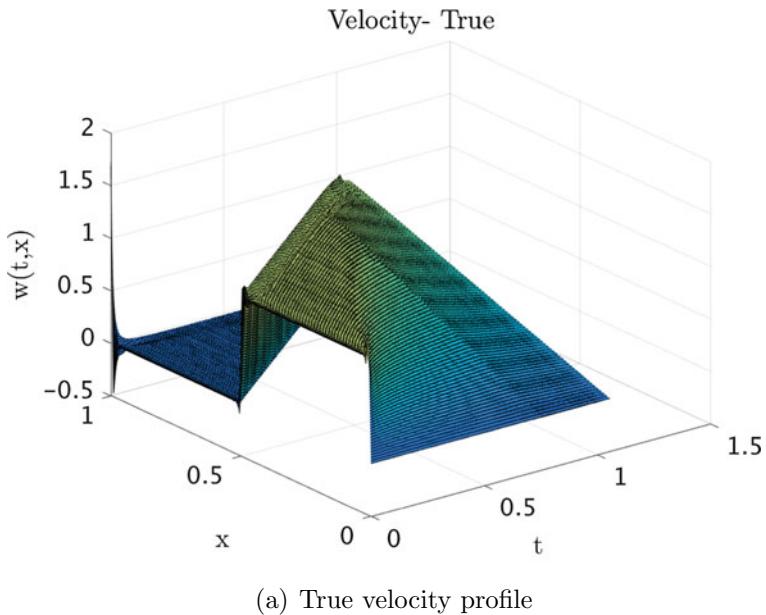
We first show for comparison purposes the true solutions, the high-fidelity (“true”) solutions obtained via the FEM. Equation (11.34) is discretized in space with piecewise linear basis functions with 100 elements, and subsequently solved with `ode15s` in `Matlab®`. The true solutions are reported in Fig. 11.1. We then report in Fig. 11.2, the solutions of the POD ROM-G (without closure model). We can see clearly on these figures that the POD ROM-G solutions are unstable, with a clear blow up of the velocity solution. We then run the ROM-CL which is the ROM with the closure model  $H$  computed from (11.27a), (11.27b), and report the corresponding solutions in Fig. 11.3. We can see clearly that the closure model stabilizes the ROM as expected (see Theorem 11.3). However, although we recover the stability of the original PDE, after adding the RL-based closure term, the performance of the ROM-CL model is rather average, as one can observe in Fig. 11.4. To improve the ROM-CL model performance in reproducing the true temperature and velocity distributions, we add an auto-tuning layer to the ROM-CL model, by using an auto-tuning extremum seeking algorithm, as explained in Sect. 11.4.

We implement the ROM-CL (11.29), where here again the closure term is given by the RL-controller (11.27a), (11.27b). The coefficients of the closure model are tuned using the first Euler discretization of (11.31), where the learning cost is defined as

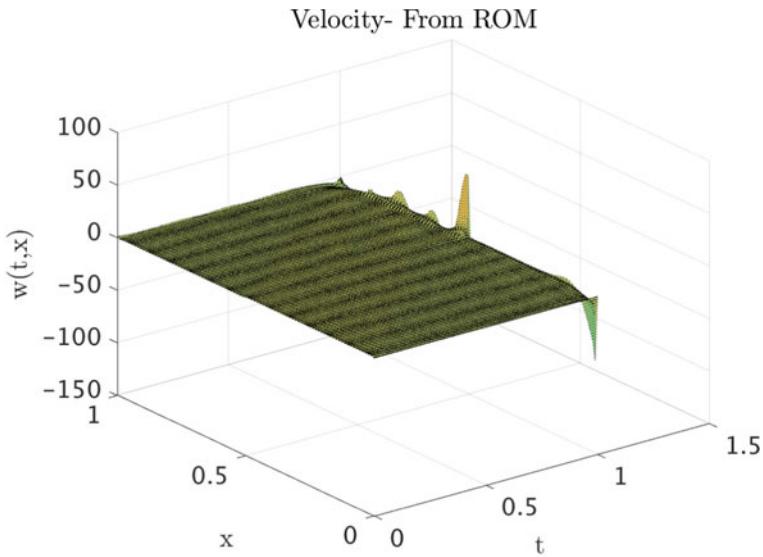
$$Q(\mu) = \int_0^{t_f} \langle e_T, e_T \rangle_{\mathcal{H}} dt + \int_0^{t_f} \langle e_v, e_v \rangle_{\mathcal{H}} dt. \quad (11.40)$$

$e_T = P_r T_n - T_n^{pod}$ ,  $e_v = P_r \mathbf{v}_n - \mathbf{v}_n^{pod}$  define the errors between the projection of the true model solution onto the POD space  $\mathcal{Z}^r$  and the POD-ROM solution for temperature and velocity, respectively. We select the following ES coefficients:  $a_1 = 8 \times 10^{-5} [-]$ ,  $\omega_1 = 10 [\frac{\text{rad}}{\text{s}}]$ ,  $a_2 = 8 \times 10^{-5} [-]$ ,  $\omega_2 = 12 [\frac{\text{rad}}{\text{s}}]$ .

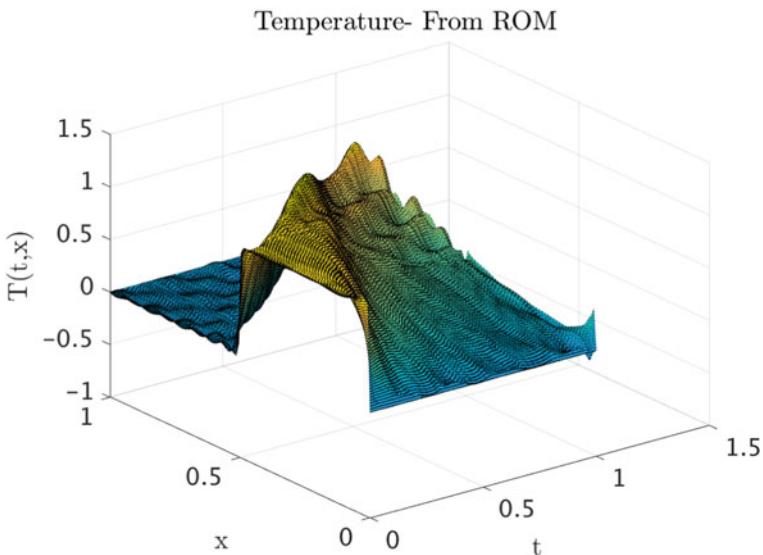
We report in Fig. 11.5, the ES learning cost over the learning iterations, where we see an improvement of the overall tracking cost function. The associated tuning coefficients estimation is shown in Fig. 11.6. Finally, the performance of the ROM-CL after tuning is shown in Figs. 11.7 and 11.8, where we see a large decrease of the tracking errors, comparatively to the errors obtained with the ROM-CL without ES tuning.



**Fig. 11.1** True solutions of (11.34)

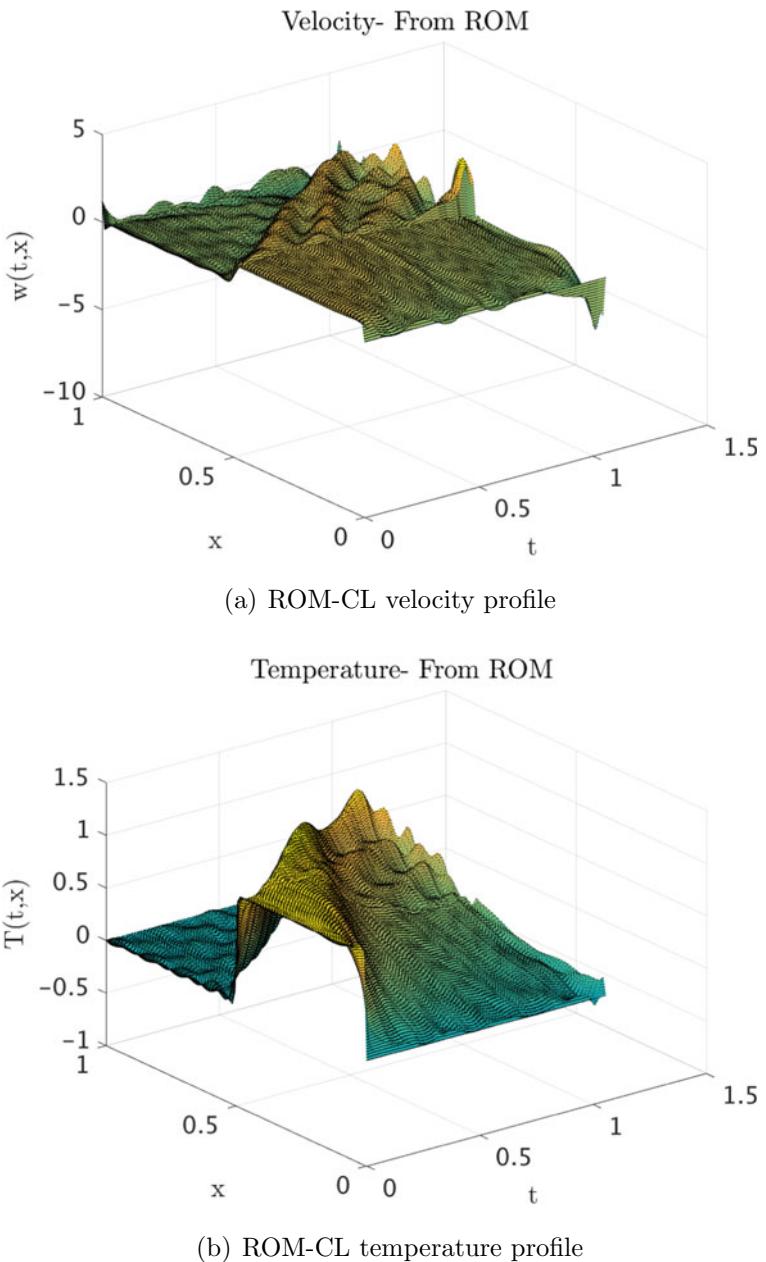


(a) Closure-model-free POD ROM velocity profile

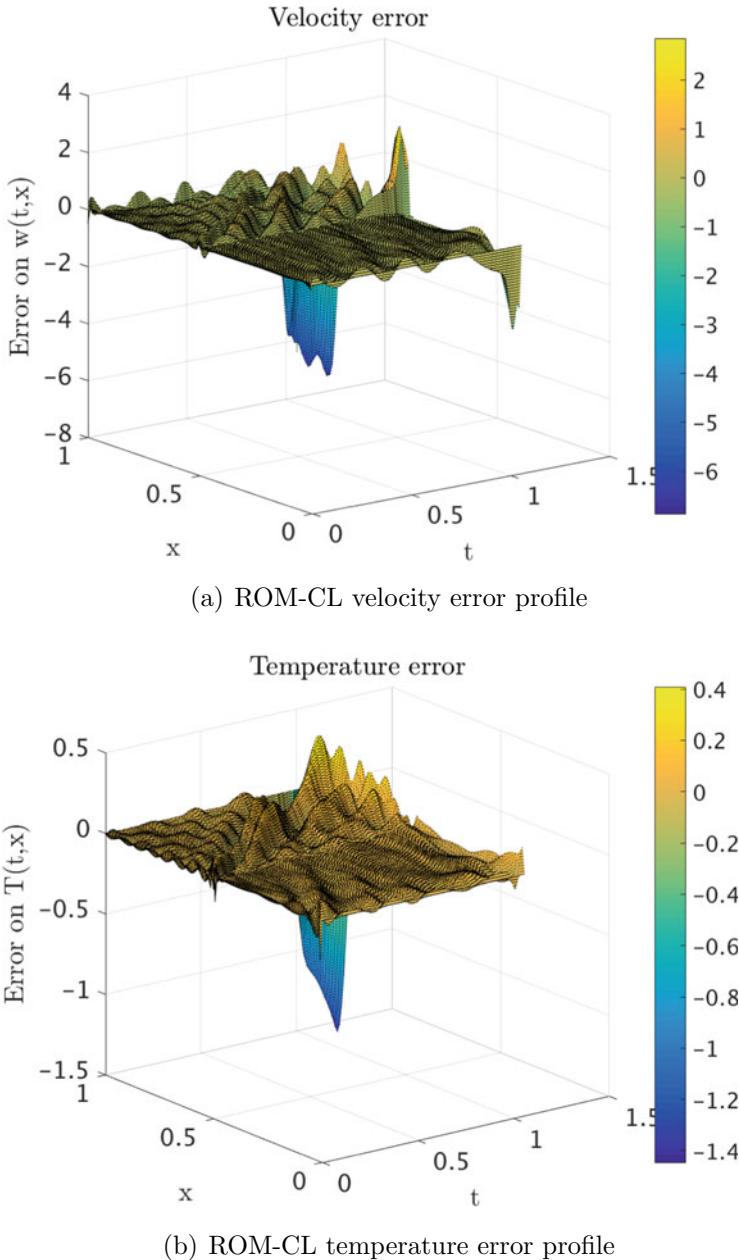


(b) Closure-model-free POD ROM temperature profile

**Fig. 11.2** Closure-model-free POD ROM solutions of (11.34)



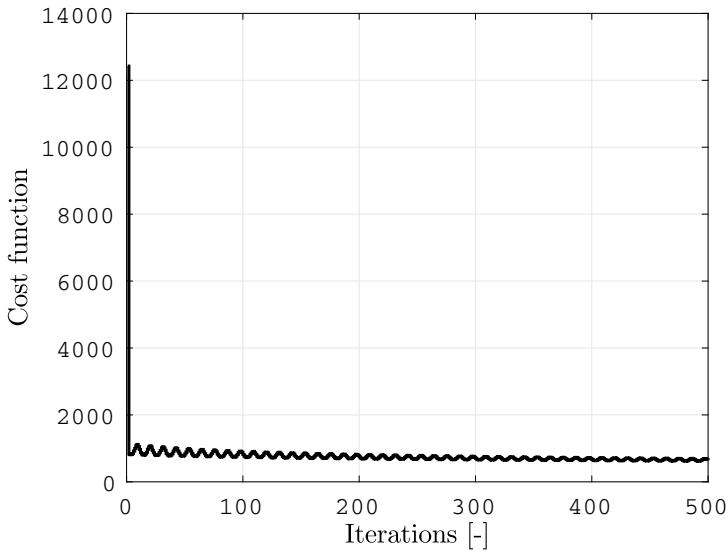
**Fig. 11.3** ROM-CL solutions of (11.34)



(a) ROM-CL velocity error profile

(b) ROM-CL temperature error profile

**Fig. 11.4** ROM-CL solutions error of (11.34)



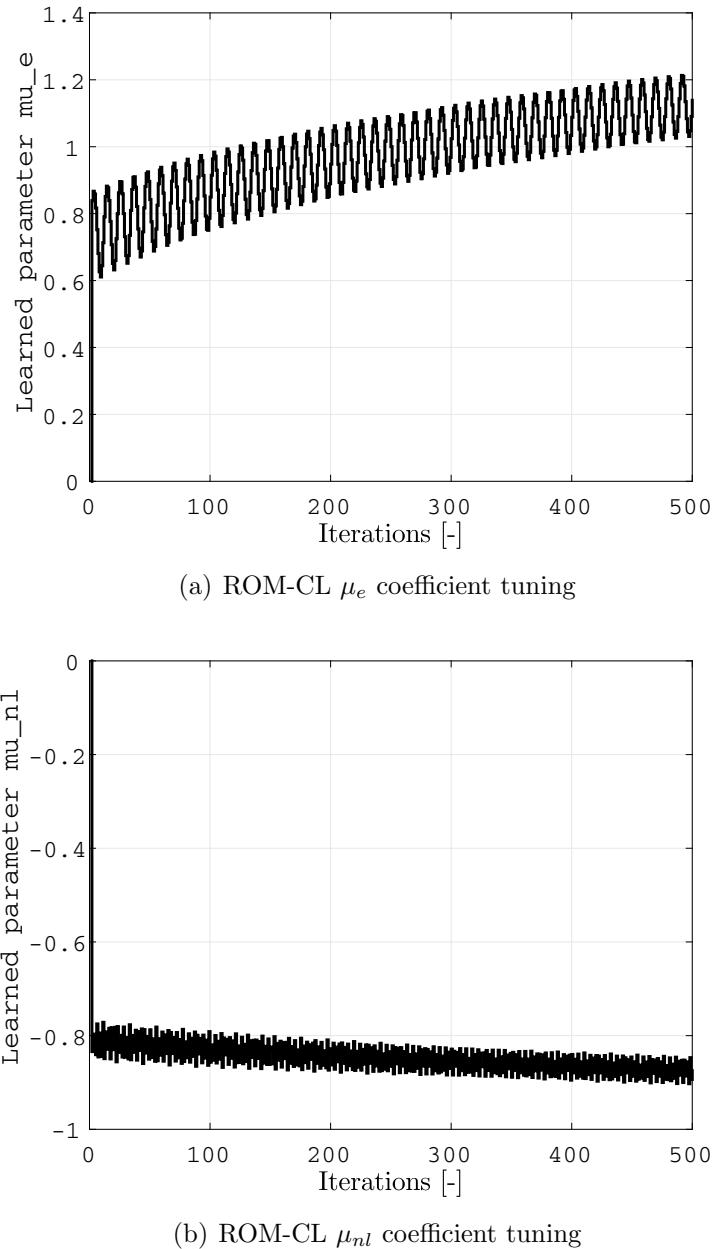
**Fig. 11.5** ROM-CL learning cost

## 11.6 Conclusion

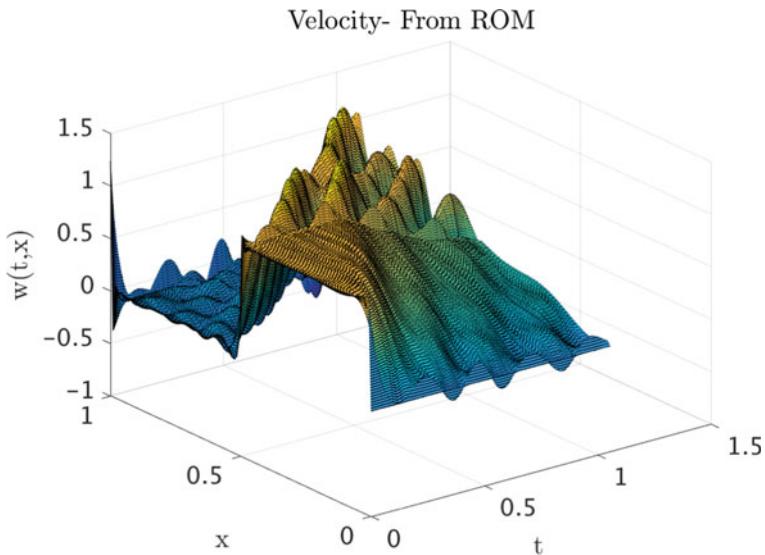
In this chapter, we have focused on the problem of model reduction of infinite-dimensional systems modeled by partial differential equations. We have proposed to use reinforcement learning (RL) control to design stabilizing closure models for reduced-order models.

The obtained stabilizing closure models are robust to model uncertainties, which makes them appealing for real-life applications, like for instance fluid dynamics modeling applications. To further improve the validity of the reduced-order models, we added a tuning layer to the proposed RL-based closure models, by tuning (possibly online) some of their free coefficients using an extremum seeking algorithm.

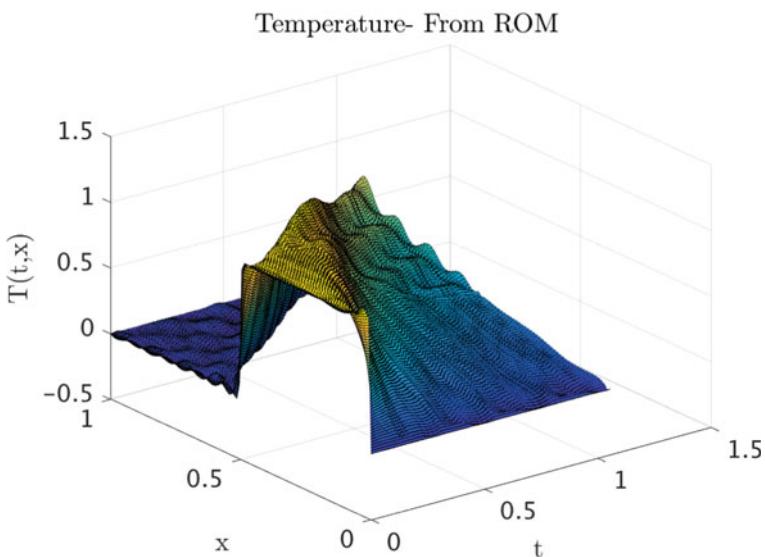
Finally, to validate this concept we simulated the coupled Burgers equation, which is a 1D simplification of the well-known Boussinesq equation that can model airflow dynamics in a room. The good performance obtained on the Burgers equation is encouraging, and motivates us to continue investigating this type of RL-based closure models but for more challenging 2D and 3D Boussinesq models.



**Fig. 11.6** ROM-CL coefficients tuning

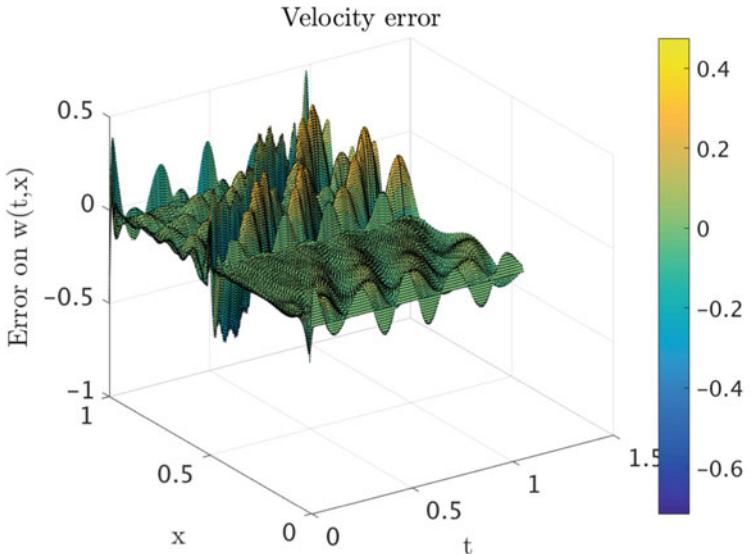


(a) ROM-CL velocity profile (with Learning)

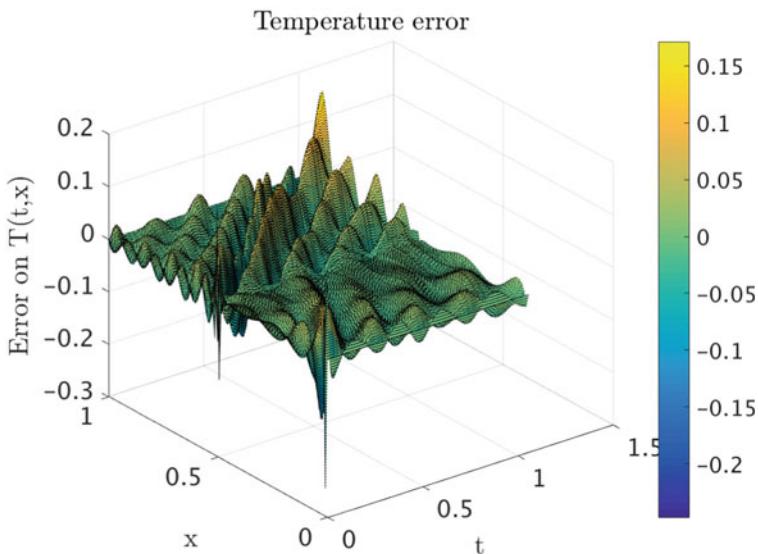


(b) ROM-CL temperature profile (with Learning)

**Fig. 11.7** ROM-CL solutions of (11.34) (with learning)



(a) ROM-CL velocity error profile (with learning)



(b) ROM-CL temperature error profile (with learning)

**Fig. 11.8** ROM-CL solutions error of (11.34) (with learning)

## References

1. Montseny, G., Audouinet, J., Matignon, D.: Fractional integro-differential boundary control of the Euler–Bernoulli beam. In: IEEE, Conference on Decision and Control, San Diego, California, pp. 4973–4978 (1997)
2. Benosman, M.: Lyapunov-based control of the sway dynamics for elevator ropes. *IEEE Trans. Control Syst. Technol.* **22**(5), 1855–1863 (2014)
3. Huges, R.: The flow of human crowds. *Annu. Rev. Fluid Mech.* **35**, 169–182 (2003)
4. Colombo, R.M., Rosini, M.D.: Pedestrian flows and non-classical shocks. *Math. Methods Appl. Sci.* **28**(13), 1553–1567 (2005)
5. Cordier, L., Noack, B., Tissot, G., Lehnasch, G., Delville, J., Balajewicz, M., Daviller, G., Niven, R.K.: Identification strategies for model-based control. *Exp. Fluids* **54**(1580), 1–21 (2013)
6. Balajewicz, M., Dowell, E., Noack, B.: Low-dimensional modelling of high-Reynolds-number shear flows incorporating constraints from the Navier-Stokes equation. *J. Fluid Mech.* **729**(1), 285–308 (2013)
7. Holmes, P., Lumley, J.L., Berkooz, G.: *Turbulence, Coherent Structures, Dynamical Systems and Symmetry*. Cambridge University Press, Cambridge (1998)
8. Couplet, M., Basdevant, C., Sagaut, P.: Calibrated reduced-order POD-Galerkin system for fluid flow modelling. *J. Comput. Phys.* **207**(1), 192–220 (2005)
9. Kalb, V.L., Deane, A.E.: An intrinsic stabilization scheme for proper orthogonal decomposition based low-dimensional models. *Phys. Fluids* **19**(5), 054106 (2007)
10. Bui-Thanh, T., Willcox, K., Ghattas, O., van Bloemen Waanders B.: Goal-oriented, model-constrained optimization for reduction of large-scale systems. *J. Comput. Phys.* **224**(2), 880–896 (2007)
11. Ilak, M., Bagheri, S., Brandt, L., Rowley, C.W., Henningson, D.S.: Model reduction of the nonlinear complex Ginzburg-Landau equation. *SIAM J. Appl. Dyn. Syst.* **9**(4), 1284–1302 (2010)
12. Kalashnikova, I., van Bloemen Waanders, B., Arunajatesan, S., Barone, M.: Stabilization of projection-based reduced order models for linear time-invariant systems via optimization-based eigenvalue reassignment. *Comput. Methods Appl. Mech. Eng.* **272**, 251–270 (2014)
13. Wang, Z., Akhtar, I., Borggaard, J., Iliescu, T.: Proper orthogonal decomposition closure models for turbulent flows: a numerical comparison. *Comput. Methods Appl. Mech. Eng.* **237–240**, 10–26 (2012)
14. San, O., Iliescu, T.: Proper orthogonal decomposition closure models for fluid flows: Burgers equation. *Int. J. Numer. Anal. Model.* **1**(1), 1–18 (2013)
15. San, O., Borggaard, J.: Basis selection and closure for POD models of convection dominated Boussinesq flows. In: 21st International Symposium on Mathematical Theory of Networks and Systems, pp. 132–139. Groningen, The Netherlands (2014)
16. Benosman, M., Kramer, B., Boufounos, P.T., Grover, P.: Learning-based reduced order model stabilization for partial differential equations: application to the coupled Burgers' equation. In: American Control Conference, pp. 1673–1678 (2016)
17. Benosman, M., Borggaard, J., San, O., Kramer, B.: Learning-based robust stabilization for reduced-order models of 2D and 3D Boussinesq equations. *Appl. Math. Model.* **49**, 162–181 (2016)
18. Benosman, M.: *Learning-Based Adaptive Control: An Extremum Seeking Approach - Theory and Applications*. Elsevier Ed., Butterworth-Heinemann, Oxford (2016)
19. Balajewicz, M.: Lyapunov stable Galerkin models of post-transient incompressible flows. Technical report (2013). [arXiv:1312.0284](https://arxiv.org/abs/1312.0284)
20. Gunzburger, M.: *Finite Element Methods for Viscous Incompressible Flows*. Academic, Cambridge (1989)
21. Kunisch, K., Volkwein, S.: Galerkin proper orthogonal decomposition methods for a general equation in fluid dynamics. *SIAM J. Numer. Anal.* **40**(2), 492–515 (2007)

22. Kramer, B., Grover, P., Boufounos, P., Nabi, S., Benosman, M.: Sparse sensing and DMD based identification of flow regimes and bifurcations in complex flows. *SIAM J. Appl. Dyn. Syst.* **16**(2), 1164–1196 (2016)
23. Veroy, K., Patera, A.: Certified real-time solution of the parametrized steady incompressible Navier-Stokes equations: rigorous reduced-basis a posteriori error bounds. *Int. J. Numer. Methods Fluids* **47**(8), 773–788 (2005)
24. Chakrabarty, A., Jha, D.K., Wang, Y.: Data-driven control policies for partially known systems via kernelized Lipschitz learning. In: Proceedings of the 2019 American Control Conference (ACC), Philadelphia, PA, USA, pp. 4192–4197 (2019)
25. Chakrabarty, A., Jha, D.K., Wang, Y., Buzzard, G.T., Vamvoudakis, K.: Safe approximate dynamic programming via kernelized Lipschitz estimation. [arXiv:1907.02151](https://arxiv.org/abs/1907.02151)
26. Lewis, L.F., Vrabie, D., Vamvoudakis, K.G.: Reinforcement learning and feedback control: using natural decision methods to design optimal adaptive controllers. *IEEE Control. Syst. Mag.* **32**(6), 76–105 (2012)
27. Liu, D., Wei, Q.: Policy iteration adaptive dynamic programming algorithm for discrete-time nonlinear systems. *IEEE Trans. Neural Netw. Learn. Syst.* **25**(3), 621–634 (2014)
28. Sanfelice, R.G., Teel, A.R.: Dynamical properties of hybrid systems simulators. *Automatica* **46**, 239–248 (2010)
29. Sanfelice, R.G.: Robust hybrid control systems. Ph.D. dissertation, University of California Santa Barbara (2007)

**Part III**

**Multi-agent Systems and RL**

## Chapter 12

# Multi-Agent Reinforcement Learning: A Selective Overview of Theories and Algorithms



Kaiqing Zhang, Zhuoran Yang, and Tamer Baar

**Abstract** Recent years have witnessed significant advances in reinforcement learning (RL), which has registered tremendous success in solving various sequential decision-making problems in machine learning. Most of the successful RL applications, e.g., the games of Go and Poker, robotics, and autonomous driving, involve the participation of more than one single agent, which naturally fall into the realm of multi-agent RL (MARL), a domain with a relatively long history, and has recently re-emerged due to advances in single-agent RL techniques. Though empirically successful, theoretical foundations for MARL are relatively lacking in the literature. In this chapter, we provide a selective overview of MARL, with focus on algorithms backed by theoretical analysis. More specifically, we review the theoretical results of MARL algorithms mainly within two representative frameworks, Markov/stochastic games and extensive-form games, in accordance with the types of tasks they address, i.e., fully cooperative, fully competitive, and a mix of the two. We also introduce several significant but challenging applications of these algorithms. Orthogonal to the existing reviews on MARL, we highlight several new angles and taxonomies of MARL theory, including learning in extensive-form games, decentralized MARL with networked agents, MARL in the mean-field regime, (non-)convergence of policy-based methods for learning in games, etc. Some of the new angles extrapolate from our own research endeavors and interests. Our overall goal with this chapter is, beyond

---

Writing of this chapter was supported in part by the US Army Research Laboratory (ARL) Cooperative Agreement W911NF-17-2-0196, and in part by the Air Force Office of Scientific Research (AFOSR) Grant FA9550-19-1-0353.

K. Zhang · T. Baar (✉)

Coordinated Science Laboratory, Department of Electrical and Computer Engineering,  
University of Illinois at Urbana-Champaign, 1308 West Main, Urbana, IL 61801, USA  
e-mail: [basar1@illinois.edu](mailto:basar1@illinois.edu)

K. Zhang

e-mail: [kzhang66@illinois.edu](mailto:kzhang66@illinois.edu)

Z. Yang

Department of Operations Research and Financial Engineering, Princeton University,  
98 Charlton St, Princeton, NJ 08540, USA  
e-mail: [zy6@princeton.edu](mailto:zy6@princeton.edu)

providing an assessment of the current state of the field on the mark, to identify fruitful future research directions on theoretical studies of MARL. We expect this chapter to serve as continuing stimulus for researchers interested in working on this exciting while challenging topic.

## 12.1 Introduction

Recent years have witnessed sensational advances of reinforcement learning (RL) in many prominent sequential decision-making problems, such as playing the game of Go [1, 2], playing real-time strategy games [3, 4], robotic control [5, 6], playing card games [7, 8], and autonomous driving [9], especially accompanied with the development of deep neural networks (DNNs) for function approximation [10]. Intriguingly, most of the successful applications involve the participation of more than one single agent/player,<sup>1</sup> which should be modeled systematically as multi-agent RL (MARL) problems. Specifically, MARL addresses the sequential decision-making problem of multiple autonomous agents that operate in a common environment, each of which aims to optimize its own long-term return by interacting with the environment and other agents [11]. Besides the aforementioned popular ones, learning in multi-agent systems finds potential applications in other subareas, including cyber-physical systems [12, 13], finance [14, 15], sensor/communication networks [16, 17], and social science [18, 19].

Largely, MARL algorithms can be placed into three groups, *fully cooperative*, *fully competitive*, and *a mix of the two*, depending on the types of settings they address. In particular, in the cooperative setting, agents collaborate to optimize a common long-term return; while in the competitive setting, the return of agents usually sum up to zero. The mixed setting involves both cooperative and competitive agents, with general-sum returns. Modeling disparate MARL settings requires frameworks spanning from optimization theory, dynamic programming, game theory, and decentralized control, see Sect. 12.2.2 for more detailed discussions. In spite of these existing multiple frameworks, several challenges in MARL are in fact common across the different settings, especially for the theoretical analysis. Specifically, first, the learning goals in MARL are *multi-dimensional*, as the objective of all agents are not necessarily aligned, which brings up the challenge of dealing with equilibrium points, as well as some additional performance criteria beyond return-optimization, such as the efficiency of communication/coordination and robustness against potential adversarial agents. Moreover, as all agents are improving their policies according to their own interests, concurrently, the environment faced by each agent becomes *non-stationary*. This breaks or invalidates the basic framework of most theoretical analyses in the single-agent setting. Furthermore, the joint action space that increases exponentially with the number of agents may cause scalability issues, known as the *combinatorial nature* of MARL [20]. Additionally, the information structure, i.e.,

---

<sup>1</sup>Hereafter, we will use *agent* and *player* interchangeably.

*who knows what*, in MARL is more involved, as each agent has limited access to the observations of others, leading to possibly sub-optimal decision rules locally. A detailed elaboration on the underlying challenges can be found in Sect. 12.3.

There has in fact been no shortage of efforts attempting to address the above challenges. See [11] for a comprehensive overview of earlier theories and algorithms on MARL. Recently, this domain has gained resurgence of interest due to the advances of single-agent RL techniques. Indeed, a huge volume of work on MARL has appeared lately, focusing on either identifying new learning criteria and/or setups [21–24], or developing new algorithms for existing setups, thanks to the development of deep learning [25–31], operations research [32–35], and multi-agent systems [36–39]. Nevertheless, not all the efforts are placed under rigorous theoretical footings, partly due to the limited understanding of even single-agent deep RL theories, and partly due to the inherent challenges in multi-agent settings. As a consequence, it is imperative to review and organize the MARL algorithms with theoretical guarantees, in order to highlight the boundary of existing research endeavors, and stimulate potential future directions on this topic.

In this chapter, we provide a selective overview of theories and algorithms in MARL, together with several significant while challenging applications. More specifically, we focus on two representative frameworks of MARL, namely, Markov/stochastic games and extensive-form games, in discrete-time settings as in standard single-agent RL. In conformity with the aforementioned three groups, we review and pay particular attention to MARL algorithms with convergence and complexity analysis, most of which are fairly recent. With this focus in mind, we note that our overview is by no means comprehensive. In fact, besides the classical reference [11], there are several other reviews on MARL that have appeared recently, due to the resurgence of MARL [20, 40–42]. We would like to emphasize that these reviews provide views and taxonomies that are complementary to ours: [40] surveys the works that are specifically devised to address *opponent-induced non-stationarity*, one of the challenges we discuss in Sect. 12.3; [20, 41] are relatively more comprehensive, but with the focal point on *deep* MARL, a subarea with scarce theories thus far; [42], on the other hand, focuses on algorithms in the *cooperative* setting only, though the review within this setting is extensive.

Finally, we spotlight several new angles and taxonomies that are comparatively underexplored in the existing MARL reviews, primarily owing to our own research endeavors and interests. First, we discuss the framework of extensive-form games in MARL, in addition to the conventional one of Markov games, or even simplified repeated games [11, 20, 40]; second, we summarize the progresses of a recently boosting subarea: decentralized MARL with *networked* agents, as an extrapolation of our early works on this [23, 43, 44]; third, we bring about the *mean-field* regime into MARL, as a remedy for the case with an extremely large population of agents; fourth, we highlight some recent advances in optimization theory, which shed lights on the (non-)convergence of policy-based methods for MARL, especially zero-sum games. We have also reviewed some of the literature on MARL in partially observed settings, but without using deep RL as heuristic solutions. We expect these new angles to help identify fruitful future research directions, and more importantly,

inspire researchers with interests in establishing rigorous theoretical foundations on MARL.

## Roadmap

The remainder of this chapter is organized as follows. In Sect. 12.2, we introduce the background of MARL: standard algorithms for single-agent RL, and the frameworks of MARL. In Sect. 12.3, we summarize several challenges in developing MARL theory, in addition to the single-agent counterparts. A series of MARL algorithms, mostly with theoretical guarantees, is reviewed and organized in Sect. 12.4, according to the types of tasks they address. In Sect. 12.5, we briefly introduce a few recent successes of MARL driven by the algorithms mentioned, followed by conclusions and several open research directions outlined in Sect. 12.6.

## 12.2 Background

In this section, we provide the necessary background on reinforcement learning, in both single- and multi-agent settings.

### 12.2.1 Single-Agent RL

A reinforcement-learning agent is modeled to perform sequential decision-making by interacting with the environment. The environment is usually formulated as an infinite-horizon discounted Markov decision process (MDP), henceforth referred to as Markov decision process<sup>2</sup>, which is formally defined as follows.

**Definition 12.1** A *Markov decision process* is defined by a tuple  $(\mathcal{S}, \mathcal{A}, \mathcal{P}, R, \gamma)$ , where  $\mathcal{S}$  and  $\mathcal{A}$  denote the state and action spaces, respectively;  $\mathcal{P} : \mathcal{S} \times \mathcal{A} \rightarrow \Delta(\mathcal{S})$  denotes the transition probability from any state  $s \in \mathcal{S}$  to any state  $s' \in \mathcal{S}$  for any given action  $a \in \mathcal{A}$ ;  $R : \mathcal{S} \times \mathcal{A} \times \mathcal{S} \rightarrow \mathbb{R}$  is the reward function that determines the immediate reward received by the agent for a transition from  $(s, a)$  to  $s'$ ;  $\gamma \in [0, 1]$  is the discount factor that trades off the instantaneous and future rewards.

As a standard model, MDP has been widely adopted to characterize the decision-making of an agent with *full observability* of the system state  $s$ .<sup>3</sup> At each time  $t$ , the agent chooses to execute an action  $a_t$  in face of the system state  $s_t$ , which causes the system to transition to  $s_{t+1} \sim \mathcal{P}(\cdot | s_t, a_t)$ . Moreover, the agent receives

---

<sup>2</sup>Note that there are several other standard formulations of MDPs, e.g., time-average-reward setting and finite-horizon episodic setting. Here, we only present the classical infinite-horizon discounted setting for ease of exposition.

<sup>3</sup>The partially observed MDP (POMDP) model is usually advocated when the agent has no access to the exact system state but only an *observation* of the state. See [45, 46] for more details on the POMDP model.

an instantaneous reward  $R(s_t, a_t, s_{t+1})$ . The goal of solving the MDP is thus to find a policy  $\pi : \mathcal{S} \rightarrow \Delta(\mathcal{A})$ , a mapping from the state space  $\mathcal{S}$  to the distribution over the action space  $\mathcal{A}$ , so that  $a_t \sim \pi(\cdot | s_t)$  and the discounted accumulated reward

$$\mathbb{E} \left[ \sum_{t \geq 0} \gamma^t R(s_t, a_t, s_{t+1}) \mid a_t \sim \pi(\cdot | s_t), s_0 \right]$$

is maximized. Accordingly, one can define the *action-value function* (Q-function) and *state-value function* (V-function) under policy  $\pi$  as

$$Q_\pi(s, a) = \mathbb{E} \left[ \sum_{t \geq 0} \gamma^t R(s_t, a_t, s_{t+1}) \mid a_t \sim \pi(\cdot | s_t), a_0 = a, s_0 = s \right],$$

$$V_\pi(s) = \mathbb{E} \left[ \sum_{t \geq 0} \gamma^t R(s_t, a_t, s_{t+1}) \mid a_t \sim \pi(\cdot | s_t), s_0 = s \right]$$

for any  $s \in \mathcal{S}$  and  $a \in \mathcal{A}$ , which are the discounted accumulated reward starting from  $(s_0, a_0) = (s, a)$  and  $s_0 = s$ , respectively. The ones corresponding to the optimal policy  $\pi^*$  are usually referred to as the *optimal Q-function* and the *optimal state-value function*, respectively.

By virtue of the Markovian property, the optimal policy can be obtained by dynamic programming/backward induction approaches, e.g., value iteration and policy iteration algorithms [47], which require the knowledge of the model, i.e., the transition probability and the form of reward function. Reinforcement learning, on the other hand, is to find such an optimal policy without knowing the model. The RL agent learns the policy from experiences collected by interacting with the environment. By and large, RL algorithms can be categorized into two mainstream types, *value-based* and *policy-based* methods.

### 12.2.1.1 Value-Based Methods

Value-based RL methods are devised to find a good estimate of the state-action value function, namely, the optimal Q-function  $Q_{\pi^*}$ . The (approximate) optimal policy can then be extracted by taking the greedy action of the Q-function estimate. One of the most popular value-based algorithms is Q-learning [48], where the agent maintains an estimate of the Q-value function  $\hat{Q}(s, a)$ . When transitioning from state-action pair  $(s, a)$  to next state  $s'$ , the agent receives a payoff  $r$  and updates the Q-function according to

$$\hat{Q}(s, a) \leftarrow (1 - \alpha) \hat{Q}(s, a) + \alpha [r + \gamma \max_{a'} \hat{Q}(s', a')], \quad (12.1)$$

where  $\alpha > 0$  is the stepsize/learning rate. Under certain conditions on  $\alpha$ , Q-learning can be proved to converge to the optimal Q-value function almost surely [48, 49],

with finite state and action spaces. Moreover, when combined with neural networks for function approximation, deep Q-learning has achieved great empirical breakthroughs in human-level control applications [10]. Another popular *on-policy* value-based method is SARSA, whose convergence was established in [50] for finite-space settings.

An alternative while popular value-based RL algorithm is Monte Carlo tree search (MCTS) [51–53], which estimates the optimal value function by constructing a search tree via Monte Carlo simulations. Tree policies that judiciously select actions to balance exploration-exploitation are used to build and update the search tree. The most common tree policy is to apply the UCB1 (UCB stands for *upper confidence bound*) algorithm, which was originally devised for stochastic multi-arm bandit problems [54, 55], to each node of the tree. This yields the popular UCT algorithm [52]. Recent research endeavors on the non-asymptotic convergence of MCTS include [56, 57].

Besides, another significant task regarding value functions in RL is to *estimate the value function associated with a given policy* (not only the optimal one). This task, usually referred to as *policy evaluation*, has been tackled by algorithms that follow a similar update as (12.1), named *temporal-difference* (TD) learning [58–60]. Some other common policy evaluation algorithms with convergence guarantees include gradient TD methods with linear [61–63], and nonlinear function approximations [64]. See [65] for a more detailed review on policy evaluation.

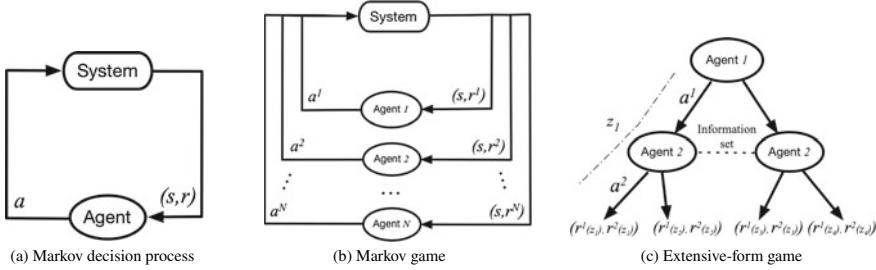
### 12.2.1.2 Policy-Based Methods

Another type of RL algorithms directly searches over the policy space, which is usually estimated by parameterized function approximators like neural networks, namely, approximating  $\pi(\cdot | s) \approx \pi_\theta(\cdot | s)$ . As a consequence, the most straightforward idea, which is to update the parameter along the gradient direction of the long-term reward, has been instantiated by the policy-gradient (PG) method. As a key premise for the idea, the closed form of PG is given as [66]

$$\nabla J(\theta) = \mathbb{E}_{a \sim \pi_\theta(\cdot | s), s \sim \eta_{\pi_\theta}(\cdot)} [Q_{\pi_\theta}(s, a) \nabla \log \pi_\theta(a | s)], \quad (12.2)$$

where  $J(\theta)$  and  $Q_{\pi_\theta}$  are the expected return and Q-function under policy  $\pi_\theta$ , respectively,  $\nabla \log \pi_\theta(a | s)$  is the score function of the policy, and  $\eta_{\pi_\theta}$  is the state-occupancy measure, either discounted or ergodic, under policy  $\pi_\theta$ . Then, various policy-gradient methods, including REINFORCE [67], G(PO)MDP [68], and actor-critic algorithms [69, 70], have been proposed by estimating the gradient in different ways. A similar idea also applies to deterministic policies in continuous-action settings, whose PG form has been derived recently by [71]. Besides gradient-based ones, several other policy optimization methods have achieved state-of-the-art performance in many applications, including PPO [72], TRPO [73], soft actor-critic [74].

Compared with value-based RL methods, policy-based ones enjoy better convergence guarantees [69, 75–77], especially with neural networks for function approximation [78, 79], which can readily handle massive or even continuous state-action



**Fig. 12.1** Schematic diagrams for the system evolution of a Markov decision process, a Markov game, and an extensive-form game, which correspond to the frameworks for single- and multi-agent RL, respectively. Specifically, in an MDP as in **(a)**, the agent observes the state  $s$  and receives reward  $r$  from the system, after outputting the action  $a$ ; in an MG as in **(b)**, all agents choose actions  $a^i$  simultaneously, after observing the system state  $s$  and receiving each individual reward  $r^i$ ; in a two-player extensive-form game as in **(c)**, the agents make decisions on choosing actions  $a^i$  alternately, and receive each individual reward  $r^i(z)$  at the end of the game, with  $z$  being the terminal history. In the imperfect information case, player 2 is uncertain about where he/she is in the game, which makes the information set non-singleton

spaces. Besides the value- and policy-based methods, there also exist RL algorithms based on the linear program formulation of an MDP, see recent efforts in [80, 81].

## 12.2.2 Multi-Agent RL Framework

In a similar vein, multi-agent RL also addresses sequential decision-making problems, but with more than one agent involved. In particular, both the evolution of the system state and the reward received by each agent are influenced by the joint actions of all agents. More intriguingly, each agent has its own long-term reward to optimize, which now becomes a function of the policies of all other agents. Such a general model finds broad applications in practice, see Sect. 12.5 for a detailed review of several prominent examples.

In general, there exist two seemingly different but closely related theoretical frameworks for MARL, Markov/stochastic games, and extensive-form games, as to be introduced next. Evolution of the systems under different frameworks are illustrated in Fig. 12.1.

### 12.2.2.1 Markov/Stochastic Games

One direct generalization of MDP that captures the intertwining of multiple agents is Markov games (MGs), also known as stochastic games [82]. Originated from the

seminal work [83], the framework of MGs<sup>4</sup> has long been used in the literature to develop MARL algorithms, see Sect. 12.4 for more details. We introduce the formal definition as below:

**Definition 12.2** A *Markov game* is defined by a tuple  $(\mathcal{N}, \mathcal{S}, \{\mathcal{A}^i\}_{i \in \mathcal{N}}, \mathcal{P}, \{R^i\}_{i \in \mathcal{N}}, \gamma)$ , where  $\mathcal{N} = \{1, \dots, N\}$  denotes the set of  $N > 1$  agents,  $\mathcal{S}$  denotes the state space observed by all agents,  $\mathcal{A}^i$  denotes the action space of agent  $i$ . Let  $\mathcal{A} := \mathcal{A}^1 \times \dots \times \mathcal{A}^N$ , then  $\mathcal{P} : \mathcal{S} \times \mathcal{A} \rightarrow \Delta(\mathcal{S})$  denotes the transition probability from any state  $s \in \mathcal{S}$  to any state  $s' \in \mathcal{S}$  for any joint action  $a \in \mathcal{A}$ ;  $R^i : \mathcal{S} \times \mathcal{A} \times \mathcal{S} \rightarrow \mathbb{R}$  is the reward function that determines the immediate reward received by agent  $i$  for a transition from  $(s, a)$  to  $s'$ ;  $\gamma \in [0, 1)$  is the discount factor.

At time  $t$ , each agent  $i \in \mathcal{N}$  executes an action  $a_t^i$ , according to the system state  $s_t$ . The system then transitions to state  $s_{t+1}$ , and rewards each agent  $i$  by  $R^i(s_t, a_t, s_{t+1})$ . The goal of agent  $i$  is to optimize its own long-term reward, by finding the policy  $\pi^i : \mathcal{S} \rightarrow \Delta(\mathcal{A}^i)$  such that  $a_t^i \sim \pi^i(\cdot | s_t)$ . As a consequence, the value function  $V^i : \mathcal{S} \rightarrow \mathbb{R}$  of agent  $i$  becomes a function of the joint policy  $\pi : \mathcal{S} \rightarrow \Delta(\mathcal{A})$  defined as  $\pi(a | s) := \prod_{i \in \mathcal{N}} \pi^i(a^i | s)$ . In particular, for any joint policy  $\pi$  and state  $s \in \mathcal{S}$ ,

$$V_{\pi^i, \pi^{-i}}^i(s) := \mathbb{E} \left[ \sum_{t \geq 0} \gamma^t R^i(s_t, a_t, s_{t+1}) \mid a_t^i \sim \pi^i(\cdot | s_t), s_0 = s \right], \quad (12.3)$$

where  $-i$  represents the indices of all agents in  $\mathcal{N}$  except agent  $i$ . Hence, the solution concept of an MG deviates from that of an MDP, since the *optimal* performance of each agent is controlled not only by its own policy, but also the choices of all other players of the game. The most common solution concept, Nash equilibrium (NE)<sup>5</sup>, is defined as follows [84, 85]:

**Definition 12.3** A *Nash equilibrium* of the Markov game  $(\mathcal{N}, \mathcal{S}, \{\mathcal{A}^i\}_{i \in \mathcal{N}}, \mathcal{P}, \{R^i\}_{i \in \mathcal{N}}, \gamma)$  is a joint policy  $\pi^* = (\pi^{1,*}, \dots, \pi^{N,*})$ , such that for any  $s \in \mathcal{S}$  and  $i \in \mathcal{N}$

$$V_{\pi^{i,*}, \pi^{-i,*}}^i(s) \geq V_{\pi^i, \pi^{-i,*}}^i(s), \quad \text{for any } \pi^i.$$

Nash equilibrium characterizes an equilibrium point  $\pi^*$ , from which none of the agents has any incentive to deviate. In other words, for any agent  $i \in \mathcal{N}$ , the policy  $\pi^{i,*}$  is the *best response* of  $\pi^{-i,*}$ . As a standard learning goal for MARL, NE always exists for finite-space infinite-horizon discounted MGs [85], but may not be unique in general. Most of the MARL algorithms are contrived to converge to such an equilibrium point, if it exists.

The framework of Markov games is general enough to umbrella various MARL settings summarized below:

---

<sup>4</sup>Similar to the single-agent setting, here we only introduce the infinite-horizon discounted setting for simplicity, though other settings of MGs, e.g., time-average-reward setting and finite-horizon episodic setting, also exist [85].

<sup>5</sup>Here, we focus only on stationary Markov Nash equilibria, for the infinite-horizon discounted MGs considered.

### Cooperative Setting

In a fully cooperative setting, all agents usually share a common reward function, i.e.,  $R^1 = R^2 = \dots = R^N = R$ . We note that this model is also referred to as *multi-agent MDPs* (MMDPs) in the AI community [86, 87], and *Markov teams/team Markov games* in the control/game theory community [88–91]. Moreover, from the game-theoretic perspective, this cooperative setting can also be viewed as a special case of Markov potential games [22, 92, 93], with the potential function being the common accumulated reward. With this model in mind, the value function and Q-function are identical to all agents, which thus enables the single-agent RL algorithms, e.g., Q-learning update (12.1), to be applied, if all agents are coordinated as one decision-maker. The global optimum for cooperation now constitutes a Nash equilibrium of the game.

Besides the common reward model, another slightly more general and surging model for cooperative MARL considers *team-average* reward [23, 94, 95]. Specifically, agents are allowed to have different reward functions, which may be kept private to each agent, while the goal for cooperation is to optimize the long-term reward corresponding to the average reward  $\bar{R}(s, a, s') := N^{-1} \cdot \sum_{i \in N} R^i(s, a, s')$  for any  $(s, a, s') \in \mathcal{S} \times \mathcal{A} \times \mathcal{S}$ . The average reward model, which allows more heterogeneity among agents, includes the model above as a special case. It also preserves privacy among agents, and facilitates the development of *decentralized* MARL algorithms [23, 94, 96]. Such heterogeneity also necessitates the incorporation of *communication* protocols into MARL, and the analysis of communication-efficient MARL algorithms.

### Competitive Setting

Fully competitive setting in MARL is typically modeled as *zero-sum* Markov games, namely,  $\sum_{i \in N} R^i(s, a, s') = 0$  for any  $(s, a, s')$ . For ease of algorithm analysis and computational tractability, most literature focused on *two* agents that compete against each other [83], where clearly the reward of one agent is exactly the loss of the other. In addition to direct applications to game-playing [2, 83, 97], zero-sum games also serve as a model for *robust* learning, since the *uncertainty* that impedes the learning process of the agent can be accounted for as a fictitious opponent in the game that is always against the agent [98–100]. Therefore, the Nash equilibrium yields a robust policy that optimizes the *worst-case* long-term reward.

### Mixed Setting

Mixed setting is also known as the *general-sum* game setting, where no restriction is imposed on the goal and relationship among agents [101, 102]. Each agent is self-interested, whose reward may be conflicting with others. Equilibrium solution concepts from game theory, such as Nash equilibrium [84], have the most significant influence on algorithms that are developed for this general setting. Furthermore, we include the setting with both fully cooperative and competitive agents, for example, two zero-sum competitive teams with cooperative agents in each team [3, 44, 103], as instances of the mixed setting as well.

### 12.2.2.2 Extensive-Form Games

Even though they constitute a classical formalism for MARL, Markov games can only handle the fully observed case, i.e., the agent has *perfect information* on the system state  $s_t$  and the executed action  $a_t$  at time  $t$ . Nonetheless, a plethora of MARL applications involve agents with only partial observability, i.e., *imperfect information* of the game. Extension of Markov games to partially observed case may be applicable, which, however, is challenging to solve, even under the cooperative setting [36, 104].<sup>6</sup>

In contrast, another framework, named *extensive-form games* [105, 106], can handily model imperfect information for multi-agent decision-making. This framework is rooted in computational game theory and has been shown to admit polynomial-time algorithms under mild conditions [107]. We briefly introduce the framework of extensive-form games as follows.

**Definition 12.4** An *extensive-form game* is defined by  $(\mathcal{N} \cup \{c\}, \mathcal{H}, \mathcal{Z}, \mathcal{A}, \{R^i\}_{i \in \mathcal{N}}, \tau, \pi^c, \mathcal{S})$ , where  $\mathcal{N} = \{1, \dots, N\}$  denotes the set of  $N > 1$  agents, and  $c$  is a special agent called *chance* or *nature*, which has a fixed stochastic policy that specifies the randomness of the environment. Besides,  $\mathcal{A}$  is the set of all possible actions that agents can take and  $\mathcal{H}$  is the set of all possible *histories*, where each history is a sequence of actions taken from the beginning of the game. Let  $\mathcal{A}(h) = \{a \mid ha \in \mathcal{H}\}$  denote the set of actions available after a nonterminal history  $h$ . Suppose an agent takes action  $a \in \mathcal{A}(h)$  given history  $h \in \mathcal{H}$ , which then leads to a new history  $ha \in \mathcal{H}$ . Among all histories,  $\mathcal{Z} \subseteq \mathcal{H}$  is a subset of *terminal histories* that represents the completion of a game. A utility is assigned to each agent  $i \in \mathcal{N}$  at a terminal history, dictated by the function  $R^i : \mathcal{Z} \rightarrow \mathbb{R}$ . Moreover,  $\tau : \mathcal{H} \rightarrow \mathcal{N} \cup \{c\}$  is the *identification* function that specifies which agent takes the action at each history. If  $\tau(h) = c$ , the chance agent takes an action  $a$  according to its policy  $\pi^c$ , i.e.,  $a \sim \pi^c(\cdot \mid h)$ . Furthermore,  $\mathcal{S}$  is the partition of  $\mathcal{H}$  such that for any  $s \in \mathcal{S}$  and any  $h, h' \in s$ , we have  $\tau(h) = \tau(h')$  and  $\mathcal{A}(h) = \mathcal{A}(h')$ . In other words, histories  $h$  and  $h'$  in the same partition are indistinguishable to the agent that is about to take action, namely  $\tau(h)$ . The elements in  $\mathcal{S}$  are referred to as *information states*.

Intuitively, the imperfect information of an extensive-form game is reflected by the fact that agents cannot distinguish between histories in the same information set. Since we have  $\tau(h) = \tau(h')$  and  $\mathcal{A}(h) = \mathcal{A}(h')$  for all  $h, h' \in s$ , for ease of presentation, in the sequel, for all  $h, h' \in s$  and  $s \in \mathcal{S}$ , we let  $\mathcal{A}(s)$  and  $\tau(s)$  denote  $\mathcal{A}(h)$  and  $\tau(h)$ , respectively. We also define a mapping  $I : \mathcal{H} \rightarrow \mathcal{S}$  by letting  $I(h) = s$  if  $h \in s$ . Moreover, we only consider games where both  $\mathcal{H}$  and  $\mathcal{A}$  are finite sets. To simplify the notation, for any two histories  $h, h' \in \mathcal{H}$ , we refer to  $h$  as a *prefix* of  $h'$ , denoted by  $h \sqsubseteq h'$ , if  $h'$  can be reached from  $h$  by taking a sequence of actions. In this case, we call  $h'$  a *suffix* of  $h$ . Furthermore, we assume throughout that the game features *perfect recall*, which implies that each agent remembers the sequence of the information states and actions that have led to its current information state. The assumption

---

<sup>6</sup>Partially observed Markov games under the cooperative setting are usually formulated as decentralized POMDP (Dec-POMDP) problems. See Sect. 12.4.1.3 for more discussions on this setting.

of perfect recall is commonly made in the literature, which enables the existence of polynomial-time algorithms for solving the game [107]. More importantly, by the celebrated Kuhn's theorem [108], under such an assumption, to find the set of Nash equilibria, it suffices to restrict the derivation to the set of *behavioral policies* which map each information set  $s \in \mathcal{S}$  to a probability distribution over  $\mathcal{A}(s)$ . For any  $i \in \mathcal{N}$ , let  $\mathcal{S}^i = \{s \in \mathcal{S}: \tau(s) = i\}$  be the set of information states of agent  $i$ . A joint policy of the agents is denoted by  $\pi = (\pi^1, \dots, \pi^N)$ , where  $\pi^i : \mathcal{S}^i \rightarrow \Delta(\mathcal{A}(s))$  is the policy of agent  $i$ . For any history  $h$  and any joint policy  $\pi$ , we define the *reach probability* of  $h$  under  $\pi$  as

$$\eta_\pi(h) = \prod_{h': h'a \sqsubseteq h} \pi^{\tau(h')}(a | I(h')) = \prod_{i \in \mathcal{N} \cup \{c\}} \prod_{h': h'a \sqsubseteq h, \tau(h')=i} \pi^i(a | I(h')), \quad (12.4)$$

which specifies the probability that  $h$  is created when all agents follow  $\pi$ . We similarly define the reach probability of an information state  $s$  under  $\pi$  as  $\eta_\pi(s) = \sum_{h \in s} \eta_\pi(h)$ . The expected utility of agent  $i \in \mathcal{N}$  is thus given by  $\sum_{z \in \mathcal{Z}} \eta_\pi(z) \cdot R^i(z)$ , which is denoted by  $R^i(\pi)$  for simplicity. Now we are ready to introduce the solution concept for extensive-form games, i.e., Nash equilibrium and its  $\epsilon$ -approximation, as follows.

**Definition 12.5** An  $\epsilon$ -*Nash equilibrium* of an extensive-form game represented by  $(\mathcal{N} \cup \{c\}, \mathcal{H}, \mathcal{Z}, \mathcal{A}, \{R^i\}_{i \in \mathcal{N}}, \tau, \pi^c, \mathcal{S})$  is a joint policy  $\pi^* = (\pi^{1,*}, \dots, \pi^{N,*})$ , such that for any  $i \in \mathcal{N}$ ,

$$R^i(\pi^{i,*}, \pi^{-i,*}) \geq R^i(\pi^i, \pi^{-i,*}) - \epsilon, \quad \text{for any policy } \pi^i \text{ of agent } i.$$

Here  $\pi^{-i}$  denotes the joint policy of agents in  $\mathcal{N} \setminus \{i\}$  where agent  $j$  adopts policy  $\pi^j$  for all  $j \in \mathcal{N} \setminus \{i\}$ . Additionally, if  $\epsilon = 0$ ,  $\pi^*$  constitutes a *Nash equilibrium*.

### Various Settings

Extensive-form games are in general used to model non-cooperative settings. Specifically, zero-sum/constant-sum utility with  $\sum_{i \in \mathcal{N}} R^i = k$  for some constant  $k$  corresponds to the fully competitive setting; general-sum utility function results in the mixed setting. More importantly, settings of different information structures can also be characterized by extensive-form games. In particular, a *perfect information* game is one where each information set is a singleton, i.e., for any  $s \in \mathcal{S}$ ,  $|s| = 1$ ; an *imperfect-information* game is one where there exists  $s \in \mathcal{S}$ ,  $|s| > 1$ . In other words, with imperfect information, the information state  $s$  used for decision-making represents more than one possible history, and the agent cannot distinguish between them.

Among various settings, the zero-sum imperfect information setting has been the main focus of theoretical studies that bridge MARL and extensive-form games [109–112]. It has also motivated MARL algorithms that revolutionized competitive setting applications like Poker AI [8, 113].

## Connection to Markov Games

Note that the two formalisms in Definitions 12.2 and 12.4 are connected. In particular, for simultaneous-move Markov games, the choices of actions by other agents are unknown to an agent, which thus leads to different histories that can be aggregated as one information state  $s$ . Histories in these games are then sequences of *joint* actions, and the discounted accumulated reward instantiates the utility at the end of the game. Conversely, by simply setting  $\mathcal{A}^j = \emptyset$  at the state  $s$  for agents  $j \neq \tau(s)$ , the extensive-form game reduces to a Markov game with *state-dependent* action spaces. See [114] for a more detailed discussion on the connection.

**Remark 12.1** (*Other MARL Frameworks*) Several other theoretical frameworks for MARL also exist in the literature, e.g., normal-form and/or repeated games [115–118], and partially observed Markov games [119–121]. However, the former framework can be viewed as a special case of MGs, with a singleton state; most early theories of MARL in this framework have been restricted to small-scale problems [116–118] only. MARL in the latter framework, on the other hand, is inherently challenging to address in general [104, 119], leading to relatively scarce theories in the literature. Due to space limitation, we do not introduce these models here in any detail. We will briefly review MARL algorithms under some of these models, especially the partially observed setting, in Sect. 12.4, though. Interested readers are referred to the early review [11] for more discussions on MARL in normal-form/repeated games.

## 12.3 Challenges in MARL Theory

Despite a general model that finds broad applications, MARL suffers from several challenges in theoretical analysis, in addition to those that arise in single-agent RL. We summarize below the challenges that we regard as fundamental in developing theories for MARL.

### 12.3.1 Non-unique Learning Goals

Unlike single-agent RL, where the goal of the agent is to maximize the long-term return efficiently, the learning goals of MARL can be vague at times. In fact, as argued in [122], the *unclarity* of the problems being addressed is the fundamental flaw in many early MARL works. Indeed, the goals that need to be considered in the analysis of MARL algorithms can be multi-dimensional. The most common goal, which has, however, been challenged in [122], is the convergence to Nash equilibrium as defined in Sect. 12.2.2. By definition, NE characterizes the point that no agent will deviate from, if any algorithm *finally* converges. This is undoubtedly a reasonable solution concept in game theory, under the assumption that the agents are all *rational*, and are capable of perfectly reasoning and infinite mutual modeling of agents. However, with *bounded rationality*, the agents may only be able to perform *finite* mutual modeling

[106]. As a result, the learning dynamics that are devised to converge to NE may not be justifiable for practical MARL agents. Instead, the goal may be focused on designing the best *learning strategy* for a given agent and *a fixed class of the other agents in the game*. In fact, these two goals are styled as *equilibrium agenda* and *AI agenda* in [122].

Besides, it has also been controversial that *convergence* (to the equilibrium point) is the dominant performance criterion for MARL algorithm analysis. In fact, it has been recognized in [123] that value-based MARL algorithms fail to converge to the *stationary* NE of general-sum Markov games, which motivated the new solution concept of *cyclic equilibrium* therein, at which the agents cycle rigidly through a set of stationary policies, i.e., not converging to any NE policy. Alternatively, [116, 124] separate the learning goal into being both *stable* and *rational*, where the former ensures the algorithm to be convergent, given a predefined, targeted class of opponents' algorithms, while the latter requires the convergence to a best response when the other agents remain stationary. If all agents are both stable and rational, convergence to NE naturally arises in this context. Moreover, the notion of *regret* introduces another angle to capture agents' rationality, which measures the performance of the algorithm compared to the best hindsight static strategy [116, 125, 126]. No-regret algorithms with asymptotically zero average regret guarantee the convergence to the equilibria of certain games [109, 125, 127], which in essence guarantee that the agent is not *exploited* by others.

In addition to the goals concerning optimizing the return, several other goals that are special to multi-agent systems have also drawn increasing attention. For example, [21, 128, 129] investigate *learning to communicate*, in order for the agents to better coordinate. Such a concern on communication protocols has naturally inspired the recent studies on *communication-efficient* MARL [130–133]. Other important goals include how to learn without over-fitting certain agents [26, 134, 135], and how to learn robustly with either malicious/adversarial or failed/dysfunctional learning agents [136–138]. Still in their infancy, some works concerning aforementioned goals provide only empirical results, leaving plenty of room for theoretical studies.

### 12.3.2 Non-stationarity

Another key challenge of MARL lies in the fact that multiple agents usually learn concurrently, causing the environment faced by each individual agent to be *non-stationary*. In particular, the action taken by one agent affects the reward of other opponent agents, and the evolution of the state. As a result, the learning agent is required to account for how the other agents behave and adapt to the *joint behavior* accordingly. This invalidates the stationarity assumption for establishing the convergence of single-agent RL algorithms, namely, the stationary Markovian property of the environment such that the individual reward and current state depend only on the previous state and action taken. This precludes the direct use of mathematical tools for single-agent RL analysis in MARL.

Indeed, theoretically, if the agent ignores this issue and optimizes its own policy assuming a stationary environment, which is usually referred to as an *independent learner*, the algorithms may fail to converge [115, 139], except for several special settings [37, 38]. Empirically, however, independent learning may achieve satisfiable performance in practice [140, 141]. As the most well-known issue in MARL, non-stationarity has long been recognized in the literature [11, 142]. A recent comprehensive survey [40] peculiarly provides an overview on how it is modeled and addressed by state-of-the-art multi-agent learning algorithms. We thus do not include any further discussion on this challenge, and refer interested readers to [40].

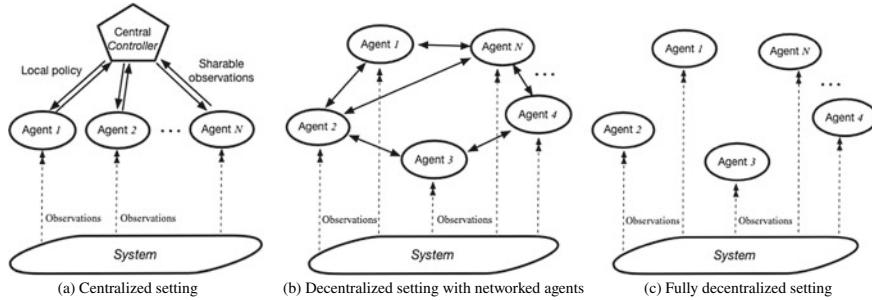
### 12.3.3 Scalability Issue

To handle non-stationarity, each individual agent may need to account for the *joint action space*, whose dimension increases exponentially with the number of agents. This is also referred to as the *combinatorial nature* of MARL [20]. Having a large number of agents complicates the theoretical analysis, especially the convergence analysis, of MARL. This argument is substantiated by the fact that theories on MARL for the two-player zero-sum setting are much more extensive and advanced than those for general-sum settings with more than two agents, see Sect. 12.4 for a detailed comparison. One possible remedy for the scalability issue is to assume additionally the *factorized* structures of either the value or reward functions with regard to the action dependence, see [143–145] for the original heuristic ideas, and [146, 147] for recent empirical progress. Relevant theoretical analysis had not been established until recently [148], which considers a special dependence structure, and develops a provably convergent model-based algorithm.

Another theoretical challenge of MARL that is brought about independently of, but worsened by, the scalability issue, is to build up theories for deep multi-agent RL. Particularly, scalability issues necessitate the use of function approximation, especially deep neural networks, in MARL. Though empirically successful, the theoretical analysis of deep MARL is an almost uncharted territory, with the currently limited understanding of deep learning theory, not alone the deep RL theory. This is included as one of the future research directions in Sect. 12.6.

### 12.3.4 Various Information Structures

Compared to the single-agent case, the information structure of MARL, namely, *who knows what* at the training and execution, is more involved. For example, in the framework of Markov games, it suffices to observe the instantaneous state  $s_t$ , in order for each agent to make decisions, since the local policy  $\pi^i$  mapping from  $\mathcal{S}$  to  $\Delta(\mathcal{A}^i)$  contains the equilibrium policy. On the other hand, for extensive-form games, each agent may need to recall the history of past decisions, under the com-



**Fig. 12.2** Three representative information structures in MARL. Specifically, in **(a)**, there exists a central controller that can aggregate information from the agents, e.g., joint actions, joint rewards, and joint observations, and even design policies for all agents. The information exchanged between the central controller and the agents can thus include both some private observations from the agents, and the local policies designed for each agent from the controller. In both **(b)** and **(c)**, there is no such a central controller, and are thus both referred to as decentralized structures. In **(b)**, agents are connected via a possibly time-varying communication network, so that the local information can spread across the network, by information exchange with only each agent's neighbors. **(b)** is more common in cooperative MARL settings. In **(c)**, the agents are full decentralized, with no explicit information exchange with each other. Instead, each agent makes decisions based on its local observations, without any coordination and/or aggregation of data. The local observations that vary across agents, however, may contain some global information, e.g., the joint actions of other agents, namely, the control sharing information structure [149]. Such a fully decentralized structure can also be found in many game-theoretic learning algorithms

mon perfect recall assumption. Furthermore, as self-interested agents, each agent can scarcely access either the *policy* or the rewards of the opponents, but at most the action samples taken by them. This partial information aggravates the issues caused by non-stationarity, as the samples can hardly recover the exact behavior of the opponents' underlying policies, which increases the non-stationarity viewed by individual agents. The extreme case is the aforementioned *independent learning* scheme, which assumes the observability of only the local action and reward, and suffers from non-convergence in general [139].

Learning schemes resulting from various information structures lead to various levels of difficulty for theoretical analysis. Specifically, to mitigate the partial information issue above, a great deal of work assumes the existence of a *central controller* that can collect information such as joint actions, joint rewards, and joint observations, and even design policies for all agents [26, 28, 119, 130, 141, 147, 150–152]. This structure gives birth to the popular learning scheme of *centralized-learning-decentralized-execution*, which stemmed from the works on planning for the partially observed setting, namely, Dec-POMDPs [119, 150, 153], and has been widely adopted in recent (deep) MARL works [26, 28, 130, 141, 147, 151]. For cooperative settings, this learning scheme greatly simplifies the analysis, allowing the use of tools for single-agent RL analysis. Though, for non-cooperative settings with heterogeneous agents, this scheme does not significantly simplify the analysis, as the learning goals of the agents are not aligned, see Sect. 12.3.1.

Nonetheless, generally such a central controller does not exist in many applications, except the ones that can easily access a simulator, such as video games and robotics. As a consequence, a *fully decentralized* learning scheme is preferred, which includes the aforementioned independent learning scheme as a special case. To address the non-convergence issue in independent learning, agents are usually allowed to exchange/share local information with their neighbors over a communication network [23, 43, 44, 94–96, 131, 154–158]. We refer to this setting as *a decentralized one with networked agents*. Theoretical analysis for convergence is then made possible in this setting, the difficulty of which sits between that of single-agent RL and general MARL algorithms. Three different information structures are depicted in Fig. 12.2.

## 12.4 MARL Algorithms with Theory

This section provides a selective review of MARL algorithms, and categorizes them according to the tasks to address. Exclusively, we review here the works that are focused on the theoretical studies only, which are mostly built upon the two representative MARL frameworks, fully observed Markov games and extensive-form games, introduced in Sect. 12.2.2. A brief summary on MARL for partially observed Markov games in *cooperative* settings, namely, the Dec-POMDP problems, is also provided below in Sect. 12.4.1, due to their relatively more mature theory than that of MARL for general partially observed Markov games.

### 12.4.1 Cooperative Setting

Cooperative MARL constitutes a great portion of MARL settings, where all agents collaborate with each other to achieve some shared goal. Most cooperative MARL algorithms backed by theoretical analysis are devised for the following more specific settings.

#### 12.4.1.1 Homogeneous Agents

A majority of cooperative MARL settings involve *homogeneous* agents with a *common* reward function that aligns all agents' interests. In the extreme case with large populations of agents, such a homogeneity also indicates that the agents play an *interchangeable* role in the system evolution, and can hardly be distinguished from each other. We elaborate more on homogeneity below.

### **Multi-Agent MDP and Markov Teams**

Consider a Markov game as in Definition 12.2 with  $R^1 = R^2 = \dots = R^N = R$ , where the reward  $R : \mathcal{S} \times \mathcal{A} \times \mathcal{S} \rightarrow \mathbb{R}$  is influenced by the joint action in  $\mathcal{A} = \mathcal{A}^1 \times \dots \times \mathcal{A}^N$ . As a result, the Q-function is identical for all agents. Hence, a straightforward algorithm proceeds by performing the standard Q-learning update (12.1) at each agent, but taking the max over the joint action space  $a' \in \mathcal{A}$ . Convergence to the optimal/equilibrium Q-function has been established in [49, 159], when both state and action spaces are finite.

However, convergence of the Q-function does not necessarily imply that of the equilibrium policy for the Markov team, as any combination of equilibrium policies extracted at each agent may not constitute an equilibrium policy, if the equilibrium policies are non-unique, and the agents fail to agree on which one to select. Hence, convergence to the NE policy is only guaranteed if either the equilibrium is assumed to be unique [159], or the agents are coordinated for equilibrium selection. The latter idea has first been validated in the cooperative repeated games setting [115], a special case of Markov teams with a singleton state, where the agents are joint-action learners (JAL), maintaining a Q-value for joint actions, and learning empirical models of all others. Convergence to equilibrium point is claimed in [115], without a formal proof. For the actual Markov teams, this coordination has been exploited in [90], which proposes *optimal adaptive learning* (OAL), the first MARL algorithm with provable convergence to the equilibrium policy. Specifically, OAL first learns the game structure, and constructs virtual games at each state that are *weakly acyclic* with respect to (w.r.t.) a biased set. OAL can be shown to converge to the NE, by introducing the biased adaptive play learning algorithm for the constructed weakly acyclic games, motivated from [160].

Apart from equilibrium selection, another subtlety special to Markov teams (compared to single-agent RL) is the necessity to address the scalability issue, see Sect. 12.3.3. As independent Q-learning may fail to converge [139], one early attempt toward developing scalable while convergent algorithms for MMDPs is [87], which advocates a distributed Q-learning algorithm that converges for *deterministic* finite MMDPs. Each agent maintains only a Q-table of state  $s$  and local action  $a^i$ , and successively takes maximization over the joint action  $a'$ . No other agent's actions and their histories can be acquired by each individual agent. Several other heuristics (with no theoretical backing) regarding either reward or value function factorization have been proposed to mitigate the scalability issue [143–147]. Very recently, [161] provides a rigorous characterization of conditions that justify this value factorization idea. Another recent theoretical work along this direction is [148], which imposes a special dependence structure, i.e., a one-directional tree, so that the (near-)optimal policy of the overall MMDP can be provably well approximated by *local policies*. More recently, [38] has studied common interest games, which includes Markov teams as an example, and develops a *decentralized* RL algorithm that relies on only states, local actions and rewards. With the same information structure as independent Q-learning [139], the algorithm is guaranteed to converge to *team optimal* equilibrium policies, and not just equilibrium policies. This is important as in general, a sub-optimal equilibrium can perform arbitrarily worse than the optimal equilibrium [38].

For policy-based methods, to our knowledge, the only convergence guarantee for this setting exists in [162]. The authors propose two-timescale actor–critic *fictitious play* algorithms, where at the slower timescale, the actor mixes the current policy and the best response one w.r.t. the local Q-value estimate, while at the faster timescale the critic performs policy evaluation, as if all agents’ policies are stationary. Convergence is established for *simultaneous-move multistage games* with a common (also zero-sum, see Sect. 12.4.2.2) reward, a special Markov team with initial and absorbing states, and each state being visited only once.

### Markov Potential Games

From a game-theoretic perspective, a more general framework to embrace cooperation is *potential games* [163], where there exists some *potential* function shared by all agents, such that if any agent changes its policy unilaterally, the change in its reward equals (or proportions to) that in the potential function. Though most potential games are stateless, an extension named *Markov potential games* (MPGs) has gained increasing attention for modeling *sequential* decision-making [22, 92], which includes Markovian states whose evolution is affected by the joint actions. Indeed, MMDPs/Markov teams constitute a particular case of MPGs, with the potential function being the common reward; such dynamic games can also be viewed as being *strategically equivalent* to Markov teams, using the terminology in, e.g., [164, Chap. 1]. Under this model, [93] provides verifiable conditions for a Markov game to be an MPG, and shows the equivalence between finding closed-loop NE in MPG and solving a single-agent optimal control problem. Hence, single-agent RL algorithms are then enabled to solve this MARL problem.

### Mean-Field Regime

Another idea toward tackling the scalability issue is to take the setting to the *mean-field* regime, with an extremely large number of homogeneous agents. Each agent’s effect on the overall multi-agent system can thus become infinitesimal, resulting in all agents being interchangeable/indistinguishable. The interaction with other agents, however, is captured simply by some mean-field quantity, e.g., the average state, or the empirical distribution of states. Each agent only needs to find the best response to the mean-field, which considerably simplifies the analysis. This mean-field view of multi-agent systems has been approached by the mean-field games (MFGs) model [165–169], the team model with mean-field sharing [170, 171], and the game model with mean-field actions [172].<sup>7</sup>

MARL in these models have not been explored until recently, mostly in the non-cooperative setting of MFGs, see Sect. 12.4.3 for a more detailed review. Regarding the cooperative setting, recent work [175] studies RL for Markov teams with mean-field sharing [170, 171, 176]. Compared to MFG, the model considers agents that share a common reward function depending only on the local state and the mean-field, which encourages cooperation among the agents. Also, the term mean-field

---

<sup>7</sup>The difference between mean-field teams and mean-field games is mainly the solution concept: optimum versus equilibrium, as the difference between general dynamic team theory [88, 173, 174] and game theory [82, 85]. Although the former can be viewed as a special case of the latter, related works are usually reviewed separately in the literature. We follow here this convention.

refers to the *empirical average* for the states of *finite* population, in contrast to the *expectation* and *probability distribution* of *infinite* population in MFGs. Based on the dynamic programming decomposition for the specified model [170], several popular RL algorithms are easily translated to address this setting [175]. More recently, [177, 178] approach the problem from a mean-field control (MFC) model, to model large-population of cooperative decision-makers. Policy gradient methods are proved to converge for linear quadratic MFCs in [177], and mean-field Q-learning is then shown to converge for general MFCs [178].

### 12.4.1.2 Decentralized Paradigm with Networked Agents

Cooperative agents in numerous practical multi-agent systems are not always homogeneous. Agents may have different preferences, i.e., reward functions, while they still form a team to maximize the return of the *team-average* reward  $\bar{R}$ , with  $\bar{R}(s, a, s') = N^{-1} \cdot \sum_{i \in N} R^i(s, a, s')$ . More subtly, the reward function is sometimes not sharable with others, as the preference is kept private to each agent. This setting finds broad applications in engineering systems as sensor networks [179], smart grid [180, 181], intelligent transportation systems [12, 182], and robotics [183].

Covering the homogeneous setting in Sect. 12.4.1.1 as a special case, the specified one definitely requires more coordination, as, for example, the global value function cannot be estimated locally without knowing other agents' reward functions. With a central controller, most MARL algorithms reviewed in Sect. 12.4.1.1 directly apply, since the controller can collect and average the rewards, and distributes the information to all agents. Nonetheless, such a controller may not exist in most aforementioned applications, due to either cost, scalability, or robustness concerns [179, 180, 184]. Instead, the agents may be able to share/exchange information with their neighbors over a possibly time-varying and sparse communication network, as illustrated in Fig. 12.2b. Though MARL under this *decentralized/distributed*<sup>8</sup> paradigm is imperative, it is relatively less-investigated, in comparison to the extensive results on distributed/consensus algorithms that solve *static/one-stage* optimization problems [185–188], which, unlike RL, involves no system *dynamics*, and does not maximize the *long-term* objective as a sequential-decision making problem.

#### Learning Optimal Policy

The most significant goal is to learn the optimal joint policy, while each agent only accesses to local and neighboring information over the network. The idea of MARL with networked agents dates back to [189]. To our knowledge, the first provably convergent MARL algorithm under this setting is due to [94], which incorporates the idea of *consensus + innovation* to the standard Q-learning algorithm, yielding the *QD-learning* algorithm with the following update

---

<sup>8</sup>Note that hereafter we use *decentralized* and *distributed* interchangeably for describing this paradigm.

$$\begin{aligned} Q_{t+1}^i(s, a) &\leftarrow Q_t^i(s, a) + \alpha_{t,s,a} \left[ R^i(s, a) + \gamma \max_{a' \in \mathcal{A}} Q_t^i(s', a') - Q_t^i(s, a) \right] \\ &\quad - \beta_{t,s,a} \sum_{j \in \mathcal{N}_t^i} [Q_t^i(s, a) - Q_t^j(s, a)], \end{aligned}$$

where  $\alpha_{t,s,a}, \beta_{t,s,a} > 0$  are stepsizes,  $\mathcal{N}_t^i$  denotes the set of neighboring agents of agent  $i$ , at time  $t$ . Compared to the Q-learning update (12.1),  $Q\mathcal{D}$ -learning appends an innovation term that captures the difference of Q-value estimates from its neighbors. With certain conditions on the stepsizes, the algorithm is guaranteed to converge to the optimum Q-function for the tabular setting.

Due to the scalability issue, function approximation is vital in MARL, which necessitates the development of policy-based algorithms. Our work [23] proposes actor–critic algorithms for this setting. Particularly, each agent parameterizes its own policy  $\pi_{\theta^i}^i : \mathcal{S} \rightarrow \Delta(\mathcal{A}^i)$  by some parameter  $\theta^i \in \mathbb{R}^{m^i}$ , the policy gradient of the return is first derived as

$$\nabla_{\theta^i} J(\theta) = \mathbb{E} [\nabla_{\theta^i} \log \pi_{\theta^i}^i(s, a^i) \cdot Q_\theta(s, a)], \quad (12.5)$$

where  $Q_\theta$  is the global value function corresponding to  $\bar{R}$  under the joint policy  $\pi_\theta$  that is defined as  $\pi_\theta(a | s) := \prod_{i \in \mathcal{N}} \pi_{\theta^i}^i(a^i | s)$ . As an analogy to (12.2), the policy gradient in (12.5) involves the expectation of the product between the local score function  $\nabla_{\theta^i} \log \pi_{\theta^i}^i(s, a^i)$  and the global Q-function  $Q_\theta$ . The latter, nonetheless, cannot be estimated individually at each agent. As a result, by parameterizing each local copy of  $Q_\theta(\cdot, \cdot)$  as  $Q_\theta(\cdot, \cdot; \omega^i)$  for agent  $i$ , we propose the following consensus-based TD learning for the critic step, i.e., for estimating  $Q_\theta(\cdot, \cdot)$ :

$$\tilde{\omega}_t^i = \omega_t^i + \beta_{\omega,t} \cdot \delta_t^i \cdot \nabla_\omega Q_t(\omega_t^i), \quad \omega_{t+1}^i = \sum_{j \in \mathcal{N}} c_t(i, j) \cdot \tilde{\omega}_t^j, \quad (12.6)$$

where  $\beta_{\omega,t} > 0$  is the stepsize,  $\delta_t^i$  is the local TD-error calculated using  $Q_\theta(\cdot, \cdot; \omega^i)$ . The first relation in (12.6) performs the standard TD update, followed by a weighted combination of the neighbors' estimates  $\tilde{\omega}_t^j$ . The weights here,  $c_t(i, j)$ , are dictated by the topology of the communication network, with nonzero values only if two agents  $i$  and  $j$  are connected at time  $t$ . They also need to satisfy the *doubly stochastic* property in expectation, so that  $\omega_t^i$  reaches a *consensual* value for all  $i \in \mathcal{N}$  if it converges. Then, each agent  $i$  updates its policy following stochastic policy gradient given by (12.5) in the actor step, using its own Q-function estimate  $Q_\theta(\cdot, \cdot; \omega_t^i)$ . A variant algorithm is also introduced in [23], relying on not the Q-function, but the state-value function approximation, to estimate the global advantage function.

With these in mind, almost sure convergence is established in [23] for these decentralized actor–critic algorithms, when linear functions are used for value function approximation. Similar ideas are also extended to the setting with continuous spaces [43], where deterministic policy-gradient (DPG) method is usually used. Off-policy exploration, namely, a stochastic behavior policy, is required for DPG, as the deter-

ministic on-policy may not be explorative enough. However, in the multi-agent setting, as the policies of other agents are unknown, the common off-policy approach for DPG [71, Sect. 4.2] does not apply. Inspired by the expected policy-gradient (EPG) method [190] which unifies stochastic PG (SPG) and DPG, we develop an algorithm that remains on-policy, but reduces the variance of general SPG [43]. In particular, we derive the multi-agent version of EPG, based on which we develop the actor step that can be implemented in a decentralized fashion, while the critic step still follows (12.6). Convergence of the algorithm is then also established when linear function approximation is used [43]. In the same vein, [158] considers the extension of [23] to an off-policy setting, building upon the emphatic temporal differences (ETD) method for the critic [191]. By incorporating the analysis of ETD( $\lambda$ ) [192] into [23], almost sure convergence guarantee has also been established. Another off-policy algorithm for the same setting is proposed concurrently by [193], where agents do not share their estimates of value function. Instead, the agents aim to reach consensus over the global optimal policy estimation. Provable convergence is then established for the algorithm, with a local critic and a consensus actor.

RL for decentralized networked agents has also been investigated in *multi-task*, in addition to the multi-agent, settings. In some sense, the former can be regarded as a simplified version of the latter, where each agent deals with an *independent MDP* that is not affected by other agents, while the goal is still to learn the optimal joint policy that accounts for the average reward of all agents. Pennesi and Paschalidis [194] propose a distributed actor–critic algorithm, assuming that the states, actions, and rewards are all local to each agent. Each agent performs a local TD-based critic step, followed by a consensus-based actor step that follows the gradient calculated using information exchanged from the neighbors. Gradient of the average return is then proved to converge to zero as the iteration goes to infinity. Macua et al. [155] has developed *Diff-DAC*, another distributed actor–critic algorithm for this setting, from duality theory. The updates resemble those in [23], but provide additional insights that actor–critic is actually an instance of the dual ascent method for solving a linear program.

Note that all the aforementioned convergence guarantees are *asymptotic*, i.e., the algorithms converge as the iteration numbers go to infinity, and are restricted to the case with linear function approximations. This fails to quantify the performance when finite iterations and/or samples are used, not to mention when nonlinear functions such as deep neural networks are utilized. As an initial step toward *finite-sample analyses* in this setting with more *general* function approximation, we consider in [44] the *batch* RL algorithms [195], specifically, decentralized variants of the fitted-Q iteration (FQI) [196, 197]. Note that we focus on FQI since it motivates the celebrated deep Q-learning algorithm [10] when deep neural networks are used for function approximation. We study FQI variants for both the cooperative setting with networked agents, and the competitive setting with two teams of such networked agents (see Sect. 12.4.2.1 for more details). In the former setting, all agents cooperate to iteratively update the global Q-function estimate, by fitting nonlinear least squares with the target values as the responses. In particular, let  $\mathcal{F}$  be the function class for Q-function approximation,  $\{(s_j, \{a_j^i\}_{i \in N}, s'_j)\}_{j \in [n]}$  be the batch transitions dataset

of size  $n$  available to all agents,  $\{r_j^i\}_{j \in [n]}$  be the *local* reward samples private to each agent, and  $y_j^i = r_j^i + \gamma \cdot \max_{a \in \mathcal{A}} Q_t^i(s'_j, a)$  be the local target value, where  $Q_t^i$  is agent  $i$ 's Q-function estimate at iteration  $t$ . Then, all agents cooperate to find a common Q-function estimate by solving

$$\min_{f \in \mathcal{F}} \frac{1}{N} \sum_{i \in N} \frac{1}{2n} \sum_{j=1}^n [y_j^i - f(s_j, a_j^1, \dots, a_j^N)]^2. \quad (12.7)$$

Since  $y_j^i$  is only known to agent  $i$ , the problem in (12.7) aligns with the formulation of *distributed/consensus optimization* in [185–188, 198, 199], whose global optimal solution can be achieved by the algorithms therein, if  $\mathcal{F}$  makes  $\sum_{j=1}^n [y_j^i - f(s_j, a_j^1, \dots, a_j^N)]^2$  convex for each  $i$ . This is indeed the case if  $\mathcal{F}$  is a linear function class. Nevertheless, with only a finite iteration of distributed optimization algorithms (common in practice), agents may not reach exact consensus, leading to an error of each agent's Q-function estimate away from the actual optimum of (12.7). Such an error also exists when nonlinear function approximation is used. Considering this error caused by decentralized computation, we follow the *error propagation* analysis stemming from single-agent batch RL [197, 200–203], to establish the finite-sample performance of the proposed algorithms, i.e., how the accuracy of the algorithms output depends on the function class  $\mathcal{F}$ , the number of samples within each iteration  $n$ , and the number of iterations  $t$ .

### Policy Evaluation

Aside from control, a series of algorithms in this setting focuses on the policy evaluation task only, namely, the critic step of the actor–critic algorithms. With the policy fixed, this task embraces a neater formulation, as the sampling distribution becomes stationary, and the objective becomes convex under linear function approximation. This facilitates the finite-time/sample analyses, in contrast to most control algorithms with only asymptotic guarantees. Specifically, under joint policy  $\pi$ , suppose each agent parameterizes the value function by a linear function class  $\{V_\omega(s) := \phi^\top(s)\omega : \omega \in \mathbb{R}^d\}$ , where  $\phi(s) \in \mathbb{R}^d$  is the feature vector at  $s \in \mathcal{S}$ , and  $\omega \in \mathbb{R}^d$  is the vector of parameters. For notational convenience, let  $\Phi := (\dots; \phi^\top(s); \dots) \in \mathbb{R}^{|\mathcal{S}| \times d}$ ,  $D = \text{diag}[\{\eta_\pi(s)\}_{s \in \mathcal{S}}] \in \mathbb{R}^{|\mathcal{S}| \times |\mathcal{S}|}$  be a diagonal matrix constructed using the state-occupancy measure  $\eta_\pi$ ,  $\bar{R}^\pi(s) = N^{-1} \cdot \sum_{i \in N} R^{i,\pi}(s)$ , where  $R^{i,\pi}(s) = \mathbb{E}_{a \sim \pi(\cdot|s), s' \sim P(\cdot|s,a)} [R^i(s, a, s')]$ , and  $P^\pi \in \mathbb{R}^{|\mathcal{S}| \times |\mathcal{S}|}$  with the  $(s, s')$  element being  $[P^\pi]_{s,s'} = \sum_{a \in \mathcal{A}} \pi(a|s) P(s'|s, a)$ . Then, the objective of all agents is to jointly minimize the mean square projected Bellman error (MSPBE) associated with the team-average reward, i.e.,

$$\min_{\omega} \text{MSPBE}(\omega) := \|\Pi_\Phi(V_\omega - \gamma P^\pi V_\omega - \bar{R}^\pi)\|_D^2 = \|A\omega - b\|_{C^{-1}}^2, \quad (12.8)$$

where  $\Pi_\Phi : \mathbb{R}^{|\mathcal{S}|} \rightarrow \mathbb{R}^{|\mathcal{S}|}$  defined as  $\Pi_\Phi := \Phi(\Phi^\top D \Phi)^{-1} \Phi^\top D$  is the projection operator onto subspace  $\{\Phi\omega : \omega \in \mathbb{R}^d\}$ ,  $A := \mathbb{E}[\phi(s)[\phi(s) - \gamma\phi(s')]^\top]$ ,  $C := \mathbb{E}[\phi(s)\phi^\top(s)]$ , and  $b := \mathbb{E}[\bar{R}^\pi(s)\phi(s)]$ . By replacing the expectation with samples

and using the Fenchel duality, the finite-sum version of (12.8) can be reformulated as a distributed saddle-point problem

$$\min_{\omega} \max_{\lambda^i, i \in \mathcal{N}} \quad \frac{1}{Nn} \sum_{i \in \mathcal{N}} \sum_{j=1}^n 2(\lambda^i)^\top A_j \omega - 2(b_j^i)^\top \lambda^i - (\lambda^i)^\top C_j \lambda^i, \quad (12.9)$$

where  $n$  is the data size,  $A_j$ ,  $C_j$  and  $b_j^i$  are empirical estimates of  $A$ ,  $C$  and  $b^i := \mathbb{E}[R^{i,\pi}(s)\phi(s)]$ , respectively, using sample  $j$ . Note that (12.9) is convex in  $\omega$  and concave in  $\{\lambda^i\}_{i \in \mathcal{N}}$ . The use of MSPBE as an objective is standard in multi-agent policy evaluation [95, 96, 154, 156, 157], and the idea of saddle-point reformulation has been adopted in [96, 154, 156, 204]. Note that in [204], a variant of MSPBE, named H-truncated  $\lambda$ -weighted MSPBE, is advocated, in order to control the bias of the solution deviated from the actual mean square Bellman error minimizer.

With the formulation (12.8) in mind, [156] develops a distributed variant of the gradient TD-based method, with asymptotic convergence established using the ordinary differential equation (ODE) method. In [96], a double averaging scheme that combines the dynamic consensus [205] and the SAG algorithm [206] has been proposed to solve the saddle-point problem (12.9) with linear rate. Cassano et al. [204] incorporates the idea of variance-reduction, specifically, AVRG in [207], into gradient TD-based policy evaluation. Achieving the same linear rate as [96], three advantages are claimed in [204]: (i) data-independent memory requirement; (ii) use of eligibility traces [208]; (iii) no need for synchronization in sampling. More recently, standard TD learning [58], instead of gradient TD, has been generalized to this MARL setting, with special focuses on finite-sample analyses [95, 157]. Distributed TD(0) is first studied in [95], using the proof techniques originated in [209], which requires a projection on the iterates, and the data samples to be independent and identically distributed (i.i.d.). Furthermore, motivated by the recent progress in [210], finite-time performance of the more general distributed TD( $\lambda$ ) algorithm is provided in [157], with neither projection nor i.i.d. noise assumption needed.

Policy evaluation for networked agents has also been investigated under the setting of *independent* agents interacting with *independent* MDPs. Macua et al. [154] studies off-policy evaluation based on the importance sampling technique. With no coupling among MDPs, an agent does not need to know the actions of the other agents. Diffusion-based distributed GTD is then proposed, and is shown to converge in the mean-square sense with a sublinear rate. In [211], two variants of the TD-learning, namely, GTD2 and TDC [62], have been designed for this setting, with weak convergence proved by the general stochastic approximation theory in [212], when agents are connected by a time-varying communication network. Note that [204] also considers the independent MDP setting, with the same results established as the actual MARL setting.

### Other Learning Goals

Several other learning goals have also been explored for decentralized MARL with networked agents. Zhang et al. [213] has considered the *optimal consensus* problem,

where each agent over the network tracks the states of its neighbors' as well as a leader's, so that the consensus error is minimized by the joint policy. A policy iteration algorithm is then introduced, followed by a practical actor–critic algorithm using neural networks for function approximation. A similar consensus error objective is also adopted in [214], under the name of *cooperative multi-agent graphical games*. A centralized-critic-decentralized-actor scheme is utilized for developing off-policy RL algorithms.

Communication efficiency, as a key ingredient in the algorithm design for this setting, has drawn increasing attention recently [130–132]. Specifically, [130] develops lazily aggregated policy gradient (LAPG), a distributed PG algorithm that can reduce the communication rounds between the agents and a central controller, by judiciously designing communication trigger rules. Ren and Haupt [132] addresses the same policy evaluation problem as [96], and develops a hierarchical distributed algorithm by proposing a mixing matrix different from the doubly stochastic one used in [23, 96, 156], which allows unidirectional information exchange among agents to save communication. In contrast, the distributed actor–critic algorithm in [131] reduces the communication by transmitting only one scaled entry of its state vector at each iteration, while preserving provable convergence as in [23].

#### 12.4.1.3 Partially Observed Model

We complete the overview for cooperative settings by briefly introducing a class of significant but challenging models where agents are faced with partial observability. Though common in practice, theoretical analysis of MARL algorithms in this setting is still relatively scarce, in contrast to the aforementioned fully observed settings. In general, this setting can be modeled by a decentralized POMDP (Dec-POMDP) [36], which shares almost all elements such as the reward function and the transition model, as the MMDP/Markov team model in Sect. 12.2.2.1, except that each agent now only has its local observations of the system state  $s$ . With no accessibility to other agents' observations, an individual agent cannot maintain a global belief state, the sufficient statistic for decision-making in single-agent POMDPs. Hence, Dec-POMDPs have been known to be NEXP-hard [104], requiring super-exponential time to solve in the worst case.

There is an increasing interest in developing planning/learning algorithms for Dec-POMDPs. Most of the algorithms are based on the *centralized-learning-decentralized-execution* scheme. In particular, the decentralized problem is first reformulated as a centralized one, which can be solved at a central controller with (a simulator that generates) the observation data of all agents. The policies are then optimized/learned using data, and distributed to all agents for execution. Finite-state controllers (FSCs) are commonly used to represent the local policy at each agent [215, 216], which map local observation histories to actions. A Bayesian nonparametric approach is proposed in [217] to determine the controller size of variable-size FSCs. To efficiently solve the centralized problem, a series of *top-down* algorithms have been proposed. In [150], the Dec-POMDP is converted to *non-observable MDP*

(NOMDP), a kind of centralized sequential decision-making problem, which is then addressed by some heuristic tree search algorithms. As an extension of the NOMDP conversion, [152, 218] convert Dec-POMDPs to *occupancy-state* MDPs (oMDPs), where the occupancy-states are distributions over hidden states and joint histories of observations. As the value functions of oMDPs enjoy the piece-wise linearity and convexity, both tractable planning [218] and value-based learning [152] algorithms have been developed.

To further improve computational efficiency, several sampling-based planning/learning algorithms have also been proposed. In particular, Monte Carlo sampling with policy iteration and the expectation-maximization algorithm, are proposed in [219, 220], respectively. Furthermore, Monte Carlo tree search has been applied to special classes of Dec-POMDPs, such as multi-agent POMDPs [121] and multi-robot active perception [221]. In addition, policy-gradient-based algorithms can also be developed for this centralized learning scheme [152], with a centralized critic and a decentralized actor. Finite-sample analysis can also be established under this scheme [222, 223], for tabular settings with finite state-action spaces.

Several attempts have also been made to enable *decentralized learning* in Dec-POMDPs. When the agents share some common information/observations, [224] proposes to reformulate the problem as a centralized POMDP, with the common information being the observations of a *virtual* central controller. This way, the centralized POMDP can be solved individually by each agent. In [225], the reformulated POMDP has been approximated by finite-state MDPs with exponentially decreasing approximation error, which are then solved by Q-learning. Very recently, [39] has developed a tree search based algorithm for solving this centralized POMDP, which, interestingly, echoes back the heuristics for solving Dec-POMDPs directly as in [121, 221], but with a more solid theoretical footing. Note that in both [39, 225], a common random number generator is used for all agents, in order to avoid communication among agents and enable a decentralized learning scheme.

### 12.4.2 Competitive Setting

Competitive settings are usually modeled as *zero-sum* games. Computationally, there exists a great barrier between solving two-player and multi-player zero-sum games. In particular, even the simplest three-player matrix games are known to be PPAD-complete [226, 227]. Thus, most existing results on competitive MARL focus on two-player zero-sum games, with  $N = \{1, 2\}$  and  $R^1 + R^2 = 0$  in Definitions 12.2 and 12.4. In the rest of this section, we review methods that provably find a Nash (equivalently, saddle-point) equilibrium in two-player Markov or extensive-form games. The existing algorithms can mainly be categorized into two classes: value-based and policy-based approaches, which are introduced separately in the sequel.

### 12.4.2.1 Value-Based Methods

Similar as in single-agent MDPs, value-based methods aim to find an optimal value function from which the joint Nash equilibrium policy can be extracted. Moreover, the optimal value function is known to be the unique fixed point of a Bellman operator, which can be obtained via dynamic programming type methods.

Specifically, for simultaneous-move Markov games, the value function defined in (12.3) satisfies  $V_{\pi^1, \pi^2}^1 = -V_{\pi^1, \pi^2}^2$ , and thus any Nash equilibrium  $\pi^* = (\pi^{1,*}, \pi^{2,*})$  satisfies

$$V_{\pi^1, \pi^{1,*}}^1(s) \leq V_{\pi^{1,*}, \pi^{2,*}}^1(s) \leq V_{\pi^{1,*}, \pi^2}^1(s), \quad \text{for any } \pi = (\pi^1, \pi^2) \text{ and } s \in \mathcal{S}. \quad (12.10)$$

Similar as the minimax theorem in normal-form/matrix zero-sum games [228], for two-player zero-sum Markov games with finite state and action spaces, one can define the optimal value function  $V^*: \mathcal{S} \rightarrow \mathbb{R}$  as

$$V^* = \max_{\pi^1} \min_{\pi^2} V_{\pi^1, \pi^2}^1 = \min_{\pi^2} \max_{\pi^1} V_{\pi^1, \pi^2}^1. \quad (12.11)$$

Then (12.10) implies that  $V_{\pi^{1,*}, \pi^{2,*}}^1$  coincides with  $V^*$  and any pair of policies  $\pi^1$  and  $\pi^2$  that attains the supremum and infimum in (12.11) constitutes a Nash equilibrium. Moreover, similar to MDPs, [82] shows that  $V^*$  is the unique solution of a Bellman equation and a Nash equilibrium can be constructed based on  $V^*$ . Specifically, for any  $V: \mathcal{S} \rightarrow \mathbb{R}$  and any  $s \in \mathcal{S}$ , we define

$$Q_V(s, a^1, a^2) = \mathbb{E}_{s' \sim \mathcal{P}(\cdot | s, a^1, a^2)} [R^1(s, a^1, a^2, s') + \gamma \cdot V(s')], \quad (12.12)$$

where  $Q_V(s, \cdot, \cdot)$  can be regarded as a matrix in  $\mathbb{R}^{|\mathcal{A}^1| \times |\mathcal{A}^2|}$ . Then we define the Bellman operator  $\mathcal{T}^*$  by solving a matrix zero-sum game regarding  $Q_V(s, \cdot, \cdot)$  as the payoff matrix, i.e., for any  $s \in \mathcal{S}$ , one can define

$$(\mathcal{T}^*V)(s) = \text{Value}[Q_V(s, \cdot, \cdot)] = \max_{u \in \Delta(\mathcal{A}^1)} \min_{v \in \Delta(\mathcal{A}^2)} \sum_{a \in \mathcal{A}^1} \sum_{b \in \mathcal{A}^2} u_a \cdot v_b \cdot Q_V(s, a, b), \quad (12.13)$$

where we use  $\text{Value}(\cdot)$  to denote the optimal value of a matrix zero-sum game, which can be obtained by solving a linear program [229]. Thus, the Bellman operator  $\mathcal{T}^*$  is  $\gamma$ -contractive in the  $\ell_\infty$ -norm and  $V^*$  in (12.11) is the unique solution to the Bellman equation  $V = \mathcal{T}^*V$ . Moreover, letting  $p_1(V)$ ,  $p_2(V)$  be any solution to the optimization problem in (12.13), we have that  $\pi^* = (p_1(V^*), p_2(V^*))$  is a Nash equilibrium specified by Definition 12.3. Thus, based on the Bellman operator  $\mathcal{T}^*$ , [82] proposes the value iteration algorithm, which creates a sequence of value functions  $\{V_t\}_{t \geq 1}$  satisfying  $V_{t+1} = \mathcal{T}^*V_t$ , that converges to  $V^*$  with a linear rate. Specifically, we have

$$\|V_{t+1} - V^*\|_\infty = \|\mathcal{T}^*V_t - \mathcal{T}^*V^*\|_\infty \leq \gamma \cdot \|V_t - V^*\|_\infty \leq \gamma^{t+1} \cdot \|V_0 - V^*\|_\infty.$$

In addition, a value iteration update can be decomposed into the two steps. In particular, letting  $\pi^1$  be any policy of player 1 and  $V$  be any value function, we define Bellman operator  $\mathcal{T}^{\pi^1}$  by

$$(\mathcal{T}^{\pi^1} V)(s) = \min_{v \in \Delta(\mathcal{A}^2)} \sum_{a \in \mathcal{A}^1} \sum_{b \in \mathcal{A}^2} \pi^1(a | s) \cdot v_b \cdot Q_V(s, a, b), \quad (12.14)$$

where  $Q_V$  is defined in (12.12). Then we can equivalently write a value iteration update as

$$\mu_{t+1} = p_1(V_t) \quad \text{and} \quad V_{t+1} = \mathcal{T}^{\mu_{t+1}} V_t. \quad (12.15)$$

Such a decomposition motivates the policy iteration algorithm for two-player zero-sum games, which has been studied in, e.g., [230–234], for different variants of such Markov games. In particular, from the perspective of player 1, policy iteration creates a sequence  $\{\mu_t, V_t\}_{t \geq 0}$  satisfying

$$\mu_{t+1} = p_1(V_t) \quad \text{and} \quad V_{t+1} = (\mathcal{T}^{\mu_{t+1}})^\infty V_t, \quad (12.16)$$

i.e.,  $V_{t+1}$  is the fixed point of  $\mathcal{T}^{\mu_{t+1}}$ . The updates for player 2 can be similarly constructed. By the definition in (12.14), the Bellman operator  $\mathcal{T}^{\mu_{t+1}}$  is  $\gamma$ -contractive and its fixed point corresponds to the value function associated with  $(\mu_{t+1}, \text{Br}(\mu_{t+1}))$ , where  $\text{Br}(\mu_{t+1})$  is the best response policy of player 2 when player 1 adopts  $\mu_{t+1}$ . Hence, in each step of policy iteration, the player first finds an improved policy  $\mu_{t+1}$  based on the current function  $V_t$ , and then obtains a conservative value function by assuming that the opponent plays the best counter policy  $\text{Br}(\mu_{t+1})$ . It has been shown in [234] that the value function sequence  $\{V_t\}_{t \geq 0}$  monotonically increases to  $V^*$  with a linear rate of convergence for turn-based zero-sum Markov games.

Notice that both the value and policy iteration algorithms are model based due to the need of computing the Bellman operator  $\mathcal{T}^{\mu_{t+1}}$  in (12.15) and (12.16). By estimating the Bellman operator via data-driven approximation, [83] has proposed minimax-Q learning, which extends the well-known Q-learning algorithm [48] for MDPs to zero-sum Markov games. In particular, minimax-Q learning is an online, off-policy, and tabular method which updates the action-value function  $Q: \mathcal{S} \times \mathcal{A} \rightarrow \mathbb{R}$  based on transition data  $\{(s_t, a_t, r_t, s'_t)\}_{t \geq 0}$ , where  $s'_t$  is the next state following  $(s_t, a_t)$  and  $r_t$  is the reward. In the  $t$ th iteration, it only updates the value of  $Q(s_t, a_t)$  and keeps other entries of  $Q$  unchanged. Specifically, we have

$$Q(s_t, a_t^1, a_t^2) \leftarrow (1 - \alpha_t) \cdot Q(s_t, a_t^1, a_t^2) + \alpha_t \cdot \{r_t + \gamma \cdot \text{Value}[Q(s'_t, \cdot, \cdot)]\}, \quad (12.17)$$

where  $\alpha_t \in (0, 1)$  is the stepsize. As shown in [49], under conditions similar to those for single-agent Q-learning [48], function  $Q$  generated by (12.17) converges to the

optimal action-value function  $Q^* = Q_{V^*}$  defined by combining (12.11) and (12.12). Moreover, with a slight abuse of notation, if we define the Bellman operator  $\mathcal{T}^*$  for action-value functions by

$$(\mathcal{T}^*Q)(s, a^1, a^2) = \mathbb{E}_{s' \sim \mathcal{P}(\cdot | s, a^1, a^2)} \{ R^1(s, a^1, a^2, s') + \gamma \cdot \text{Value}[Q(s', \cdot, \cdot)] \}, \quad (12.18)$$

then we have  $Q^*$  as the unique fixed point of  $\mathcal{T}^*$ . Since the target value  $r_t + \gamma \cdot \text{Value}[Q(s'_t, \cdot, \cdot)]$  in (12.17) is an estimator of  $(\mathcal{T}^*Q)(s_t, a_t^1, a_t^2)$ , minimax-Q learning can be viewed as a stochastic approximation algorithm for computing the fixed point of  $\mathcal{T}^*$ . Following [83], minimax-Q learning has been further extended to the function approximation setting where  $Q$  in (12.17) is approximated by a class of parametrized functions. However, convergence guarantees for this minimax-Q learning with even linear function approximation have not been well understood. Such a linear value function approximation also applies to a significant class of zero-sum MG instances with continuous state-action spaces, i.e., linear quadratic (LQ) zero-sum games [99, 238, 239], where the reward function is quadratic with respect to the states and actions, while the transition model follows linear dynamics. In this setting, Q-learning based algorithm can be guaranteed to converge to the NE [239].

To embrace general function classes, the framework of batch RL [197, 200–202, 235, 240] can be adapted to the multi-agent settings, as in the recent works [44, 241]. As mentioned in Sect. 12.4.1.2 for cooperative batch MARL, each agent iteratively updates the Q-function by fitting least-squares using the target values. Specifically, let  $\mathcal{F}$  be the function class of interest and let  $\{(s_i, a_i^1, a_i^2, r_i, s'_i)\}_{i \in [n]}$  be the dataset. For any  $t \geq 0$ , let  $Q_t$  be the current iterate in the  $t$ th iteration, and define  $y_i = r_i + \gamma \cdot \text{Value}[Q_t(s'_i, \cdot, \cdot)]$  for all  $i \in [n]$ . Then we update  $Q_t$  by solving a least-squares regression problem in  $\mathcal{F}$ , that is,

$$Q_{t+1} \leftarrow \underset{f \in \mathcal{F}}{\operatorname{argmin}} \frac{1}{2n} \sum_{i=1}^n [y_i - f(s_i, a_i^1, a_i^2)]^2. \quad (12.19)$$

In such a two-player zero-sum Markov game setting, a finite-sample error bound on the Q-function estimate is established in [241].

Regarding other finite-sample analyses, very recently, [242] has studied zero-sum turn-based stochastic games (TBSG), a simplified zero-sum MG when the transition model is embedded in some feature space and a generative model is available. Two Q-learning based algorithms have been proposed and analyzed for this setting. Sidford et al. [35] have proposed algorithms that achieve near-optimal sample complexity for general zero-sum TBSGs with a generative model, by extending the previous near-optimal Q-learning algorithm for MDPs [243]. In the online setting, where the learner controls only one of the players that plays against an arbitrary opponent, [244] has proposed UCSG, an algorithm for the *average reward* zero-sum MGs, using the principle of *optimism in the face of uncertainty* [245, 246]. UCSG is shown to achieve a sublinear regret compared to the game value when competing

with an arbitrary opponent, and also achieve  $\tilde{O}(\text{poly}(1/\epsilon))$  sample complexity if the opponent plays an optimistic best response.

Furthermore, when it comes to zero-sum games with imperfect information, [247–250] have proposed to transform extensive-form games into normal-form games using the *sequence-form* representation, which enables equilibrium finding via linear programming. In addition, by lifting the state space to the space of belief states, [119, 251–254] have applied dynamic programming methods to zero-sum stochastic games. Both of these approaches guarantee finding of a Nash equilibrium but are only efficient for small-scale problems. Finally, MCTS with UCB-type action selection rule [51–53] can also be applied to two-player turn-based games with incomplete information [52, 255–258], which lays the foundation for the recent success of deep RL for the game of Go [1]. Moreover, these methods are shown to converge to the minimax solution of the game, thus can be viewed as a counterpart of minimax-Q learning with Monte Carlo sampling.

#### 12.4.2.2 Policy-Based Methods

Policy-based reinforcement-learning methods introduced in Sect. 12.2.1.2 can also be extended to the multi-agent setting. Instead of finding the fixed point of the Bellman operator, a fair amount of methods only focus on a single agent and aim to maximize the expected return of that agent, disregarding the other agents' policies. Specifically, from the perspective of a single agent, the environment is time-varying as the other agents also adjust their policies. Policy-based methods aim to achieve the optimal performance when other agents play arbitrarily by minimizing the (*external*) *regret*, that is, find a sequence of actions that perform nearly as well as the optimal fixed policy in hindsight. An algorithm with negligible average overall regret is called *no-regret* or *Hannan consistent* [259]. Any Hannan consistent algorithm is known to have the following two desired properties in repeated normal-form games. First, when other agents adopt stationary policies, the time-average policy constructed by the algorithm converges to the best response policy (against the ones used by the other agents). Second, more interestingly, in two-player zero-sum games, when both players adopt Hannan consistent algorithms and both their average overall regrets are no more than  $\epsilon$ , their time-average policies constitute a  $2\epsilon$ -approximate Nash equilibrium [126]. Thus, any Hannan consistent single-agent reinforcement-learning algorithm can be applied to find the Nash equilibria of two-player zero-sum games via *self-play*. Most of these methods belong to one of the following two families: fictitious play [260, 261], and counterfactual regret minimization [109], which will be summarized below.

Fictitious play is a classic algorithm studied in game theory, where the players play the game repeatedly and each player adopts a policy that best responds to the average policy of the other agents. This method was originally proposed for solving *normal-form games*, which are a simplification of the Markov games defined in Definition 12.2 with  $S$  being a singleton and  $\gamma = 0$ . In particular, for any joint policy  $\pi \in \Delta(\mathcal{A})$  of the  $N$  agents, we let  $\pi^{-i}$  be the marginal policy of all players except

player  $i$ . For any  $t \geq 1$ , suppose the agents have played  $\{a_\tau : .1 \leq \tau \leq t\}$  in the first  $t$  stages. We define  $x_t$  as the empirical distribution of  $\{a_\tau : .1 \leq \tau \leq t\}$ , i.e.,  $x_t(a) = t^{-1} \sum_{\tau=1}^t \mathbb{1}\{a_\tau = a\}$  for any  $a \in \mathcal{A}$ . Then, in the  $t$ -th stage, each agent  $i$  takes action  $a_t^i \in \mathcal{A}^i$  according to the best response policy against  $x_t^{-i}$ . In other words, each agent plays the best counter policy against the policy of the other agents inferred from history data. Here, for any  $\epsilon > 0$  and any  $\pi \in \Delta(\mathcal{A})$ , we denote by  $\text{Br}_\epsilon(\pi^{-i})$  the  $\epsilon$ -best response policy of player  $i$ , which satisfies

$$R^i(\text{Br}_\epsilon(\pi^{-i}), \pi^{-i}) \geq \sup_{\mu \in \Delta(\mathcal{A}^i)} R^i(\mu, \pi^{-i}) - \epsilon. \quad (12.20)$$

Moreover, we define  $\text{Br}_\epsilon(\pi)$  as the joint policy  $(\text{Br}_\epsilon(\pi^{-1}), \dots, \text{Br}_\epsilon(\pi^{-N})) \in \Delta(\mathcal{A})$  and suppress the subscript  $\epsilon$  in  $\text{Br}_\epsilon$  if  $\epsilon = 0$ . By this notation, regarding each  $a \in \mathcal{A}$  as a vertex of  $\Delta(\mathcal{A})$ , we can equivalently write the fictitious process as

$$x_t - x_{t-1} = (1/t) \cdot (a_t - x_{t-1}), \quad \text{where} \quad a_t \sim \text{Br}(x_{t-1}). \quad (12.21)$$

As  $t \rightarrow \infty$ , the updates in (12.21) can be approximately characterized by a differential inclusion [262]

$$\frac{dx(t)}{dt} \in \text{Br}(x(t)) - x(t), \quad (12.22)$$

which is known as the continuous-time fictitious play. Although it is well known that the discrete-time fictitious play in (12.21) is not Hannan consistent [160, 263], it is shown in [264, 265] that the continuous-time fictitious play in (12.22) is Hannan consistent. Moreover, using tools from stochastic approximation [263, 266], various modifications of discrete-time fictitious play based on techniques such as smoothing or stochastic perturbations have been shown to converge to the continuous-time fictitious play and are thus Hannan consistent [267–271]. As a result, applying these methods with self-play provably finds a Nash equilibrium of a two-player zero-sum normal-form game.

Furthermore, fictitious play methods have also been extended to RL settings without the model knowledge. Specifically, using sequence-form representation, [110] has proposed the first fictitious play algorithm for extensive-form games which is realization-equivalent to the *generalized weakened fictitious play* [269] for normal-form games. The pivotal insight is that a convex combination of normal-form policies can be written as a weighted convex combination of behavioral policies using realization probabilities. Specifically, recall that the set of information states of agent  $i$  was denoted by  $\mathcal{S}^i$ . When the game has perfect recall, each  $s^i \in \mathcal{S}^i$  uniquely defines a sequence  $\sigma_{s^i}$  of actions played by agent  $i$  for reaching state  $s^i$ . Then any behavioral policy  $\pi^i$  of agent  $i$  induces a *realization probability*  $\text{Rp}(\pi^i; \cdot)$  for each sequence  $\sigma$  of agent  $i$ , which is defined by  $\text{Rp}(\pi^i; \sigma) = \prod_{(s', a) \in \sigma} \pi^i(a | s')$ , where the product is taken over all  $s' \in \mathcal{S}^i$  and  $a \in \mathcal{A}^i$  such that  $(\sigma_{s'}, a)$  is a subsequence of  $\sigma$ . Using

the notation of realization probability, for any two behavioral policies  $\pi$  and  $\tilde{\pi}$  of agent  $i$ , the sum

$$\frac{\lambda \cdot \text{Rp}(\pi, \sigma_{s^i}) \cdot \pi(\cdot | s^i)}{\lambda \cdot \text{Rp}(\pi, \sigma_{s^i}) + (1 - \lambda) \cdot \text{Rp}(\tilde{\pi}, \sigma_{s^i})} + \frac{(1 - \lambda) \cdot \text{Rp}(\tilde{\pi}, \sigma_{s^i}) \cdot \tilde{\pi}(\cdot | s^i)}{\lambda \cdot \text{Rp}(\pi, \sigma_{s^i}) + (1 - \lambda) \cdot \text{Rp}(\tilde{\pi}, \sigma_{s^i})}, \quad \forall s^i \in \mathcal{S}^i, \quad (12.23)$$

is the mixture policy of  $\pi$  and  $\tilde{\pi}$  with weights  $\lambda \in (0, 1)$  and  $1 - \lambda$ , respectively. Then, combining (12.20) and (12.23), the fictitious play algorithm in [110] computes a sequence of policies  $\{\pi_t\}_{t \geq 1}$ . In particular, in the  $t$ -th iteration, any agent  $i$  first computes the  $\epsilon_{t+1}$ -best response policy  $\tilde{\pi}_{t+1}^i \in \text{Br}_{\epsilon_{t+1}}(\pi_t^{-i})$  and then constructs  $\pi_{t+1}^i$  as the mixture policy of  $\pi_t^i$  and  $\tilde{\pi}_{t+1}^i$  with weights  $1 - \alpha_{t+1}$  and  $\alpha_{t+1}$ , respectively. Here, both  $\epsilon_t$  and  $\alpha_t$  are taken to converge to zero as  $t$  goes to infinity, and we further have  $\sum_{t \geq 1} \alpha_t = \infty$ . We note, however, that although such a method provably converges to a Nash equilibrium of a zero-sum game via self-play, it suffers from the curse of dimensionality due to the need to iterate all states of the game in each iteration. For computational efficiency, [110] has also proposed a data-driven fictitious self-play framework where the best response is computed via fitted Q-iteration [200, 272] for the single-agent RL problem, with the policy mixture being learned through supervised learning. This framework was later adopted by [25, 30, 31, 273] to incorporate other single RL methods such as deep Q-network [10] and Monte Carlo tree search [52, 53, 274]. Moreover, in a more recent work, [162] has proposed a smooth fictitious play algorithm [267] for zero-sum multi-stage games with simultaneous moves. Their algorithm combines the actor-critic framework [69] with fictitious self-play, and infers the opponent's policy implicitly via policy evaluation. Specifically, when the two players adopt a joint policy  $\pi = (\pi^1, \pi^2)$ , from the perspective of player 1, it infers  $\pi^2$  implicitly by estimating  $\bar{Q}_{\pi^1, \pi^2}$  via temporal-difference learning [237], where  $\bar{Q}_{\pi^1, \pi^2}: \mathcal{S} \times \mathcal{A}^1 \rightarrow \mathbb{R}$  is defined as

$$\bar{Q}_{\pi^1, \pi^2}(s, a^1) = \mathbb{E} \left[ \sum_{t \geq 0} \gamma^t R^1(s_t, a_t, s_{t+1}) \mid s_0 = s, a_0^1 = a^1, a_0^2 \sim \pi^2(\cdot | s), a_t \sim \pi(\cdot | s_t), \forall t \geq 1 \right],$$

which is the action-value function of player 1 marginalized by  $\pi^2$ . Besides, the best response policy is obtained by taking the soft-greedy policy with respect to  $\bar{Q}_{\pi^1, \pi^2}$ , i.e.,

$$\pi^1(a^1 | s) \leftarrow \frac{\exp[\eta^{-1} \cdot \bar{Q}_{\pi^1, \pi^2}(s, a^1)]}{\sum_{a^1 \in \mathcal{A}^1} \exp[\eta^{-1} \cdot \bar{Q}_{\pi^1, \pi^2}(s, a^1)]}, \quad (12.24)$$

where  $\eta > 0$  is the smoothing parameter. Finally, the algorithm is obtained by performing both policy evaluation and policy update in (12.24) simultaneously using two-timescale updates [266, 275], which ensure that the policy updates, when using self-play, can be characterized by an ordinary differential equation whose asymptotically stable solution is a smooth Nash equilibrium of the game.

Another family of popular policy-based methods is based on the idea of *counterfactual regret minimization* (CFR), first proposed in [109], which has been a breakthrough in the effort to solve large-scale extensive-form games. Moreover, from a theoretical perspective, compared with fictitious play algorithms whose convergence is analyzed asymptotically via stochastic approximation, explicit regret bounds can be established using tools from online learning [276], which yield rates of convergence to the Nash equilibrium. Specifically, when  $N$  agents play the extensive-form game for  $T$  rounds with  $\{\pi_t : 1 \leq t \leq T\}$ , the *regret* of player  $i$  is defined as

$$\text{Reg}_T^i = \max_{\pi^i} \sum_{t=1}^T [R^i(\pi^i, \pi_t^{-i}) - R^i(\pi_t^i, \pi_t^{-i})], \quad (12.25)$$

where the maximum is taken over all possible policies of player  $i$ . In the following, before we define the notion of counterfactual regret, we first introduce a few notations. Recall that we had defined the reach probability  $\eta_\pi(h)$  in (12.4), which can be factorized into the product of each agent's contribution. That is, for each  $i \in \mathcal{U} \cup \{c\}$ , we can group the probability terms involving  $\pi^i$  into  $\eta_\pi^i(h)$  and write  $\eta_\pi(h) = \prod_{i \in \mathcal{N} \cup \{c\}} \eta_\pi^i(h) = \eta_\pi^i(h) \cdot \eta_\pi^{-i}(h)$ . Moreover, for any two histories  $h, h' \in \mathcal{H}$  satisfying  $h \sqsubseteq h'$ , we define the *conditional reach probability*  $\eta_\pi(h' | h)$  as  $\eta_\pi(h')/\eta_\pi(h)$  and define  $\eta_\pi^i(h' | h)$  similarly. For any  $s \in \mathcal{S}^i$  and any  $a \in \mathcal{A}(s)$ , we define  $\mathcal{Z}(s, a) = \{(h, z) \in \mathcal{H} \times \mathcal{Z} | h \in s, ha \sqsubseteq z\}$ , which contains all possible pairs of history in information state  $s$  and terminal history after taking action  $a$  at  $s$ . Then, the *counterfactual value function* is defined as

$$Q_{\text{CF}}^i(\pi, s, a) = \sum_{(h, z) \in \mathcal{Z}(s, a)} \eta_\pi^{-i}(h) \cdot \eta_\pi(z | ha) \cdot R^i(z), \quad (12.26)$$

which is the expected utility obtained by agent  $i$  given that it has played to reach state  $s$ . We also define  $V_{\text{CF}}^i(\pi, s) = \sum_{a \in \mathcal{A}(s)} Q_{\text{CF}}^i(\pi, s, a) \cdot \pi^i(a | s)$ . Then the difference between  $Q_{\text{CF}}^i(\pi, s, a)$  and  $V_{\text{CF}}^i(\pi, s)$  can be viewed as the value of action  $a$  at information state  $s \in \mathcal{S}^i$ , and *counterfactual regret* of agent  $i$  at state  $s$  is defined as

$$\text{Reg}_T^i(s) = \max_{a \in \mathcal{A}(s)} \sum_{i=1}^T [Q_{\text{CF}}^i(\pi_t, s, a) - V_{\text{CF}}^i(\pi_t, s)], \quad \forall s \in \mathcal{S}^i. \quad (12.27)$$

Moreover, as shown in Theorem 3 of [109], counterfactual regrets defined in (12.27) provide an upper bound for the total regret in (12.25):

$$\text{Reg}_T^i \leq \sum_{s \in \mathcal{S}^i} \text{Reg}_T^{i,+}(s), \quad (12.28)$$

where we let  $x^+$  denote  $\max\{x, 0\}$  for any  $x \in \mathbb{R}$ . This bound lays the foundation of *counterfactual regret minimization* algorithms. Specifically, to minimize the total regret in (12.25), it suffices to minimize the counterfactual regret for each information state, which can be obtained by any online learning algorithm, such as EXP3 [277], Hedge [278–280], and regret matching [281]. All these methods ensure that the counterfactual regret is  $O(\sqrt{T})$  for all  $s \in \mathcal{S}^i$ , which leads to an  $O(\sqrt{T})$  upper bound of the total regret. Thus, applying CFR-type methods with self-play to a zero-sum two-play extensive-form game, the average policy is an  $O(\sqrt{1/T})$ -approximate Nash equilibrium after  $T$  steps. In particular, the vanilla CFR algorithm updates the policies via regret matching, which yields that  $\text{Reg}_T^i(s) \leq R_{\max}^i \cdot \sqrt{A^i \cdot T}$  for all  $s \in \mathcal{S}^i$ , where we have introduced

$$R_{\max}^i = \max_{z \in \mathcal{Z}} R^i(z) - \min_{z \in \mathcal{Z}} R^i(z), \quad A_i = \max_{h: \tau(h)=i} |\mathcal{A}(h)|.$$

Thus, by (12.28), the total regret of agent  $i$  is bounded by  $R_{\max}^i \cdot |\mathcal{S}^i| \cdot \sqrt{A^i \cdot T}$ .

One drawback of vanilla CFR is that the entire game tree needs to be traversed in each iteration, which can be computationally prohibitive. A number of CFR variants have been proposed since the pioneering work [109] for improving computational efficiency. For example, [282–287] combine CFR with Monte Carlo sampling; [288–290] propose to estimate the counterfactual value functions via regression; [291–293] improve the computational efficiency by pruning sub-optimal paths in the game tree; [294–296] analyze the performance of a modification named CFR<sup>+</sup>, and [297] proposes lazy updates with a near-optimal regret upper bound.

Furthermore, it has been shown recently in [111] that CFR is closely related to policy-gradient methods. To see this, for any joint policy  $\pi$  and any  $i \in \mathcal{N}$ , we define the action-value function of agent  $i$ , denoted by  $Q_\pi^i$ , as

$$Q_\pi^i(s, a) = \frac{1}{\eta_\pi(s)} \cdot \sum_{(h, z) \in \mathcal{Z}(s, a)} \eta_\pi(h) \cdot \eta_\pi(z | ha) \cdot R^i(z), \quad \forall s \in \mathcal{S}^i, \forall a \in \mathcal{A}(s). \quad (12.29)$$

That is,  $Q_\pi^i(s, a)$  is the expected utility of agent  $i$  when the agents follow policy  $\pi$  and agent  $i$  takes action  $a$  at information state  $s \in \mathcal{S}^i$ , conditioning on  $s$  being reached. It has been shown in [111] that the  $Q_{\text{CF}}^i$  in (12.26) is connected with  $Q_\pi^i$  in (12.29) via  $Q_{\text{CF}}^i(\pi, s, a) = Q_\pi^i(s, a) \cdot [\sum_{h \in s} \eta_\pi^{-i}(h)]$ . Moreover, in the tabular setting where we regard the joint policy  $\pi$  as a table  $\{\pi^i(a | s) : s \in \mathcal{S}^i, a \in \mathcal{A}(s)\}$ , for any  $s \in \mathcal{S}^i$  and any  $a \in \mathcal{A}(s)$ , the policy-gradient of  $R^i(\pi)$  can be written as

$$\frac{\partial R^i(\pi)}{\partial \pi^i(a | s)} = \eta_\pi(s) \cdot Q_\pi^i(s, a) = \eta_\pi^i(s) \cdot Q_{\text{CF}}^i(\pi, s, a), \quad \forall s \in \mathcal{S}^i, \forall a \in \mathcal{A}(s).$$

As a result, the advantage actor–critic (A2C) algorithm [69] is equivalent to a particular CFR algorithm, where the policy update rule is specified by the *generalized*

*infinitesimal gradient ascent* algorithm [298]. Thus, [111] proves that the regret of the tabular A2C algorithm is bounded by  $|\mathcal{S}^i| \cdot [1 + A^i \cdot (R_{\max}^i)^2] \cdot \sqrt{T}$ . Following this work, [112] shows that A2C where the policy is tabular and is parametrized by a softmax function is equivalent to CFR that uses Hedge to update the policy. Moreover, [299] proposes a policy optimization method known as *exploitability descent*, where the policy is updated using actor–critic, assuming the opponent plays the best counter-policy. This method is equivalent to the CFR-BR algorithm [300] with Hedge. Thus, [111, 112, 299] show that actor–critic and policy-gradient methods for MARL can be formulated as CFR methods and thus convergence to a Nash equilibrium of a zero-sum extensive-form game is guaranteed.

In addition, besides fictitious play and CFR methods introduced above, multiple policy optimization methods have been proposed for special classes of two-player zero-sum stochastic games or extensive-form games. For example, Monte Carlo tree search methods have been proposed for perfect-information extensive games with simultaneous moves. It has been shown in [301] that the MCTS methods with UCB-type action selection rules, introduced in Sect. 12.4.2.1, fail to converge to a Nash equilibrium in simultaneous-move games, as UCB does not take into consideration the possibly adversarial moves of the opponent. To remedy this issue, [302–305] have proposed to adopt stochastic policies and using Hannan consistent methods such as EXP3 [277] and regret matching [281] to update the policies. With self-play, [303] shows that the average policy obtained by MCTS with any  $\epsilon$ -Hannan consistent policy update method converges to an  $O(D^2 \cdot \epsilon)$ -Nash equilibrium, where  $D$  is the maximal depth.

Finally, there are surging interests in investigating policy-gradient-based methods in *continuous* games, i.e., the games with continuous state-action spaces. With policy parameterization, finding the NE of zero-sum Markov games becomes a nonconvex–nonconcave saddle-point problem in general [32, 34, 306, 307]. This hardness is inherent, even in the simplest linear quadratic setting with linear function approximation [34, 307]. As a consequence, most of the convergence results are *local* [32, 33, 306, 308–312], in the sense that they address the convergence behavior around local NE points. Still, it has been shown that the vanilla gradient-descent-ascent (GDA) update, which is equivalent to the policy-gradient update in MARL, fails to converge to local NEs, for either the non-convergent behaviors such as limit cycling [308, 310, 311, 313], or the existence of non-Nash stable limit points for the GDA dynamics [306, 309]. Consensus optimization [308], symplectic gradient adjustment [313], and extragradient method [311] have been advocated to mitigate the oscillatory behaviors around the equilibria; while [306, 309] exploit the curvature information so that all the stable limit points of the proposed updates are local NEs. Going beyond Nash equilibria, [33, 312] consider gradient-based learning for *Stackelberg equilibria*, which correspond to only the one-sided equilibrium solution in zero-sum games, i.e., either minimax or maximin, as the order of which player acts first is vital in nonconvex–nonconcave problems. Jin et al. [33] introduces the concept of *local minimax* point as the solution, and shows that GDA converges to local minimax points under mild conditions. Fiez et al. [312] proposes a two-timescale algorithm where the follower uses a gradient-play update rule, instead of an exact

best response strategy, which has been shown to converge to the Stackelberg equilibria. Under a stronger assumption of *gradient dominance*, [314, 315] have shown that nested gradient descent methods converge to the stationary points of the outer loop, i.e., minimax, problem at a sublinear rate.

We note that these convergence results have been developed for *general* continuous games with *agnostic cost/reward* functions, meaning that the functions may have various forms, so long as they are *differentiable*, sometimes even (*Lipschitz*) *smooth*, w.r.t. each agent's policy parameter. For MARL, this is equivalent to requiring differentiability/smoothness of the long-term *return*, which relies on the properties of the game, as well as of the policy parameterization. Such an assumption is generally very restrictive. For example, the Lipschitz smoothness assumption fails to hold globally for LQ games [34, 307, 316], a special type of MGs. Fortunately, thanks to the special structure of the LQ setting, [34] has proposed several projected nested policy-gradient methods that are guaranteed to have *global* convergence to the NE, with convergence rates established. This appears to be the first-of-its-kind result in MARL. Very recently, [307] improves the results by removing the projection step in the updates, for a more general class of such games.

### 12.4.3 Mixed Setting

In stark contrast with the fully collaborative and fully competitive settings, the mixed setting is notoriously challenging and thus rather less well understood. Even in the simplest case of a two-player general-sum normal-form game, finding a Nash equilibrium is PPAD-complete [317]. Moreover, [123] has proved that value iteration methods fail to find stationary Nash or correlated equilibria for general-sum Markov games. Recently, it is shown that vanilla policy-gradient methods avoid a non-negligible subset of Nash equilibria in general-sum continuous games [32], including the LQ general-sum games [316]. Thus, additional structures on either the games or the algorithms need to be exploited, to ascertain provably convergent MARL in the mixed setting.

#### Value-Based Methods

Under relatively stringent assumptions, several value-based methods that extend Q-learning [48] to the mixed setting are guaranteed to find an equilibrium. In particular, [101] has proposed the Nash-Q learning algorithm for general-sum Markov games, where one maintains  $N$  action-value functions  $\mathcal{Q}_N = (Q^1, \dots, Q^N) : \mathcal{S} \times \mathcal{A} \rightarrow \mathbb{R}^N$  for all  $N$  agents, which are updated using sample-based estimator of a Bellman operator. Specifically, letting  $R_N = (R^1, \dots, R^N)$  denote the reward functions of the agents, Nash-Q uses the following Bellman operator:

$$(\mathcal{T}^* \mathcal{Q}_N)(s, a) = \mathbb{E}_{s' \sim \mathcal{P}(\cdot | s, a)} \{ R_N(s, a, s') + \gamma \cdot \text{Nash}[\mathcal{Q}_N(s', \cdot)] \}, \quad \forall (s, a) \in \mathcal{S} \times \mathcal{A}, \quad (12.30)$$

where  $\text{Nash}[Q_N(s', \cdot)]$  is the objective value of the Nash equilibrium of the stage game with rewards  $\{Q_N(s', a)\}_{a \in \mathcal{A}}$ . For zero-sum games, we have  $Q^1 = -Q^2$  and thus the Bellman operator defined in (12.30) is equivalent to the one in (12.18) used by minimax-Q learning [83]. Moreover, [101] establishes convergence to Nash equilibrium under the restrictive assumption that  $\text{Nash}[Q_N(s', \cdot)]$  in each iteration of the algorithm has unique Nash equilibrium. In addition, [102] has proposed the Friend-or-Foe Q-learning algorithm where each agent views the other agent as either a “friend” or a “foe”. In this case,  $\text{Nash}[Q_N(s', \cdot)]$  can be efficiently computed via linear programming. This algorithm can be viewed as a generalization of minimax-Q learning, and Nash equilibrium convergence is guaranteed for two-player zero-sum games and coordination games with a unique equilibrium. Furthermore, [318] has proposed correlated Q-learning, which replaces  $\text{Nash}[Q_N(s', \cdot)]$  in (12.30) by computing a correlated equilibrium [319], a more general equilibrium concept than Nash equilibrium. In a recent work, [320] has proposed a batch RL method to find an approximate Nash equilibrium via Bellman residue minimization [321]. They have proved that the global minimizer of the empirical Bellman residue is an approximate Nash equilibrium, followed by the error propagation analysis for the algorithm. Also in the batch RL regime, [44] has considered a simplified mixed setting for decentralized MARL: two teams of cooperative networked agents compete in a zero-sum Markov game. A decentralized variant of FQI, where the agents within one team cooperate to solve (12.7) while the two teams essentially solve (12.19), is proposed. Finite-sample error bounds have then been established for the proposed algorithm.

To address the scalability issue, independent learning is preferred, which, however, fails to converge in general [139]. Arslan and Yüksel [37] has proposed *decentralized Q-learning*, a two timescale modification of Q-learning, that is guaranteed to converge to the equilibrium for *weakly acyclic Markov games* almost surely. Each agent therein only observes local action and reward, and neither observes nor keeps track of others’ actions. All agents are instructed to use the same stationary *baseline policy* for many consecutive stages, named *exploration phase*. At the end of the *exploration phase*, all agents are *synchronized* to update their baseline policies, which makes the environment stationary for long enough, and enables the convergence of Q-learning based methods. Note that these algorithms can also be applied to the cooperative setting, as these games include Markov teams as a special case.

### Policy-Based Methods

For continuous games, due to the general negative results therein, [32] introduces a new class of games, *Morse–Smale games*, for which the gradient dynamics correspond to gradient-like flows. Then, definitive statements on almost sure convergence of PG methods to either limit cycles, Nash equilibria, or non-Nash fixed points can be made, using tools from dynamical systems theory. Moreover, [313, 322] have studied the second-order structure of game dynamics, by decomposing it into two components. The first one, named symmetric component, relates to potential games, which yields gradient descent on some implicit function; the second one, named antisymmetric component, relates to *Hamiltonian games* that follows some conservation law, motivated by classical mechanical systems analysis. The fact that gradient descent

converges to the Nash equilibrium of both types of games motivates the development of the symplectic gradient adjustment (SGA) algorithm that finds *stable fixed points* of the game, which constitute all local Nash equilibria for zero-sum games, and only a subset of local NE for general-sum games. Chasnov et al. [323] provides finite-time local convergence guarantees to a neighborhood of a *stable* local Nash equilibrium of continuous games, in both deterministic setting, with exact PG, and stochastic setting, with unbiased PG estimates. Additionally, [323] has also explored the effects of *non-uniform* learning rates on the learning dynamics and convergence rates. Fiez et al. [312] has also considered *general-sum* Stackelberg games, and shown that the same two-timescale algorithm update as in the zero-sum case now converges almost surely to the stable attractors only. It has also established finite-time performance for local convergence to a neighborhood of a stable Stackelberg equilibrium. In complete analogy to the zero-sum class, these convergence results for continuous games do not apply to MARL in Markov games directly, as they are built upon the differentiability/smoothness of the long-term return, which may not hold for general MGs, for example, LQ games [316]. Moreover, most of these convergence results are local instead of global.

Other than continuous games, the policy-based methods summarized in Sect. 12.4.2.2 can also be applied to the mixed setting via self-play. The validity of such an approach is based on a fundamental connection between game theory and online learning—if the external regret of each agent is no more than  $\epsilon$ , then their average policies constitute an  $\epsilon$ -approximate *coarse correlated equilibrium* [263, 281, 324] of the general-sum normal-form games. Thus, although in general we are unable to find a Nash equilibrium, policy optimization with self-play guarantees to find a coarse correlated equilibrium.

### Mean-Field Regime

The scalability issue in the non-cooperative setting can also be alleviated in the mean-field regime, as the cooperative setting discussed in Sect. 12.4.1.1. For general-sum games, [172] has proposed a modification of the Nash-Q learning algorithm where the actions of other agents are approximated by their empirical average. That is, the action-value function of each agent  $i$  is parametrized by  $Q^i(s, a^i, \mu_{a^{-i}})$ , where  $\mu_{a^{-i}}$  is the empirical distribution of  $\{a_j : j \neq i\}$ . Asymptotic convergence of this mean-field Nash-Q learning algorithm has also been established.

Besides, most mean-field RL algorithms are focused on addressing the *mean-field game* model. In mean-field games, each agent  $i$  has a local state  $s^i \in \mathcal{S}$  and a local action  $a^i \in \mathcal{A}$ , and the interaction among other agents is captured by an aggregated effect  $\mu$ , also known as the *mean-field term*, which is a functional of the empirical distribution of the local states and actions of the agents. Specifically, at the  $t$ -th time-step, when agent  $i$  takes action  $a_t^i$  at state  $s_t^i$  and the mean-field term is  $\mu_t$ , it receives an immediate reward  $R(s_t^i, a_t^i, \mu_t)$  and its local state evolves into  $s_{t+1}^i \sim \mathcal{P}(\cdot | s_t^i, a_t^i, \mu_t) \in \Delta(\mathcal{S})$ . Thus, from the perspective of agent  $i$ , instead of participating in a multi-agent game, it is faced with a time-varying MDP parameterized by the sequence of mean-field terms  $\{\mu_t\}_{t \geq 0}$ , which in turn is determined by the states and actions of all agents. The solution concept in MFGs is the *mean-field equilibrium*.

*rium*, which is a sequence of *pairs* of policy and mean-field terms  $\{\pi_t^*, \mu_t^*\}_{t \geq 0}$  that satisfy the following two conditions: (1)  $\pi^* = \{\pi_t^*\}_{t \geq 0}$  is the optimal policy for the time-varying MDP specified by  $\mu^* = \{\mu_t^*\}_{t \geq 0}$ , and (2)  $\mu^*$  is generated when each agent follows policy  $\pi^*$ . The existence of the mean-field equilibrium for discrete-time MFGs has been studied in [325–328] and their constructive proofs exhibit that the mean-field equilibrium can be obtained via a fixed-point iteration. Specifically, one can construct a sequence of policies and mean-field terms  $\{\pi^{(i)}\}_{t \geq 0}$  and  $\{\mu^{(i)}\}_{t \geq 0}$  such that  $\{\pi^{(i)}\}_{t \geq 0}$  solves the time-varying MDP specified by  $\mu^{(i)}$ , and  $\mu^{(i+1)}$  is generated when all players adopt policy  $\pi^{(i)}$ . Following this agenda, various model-free RL methods are proposed for solving MFGs where  $\{\pi^{(i)}\}_{t \geq 0}$  is approximately solved via single-agent RL such as Q-learning [329] and policy-based methods [24, 330], with  $\{\mu^{(i)}\}_{t \geq 0}$  being estimated via sampling. In addition, [331, 332] recently propose fictitious play updates for the mean-field state where we have  $\mu^{(i+1)} = (1 - \alpha^{(i)}) \cdot \mu^{(i)} + \alpha^{(i)} \cdot \hat{\mu}^{(i+1)}$ , with  $\alpha^{(i)}$  being the learning rate and  $\hat{\mu}^{(i+1)}$  being the mean-field term generated by policy  $\pi^{(i)}$ . Note that the aforementioned works focus on the settings with either *finite* horizon [331, 332] or *stationary* mean-field equilibria [24, 329, 330] only. Instead, recent works [333, 334] consider possibly non-stationary mean-field equilibrium in infinite horizon settings, and develop equilibrium computation algorithms that have laid foundations for model-free RL algorithms.

## 12.5 Application Highlights

In this section, we briefly review the recent empirical successes of MARL driven by the methods introduced in the previous section. In the following, we focus on the three MARL settings reviewed in Sect. 12.4 and highlight four representative and practical applications in each setting, as illustrated in Fig. 12.3.

### 12.5.1 Cooperative Setting

#### Unmanned Aerial Vehicles



**Fig. 12.3** Four representative applications of recent successes of MARL: unmanned aerial vehicles, game of Go, Poker games, and team-battle video games

One prominent application of MARL is the control of practical multi-agent systems, most of which are cooperative and decentralized. Examples of the scenarios include robot team navigation [183], smart grid operation [180], and control of mobile sensor networks [16]. Here we choose unmanned aerial vehicles (UAVs) [335–340], a recently surging application scenario of multi-agent autonomous systems, as one representative example. Specifically, a team of UAVs are deployed to accomplish a cooperation task, usually without the coordination of any central controller, i.e., in a decentralized fashion. Each UAV is normally equipped with communication devices, so that they can exchange information with some of their teammates, provided that they are inside its sensing and coverage range. As a consequence, this application naturally fits in the decentralized paradigm with networked agents we advocated in Sect. 12.4.1.2, which is also illustrated in Fig. 12.2b. Due to the high-mobility of UAVs, the communication links among agents are indeed *time-varying* and fragile, making (online) cooperation extremely challenging. Various challenges thus arise in the context of cooperative UAVs, some of which have recently been addressed by MARL.

In [335], the UAVs' optimal links discovery and selection problem is considered. Each UAV  $u \in \mathcal{U}$ , where  $\mathcal{U}$  is the set of all UAVs, has the capability to perceive the local available channels and select a connected link over a common channel shared by another agent  $v \in \mathcal{U}$ . Each UAV  $u$  has its local set of channels  $C_u$  with  $C_u \cap C_v \neq \emptyset$  for any  $u, v$ , and a connected link between two adjacent UAVs is built if they announce their messages on the same channel simultaneously. Each UAV's local state is whether the previous message has been successfully sent, and its action is to choose a pair  $(v, ch_u)$ , with  $v \in \mathcal{T}_u$  and  $ch_u \in C_u$ , where  $\mathcal{T}_u$  is the set of teammates that agent  $u$  can reach. The availability of local channels  $ch_u \in C_u$  is modeled as probabilistic, and the reward  $R^u$  is calculated by the number of messages that are successfully sent. Essentially, the algorithm in [335] is based on independent Q-learning [139], but with two heuristics to improve the tractability and convergence performance; by *fractional slicing*, it treats each dimension (fraction) of the action space independently, and estimates the actual Q-value by the average of that for all fractions; by *mutual sampling*, it shares both state-action pairs and a mutual Q-function parameter. Pham et al. [336] addresses the problem of *field coverage*, where the UAVs aim to provide a full coverage of an unknown field, while minimizing the overlapping sections among their field of views. Modeled as a Markov team, the overall state  $s$  is the concatenation of all local states  $s_i$ , which are defined as its 3D position coordinates in the environment. Each agent chooses to either head different directions, or go up and down, yielding 6 possible actions. A multi-agent Q-learning over the *joint* action space is developed, with linear function approximation. In contrast, [338] focuses on spectrum sharing among a network of UAVs. Under a remote sensing task, the UAVs are categorized into two clusters: the relaying ones that provide relay services and the other ones that gain spectrum access for the remaining ones, which perform the sensing task. Such a problem can be modeled as a *deterministic* MMDP, which can thus be solved by distributed Q-learning proposed in [87], with optimality guaranteed. Moreover, [340] considers the problem of *simultaneous* target assignment and path planning

for multiple UAVs. In particular, a team of UAVs  $U_i \in \mathbf{U}$ , with each  $U_i$ 's position at time  $t$  given by  $(x_i^U(t), y_i^U(t))$ , aim to cover all the targets  $T_j \in \mathbf{T}$  without collision with the threat areas  $D_i \in \mathbf{D}$ , as well as with other UAVs. For each  $U_i$ , a path  $P_i$  is planned as  $P_i = \{(x_i^U(0), y_i^U(0), \dots, x_i^U(n), y_i^U(n))\}$ , and the length of  $P_i$  is denoted by  $d_i$ . Thus, the goal is to minimize  $\sum_i d_i$  while the collision-free constraints are satisfied. By penalizing the collision in the reward function, such a problem can be characterized as one with a mixed MARL setting that contains both cooperative and competitive agents. Hence, the MADDPG algorithm proposed in [26] is adopted, with centralized-learning-decentralized-execution. Two other tasks that can be tackled by MARL include resource allocation in UAV-enabled communication networks, using Q-learning based method [339], aerial surveillance and base defense in UAV fleet control, using policy optimization method in a purely centralized fashion [337].

### Learning to Communicate

Another application of cooperative MARL aims to foster communication and coordination among a team of agents without explicit human supervision. Such a type of problems is usually formulated as a Dec-POMDP involving  $N$  agents, which is similar to the Markov game introduced in Definition 12.2 except that each agent cannot observe the state  $s \in \mathcal{S}$  and that each agent has the same reward function  $\mathcal{R}$ . More specifically, we assume that each agent  $i \in \mathcal{N}$  receives observations from set  $\mathcal{Y}^i$  via a noisy observation channel  $O^i : \mathcal{S} \rightarrow \mathcal{P}(\mathcal{Y}_i)$  such that agent  $i$  observes a random variable  $y^i \sim O^i(\cdot | s)$  when the environment is at state  $s$ . Note that this model can be viewed as a POMDP when there is a central planner that collects the observations of each agent and decides the actions for each agent. Due to the noisy observation channels, in such a model the agents need to communicate with each other so as to better infer the underlying state and make decisions that maximize the expected return shared by all agents. Let  $\mathcal{N}_t^i \subseteq \mathcal{N}$  be the neighbors of agent  $i$  at the  $t$ -th time-step, that is, agent  $i$  is able to receive a message  $m_t^{j \rightarrow i}$  from any agent  $j \in \mathcal{N}_t^i$  at time  $t$ . We let  $I_t^i$  denote the information agent  $i$  collects up to time  $t$ , which is defined as

$$I_t^i = \left\{ \left( o_\ell^i, \{a_\ell^j\}_{j \in \mathcal{N}}, \{m_\ell^{j \rightarrow i}\}_{j \in \mathcal{N}_t^i} \right) : \ell \in \{0, \dots, t-1\} \right\} \bigcup \{o_t^i\}, \quad (12.31)$$

which contains its history collected in previous time steps and the observation received at time  $t$ . With the information  $I_t^i$ , agent  $i$  takes an action  $a_t^i \in \mathcal{A}^i$  and also broadcasts messages  $m_t^{i \rightarrow j}$  to all agents  $j$  with  $i \in \mathcal{N}_t^j$ . That is, the policy  $\pi_t^i$  of agent  $i$  is a mapping from  $I_t^i$  to a (random) action  $\tilde{a}_t^i = (a_t^i, \{m_t^{i \rightarrow j} : i \in \mathcal{N}_t^j\})$ , i.e.,  $\tilde{a}_t^i \sim \pi_t^i(\cdot | I_t^i)$ . Notice that the size of information set  $I_t^i$  grows as  $t$  grows. To handle the memory issue, it is common to first embed  $I_t^i$  in a fixed latent space via recurrent neural network (RNN) or long short-term memory (LSTM) [341] and define the value and policy functions on top of the embedded features. Moreover, most existing works in this line of research adopt the paradigm of centralized learning and utilize techniques such as weight-sharing or attention mechanism [342] to increase computational efficiency. With centralized learning, single-agent RL algorithms such as Q-learning and actor-critic are readily applicable.

In particular, [21] first proposes to tackle the problem of learning to communicate via deep Q-learning. They propose to use two Q networks that govern taking action  $a^i \in \mathcal{A}$  and producing messages separately. Their training algorithm is an extension of the deep recurrent Q-learning (DRQN) [343], which combines RNN and deep Q-learning [10]. Following [21], various works [344–355] have proposed a variety of neural network architectures to foster communication among agents. These works combine single-agent RL methods with novel developments in deep learning, and demonstrate their performances via empirical studies. Among these works, [346, 347, 349, 354, 355] have reported the emergence of computational communication protocols among the agents when the RL algorithm is trained from scratch with text or image inputs. We remark that the algorithms used in these works are more akin to single-agent RL due to centralized learning. For more details overviews of multi-agent communication, we refer the interested readers to Sect. 6 of [42] and Sect. 3 of [20].

### 12.5.2 Competitive Setting

Regarding the competitive setting, in the following, we highlight the recent applications of MARL to *the game of Go* and *Texas hold'em poker*, which are archetypal instances of two-player perfect-information and partial-information extensive-form games, respectively.

#### The Game of Go

The game of Go is a board game played by two competing players, with the goal of surrounding more territory on the board than the opponent. These two players have access to white or black stones, respectively, and take turns placing their stones on a  $19 \times 19$  board, representing their territories. In each move, a player can place a stone to any of the total 361 positions on the board that is not already taken by a stone. Once placed on the board, the stones cannot be moved. But the stones will be removed from the board when completely surrounded by opposing stones. The game terminates when neither of the players is unwilling or unable to make a further move, and the winner is determined by counting the area of the territory and the number of stones captured by the players.

The game of Go can be viewed as a two-player zero-sum Markov game with deterministic state transitions, and the reward only appears at the end of the game. The state of this Markov game is the current configuration of the board and the reward is either one or minus one, representing either a win or a loss, respectively. Specifically, we have  $r^1(s) + r^2(s) = 0$  for any state  $s \in \mathcal{S}$ , and  $r^1(s), r^2(s) \in \{1, -1\}$  when  $s$  is a terminating state, and  $r^1(s) = r^2(s) = 0$  otherwise. Let  $V_*^i(s)$  denote the optimal value function of player  $i \in \{1, 2\}$ . Thus, in this case,  $[1 + V^i(s)]/2$  is the probability of player  $i \in \{1, 2\}$  winning the game when the current state is  $s$  and both players follow the Nash equilibrium policies thereafter. Moreover, as this Markov game is turn-based, it is known that the Nash equilibrium policies of the two players

are deterministic [234]. Furthermore, since each configuration of the board can be constructed from a sequence of moves of the two players due to deterministic transitions, we can also view the game of Go as an extensive-form game with perfect information. This problem is notoriously challenging due to the gigantic state space. It is estimated in [356] that the size of state space exceeds  $10^{360}$ , which forbids the usage of any traditional reinforcement-learning or searching algorithms.

A significant breakthrough has been made by the *AlphaGo* introduced in [1], which is the first computer Go program that defeats a human professional player on a full-sized board. *AlphaGo* integrates a variety of ideas from deep learning and reinforcement learning, and tackles the challenge of huge state space by representing the policy and value functions using deep convolutional neural networks (CNN) [357]. Specifically, both the policy and value networks are 13-layer CNNs with the same architecture, and a board configuration is represented by 48 features. Thus, both the policy and value networks take inputs of size  $19 \times 19 \times 48$ . These two networks are trained through a novel combination of supervised learning from human expert data and reinforcement learning from Monte Carlo tree search (MCTS) and self-play. Specifically, in the first stage, the policy network is trained by supervised learning to predict the actions made by the human players, where the dataset consists of 30 million positions from the KGS Go server. That is, for any state-action pair  $(s, a)$  in the dataset, the action  $a$  is treated as the response variable and the state  $s$  is regarded as the covariate. The weights of the policy network is trained via stochastic gradient ascent to maximize the likelihood function. After initializing the policy network via supervised learning, in the second stage of the pipeline, both the policy and value networks are trained via reinforcement learning and self-play. In particular, new data are generated by games played between the current policy network and a random previous iteration of the policy network. Moreover, the policy network is updated following policy gradient, and the value network aims to find the value function associated with the policy network and is updated by minimizing the mean-squared prediction error. Finally, when playing the game, the current iterates of the policy and value networks are combined to produce an improved policy by lookahead search via MCTS. The actual action taken by *AlphaGo* is determined by such an MCTS policy. Moreover, to improve computational efficiency, *AlphaGo* uses an asynchronous and distributed version of MCTS to speed up simulation.

Since the advent of *AlphaGo*, an improved version, known as *AlphaGo Zero*, has been proposed in [2]. Compared with the vanilla *AlphaGo*, *AlphaGo Zero* does not use supervised learning to initialize the policy network. Instead, both the policy and value networks are trained from scratch solely via reinforcement learning and self-play. Besides, instead of having separate policy and value functions share the same network architecture, in *AlphaGo Zero*, these two networks are aggregated into a single neural network structure. Specifically, the policy and value functions are represented by  $(p(s), V(s)) = f_\theta(s)$ , where  $s \in \mathcal{S}$  is the state which represents the current board,  $f_\theta$  is a deep CNN with parameter  $\theta$ ,  $V(s)$  is a scalar that corresponds to the value function, and  $p(s)$  is a vector which represents the policy, i.e., for each entry  $a \in \mathcal{A}$ ,  $p_a(s)$  is the probability of taking action  $a$  at state  $s$ . Thus, under such a network structure, the policy and value networks automatically share the same

low-level representations of the states. Moreover, the parameter  $\theta$  of network  $f_\theta$  is trained via self-play and MCTS. Specifically, at each time-step  $t$ , based on the policy  $p$  and value  $V$  given by  $f_{\theta_t}$ , an MCTS policy  $\pi_t$  can be obtained and a move is executed following policy  $\pi_t(s_t)$ . Such a simulation procedure continues until the current game terminates. Then the outcome of the  $t$ -th time-step,  $z_t \in \{1, -1\}$ , is recorded, according to the perspective of the player at time-step  $t$ . Then the parameter  $\theta$  is updated by following a stochastic gradient step on a loss function  $\ell_t$ , which is defined as

$$\ell_t(\theta) = [z - V(s_t)]^2 - \pi_t^\top \log p(s_t) + c \cdot \|\theta\|_2^2, \quad (p(\cdot), V(\cdot)) = f_\theta(\cdot).$$

Thus,  $\ell_t$  is the sum of the mean-squared prediction error of the value function, cross-entropy loss between the policy network and the MCTS policy, and a weight-decay term for regularization. It is reported that AlphaGo Zero has defeated the strongest versions of the previous AlphaGo and that it also has demonstrated non-standard Go strategies that had not been discovered before. Finally, the techniques adopted in AlphaGo Zero have been generalized to other challenging board games. Specifically, [358] proposes the AlphaZero program that is trained by self-play and reinforcement learning with zero human knowledge, and achieves superhuman performances in the games of chess, shogi, and Go.

### Texas Hold'em Poker

Another remarkable applicational achievement of MARL in the competitive setting focuses on developing artificial intelligence in the Texas hold'em poker, which is one of the most popular variations of the poker. Texas hold'em is usually played by a group of two or more players, where each player is first dealt with two *private cards* face down. Then five *community cards* are dealt face up in three rounds. In each round, each player has four possible actions—*check*, *call*, *raise*, and *fold*. After all the cards are dealt, each player who has not folded has seven cards in total, consisting of five community cards and two private cards. Each of these players then finds the best five-card poker hand out of all combinations of the seven cards. The player with the best hand is the winner and wins all the money that the players wager for that hand, which is also known as the *pot*. Note that each hand of Texas hold'em terminates after three rounds, and the payoffs of the player are only known after the hand ends. Also notice that each player is unaware of the private cards of the rest of the players. Thus, Texas hold'em is an instance of multi-player extensive-form game with incomplete information. The game is called *heads-up* when there are only two players. When both the bet sizes and the amount of allowed raises are fixed, the game is called *limit hold'em*. In the no-limit hold'em, however, each player may bet or raise any amount up to all of the money the player has at the table, as long as it exceeds the previous bet or raise.

There has been quest for developing superhuman computer poker programs for over two decades [113, 359]. Various methods have been shown successful for simple variations of poker such as Kuhn poker [360] and Leduc hold'em [361]. However, the full-fledged Texas hold'em is much more challenging and several breakthroughs have

been achieved only recently. The simplest version of Texas hold’em is *heads-up limit hold’em* (HULHE), which has  $3.9 \times 10^{14}$  information sets in total [362], where a player is required to take an action at each information set. Bowling et al. [362] has for the first time reported solving HULHE to approximate Nash equilibrium via CFR<sup>+</sup> [294, 295], a variant of counterfactual regret minimization [109]. Subsequently, other methods such as Neural Fictitious Self-Play [25] and Monte Carlo tree search with self-play [363] have also been adopted to successfully solve HULHE.

Despite these breakthroughs, solving *heads-up no-limit hold’em* (HUNL) with artificial intelligence has remained open until recently, which has more than  $6 \times 10^{161}$  information sets, an astronomical number. Thus, in HUNL, it is impossible (in today’s computational power) to traverse all information sets, making it inviable to apply CFR<sup>+</sup> as in [362]. Ground-breaking achievements have recently been made by *DeepStack* [364] and *Libratus* [365], two computer poker programs developed independently, which defeat human professional poker players in HUNL for the first time. Both of these programs adopt CFR as the backbone of their algorithmic frameworks, but adopt different strategies for handling the gigantic size of the game. In particular, DeepStack applies deep learning to learn good representations of the game and proposes *deep counterfactual value networks* to integrate deep learning and CFR. Moreover, DeepStack adopts limited depth lookahead planning to reduce the gigantic  $6 \times 10^{161}$  information sets to no more than  $10^{17}$  information sets, thus making it possible to enumerate all information sets. In contrast, Libratus does not utilize any deep learning techniques. Instead, it reduces the size of the game by computation of an abstraction of the game, which is possible since many of the information sets are very similar. Moreover, it further reduces the complexity by using the sub-game decomposition technique [366–368] for imperfect-information games and by constructing fine-grained abstractions of the sub-games. When the abstractions are constructed, an improved version of the Monte Carlo CFR [282–284] is utilized to compute the policy. Furthermore, very recently, based upon *Libratus*, [8] has proposed *Pluribus*, a computer poker program that has been shown to be stronger than top human professionals in no-limit Texas hold’em poker with six players. The success of *Pluribus* is attributed to the following techniques that have appeared in the literature: abstraction and sub-game decomposition for large-scale imperfect-information games, Monte Carlo CFR, self-play, and depth-limited search.

### Other Applications

Furthermore, another popular testbed of MARL is the StarCraft II [369], which is an immensely popular multi-player real-strategy computer game. This game can be formulated as a multi-agent Markov game with partial observation, where each player has only limited information of the game state. Designing reinforcement-learning systems for StarCraft II is extremely challenging due to the needs to make decisions under uncertainty and incomplete information, to consider the optimal strategy in the long-run, and to design good reward functions that elicit learning. Since released, both the full-game and sub-game versions of StarCraft II have gained tremendous research interest. A breakthrough in this game was achieved by *AlphaStar*, recently proposed in [370], which has demonstrated superhuman performance in zero-sum

two-player full-game StarCraft II. Its reinforcement-learning algorithm combines LSTM for the parametrization of policy and value functions, asynchronous actor-critic [371] for policy updates, and Neural Fictitious Self-Play [25] for equilibrium finding.

### 12.5.3 Mixed Settings

Compared to the cooperative and competitive settings, research on MARL under the mixed setting is rather less explored. One application in this setting is multi-player poker. As we have mentioned in Sect. 12.5.2, Pluribus introduced in [8] has demonstrated superhuman performance in six-player no-limit Texas hold’em. In addition, as an extension of the problem of learning to communicate, introduced in Sect. 12.5.1, there is a line of research that aims to apply MARL to tackle learning social dilemmas, which is usually formulated as a multi-agent stochastic game with partial information. Thus, most of the algorithms proposed under these settings incorporate RNN or LSTM for learning representations of the histories experienced by the agent, and the performances of these algorithms are usually exhibited using experimental results, see, e.g., [19, 372, 373], and the references therein.

Moreover, another example of the mixed setting is the case where the agents are divided into two opposing teams that play zero-sum games. The reward of a team is shared by each player within this team. Compared with two-player zero-sum games, this setting is more challenging in that both cooperation among teammates and competition against the opposing team need to be taken into consideration. A prominent testbed of this case is the *Dota 2* video game, where each of the two teams, each with five players, aims to conquer the base of the other team and defend its own base. Each player independently controls a powerful character known as the *hero*, and only observes the state of the game via the video output on the screen. Thus, Dota 2 is a zero-sum Markov game played by two teams, with each agent having imperfect information of the game. For this challenging problem, in 2018, *OpenAI* has proposed the *OpenAI Five* AI system [3], which enjoys superhuman performance and has defeated human world champions in an e-sports game. The algorithmic framework integrates LSTM for learning good representations and proximal policy optimization [72] with self-play for policy learning. Moreover, to balance between effective coordination and communication cost, instead of having explicit communication channels among the teams, OpenAI Five utilizes reward shaping by having a hyperparameter, named “team spirit”, to balance the relative importance between each hero’s individual reward function and the average of the team’s reward function.

## 12.6 Conclusions and Future Directions

Multi-agent RL has long been an active and significant research area in reinforcement learning, in view of the ubiquity of sequential decision-making with multiple agents coupled in their actions and information. In stark contrast to its great empirical success, theoretical understanding of MARL algorithms is well recognized to be challenging and relatively lacking in the literature. Indeed, establishing an encompassing theory for MARL requires tools spanning dynamic programming, game theory, optimization theory, and statistics, which are non-trivial to unify and investigate within one context.

In this chapter, we have provided a selective overview of mostly recent MARL algorithms, backed by theoretical analysis, followed by several high-profile but challenging applications that have been addressed lately. Following the classical overview [11], we have categorized the algorithms into three groups: those solving problems that are fully cooperative, fully competitive, and a mix of the two. Orthogonal to the existing reviews on MARL, this chapter has laid emphasis on several new angles and taxonomies of MARL theory, some of which have been drawn from our own research endeavors and interests. We note that our overview should not be viewed as a comprehensive one, but instead as a focused one dictated by our own interests and expertise, which should appeal to researchers of similar interests, and provide a stimulus for future research directions in this general topical area. Accordingly, we have identified the following paramount while open avenues for future research on MARL theory.

### Partially Observed Settings

Partial observability of the system states and the actions of other agents is quintessential and inevitable in many practical MARL applications. In general, these settings can be modeled as a partially observed stochastic game (POSG), which includes the cooperative setting with a common reward function, i.e., the Dec-POMDP model, as a special case. Nevertheless, as pointed out in Sect. 12.4.1.3, even the cooperative task is NEXP-hard [104] and difficult to solve. In fact, the information state for optimal decision-making in POSGs can be very complicated and involve belief generation over the opponents' policies [119], compared to that in POMDPs, which requires belief on only states. This difficulty essentially stems from the heterogenous beliefs of agents resulting from their own observations obtained from the model, an inherent challenge of MARL mentioned in Sect. 12.3 due to various information structures. It might be possible to start by generalizing the centralized-learning-decentralized-execution scheme for solving Dec-POMDPs [121, 152] to solving POSGs.

### Deep MARL Theory

As mentioned in Sect. 12.3.3, using deep neural networks for function approximation can address the scalability issue in MARL. In fact, most of the recent empirical successes in MARL result from the use of DNNs [25–29]. Nonetheless, because of lack of theoretical backings, we have not included details of these algorithms in this chapter. Very recently, a few attempts have been made to understand the

global convergence of several single-agent deep RL algorithms, such as neural TD learning [374] and neural policy optimization [78, 79], when overparameterized neural networks [375, 376] are used. It is thus promising to extend these results to multi-agent settings, as initial steps toward theoretical understanding of deep MARL.

### Model-Based MARL

It may be slightly surprising that very few MARL algorithms in the literature are *model-based*, in the sense that the MARL model is first estimated, and then used as a nominal one to design algorithms. To the best of our knowledge, the only existing model-based MARL algorithms include the early one in [377] that solves single-controller-stochastic games, a special zero-sum MG; and the later improved one in [378], named R-MAX, for zero-sum MGs. These algorithms are also built upon the principle of optimism in the face of uncertainty [245, 246], as several aforementioned model-free ones. Considering recent progresses in model-based RL, especially its provable advantages over model-free ones in certain regimes [379, 380], it is worth generalizing these results to MARL to improve its sample efficiency.

### Convergence of Policy Gradient Methods

As mentioned in Sect. 12.4.3, the convergence result of vanilla policy-gradient method in general MARL is mostly negative, i.e., it may avoid even the local NE points in many cases. This is essentially related to the challenge of non-stationarity in MARL, see Sect. 12.3.2. Even though some remedies have been advocated [312, 313, 322, 323] to stabilize the convergence in *general continuous* games, these assumptions are not easily verified/satisfied in MARL, e.g., even in the simplest LQ setting [316], as they depend not only on the model, but also on the policy parameterization. Due to this subtlety, it may be interesting to explore the (global) convergence of policy-based methods for MARL, probably starting with the simple LQ setting, i.e., general-sum LQ games, in analogy to that for the zero-sum counterpart [34, 307]. Such an exploration may also benefit from the recent advances of nonconvex-(non)concave optimization [33, 315, 381].

### MARL with Robustness/Safety Concerns

Concerning the challenge of non-unique learning goals in MARL (see Sect. 12.3.1), we believe it is of merit to consider robustness and/or safety constraints in MARL. To the best of our knowledge, this is still a relatively uncharted territory. In fact, safe RL has been recognized as one of the most significant challenges in the single-agent setting [382]. With more than one agents that may have conflicted objectives, guaranteeing safety becomes more involved, as the safety requirement now concerns the coupling of all agents. One straightforward model is constrained multi-agent MDPs/Markov games, with the constraints characterizing the safety requirement. Learning with provably safety guarantees in this setting is non-trivial, but necessary for some safety-critical MARL applications as autonomous driving [9] and robotics [5]. In addition, it is also natural to think of robustness against adversarial agents, especially in the decentralized/distributed cooperative MARL settings as in [23, 96, 130], where the adversary may disturb the learning process in an anonymous way—a common scenario in distributed systems. Recent development of robust distributed

supervised learning against Byzantine adversaries [383, 384] may be useful in this context.

## References

1. Silver, D., Huang, A., Maddison, C.J., Guez, A., Sifre, L., Van Den Driessche, G., Schrittwieser, J., Antonoglou, I., Panneershelvam, V., Lanctot, M., et al.: Mastering the game of Go with deep neural networks and tree search. *Nature* **529**(7587), 484–489 (2016)
2. Silver, D., Schrittwieser, J., Simonyan, K., Antonoglou, I., Huang, A., Guez, A., Hubert, T., Baker, L., Lai, M., Bolton, A., et al.: Mastering the game of Go without human knowledge. *Nature* **550**(7676), 354 (2017)
3. OpenAI: Openai five. <https://blog.openai.com/openai-five/> (2018)
4. Vinyals, O., Babuschkin, I., Chung, J., Mathieu, M., Jaderberg, M., Czarnecki, W.M., Dudzik, A., Huang, A., Georgiev, P., Powell, R., Ewalds, T., Horgan, D., Kroiss, M., Danihelka, I., Agapiou, J., Oh, J., Dalibard, V., Choi, D., Sifre, L., Sulsky, Y., Vezhnevets, S., Molloy, J., Cai, T., Budden, D., Paine, T., Gulcehre, C., Wang, Z., Pfaff, T., Pohlen, T., Wu, Y., Yogatama, D., Cohen, J., McKinney, K., Smith, O., Schaul, T., Lillicrap, T., Apps, C., Kavukcuoglu, K., Hassabis, D., Silver, D.: AlphaStar: mastering the real-time strategy game starcraft II. <https://deepmind.com/blog/alphastar-mastering-real-time-strategy-game-starcraft-ii/> (2019)
5. Kober, J., Bagnell, J.A., Peters, J.: Reinforcement learning in robotics: a survey. *Int. J. Robot. Res.* **32**(11), 1238–1274 (2013)
6. Lillicrap, T.P., Hunt, J.J., Pritzel, A., Heess, N., Erez, T., Tassa, Y., Silver, D., Wierstra, D.: Continuous control with deep reinforcement learning. In: International Conference on Learning Representations (2016)
7. Brown, N., Sandholm, T.: Libratus: the superhuman AI for no-limit Poker. In: International Joint Conference on Artificial Intelligence, pp. 5226–5228 (2017)
8. Brown, N., Sandholm, T.: Superhuman AI for multiplayer poker. *Science* **365**, 885–890 (2019)
9. Shalev-Shwartz, S., Shamir, S., Shashua, A.: Safe, multi-agent, reinforcement learning for autonomous driving (2016). arXiv preprint [arXiv:1610.03295](https://arxiv.org/abs/1610.03295)
10. Mnih, V., Kavukcuoglu, K., Silver, D., Rusu, A.A., Veness, J., Bellemare, M.G., Graves, A., Riedmiller, M., Fidjeland, A.K., Ostrovski, G., et al.: Human-level control through deep reinforcement learning. *Nature* **518**(7540), 529–533 (2015)
11. Busoniu, L., Babuska, R., De Schutter, B., et al.: A comprehensive survey of multiagent reinforcement learning. *IEEE Trans. Syst. Man Cybern. Part C* **38**(2), 156–172 (2008)
12. Adler, J.L., Blue, V.J.: A cooperative multi-agent transportation management and route guidance system. *Transp. Res. Part C: Emerg. Technol.* **10**(5), 433–454 (2002)
13. Wang, S., Wan, J., Zhang, D., Li, D., Zhang, C.: Towards smart factory for industry 4.0: a self-organized multi-agent system with big data based feedback and coordination. *Comput. Netw.* **101**, 158–168 (2016)
14. Jangmin, O., Lee, J.W., Zhang, B.T.: Stock trading system using reinforcement learning with cooperative agents. In: International Conference on Machine Learning, pp. 451–458 (2002)
15. Lee, J.W., Park, J., Jangmin, O., Lee, J., Hong, E.: A multiagent approach to  $Q$ -learning for daily stock trading. *IEEE Trans. Syst. Man Cybern.-Part A: Syst. Hum.* **37**(6), 864–877 (2007)
16. Cortes, J., Martinez, S., Karatas, T., Bullo, F.: Coverage control for mobile sensing networks. *IEEE Trans. Robot. Autom.* **20**(2), 243–255 (2004)
17. Choi, J., Oh, S., Horowitz, R.: Distributed learning and cooperative control for multi-agent systems. *Automatica* **45**(12), 2802–2814 (2009)
18. Castelfranchi, C.: The theory of social functions: challenges for computational social science and multi-agent learning. *Cogn. Syst. Res.* **2**(1), 5–38 (2001)
19. Leibo, J.Z., Zambaldi, V., Lanctot, M., Marecki, J., Graepel, T.: Multi-agent reinforcement learning in sequential social dilemmas. In: International Conference on Autonomous Agents and Multi-Agent Systems, pp. 464–473 (2017)

20. Hernandez-Leal, P., Kartal, B., Taylor, M.E.: A survey and critique of multiagent deep reinforcement learning (2018). arXiv preprint [arXiv:1810.05587](https://arxiv.org/abs/1810.05587)
21. Foerster, J., Assael, Y.M., de Freitas, N., Whiteson, S.: Learning to communicate with deep multi-agent reinforcement learning. In: Advances in Neural Information Processing Systems, pp. 2137–2145 (2016)
22. Zazo, S., Macua, S.V., Sánchez-Fernández, M., Zazo, J.: Dynamic potential games with constraints: fundamentals and applications in communications. IEEE Trans. Signal Process. **64**(14), 3806–3821 (2016)
23. Zhang, K., Yang, Z., Liu, H., Zhang, T., Başar, T.: Fully decentralized multi-agent reinforcement learning with networked agents. In: International Conference on Machine Learning, pp. 5867–5876 (2018)
24. Subramanian, J., Mahajan, A.: Reinforcement learning in stationary mean-field games. In: International Conference on Autonomous Agents and Multi-Agent Systems, pp. 251–259 (2019)
25. Heinrich, J., Silver, D.: Deep reinforcement learning from self-play in imperfect-information games (2016). arXiv preprint [arXiv:1603.01121](https://arxiv.org/abs/1603.01121)
26. Lowe, R., Wu, Y., Tamar, A., Harb, J., Abbeel, P., Mordatch, I.: Multi-agent actor-critic for mixed cooperative-competitive environments. In: Advances in Neural Information Processing Systems, pp. 6379–6390 (2017)
27. Foerster, J., Farquhar, G., Afouras, T., Nardelli, N., Whiteson, S.: Counterfactual multi-agent policy gradients (2017). arXiv preprint [arXiv:1705.08926](https://arxiv.org/abs/1705.08926)
28. Gupta, J.K., Egorov, M., Kochenderfer, M.: Cooperative multi-agent control using deep reinforcement learning. In: International Conference on Autonomous Agents and Multi-Agent Systems, pp. 66–83 (2017)
29. Omidshafiei, S., Pazis, J., Amato, C., How, J.P., Vian, J.: Deep decentralized multi-task multi-agent reinforcement learning under partial observability. In: International Conference on Machine Learning, pp. 2681–2690 (2017)
30. Kawamura, K., Mizukami, N., Tsuruoka, Y.: Neural fictitious self-play in imperfect information games with many players. In: Workshop on Computer Games, pp. 61–74 (2017)
31. Zhang, L., Wang, W., Li, S., Pan, G.: Monte Carlo neural fictitious self-play: Approach to approximate Nash equilibrium of imperfect-information games (2019). arXiv preprint [arXiv:1903.09569](https://arxiv.org/abs/1903.09569)
32. Mazumdar, E., Ratliff, L.J.: On the convergence of gradient-based learning in continuous games (2018). arXiv preprint [arXiv:1804.05464](https://arxiv.org/abs/1804.05464)
33. Jin, C., Netrapalli, P., Jordan, M.I.: Minmax optimization: stable limit points of gradient descent ascent are locally optimal (2019). arXiv preprint [arXiv:1902.00618](https://arxiv.org/abs/1902.00618)
34. Zhang, K., Yang, Z., Başar, T.: Policy optimization provably converges to Nash equilibria in zero-sum linear quadratic games. In: Advances in Neural Information Processing Systems (2019)
35. Sidford, A., Wang, M., Yang, L.F., Ye, Y.: Solving discounted stochastic two-player games with near-optimal time and sample complexity (2019). arXiv preprint [arXiv:1908.11071](https://arxiv.org/abs/1908.11071)
36. Oliehoek, F.A., Amato, C.: A Concise Introduction to Decentralized POMDPs, vol. 1. Springer, Berlin (2016)
37. Arslan, G., Yüksel, S.: Decentralized Q-learning for stochastic teams and games. IEEE Trans. Autom. Control **62**(4), 1545–1558 (2017)
38. Yongacoglu, B., Arslan, G., Yüksel, S.: Learning team-optimality for decentralized stochastic control and dynamic games (2019). arXiv preprint [arXiv:1903.05812](https://arxiv.org/abs/1903.05812)
39. Zhang, K., Miehling, E., Başar, T.: Online planning for decentralized stochastic control with partial history sharing. In: IEEE American Control Conference, pp. 167–172 (2019)
40. Hernandez-Leal, P., Kaisers, M., Baarslag, T., de Cote, E.M.: A survey of learning in multiagent environments: dealing with non-stationarity (2017). arXiv preprint [arXiv:1707.09183](https://arxiv.org/abs/1707.09183)
41. Nguyen, T.T., Nguyen, N.D., Nahavandi, S.: Deep reinforcement learning for multiagent systems: a review of challenges, solutions and applications (2018). arXiv preprint [arXiv:1812.11794](https://arxiv.org/abs/1812.11794)

42. Oroojlooy Javid, A., Hajinezhad, D.: A review of cooperative multi-agent deep reinforcement learning (2019). arXiv preprint [arXiv:1908.03963](https://arxiv.org/abs/1908.03963)
43. Zhang, K., Yang, Z., Başar, T.: Networked multi-agent reinforcement learning in continuous spaces. In: IEEE Conference on Decision and Control, pp. 2771–2776 (2018)
44. Zhang, K., Yang, Z., Liu, H., Zhang, T., Başar, T.: Finite-sample analyses for fully decentralized multi-agent reinforcement learning (2018). arXiv preprint [arXiv:1812.02783](https://arxiv.org/abs/1812.02783)
45. Monahan, G.E.: State of the art-a survey of partially observable Markov decision processes: theory, models, and algorithms. *Manag. Sci.* **28**(1), 1–16 (1982)
46. Cassandra, A.R.: Exact and approximate algorithms for partially observable Markov decision processes. Brown University (1998)
47. Bertsekas, D.P.: *Dynamic Programming and Optimal Control*, vol. 1. Athena Scientific, Belmont (2005)
48. Watkins, C.J., Dayan, P.: Q-learning. *Mach. Learn.* **8**(3–4), 279–292 (1992)
49. Szepesvári, C., Littman, M.L.: A unified analysis of value-function-based reinforcement-learning algorithms. *Neural Comput.* **11**(8), 2017–2060 (1999)
50. Singh, S., Jaakkola, T., Littman, M.L., Szepesvári, C.: Convergence results for single-step on-policy reinforcement-learning algorithms. *Mach. Learn.* **38**(3), 287–308 (2000)
51. Chang, H.S., Fu, M.C., Hu, J., Marcus, S.I.: An adaptive sampling algorithm for solving Markov decision processes. *Oper. Res.* **53**(1), 126–139 (2005)
52. Kocsis, L., Szepesvári, C.: Bandit based Monte-Carlo planning. In: European Conference on Machine Learning, pp. 282–293. Springer (2006)
53. Coulom, R.: Efficient selectivity and backup operators in Monte-Carlo tree search. In: International Conference on Computers and Games, pp. 72–83 (2006)
54. Agrawal, R.: Sample mean based index policies by  $O(\log n)$  regret for the multi-armed bandit problem. *Adv. Appl. Probab.* **27**(4), 1054–1078 (1995)
55. Auer, P., Cesa-Bianchi, N., Fischer, P.: Finite-time analysis of the multiarmed bandit problem. *Mach. Learn.* **47**(2–3), 235–256 (2002)
56. Jiang, D., Ekwedike, E., Liu, H.: Feedback-based tree search for reinforcement learning. In: International Conference on Machine Learning, pp. 2284–2293 (2018)
57. Shah, D., Xie, Q., Xu, Z.: On reinforcement learning using Monte-Carlo tree search with supervised learning: non-asymptotic analysis (2019). arXiv preprint [arXiv:1902.05213](https://arxiv.org/abs/1902.05213)
58. Tesauro, G.: Temporal difference learning and TD-Gammon. *Commun. ACM* **38**(3), 58–68 (1995)
59. Tsitsiklis, J.N., Van Roy, B.: Analysis of temporal-difference learning with function approximation. In: Advances in Neural Information Processing Systems, pp. 1075–1081 (1997)
60. Sutton, R.S., Barto, A.G.: *Reinforcement Learning: An Introduction*. MIT Press, Cambridge (2018)
61. Sutton, R.S., Szepesvári, C., Maei, H.R.: A convergent  $O(n)$  algorithm for off-policy temporal-difference learning with linear function approximation. In: Advances in Neural Information Processing Systems, vol. 21(21), pp. 1609–1616 (2008)
62. Sutton, R.S., Maei, H.R., Precup, D., Bhatnagar, S., Silver, D., Szepesvári, C., Wiewiora, E.: Fast gradient-descent methods for temporal-difference learning with linear function approximation. In: International Conference on Machine Learning, pp. 993–1000 (2009)
63. Liu, B., Liu, J., Ghavamzadeh, M., Mahadevan, S., Petrik, M.: Finite-sample analysis of proximal gradient TD algorithms. In: Conference on Uncertainty in Artificial Intelligence, pp. 504–513 (2015)
64. Bhatnagar, S., Precup, D., Silver, D., Sutton, R.S., Maei, H.R., Szepesvári, C.: Convergent temporal-difference learning with arbitrary smooth function approximation. In: Advances in Neural Information Processing Systems, pp. 1204–1212 (2009)
65. Dann, C., Neumann, G., Peters, J., et al.: Policy evaluation with temporal differences: a survey and comparison. *J. Mach. Learn. Res.* **15**, 809–883 (2014)
66. Sutton, R.S., McAllester, D.A., Singh, S.P., Mansour, Y.: Policy gradient methods for reinforcement learning with function approximation. In: Advances in Neural Information Processing Systems, pp. 1057–1063 (2000)

67. Williams, R.J.: Simple statistical gradient-following algorithms for connectionist reinforcement learning. *Mach. Learn.* **8**(3–4), 229–256 (1992)
68. Baxter, J., Bartlett, P.L.: Infinite-horizon policy-gradient estimation. *J. Artif. Intell. Res.* **15**, 319–350 (2001)
69. Konda, V.R., Tsitsiklis, J.N.: Actor-critic algorithms. In: *Advances in Neural Information Processing Systems*, pp. 1008–1014 (2000)
70. Bhatnagar, S., Sutton, R., Ghavamzadeh, M., Lee, M.: Natural actor-critic algorithms. *Automatica* **45**(11), 2471–2482 (2009)
71. Silver, D., Lever, G., Heess, N., Degris, T., Wierstra, D., Riedmiller, M.: Deterministic policy gradient algorithms. In: *International Conference on Machine Learning*, pp. 387–395 (2014)
72. Schulman, J., Wolski, F., Dhariwal, P., Radford, A., Klimov, O.: Proximal policy optimization algorithms (2017). arXiv preprint [arXiv:1707.06347](https://arxiv.org/abs/1707.06347)
73. Schulman, J., Levine, S., Abbeel, P., Jordan, M., Moritz, P.: Trust region policy optimization. In: *International Conference on Machine Learning*, pp. 1889–1897 (2015)
74. Haarnoja, T., Zhou, A., Abbeel, P., Levine, S.: Soft actor-critic: off-policy maximum entropy deep reinforcement learning with a stochastic actor (2018). arXiv preprint [arXiv:1801.01290](https://arxiv.org/abs/1801.01290)
75. Yang, Z., Zhang, K., Hong, M., Başar, T.: A finite sample analysis of the actor-critic algorithm. In: *IEEE Conference on Decision and Control*, pp. 2759–2764 (2018)
76. Zhang, K., Koppel, A., Zhu, H., Başar, T.: Global convergence of policy gradient methods to (almost) locally optimal policies (2019). arXiv preprint [arXiv:1906.08383](https://arxiv.org/abs/1906.08383)
77. Agarwal, A., Kakade, S.M., Lee, J.D., Mahajan, G.: Optimality and approximation with policy gradient methods in Markov decision processes (2019). arXiv preprint [arXiv:1908.00261](https://arxiv.org/abs/1908.00261)
78. Liu, B., Cai, Q., Yang, Z., Wang, Z.: Neural proximal/trust region policy optimization attains globally optimal policy (2019). arXiv preprint [arXiv:1906.10306](https://arxiv.org/abs/1906.10306)
79. Wang, L., Cai, Q., Yang, Z., Wang, Z.: Neural policy gradient methods: global optimality and rates of convergence (2019). arXiv preprint [arXiv:1909.01150](https://arxiv.org/abs/1909.01150)
80. Chen, Y., Wang, M.: Stochastic primal-dual methods and sample complexity of reinforcement learning (2016). arXiv preprint [arXiv:1612.02516](https://arxiv.org/abs/1612.02516)
81. Wang, M.: Primal-dual  $\pi$  learning: sample complexity and sublinear run time for ergodic Markov decision problems (2017). arXiv preprint [arXiv:1710.06100](https://arxiv.org/abs/1710.06100)
82. Shapley, L.S.: Stochastic games. *Proc. Natl. Acad. Sci.* **39**(10), 1095–1100 (1953)
83. Littman, M.L.: Markov games as a framework for multi-agent reinforcement learning. In: *International Conference on Machine Learning*, pp. 157–163 (1994)
84. Başar, T., Olsder, G.J.: *Dynamic Noncooperative Game Theory*, vol. 23. SIAM, Philadelphia (1999)
85. Filar, J., Vrieze, K.: *Competitive Markov Decision Processes*. Springer Science & Business Media, Berlin (2012)
86. Boutilier, C.: Planning, learning and coordination in multi-agent decision processes. In: *Conference on Theoretical Aspects of Rationality and Knowledge*, pp. 195–210 (1996)
87. Lauer, M., Riedmiller, M.: An algorithm for distributed reinforcement learning in cooperative multi-agent systems. In: *International Conference on Machine Learning* (2000)
88. Yoshikawa, T.: Decomposition of dynamic team decision problems. *IEEE Trans. Autom. Control* **23**(4), 627–632 (1978)
89. Ho, Y.C.: Team decision theory and information structures. *Proc. IEEE* **68**(6), 644–654 (1980)
90. Wang, X., Sandholm, T.: Reinforcement learning to play an optimal Nash equilibrium in team Markov games. In: *Advances in Neural Information Processing Systems*, pp. 1603–1610 (2003)
91. Mahajan, A.: Sequential decomposition of sequential dynamic teams: applications to real-time communication and networked control systems. Ph.D. thesis, University of Michigan (2008)
92. González-Sánchez, D., Hernández-Lerma, O.: *Discrete-Time Stochastic Control and Dynamic Potential Games: The Euler-Equation Approach*. Springer Science & Business Media, Berlin (2013)

93. Valcarcel Macua, S., Zazo, J., Zazo, S.: Learning parametric closed-loop policies for Markov potential games. In: International Conference on Learning Representations (2018)
94. Kar, S., Moura, J.M., Poor, H.V.: QD-learning: a collaborative distributed strategy for multi-agent reinforcement learning through consensus + innovations. *IEEE Trans. Signal Process.* **61**(7), 1848–1862 (2013)
95. Doan, T., Maguluri, S., Romberg, J.: Finite-time analysis of distributed TD (0) with linear function approximation on multi-agent reinforcement learning. In: International Conference on Machine Learning, pp. 1626–1635 (2019)
96. Wai, H.T., Yang, Z., Wang, Z., Hong, M.: Multi-agent reinforcement learning via double averaging primal-dual optimization. In: Advances in Neural Information Processing Systems, pp. 9649–9660 (2018)
97. OpenAI: Openai dota 2 1v1 bot. <https://openai.com/the-international/> (2017)
98. Jacobson, D.: Optimal stochastic linear systems with exponential performance criteria and their relation to deterministic differential games. *IEEE Trans. Autom. Control* **18**(2), 124–131 (1973)
99. Başar, T., Bernhard, P.:  $H_\infty$  Optimal Control and Related Minimax Design Problems: A Dynamic Game Approach. Birkhäuser, Boston (1995)
100. Zhang, K., Hu, B., Başar, T.: Policy optimization for  $\mathcal{H}_2$  linear control with  $\mathcal{H}_\infty$  robustness guarantee: implicit regularization and global convergence (2019). arXiv preprint [arXiv:1910.09496](https://arxiv.org/abs/1910.09496)
101. Hu, J., Wellman, M.P.: Nash Q-learning for general-sum stochastic games. *J. Mach. Learn. Res.* **4**, 1039–1069 (2003)
102. Littman, M.L.: Friend-or-foe Q-learning in general-sum games. In: International Conference on Machine Learning, pp. 322–328 (2001)
103. Lagoudakis, M.G., Parr, R.: Learning in zero-sum team Markov games using factored value functions. In: Advances in Neural Information Processing Systems, pp. 1659–1666 (2003)
104. Bernstein, D.S., Givan, R., Immerman, N., Zilberstein, S.: The complexity of decentralized control of Markov decision processes. *Math. Oper. Res.* **27**(4), 819–840 (2002)
105. Osborne, M.J., Rubinstein, A.: A Course in Game Theory. MIT Press, Cambridge (1994)
106. Shoham, Y., Leyton-Brown, K.: Multiagent Systems: Algorithmic, Game-Theoretic, and Logical Foundations. Cambridge University Press, Cambridge (2008)
107. Koller, D., Megiddo, N.: The complexity of two-person zero-sum games in extensive form. *Games Econ. Behav.* **4**(4), 528–552 (1992)
108. Kuhn, H.: Extensive games and the problem of information. *Contrib. Theory Games* **2**, 193–216 (1953)
109. Zinkevich, M., Johanson, M., Bowling, M., Piccione, C.: Regret minimization in games with incomplete information. In: Advances in Neural Information Processing Systems, pp. 1729–1736 (2008)
110. Heinrich, J., Lanctot, M., Silver, D.: Fictitious self-play in extensive-form games. In: International Conference on Machine Learning, pp. 805–813 (2015)
111. Srinivasan, S., Lanctot, M., Zambaldi, V., Pérolat, J., Tuyls, K., Munos, R., Bowling, M.: Actor-critic policy optimization in partially observable multiagent environments. In: Advances in Neural Information Processing Systems, pp. 3422–3435 (2018)
112. Omidshafiei, S., Hennes, D., Morrill, D., Munos, R., Perolat, J., Lanctot, M., Gruslys, A., Lespiau, J.B., Tuyls, K.: Neural replicator dynamics (2019). arXiv preprint [arXiv:1906.00190](https://arxiv.org/abs/1906.00190)
113. Rubin, J., Watson, I.: Computer Poker: a review. *Artif. Intell.* **175**(5–6), 958–987 (2011)
114. Lanctot, M., Lockhart, E., Lespiau, J.B., Zambaldi, V., Upadhyay, S., Pérolat, J., Srinivasan, S., Timbers, F., Tuyls, K., Omidshafiei, S., et al.: Openspiel: a framework for reinforcement learning in games (2019). arXiv preprint [arXiv:1908.09453](https://arxiv.org/abs/1908.09453)
115. Claus, C., Boutilier, C.: The dynamics of reinforcement learning in cooperative multiagent systems. In: AAAI Conference on Artificial Intelligence, vol. 1998, pp. 746–752, 2p. (1998)
116. Bowling, M., Veloso, M.: Rational and convergent learning in stochastic games. In: International Joint Conference on Artificial Intelligence, vol. 17, pp. 1021–1026 (2001)

117. Kapetanakis, S., Kudenko, D.: Reinforcement learning of coordination in cooperative multi-agent systems. In: AAAI Conference on Artificial Intelligence, vol. 2002, pp. 326–331 (2002)
118. Conitzer, V., Sandholm, T.: Awesome: a general multiagent learning algorithm that converges in self-play and learns a best response against stationary opponents. *Mach. Learn.* **67**(1–2), 23–43 (2007)
119. Hansen, E.A., Bernstein, D.S., Zilberstein, S.: Dynamic programming for partially observable stochastic games. In: AAAI Conference on Artificial Intelligence, pp. 709–715 (2004)
120. Amato, C., Chowdhary, G., Geramifard, A., Üre, N.K., Kochenderfer, M.J.: Decentralized control of partially observable Markov decision processes. In: IEEE Conference on Decision and Control, pp. 2398–2405 (2013)
121. Amato, C., Oliehoek, F.A.: Scalable planning and learning for multiagent POMDPs. In: AAAI Conference on Artificial Intelligence (2015)
122. Shoham, Y., Powers, R., Grenager, T.: Multi-agent reinforcement learning: a critical survey. Technical Report (2003)
123. Zinkevich, M., Greenwald, A., Littman, M.L.: Cyclic equilibria in Markov games. In: Advances in Neural Information Processing Systems, pp. 1641–1648 (2006)
124. Bowling, M., Veloso, M.: Multiagent learning using a variable learning rate. *Artif. Intell.* **136**(2), 215–250 (2002)
125. Bowling, M.: Convergence and no-regret in multiagent learning. In: Advances in Neural Information Processing Systems, pp. 209–216 (2005)
126. Blum, A., Mansour, Y.: Learning, regret minimization, and equilibria. In: Algorithmic Game Theory, pp. 79–102 (2007)
127. Hart, S., Mas-Colell, A.: A reinforcement procedure leading to correlated equilibrium. In: Economics Essays, pp. 181–200. Springer, Berlin (2001)
128. Kasai, T., Tenmoto, H., Kamiya, A.: Learning of communication codes in multi-agent reinforcement learning problem. In: IEEE Conference on Soft Computing in Industrial Applications, pp. 1–6 (2008)
129. Kim, D., Moon, S., Hostallero, D., Kang, W.J., Lee, T., Son, K., Yi, Y.: Learning to schedule communication in multi-agent reinforcement learning. In: International Conference on Learning Representations (2019)
130. Chen, T., Zhang, K., Giannakis, G.B., Başar, T.: Communication-efficient distributed reinforcement learning (2018). arXiv preprint [arXiv:1812.03239](https://arxiv.org/abs/1812.03239)
131. Lin, Y., Zhang, K., Yang, Z., Wang, Z., Başar, T., Sandhu, R., Liu, J.: A communication-efficient multi-agent actor-critic algorithm for distributed reinforcement learning. In: IEEE Conference on Decision and Control (2019)
132. Ren, J., Haupt, J.: A communication efficient hierarchical distributed optimization algorithm for multi-agent reinforcement learning. In: Real-World Sequential Decision Making Workshop at International Conference on Machine Learning (2019)
133. Kim, W., Cho, M., Sung, Y.: Message-dropout: an efficient training method for multi-agent deep reinforcement learning. In: AAAI Conference on Artificial Intelligence (2019)
134. He, H., Boyd-Graber, J., Kwok, K., Daumé III, H.: Opponent modeling in deep reinforcement learning. In: International Conference on Machine Learning, pp. 1804–1813 (2016)
135. Grover, A., Al-Shedivat, M., Gupta, J., Burda, Y., Edwards, H.: Learning policy representations in multiagent systems. In: International Conference on Machine Learning, pp. 1802–1811 (2018)
136. Gao, C., Mueller, M., Hayward, R.: Adversarial policy gradient for alternating Markov games. In: Workshop at International Conference on Learning Representations (2018)
137. Li, S., Wu, Y., Cui, X., Dong, H., Fang, F., Russell, S.: Robust multi-agent reinforcement learning via minimax deep deterministic policy gradient. In: AAAI Conference on Artificial Intelligence (2019)
138. Zhang, X., Zhang, K., Miehling, E., Basar, T.: Non-cooperative inverse reinforcement learning. In: Advances in Neural Information Processing Systems, pp. 9482–9493 (2019)
139. Tan, M.: Multi-agent reinforcement learning: Independent vs. cooperative agents. In: International Conference on Machine Learning, pp. 330–337 (1993)

140. Matignon, L., Laurent, G.J., Le Fort-Piat, N.: Independent reinforcement learners in cooperative Markov games: a survey regarding coordination problems. *Knowl. Eng. Rev.* **27**(1), 1–31 (2012)
141. Foerster, J., Nardelli, N., Farquhar, G., Torr, P., Kohli, P., Whiteson, S., et al.: Stabilising experience replay for deep multi-agent reinforcement learning. In: International Conference of Machine Learning, pp. 1146–1155 (2017)
142. Tuyls, K., Weiss, G.: Multiagent learning: basics, challenges, and prospects. *AI Mag.* **33**(3), 41 (2012)
143. Guestrin, C., Lagoudakis, M., Parr, R.: Coordinated reinforcement learning. In: International Conference on Machine Learning, pp. 227–234 (2002)
144. Guestrin, C., Koller, D., Parr, R.: Multiagent planning with factored MDPs. In: Advances in Neural Information Processing Systems, pp. 1523–1530 (2002)
145. Kok, J.R., Vlassis, N.: Sparse cooperative Q-learning. In: International Conference on Machine learning, pp. 61–69 (2004)
146. Sunehag, P., Lever, G., Gruslys, A., Czarnecki, W.M., Zambaldi, V., Jaderberg, M., Lanctot, M., Sonnerat, N., Leibo, J.Z., Tuyls, K., et al.: Value-decomposition networks for cooperative multi-agent learning based on team reward. In: International Conference on Autonomous Agents and Multi-Agent Systems, pp. 2085–2087 (2018)
147. Rashid, T., Samvelyan, M., De Witt, C.S., Farquhar, G., Foerster, J., Whiteson, S.: QMIX: monotonic value function factorisation for deep multi-agent reinforcement learning. In: International Conference on Machine Learning, pp. 681–689 (2018)
148. Qu, G., Li, N.: Exploiting fast decaying and locality in multi-agent MDP with tree dependence structure. In: IEEE Conference on Decision and Control (2019)
149. Mahajan, A.: Optimal decentralized control of coupled subsystems with control sharing. *IEEE Trans. Autom. Control* **58**(9), 2377–2382 (2013)
150. Oliehoek, F.A., Amato, C.: Dec-POMDPs as non-observable MDPs. IAS Technical Report (IAS-UVA-14-01) (2014)
151. Foerster, J.N., Farquhar, G., Afouras, T., Nardelli, N., Whiteson, S.: Counterfactual multi-agent policy gradients. In: AAAI Conference on Artificial Intelligence (2018)
152. Dibangoye, J., Buffet, O.: Learning to act in decentralized partially observable MDPs. In: International Conference on Machine Learning, pp. 1233–1242 (2018)
153. Kraemer, L., Banerjee, B.: Multi-agent reinforcement learning as a rehearsal for decentralized planning. *Neurocomputing* **190**, 82–94 (2016)
154. Macua, S.V., Chen, J., Zazo, S., Sayed, A.H.: Distributed policy evaluation under multiple behavior strategies. *IEEE Trans. Autom. Control* **60**(5), 1260–1274 (2015)
155. Macua, S.V., Tukiainen, A., Hernández, D.G.O., Baldazo, D., de Cote, E.M., Zazo, S.: Diff-dac: Distributed actor-critic for average multitask deep reinforcement learning (2017). arXiv preprint [arXiv:1710.10363](https://arxiv.org/abs/1710.10363)
156. Lee, D., Yoon, H., Hovakimyan, N.: Primal-dual algorithm for distributed reinforcement learning: distributed GTD. In: IEEE Conference on Decision and Control, pp. 1967–1972 (2018)
157. Doan, T.T., Maguluri, S.T., Romberg, J.: Finite-time performance of distributed temporal difference learning with linear function approximation (2019). arXiv preprint [arXiv:1907.12530](https://arxiv.org/abs/1907.12530)
158. Suttle, W., Yang, Z., Zhang, K., Wang, Z., Başar, T., Liu, J.: A multi-agent off-policy actor-critic algorithm for distributed reinforcement learning (2019). arXiv preprint [arXiv:1903.06372](https://arxiv.org/abs/1903.06372)
159. Littman, M.L.: Value-function reinforcement learning in Markov games. *Cogn. Syst. Res.* **2**(1), 55–66 (2001)
160. Young, H.P.: The evolution of conventions. *Econ.: J. Econ. Soc.* 57–84 (1993)
161. Son, K., Kim, D., Kang, W.J., Hostallero, D.E., Yi, Y.: QTRAN: Learning to factorize with transformation for cooperative multi-agent reinforcement learning. In: International Conference on Machine Learning, pp. 5887–5896 (2019)
162. Perolat, J., Piot, B., Pietquin, O.: Actor-critic fictitious play in simultaneous move multistage games. In: International Conference on Artificial Intelligence and Statistics (2018)

163. Monderer, D., Shapley, L.S.: Potential games. *Games Econ. Behav.* **14**(1), 124–143 (1996)
164. Başar, T., Zaccour, G.: *Handbook of Dynamic Game Theory*. Springer, Berlin (2018)
165. Huang, M., Caines, P.E., Malhamé, R.P.: Individual and mass behaviour in large population stochastic wireless power control problems: centralized and Nash equilibrium solutions. In: IEEE Conference on Decision and Control, pp. 98–103 (2003)
166. Huang, M., Malhamé, R.P., Caines, P.E., et al.: Large population stochastic dynamic games: closed-loop McKean-Vlasov systems and the Nash certainty equivalence principle. *Commun. Inf. Syst.* **6**(3), 221–252 (2006)
167. Lasry, J.M., Lions, P.L.: Mean field games. *Jpn. J. Math.* **2**(1), 229–260 (2007)
168. Bensoussan, A., Frehse, J., Yam, P., et al.: *Mean Field Games and Mean Field Type Control Theory*, vol. 101. Springer, Berlin (2013)
169. Tembine, H., Zhu, Q., Başar, T.: Risk-sensitive mean-field games. *IEEE Trans. Autom. Control* **59**(4), 835–850 (2013)
170. Arabneydi, J., Mahajan, A.: Team optimal control of coupled subsystems with mean-field sharing. In: IEEE Conference on Decision and Control, pp. 1669–1674 (2014)
171. Arabneydi, J.: New concepts in team theory: Mean field teams and reinforcement learning. Ph.D. thesis, McGill University (2017)
172. Yang, Y., Luo, R., Li, M., Zhou, M., Zhang, W., Wang, J.: Mean field multi-agent reinforcement learning. In: International Conference on Machine Learning, pp. 5571–5580 (2018)
173. Witsenhausen, H.S.: Separation of estimation and control for discrete time systems. *Proc. IEEE* **59**(11), 1557–1566 (1971)
174. Yüksel, S., Başar, T.: *Stochastic Networked Control Systems: Stabilization and Optimization Under Information Constraints*. Springer Science & Business Media, Berlin (2013)
175. Subramanian, J., Seraj, R., Mahajan, A.: Reinforcement learning for mean-field teams. In: Workshop on Adaptive and Learning Agents at International Conference on Autonomous Agents and Multi-Agent Systems (2018)
176. Arabneydi, J., Mahajan, A.: Linear quadratic mean field teams: optimal and approximately optimal decentralized solutions (2016). arXiv preprint [arXiv:1609.00056](https://arxiv.org/abs/1609.00056)
177. Carmona, R., Laurière, M., Tan, Z.: Linear-quadratic mean-field reinforcement learning: convergence of policy gradient methods (2019). arXiv preprint [arXiv:1910.04295](https://arxiv.org/abs/1910.04295)
178. Carmona, R., Laurière, M., Tan, Z.: Model-free mean-field reinforcement learning: mean-field MDP and mean-field Q-learning (2019). arXiv preprint [arXiv:1910.12802](https://arxiv.org/abs/1910.12802)
179. Rabbat, M., Nowak, R.: Distributed optimization in sensor networks. In: International Symposium on Information Processing in Sensor Networks, pp. 20–27 (2004)
180. Dall’Anese, E., Zhu, H., Giannakis, G.B.: Distributed optimal power flow for smart microgrids. *IEEE Trans. Smart Grid* **4**(3), 1464–1475 (2013)
181. Zhang, K., Shi, W., Zhu, H., Dall’Anese, E., Başar, T.: Dynamic power distribution system management with a locally connected communication network. *IEEE J. Sel. Top. Signal Process.* **12**(4), 673–687 (2018)
182. Zhang, K., Lu, L., Lei, C., Zhu, H., Ouyang, Y.: Dynamic operations and pricing of electric unmanned aerial vehicle systems and power networks. *Transp. Res. Part C: Emerg. Technol.* **92**, 472–485 (2018)
183. Corke, P., Peterson, R., Rus, D.: Networked robots: flying robot navigation using a sensor net. *Robot. Res.* 234–243 (2005)
184. Zhang, K., Liu, Y., Liu, J., Liu, M., Başar, T.: Distributed learning of average belief over networks using sequential observations. *Automatica* (2019)
185. Nedic, A., Ozdaglar, A.: Distributed subgradient methods for multi-agent optimization. *IEEE Trans. Autom. Control* **54**(1), 48–61 (2009)
186. Agarwal, A., Duchi, J.C.: Distributed delayed stochastic optimization. In: Advances in Neural Information Processing Systems, pp. 873–881 (2011)
187. Jakovetic, D., Xavier, J., Moura, J.M.: Cooperative convex optimization in networked systems: augmented Lagrangian algorithms with directed gossip communication. *IEEE Trans. Signal Process.* **59**(8), 3889–3902 (2011)

188. Tu, S.Y., Sayed, A.H.: Diffusion strategies outperform consensus strategies for distributed estimation over adaptive networks. *IEEE Trans. Signal Process.* **60**(12), 6217–6234 (2012)
189. Varshavskaya, P., Kaelbling, L.P., Rus, D.: Efficient distributed reinforcement learning through agreement. In: *Distributed Autonomous Robotic Systems*, pp. 367–378 (2009)
190. Ciosek, K., Whiteson, S.: Expected policy gradients for reinforcement learning (2018). arXiv preprint [arXiv:1801.03326](https://arxiv.org/abs/1801.03326)
191. Sutton, R.S., Mahmood, A.R., White, M.: An emphatic approach to the problem of off-policy temporal-difference learning. *J. Mach. Learn. Res.* **17**(1), 2603–2631 (2016)
192. Yu, H.: On convergence of emphatic temporal-difference learning. In: *Conference on Learning Theory*, pp. 1724–1751 (2015)
193. Zhang, Y., Zavlanos, M.M.: Distributed off-policy actor-critic reinforcement learning with policy consensus (2019). arXiv preprint [arXiv:1903.09255](https://arxiv.org/abs/1903.09255)
194. Pennesi, P., Paschalidis, I.C.: A distributed actor-critic algorithm and applications to mobile sensor network coordination problems. *IEEE Trans. Autom. Control* **55**(2), 492–497 (2010)
195. Lange, S., Gabel, T., Riedmiller, M.: Batch reinforcement learning. In: *Reinforcement Learning*, pp. 45–73. Springer, Berlin (2012)
196. Riedmiller, M.: Neural fitted Q iteration–first experiences with a data efficient neural reinforcement learning method. In: *European Conference on Machine Learning*, pp. 317–328 (2005)
197. Antos, A., Szepesvári, C., Munos, R.: Fitted Q-iteration in continuous action-space MDPs. In: *Advances in Neural Information Processing Systems*, pp. 9–16 (2008)
198. Hong, M., Chang, T.H.: Stochastic proximal gradient consensus over random networks. *IEEE Trans. Signal Process.* **65**(11), 2933–2948 (2017)
199. Nedic, A., Olshevsky, A., Shi, W.: Achieving geometric convergence for distributed optimization over time-varying graphs. *SIAM J. Optim.* **27**(4), 2597–2633 (2017)
200. Munos, R.: Performance bounds in  $\ell_p$ -norm for approximate value iteration. *SIAM J. Control Optim.* **46**(2), 541–561 (2007)
201. Munos, R., Szepesvári, C.: Finite-time bounds for fitted value iteration. *J. Mach. Learn. Res.* **9**(May), 815–857 (2008)
202. Antos, A., Szepesvári, C., Munos, R.: Learning near-optimal policies with Bellman-residual minimization based fitted policy iteration and a single sample path. *Mach. Learn.* **71**(1), 89–129 (2008)
203. Farahmand, A.M., Szepesvári, C., Munos, R.: Error propagation for approximate policy and value iteration. In: *Advances in Neural Information Processing Systems*, pp. 568–576 (2010)
204. Cassano, L., Yuan, K., Sayed, A.H.: Multi-agent fully decentralized off-policy learning with linear convergence rates (2018). arXiv preprint [arXiv:1810.07792](https://arxiv.org/abs/1810.07792)
205. Qu, G., Li, N.: Harnessing smoothness to accelerate distributed optimization. *IEEE Trans. Control Netw. Syst.* **5**(3), 1245–1260 (2017)
206. Schmidt, M., Le Roux, N., Bach, F.: Minimizing finite sums with the stochastic average gradient. *Math. Program.* **162**(1–2), 83–112 (2017)
207. Ying, B., Yuan, K., Sayed, A.H.: Convergence of variance-reduced learning under random reshuffling. In: *IEEE International Conference on Acoustics, Speech and Signal Processing*, pp. 2286–2290 (2018)
208. Singh, S.P., Sutton, R.S.: Reinforcement learning with replacing eligibility traces. *Mach. Learn.* **22**(1–3), 123–158 (1996)
209. Bhandari, J., Russo, D., Singal, R.: A finite time analysis of temporal difference learning with linear function approximation. In: *Conference On Learning Theory*, pp. 1691–1692 (2018)
210. Srikant, R., Ying, L.: Finite-time error bounds for linear stochastic approximation and TD learning. In: *Conference on Learning Theory*, pp. 2803–2830 (2019)
211. Stanković, M.S., Stanković, S.S.: Multi-agent temporal-difference learning with linear function approximation: weak convergence under time-varying network topologies. In: *IEEE American Control Conference*, pp. 167–172 (2016)
212. Stanković, M.S., Ilić, N., Stanković, S.S.: Distributed stochastic approximation: weak convergence and network design. *IEEE Trans. Autom. Control* **61**(12), 4069–4074 (2016)

213. Zhang, H., Jiang, H., Luo, Y., Xiao, G.: Data-driven optimal consensus control for discrete-time multi-agent systems with unknown dynamics using reinforcement learning method. *IEEE Trans. Ind. Electron.* **64**(5), 4091–4100 (2016)
214. Zhang, Q., Zhao, D., Lewis, F.L.: Model-free reinforcement learning for fully cooperative multi-agent graphical games. In: International Joint Conference on Neural Networks, pp. 1–6 (2018)
215. Bernstein, D.S., Amato, C., Hansen, E.A., Zilberstein, S.: Policy iteration for decentralized control of Markov decision processes. *J. Artif. Intell. Res.* **34**, 89–132 (2009)
216. Amato, C., Bernstein, D.S., Zilberstein, S.: Optimizing fixed-size stochastic controllers for POMDPs and decentralized POMDPs. *Auton. Agents Multi-Agent Syst.* **21**(3), 293–320 (2010)
217. Liu, M., Amato, C., Liao, X., Carin, L., How, J.P.: Stick-breaking policy learning in Dec-POMDPs. In: International Joint Conference on Artificial Intelligence (2015)
218. Dibangoye, J.S., Amato, C., Buffet, O., Charillet, F.: Optimally solving Dec-POMDPs as continuous-state MDPs. *J. Artif. Intell. Res.* **55**, 443–497 (2016)
219. Wu, F., Zilberstein, S., Chen, X.: Rollout sampling policy iteration for decentralized POMDPs. In: Conference on Uncertainty in Artificial Intelligence (2010)
220. Wu, F., Zilberstein, S., Jennings, N.R.: Monte-Carlo expectation maximization for decentralized POMDPs. In: International Joint Conference on Artificial Intelligence (2013)
221. Best, G., Cliff, O.M., Patten, T., Mettu, R.R., Fitch, R.: Dec-MCTS: decentralized planning for multi-robot active perception. *Int. J. Robot. Res.* 1–22 (2018)
222. Amato, C., Zilberstein, S.: Achieving goals in decentralized POMDPs. In: International Conference on Autonomous Agents and Multi-Agent Systems, pp. 593–600 (2009)
223. Banerjee, B., Lyle, J., Kraemer, L., Yellamraju, R.: Sample bounded distributed reinforcement learning for decentralized POMDPs. In: AAAI Conference on Artificial Intelligence (2012)
224. Nayyar, A., Mahajan, A., Teneketzis, D.: Decentralized stochastic control with partial history sharing: a common information approach. *IEEE Trans. Autom. Control* **58**(7), 1644–1658 (2013)
225. Arabneydi, J., Mahajan, A.: Reinforcement learning in decentralized stochastic control systems with partial history sharing. In: IEEE American Control Conference, pp. 5449–5456 (2015)
226. Papadimitriou, C.H.: On inefficient proofs of existence and complexity classes. In: *Annals of Discrete Mathematics*, vol. 51, pp. 245–250. Elsevier (1992)
227. Daskalakis, C., Goldberg, P.W., Papadimitriou, C.H.: The complexity of computing a Nash equilibrium. *SIAM J. Comput.* **39**(1), 195–259 (2009)
228. Von Neumann, J., Morgenstern, O., Kuhn, H.W.: *Theory of Games and Economic Behavior* (commemorative edition). Princeton University Press, Princeton (2007)
229. Vanderbei, R.J., et al.: *Linear Programming*. Springer, Berlin (2015)
230. Hoffman, A.J., Karp, R.M.: On nonterminating stochastic games. *Manag. Sci.* **12**(5), 359–370 (1966)
231. Van Der Wal, J.: Discounted Markov games: generalized policy iteration method. *J. Optim. Theory Appl.* **25**(1), 125–138 (1978)
232. Rao, S.S., Chandrasekaran, R., Nair, K.: Algorithms for discounted stochastic games. *J. Optim. Theory Appl.* **11**(6), 627–637 (1973)
233. Patek, S.D.: Stochastic and shortest path games: theory and algorithms. Ph.D. thesis, Massachusetts Institute of Technology (1997)
234. Hansen, T.D., Miltersen, P.B., Zwick, U.: Strategy iteration is strongly polynomial for 2-player turn-based stochastic games with a constant discount factor. *J. ACM* **60**(1), 1 (2013)
235. Lagoudakis, M.G., Parr, R.: Value function approximation in zero-sum Markov games. In: Conference on Uncertainty in Artificial Intelligence, pp. 283–292 (2002)
236. Zou, S., Xu, T., Liang, Y.: Finite-sample analysis for SARSA with linear function approximation (2019). arXiv preprint [arXiv:1902.02234](https://arxiv.org/abs/1902.02234)
237. Sutton, R.S., Barto, A.G.: A temporal-difference model of classical conditioning. In: Proceedings of the Annual Conference of the Cognitive Science Society, pp. 355–378 (1987)

238. Al-Tamimi, A., Abu-Khalaf, M., Lewis, F.L.: Adaptive critic designs for discrete-time zero-sum games with application to  $\mathcal{H}_\infty$  control. *IEEE Trans. Syst. Man Cybern. Part B* **37**(1), 240–247 (2007)
239. Al-Tamimi, A., Lewis, F.L., Abu-Khalaf, M.: Model-free Q-learning designs for linear discrete-time zero-sum games with application to  $\mathcal{H}_\infty$  control. *Automatica* **43**(3), 473–481 (2007)
240. Farahmand, A.M., Ghavamzadeh, M., Szepesvári, C., Mannor, S.: Regularized policy iteration with nonparametric function spaces. *J. Mach. Learn. Res.* **17**(1), 4809–4874 (2016)
241. Yang, Z., Xie, Y., Wang, Z.: A theoretical analysis of deep Q-learning (2019). arXiv preprint [arXiv:1901.00137](https://arxiv.org/abs/1901.00137)
242. Jia, Z., Yang, L.F., Wang, M.: Feature-based Q-learning for two-player stochastic games (2019). arXiv preprint [arXiv:1906.00423](https://arxiv.org/abs/1906.00423)
243. Sidford, A., Wang, M., Wu, X., Yang, L., Ye, Y.: Near-optimal time and sample complexities for solving Markov decision processes with a generative model. In: Advances in Neural Information Processing Systems, pp. 5186–5196 (2018)
244. Wei, C.Y., Hong, Y.T., Lu, C.J.: Online reinforcement learning in stochastic games. In: Advances in Neural Information Processing Systems, pp. 4987–4997 (2017)
245. Auer, P., Ortner, R.: Logarithmic online regret bounds for undiscounted reinforcement learning. In: Advances in Neural Information Processing Systems, pp. 49–56 (2007)
246. Jaksch, T., Ortner, R., Auer, P.: Near-optimal regret bounds for reinforcement learning. *J. Mach. Learn. Res.* **11**, 1563–1600 (2010)
247. Koller, D., Megiddo, N., von Stengel, B.: Fast algorithms for finding randomized strategies in game trees. *Computing* **750**, 759 (1994)
248. Von Stengel, B.: Efficient computation of behavior strategies. *Games Econ. Behav.* **14**(2), 220–246 (1996)
249. Koller, D., Megiddo, N., Von Stengel, B.: Efficient computation of equilibria for extensive two-person games. *Games Econ. Behav.* **14**(2), 247–259 (1996)
250. Von Stengel, B.: Computing equilibria for two-person games. *Handbook of Game Theory with Economic Applications* **3**, 1723–1759 (2002)
251. Parr, R., Russell, S.: Approximating optimal policies for partially observable stochastic domains. In: International Joint Conference on Artificial Intelligence, pp. 1088–1094 (1995)
252. Rodriguez, A.C., Parr, R., Koller, D.: Reinforcement learning using approximate belief states. In: Advances in Neural Information Processing Systems, pp. 1036–1042 (2000)
253. Hauskrecht, M.: Value-function approximations for partially observable Markov decision processes. *J. Artif. Intell. Res.* **13**, 33–94 (2000)
254. Buter, B.J.: Dynamic programming for extensive form games with imperfect information. Ph.D. thesis, Universiteit van Amsterdam (2012)
255. Cowling, P.I., Powley, E.J., Whitehouse, D.: Information set Monte Carlo tree search. *IEEE Trans. Comput. Intell. AI Games* **4**(2), 120–143 (2012)
256. Teraoka, K., Hatano, K., Takimoto, E.: Efficient sampling method for Monte Carlo tree search problem. *IEICE Trans. Inf. Syst.* **97**(3), 392–398 (2014)
257. Whitehouse, D.: Monte Carlo tree search for games with hidden information and uncertainty. Ph.D. thesis, University of York (2014)
258. Kaufmann, E., Koolen, W.M.: Monte-Carlo tree search by best arm identification. In: Advances in Neural Information Processing Systems, pp. 4897–4906 (2017)
259. Hannan, J.: Approximation to Bayes risk in repeated play. *Contrib. Theory Games* **3**, 97–139 (1957)
260. Brown, G.W.: Iterative solution of games by fictitious play. *Act. Anal. Prod. Allo.* **13**(1), 374–376 (1951)
261. Robinson, J.: An iterative method of solving a game. *Ann. Math.* 296–301 (1951)
262. Benaïm, M., Hofbauer, J., Sorin, S.: Stochastic approximations and differential inclusions. *SIAM J. Control Optim.* **44**(1), 328–348 (2005)
263. Hart, S., Mas-Colell, A.: A general class of adaptive strategies. *J. Econ. Theory* **98**(1), 26–54 (2001)

264. Monderer, D., Samet, D., Sela, A.: Belief affirming in learning processes. *J. Econ. Theory* **73**(2), 438–452 (1997)
265. Viossat, Y., Zapechelnyuk, A.: No-regret dynamics and fictitious play. *J. Econ. Theory* **148**(2), 825–842 (2013)
266. Kushner, H.J., Yin, G.G.: Stochastic Approximation and Recursive Algorithms and Applications. Springer, New York (2003)
267. Fudenberg, D., Levine, D.K.: Consistency and cautious fictitious play. *J. Econ. Dyn. Control* **19**(5–7), 1065–1089 (1995)
268. Hofbauer, J., Sandholm, W.H.: On the global convergence of stochastic fictitious play. *Econometrica* **70**(6), 2265–2294 (2002)
269. Leslie, D.S., Collins, E.J.: Generalised weakened fictitious play. *Games Econ. Behav.* **56**(2), 285–298 (2006)
270. Benaim, M., Faure, M.: Consistency of vanishingly smooth fictitious play. *Math. Oper. Res.* **38**(3), 437–450 (2013)
271. Li, Z., Tewari, A.: Sampled fictitious play is Hannan consistent. *Games Econ. Behav.* **109**, 401–412 (2018)
272. Ernst, D., Geurts, P., Wehenkel, L.: Tree-based batch mode reinforcement learning. *J. Mach. Learn. Res.* **6**(Apr), 503–556 (2005)
273. Heinrich, J., Silver, D.: Self-play Monte-Carlo tree search in computer Poker. In: Workshops at AAAI Conference on Artificial Intelligence (2014)
274. Browne, C.B., Powley, E., Whitehouse, D., Lucas, S.M., Cowling, P.I., Rohlfsen, P., Tavener, S., Perez, D., Samothrakis, S., Colton, S.: A survey of Monte Carlo tree search methods. *IEEE Trans. Comput. Intell. AI Games* **4**(1), 1–43 (2012)
275. Borkar, V.S.: Stochastic Approximation: A Dynamical Systems Viewpoint. Cambridge University Press, Cambridge (2008)
276. Cesa-Bianchi, N., Lugosi, G.: Prediction, Learning, and Games. Cambridge University Press, Cambridge (2006)
277. Auer, P., Cesa-Bianchi, N., Freund, Y., Schapire, R.E.: The nonstochastic multiarmed bandit problem. *SIAM J. Comput.* **32**(1), 48–77 (2002)
278. Vovk, V.G.: Aggregating strategies. In: Proceedings of Computational Learning Theory (1990)
279. Littlestone, N., Warmuth, M.K.: The weighted majority algorithm. *Inf. Comput.* **108**(2), 212–261 (1994)
280. Freund, Y., Schapire, R.E.: Adaptive game playing using multiplicative weights. *Games Econ. Behav.* **29**(1–2), 79–103 (1999)
281. Hart, S., Mas-Colell, A.: A simple adaptive procedure leading to correlated equilibrium. *Econometrica* **68**(5), 1127–1150 (2000)
282. Lanctot, M., Waugh, K., Zinkevich, M., Bowling, M.: Monte Carlo sampling for regret minimization in extensive games. In: Advances in Neural Information Processing Systems, pp. 1078–1086 (2009)
283. Burch, N., Lanctot, M., Szafron, D., Gibson, R.G.: Efficient Monte Carlo counterfactual regret minimization in games with many player actions. In: Advances in Neural Information Processing Systems, pp. 1880–1888 (2012)
284. Gibson, R., Lanctot, M., Burch, N., Szafron, D., Bowling, M.: Generalized sampling and variance in counterfactual regret minimization. In: AAAI Conference on Artificial Intelligence (2012)
285. Johanson, M., Bard, N., Lanctot, M., Gibson, R., Bowling, M.: Efficient Nash equilibrium approximation through Monte Carlo counterfactual regret minimization. In: International Conference on Autonomous Agents and Multi-Agent Systems, pp. 837–846 (2012)
286. Lisý, V., Lanctot, M., Bowling, M.: Online Monte Carlo counterfactual regret minimization for search in imperfect information games. In: International Conference on Autonomous Agents and Multi-Agent Systems, pp. 27–36 (2015)
287. Schmid, M., Burch, N., Lanctot, M., Moravcik, M., Kadlec, R., Bowling, M.: Variance reduction in Monte Carlo counterfactual regret minimization (VR-MCCFR) for extensive form games using baselines. In: AAAI Conference on Artificial Intelligence, vol. 33, pp. 2157–2164 (2019)

288. Waugh, K., Morrill, D., Bagnell, J.A., Bowling, M.: Solving games with functional regret estimation. In: AAAI Conference on Artificial Intelligence (2015)
289. Morrill, D.: Using regret estimation to solve games compactly. Ph.D. thesis, University of Alberta (2016)
290. Brown, N., Lerer, A., Gross, S., Sandholm, T.: Deep counterfactual regret minimization. In: International Conference on Machine Learning, pp. 793–802 (2019)
291. Brown, N., Sandholm, T.: Regret-based pruning in extensive-form games. In: Advances in Neural Information Processing Systems, pp. 1972–1980 (2015)
292. Brown, N., Kroer, C., Sandholm, T.: Dynamic thresholding and pruning for regret minimization. In: AAAI Conference on Artificial Intelligence (2017)
293. Brown, N., Sandholm, T.: Reduced space and faster convergence in imperfect-information games via pruning. In: International Conference on Machine Learning, pp. 596–604 (2017)
294. Tammelin, O.: Solving large imperfect information games using CFR+ (2014). arXiv preprint [arXiv:1407.5042](https://arxiv.org/abs/1407.5042)
295. Tammelin, O., Burch, N., Johanson, M., Bowling, M.: Solving heads-up limit Texas Hold’em. In: International Joint Conference on Artificial Intelligence (2015)
296. Burch, N., Moravcik, M., Schmid, M.: Revisiting CFR+ and alternating updates. *J. Artif. Intell. Res.* **64**, 429–443 (2019)
297. Zhou, Y., Ren, T., Li, J., Yan, D., Zhu, J.: Lazy-CFR: a fast regret minimization algorithm for extensive games with imperfect information (2018). arXiv preprint [arXiv:1810.04433](https://arxiv.org/abs/1810.04433)
298. Zinkevich, M.: Online convex programming and generalized infinitesimal gradient ascent. In: International Conference on Machine Learning, pp. 928–936 (2003)
299. Lockhart, E., Lanctot, M., Pérolat, J., Lespiau, J.B., Morrill, D., Timbers, F., Tuyls, K.: Computing approximate equilibria in sequential adversarial games by exploitability descent (2019). arXiv preprint [arXiv:1903.05614](https://arxiv.org/abs/1903.05614)
300. Johanson, M., Bard, N., Burch, N., Bowling, M.: Finding optimal abstract strategies in extensive-form games. In: AAAI Conference on Artificial Intelligence, pp. 1371–1379 (2012)
301. Schaeffer, M.S., Sturtevant, N., Schaeffer, J.: Comparing UCT versus CFR in simultaneous games (2009)
302. Lanctot, M., Lisý, V., Winands, M.H.: Monte Carlo tree search in simultaneous move games with applications to Goofspiel. In: Workshop on Computer Games, pp. 28–43 (2013)
303. Lisý, V., Kovářík, V., Lanctot, M., Bošanský, B.: Convergence of Monte Carlo tree search in simultaneous move games. In: Advances in Neural Information Processing Systems, pp. 2112–2120 (2013)
304. Tak, M.J., Lanctot, M., Winands, M.H.: Monte Carlo tree search variants for simultaneous move games. In: IEEE Conference on Computational Intelligence and Games, pp. 1–8 (2014)
305. Kovářík, V., Lisý, V.: Analysis of Hannan consistent selection for Monte Carlo tree search in simultaneous move games (2018). arXiv preprint [arXiv:1804.09045](https://arxiv.org/abs/1804.09045)
306. Mazumdar, E.V., Jordan, M.I., Sastry, S.S.: On finding local Nash equilibria (and only local Nash equilibria) in zero-sum games (2019). arXiv preprint [arXiv:1901.00838](https://arxiv.org/abs/1901.00838)
307. Bu, J., Ratliff, L.J., Mesbahi, M.: Global convergence of policy gradient for sequential zero-sum linear quadratic dynamic games (2019). arXiv preprint [arXiv:1911.04672](https://arxiv.org/abs/1911.04672)
308. Mescheder, L., Nowozin, S., Geiger, A.: The numerics of GANs. In: Advances in Neural Information Processing Systems, pp. 1825–1835 (2017)
309. Adolphs, L., Daneshmand, H., Lucchi, A., Hofmann, T.: Local saddle point optimization: a curvature exploitation approach (2018). arXiv preprint [arXiv:1805.05751](https://arxiv.org/abs/1805.05751)
310. Daskalakis, C., Panageas, I.: The limit points of (optimistic) gradient descent in min-max optimization. In: Advances in Neural Information Processing Systems, pp. 9236–9246 (2018)
311. Mertikopoulos, P., Zenati, H., Lecouat, B., Foo, C.S., Chandrasekhar, V., Piliouras, G.: Optimistic mirror descent in saddle-point problems: going the extra (gradient) mile. In: International Conference on Learning Representations (2019)
312. Fiez, T., Chasnov, B., Ratliff, L.J.: Convergence of learning dynamics in Stackelberg games (2019). arXiv preprint [arXiv:1906.01217](https://arxiv.org/abs/1906.01217)

313. Balduzzi, D., Racaniere, S., Martens, J., Foerster, J., Tuyls, K., Graepel, T.: The mechanics of n-player differentiable games. In: International Conference on Machine Learning, pp. 363–372 (2018)
314. Sanjabi, M., Razaviyayn, M., Lee, J.D.: Solving non-convex non-concave min-max games under Polyak-Łojasiewicz condition (2018). arXiv preprint [arXiv:1812.02878](https://arxiv.org/abs/1812.02878)
315. Nouiehed, M., Sanjabi, M., Lee, J.D., Razaviyayn, M.: Solving a class of non-convex min-max games using iterative first order methods (2019). arXiv preprint [arXiv:1902.08297](https://arxiv.org/abs/1902.08297)
316. Mazumdar, E., Ratliff, L.J., Jordan, M.I., Sastry, S.S.: Policy-gradient algorithms have no guarantees of convergence in continuous action and state multi-agent settings (2019). arXiv preprint [arXiv:1907.03712](https://arxiv.org/abs/1907.03712)
317. Chen, X., Deng, X., Teng, S.H.: Settling the complexity of computing two-player Nash equilibria. *J. ACM* **56**(3), 14 (2009)
318. Greenwald, A., Hall, K., Serrano, R.: Correlated Q-learning. In: International Conference on Machine Learning, pp. 242–249 (2003)
319. Aumann, R.J.: Subjectivity and correlation in randomized strategies. *J. Math. Econ.* **1**(1), 67–96 (1974)
320. Perolat, J., Strub, F., Piot, B., Pietquin, O.: Learning Nash equilibrium for general-sum Markov games from batch data. In: International Conference on Artificial Intelligence and Statistics, pp. 232–241 (2017)
321. Maillard, O.A., Munos, R., Lazaric, A., Ghavamzadeh, M.: Finite-sample analysis of Bellman residual minimization. In: Asian Conference on Machine Learning, pp. 299–314 (2010)
322. Letcher, A., Balduzzi, D., Racanière, S., Martens, J., Foerster, J.N., Tuyls, K., Graepel, T.: Differentiable game mechanics. *J. Mach. Learn. Res.* **20**(84), 1–40 (2019)
323. Chasnov, B., Ratliff, L.J., Mazumdar, E., Burden, S.A.: Convergence analysis of gradient-based learning with non-uniform learning rates in non-cooperative multi-agent settings (2019). arXiv preprint [arXiv:1906.00731](https://arxiv.org/abs/1906.00731)
324. Hart, S., Mas-Colell, A.: Uncoupled dynamics do not lead to Nash equilibrium. *Am. Econ. Rev.* **93**(5), 1830–1836 (2003)
325. Saldi, N., Başar, T., Raginsky, M.: Markov-Nash equilibria in mean-field games with discounted cost. *SIAM J. Control Optim.* **56**(6), 4256–4287 (2018)
326. Saldi, N., Başar, T., Raginsky, M.: Approximate Nash equilibria in partially observed stochastic games with mean-field interactions. *Math. Oper. Res.* (2019)
327. Saldi, N.: Discrete-time average-cost mean-field games on Polish spaces (2019). arXiv preprint [arXiv:1908.08793](https://arxiv.org/abs/1908.08793)
328. Saldi, N., Başar, T., Raginsky, M.: Discrete-time risk-sensitive mean-field games (2018). arXiv preprint [arXiv:1808.03929](https://arxiv.org/abs/1808.03929)
329. Guo, X., Hu, A., Xu, R., Zhang, J.: Learning mean-field games (2019). arXiv preprint [arXiv:1901.09585](https://arxiv.org/abs/1901.09585)
330. Fu, Z., Yang, Z., Chen, Y., Wang, Z.: Actor-critic provably finds Nash equilibria of linear-quadratic mean-field games (2019). arXiv preprint [arXiv:1910.07498](https://arxiv.org/abs/1910.07498)
331. Hadikhanloo, S., Silva, F.J.: Finite mean field games: fictitious play and convergence to a first order continuous mean field game. *J. Math. Pures Appl.* (2019)
332. Elie, R., Pérolat, J., Laurière, M., Geist, M., Pietquin, O.: Approximate fictitious play for mean field games (2019). arXiv preprint [arXiv:1907.02633](https://arxiv.org/abs/1907.02633)
333. Anahtarci, B., Kariksiz, C.D., Saldi, N.: Value iteration algorithm for mean-field games (2019). arXiv preprint [arXiv:1909.01758](https://arxiv.org/abs/1909.01758)
334. Zaman, M.A.u., Zhang, K., Miehling, E., Başar, T.: Approximate equilibrium computation for discrete-time linear-quadratic mean-field games. Submitted to IEEE American Control Conference (2020)
335. Yang, B., Liu, M.: Keeping in touch with collaborative UAVs: a deep reinforcement learning approach. In: International Joint Conference on Artificial Intelligence, pp. 562–568 (2018)
336. Pham, H.X., La, H.M., Feil-Seifer, D., Nefian, A.: Cooperative and distributed reinforcement learning of drones for field coverage (2018). arXiv preprint [arXiv:1803.07250](https://arxiv.org/abs/1803.07250)

337. Tožička, J., Szulyovszky, B., de Chambrier, G., Sarwal, V., Wani, U., Gribulis, M.: Application of deep reinforcement learning to UAV fleet control. In: SAI Intelligent Systems Conference, pp. 1169–1177 (2018)
338. Shamsoshoara, A., Khaledi, M., Afghah, F., Razi, A., Ashdown, J.: Distributed cooperative spectrum sharing in UAV networks using multi-agent reinforcement learning. In: IEEE Annual Consumer Communications & Networking Conference, pp. 1–6 (2019)
339. Cui, J., Liu, Y., Nallanathan, A.: The application of multi-agent reinforcement learning in UAV networks. In: IEEE International Conference on Communications Workshops, pp. 1–6 (2019)
340. Qie, H., Shi, D., Shen, T., Xu, X., Li, Y., Wang, L.: Joint optimization of multi-UAV target assignment and path planning based on multi-agent reinforcement learning. IEEE Access (2019)
341. Hochreiter, S., Schmidhuber, J.: Long short-term memory. *Neural Comput.* **9**(8), 1735–1780 (1997)
342. Vaswani, A., Shazeer, N., Parmar, N., Uszkoreit, J., Jones, L., Gomez, A.N., Kaiser, Ł., Polosukhin, I.: Attention is all you need. In: Advances in neural information processing systems, pp. 5998–6008 (2017)
343. Hausknecht, M., Stone, P.: Deep recurrent Q-learning for partially observable MDPs. In: 2015 AAAI Fall Symposium Series (2015)
344. Jorge, E., Kågebäck, M., Johansson, F.D., Gustavsson, E.: Learning to play guess who? and inventing a grounded language as a consequence (2016). arXiv preprint [arXiv:1611.03218](https://arxiv.org/abs/1611.03218)
345. Sukhbaatar, S., Fergus, R., et al.: Learning multiagent communication with backpropagation. In: Advances in Neural Information Processing Systems, pp. 2244–2252 (2016)
346. Havrylov, S., Titov, I.: Emergence of language with multi-agent games: learning to communicate with sequences of symbols. In: Advances in Neural Information Processing Systems, pp. 2149–2159 (2017)
347. Das, A., Kottur, S., Moura, J.M., Lee, S., Batra, D.: Learning cooperative visual dialog agents with deep reinforcement learning. In: Proceedings of the IEEE International Conference on Computer Vision, pp. 2951–2960 (2017)
348. Peng, P., Wen, Y., Yang, Y., Yuan, Q., Tang, Z., Long, H., Wang, J.: Multiagent bidirectionally-coordinated nets: emergence of human-level coordination in learning to play starcraft combat games (2017). arXiv preprint [arXiv:1703.10069](https://arxiv.org/abs/1703.10069)
349. Mordatch, I., Abbeel, P.: Emergence of grounded compositional language in multi-agent populations. In: AAAI Conference on Artificial Intelligence (2018)
350. Jiang, J., Lu, Z.: Learning attentional communication for multi-agent cooperation. In: Advances in Neural Information Processing Systems, pp. 7254–7264 (2018)
351. Jiang, J., Dun, C., Lu, Z.: Graph convolutional reinforcement learning for multi-agent cooperation. **2**(3) (2018). arXiv preprint [arXiv:1810.09202](https://arxiv.org/abs/1810.09202)
352. Celikyilmaz, A., Bosselut, A., He, X., Choi, Y.: Deep communicating agents for abstractive summarization (2018). arXiv preprint [arXiv:1803.10357](https://arxiv.org/abs/1803.10357)
353. Das, A., Gervet, T., Romoff, J., Batra, D., Parikh, D., Rabbat, M., Pineau, J.: TarMAC: targeted multi-agent communication (2018). arXiv preprint [arXiv:1810.11187](https://arxiv.org/abs/1810.11187)
354. Lazaridou, A., Hermann, K.M., Tuyts, K., Clark, S.: Emergence of linguistic communication from referential games with symbolic and pixel input (2018). arXiv preprint [arXiv:1804.03984](https://arxiv.org/abs/1804.03984)
355. Cogswell, M., Lu, J., Lee, S., Parikh, D., Batra, D.: Emergence of compositional language with deep generational transmission (2019). arXiv preprint [arXiv:1904.09067](https://arxiv.org/abs/1904.09067)
356. Allis, L.: Searching for solutions in games and artificial intelligence. Ph.D. thesis, Maastricht University (1994)
357. Krizhevsky, A., Sutskever, I., Hinton, G.E.: Imagenet classification with deep convolutional neural networks. In: Advances in Neural Information Processing Systems, pp. 1097–1105 (2012)
358. Silver, D., Hubert, T., Schrittwieser, J., Antonoglou, I., Lai, M., Guez, A., Lanctot, M., Sifre, L., Kumaran, D., Graepel, T., Lillicrap, T., Simonyan, K., Hassabis, D.: A general reinforcement learning algorithm that masters chess, shogi, and Go through self-play. *Science* **362**(6419), 1140–1144 (2018)

359. Billings, D., Davidson, A., Schaeffer, J., Szafron, D.: The challenge of Poker. *Artif. Intell.* **134**(1–2), 201–240 (2002)
360. Kuhn, H.W.: A simplified two-person Poker. *Contrib. Theory Games* **1**, 97–103 (1950)
361. Southey, F., Bowling, M., Larson, B., Piccione, C., Burch, N., Billings, D., Rayner, C.: Bayes' bluff: opponent modelling in Poker. In: Proceedings of the 21st Conference on Uncertainty in Artificial Intelligence, pp. 550–558. AUAI Press (2005)
362. Bowling, M., Burch, N., Johanson, M., Tammelin, O.: Heads-up limit hold'em Poker is solved. *Science* **347**(6218), 145–149 (2015)
363. Heinrich, J., Silver, D.: Smooth UCT search in computer Poker. In: 24th International Joint Conference on Artificial Intelligence (2015)
364. Moravčík, M., Schmid, M., Burch, N., Lisý, V., Morrill, D., Bard, N., Davis, T., Waugh, K., Johanson, M., Bowling, M.: Deepstack: expert-level artificial intelligence in heads-up no-limit Poker. *Science* **356**(6337), 508–513 (2017)
365. Brown, N., Sandholm, T.: Superhuman AI for heads-up no-limit Poker: Libratus beats top professionals. *Science* **359**(6374), 418–424 (2018)
366. Burch, N., Johanson, M., Bowling, M.: Solving imperfect information games using decomposition. In: 28th AAAI Conference on Artificial Intelligence (2014)
367. Moravčík, M., Schmid, M., Ha, K., Hladík, M., Gaukrodger, S.J.: Refining subgames in large imperfect information games. In: 30th AAAI Conference on Artificial Intelligence (2016)
368. Brown, N., Sandholm, T.: Safe and nested subgame solving for imperfect-information games. In: Advances in Neural Information Processing Systems, pp. 689–699 (2017)
369. Vinyals, O., Ewalds, T., Bartunov, S., Georgiev, P., Vezhnevets, A.S., Yeo, M., Makhzani, A., Küttler, H., Agapiou, J., Schrittwieser, J., et al.: Starcraft II: a new challenge for reinforcement learning (2017). arXiv preprint [arXiv:1708.04782](https://arxiv.org/abs/1708.04782)
370. Vinyals, O., Babuschkin, I., Czarnecki, W.M., Mathieu, M., Dudzik, A., Chung, J., Choi, D.H., Powell, R., Ewalds, T., Georgiev, P., et al.: Grandmaster level in Starcraft II using multi-agent reinforcement learning. *Nature* **1**–5 (2019)
371. Mnih, V., Badia, A.P., Mirza, M., Graves, A., Lillicrap, T., Harley, T., Silver, D., Kavukcuoglu, K.: Asynchronous methods for deep reinforcement learning. In: International Conference on Machine Learning, pp. 1928–1937 (2016)
372. Lerer, A., Peysakhovich, A.: Maintaining cooperation in complex social dilemmas using deep reinforcement learning (2017). arXiv preprint [arXiv:1707.01068](https://arxiv.org/abs/1707.01068)
373. Hughes, E., Leibo, J.Z., Phillips, M., Tuyts, K., Dueñez-Guzman, E., Castañeda, A.G., Dunning, I., Zhu, T., McKee, K., Koster, R., et al.: Inequity aversion improves cooperation in intertemporal social dilemmas. In: Advances in Neural Information Processing Systems, pp. 3326–3336 (2018)
374. Cai, Q., Yang, Z., Lee, J.D., Wang, Z.: Neural temporal-difference learning converges to global optima (2019). arXiv preprint [arXiv:1905.10027](https://arxiv.org/abs/1905.10027)
375. Arora, S., Cohen, N., Hazan, E.: On the optimization of deep networks: implicit acceleration by overparameterization (2018). arXiv preprint [arXiv:1802.06509](https://arxiv.org/abs/1802.06509)
376. Li, Y., Liang, Y.: Learning overparameterized neural networks via stochastic gradient descent on structured data. In: Advances in Neural Information Processing Systems, pp. 8157–8166 (2018)
377. Brafman, R.I., Tennenholtz, M.: A near-optimal polynomial time algorithm for learning in certain classes of stochastic games. *Artif. Intell.* **121**(1–2), 31–47 (2000)
378. Brafman, R.I., Tennenholtz, M.: R-max-a general polynomial time algorithm for near-optimal reinforcement learning. *J. Mach. Learn. Res.* **3**, 213–231 (2002)
379. Tu, S., Recht, B.: The gap between model-based and model-free methods on the linear quadratic regulator: an asymptotic viewpoint (2018). arXiv preprint [arXiv:1812.03565](https://arxiv.org/abs/1812.03565)
380. Sun, W., Jiang, N., Krishnamurthy, A., Agarwal, A., Langford, J.: Model-based RL in contextual decision processes: PAC bounds and exponential improvements over model-free approaches. In: Conference on Learning Theory, pp. 2898–2933 (2019)
381. Lin, Q., Liu, M., Rafique, H., Yang, T.: Solving weakly-convex-weakly-concave saddle-point problems as weakly-monotone variational inequality (2018). arXiv preprint [arXiv:1810.10207](https://arxiv.org/abs/1810.10207)

382. García, J., Fernández, F.: A comprehensive survey on safe reinforcement learning. *J. Mach. Learn. Res.* **16**(1), 1437–1480 (2015)
383. Chen, Y., Su, L., Xu, J.: Distributed statistical machine learning in adversarial settings: Byzantine gradient descent. *Proc. ACM Meas. Anal. Comput. Syst.* **1**(2), 44 (2017)
384. Yin, D., Chen, Y., Ramchandran, K., Bartlett, P.: Byzantine-robust distributed learning: towards optimal statistical rates (2018). arXiv preprint [arXiv:1803.01498](https://arxiv.org/abs/1803.01498)

# Chapter 13

## Computational Intelligence in Uncertainty Quantification for Learning Control and Differential Games



Mushuang Liu, Yan Wan, Zongli Lin, Frank L. Lewis, Junfei Xie,  
and Brian A. Jalaian

**Abstract** Multi-dimensional uncertainties often modulate modern system dynamics in a complicated fashion. They lead to challenges for real-time control, considering the significant computation load needed to evaluate them in real-time decision processes. This chapter describes the use of computationally effective uncertainty evaluation methods for adaptive optimal control, including learning control and differential games. Two uncertainty evaluation methods are described, the multivariate probabilistic collocation method (MPCM) and its extension the MPCM-OFFD that integrates the MPCM with the orthogonal fractional factorial design (OFFD) to break the curse of dimensionality. These scalable uncertainty evaluation methods are then developed for reinforcement learning (RL)-based adaptive optimal control. Stochastic differential games, including the two-player zero-sum and multi-player nonzero-sum games, are formulated and investigated. Nash equilibrium solutions for these games are found in real time using the MPCM-based on-policy/off-policy RL methods. Real-world applications on broad-band long-distance aerial networking and strategic air traffic management demonstrate the practical use of MPCM- and MPCM-OFFD-based learning control for uncertain systems.

---

M. Liu · Y. Wan (✉) · F. L. Lewis  
University of Texas at Arlington, Arlington, USA  
e-mail: [yan.wan@uta.edu](mailto:yan.wan@uta.edu)

M. Liu  
e-mail: [mushuang.liu@mavs.uta.edu](mailto:mushuang.liu@mavs.uta.edu)

F. L. Lewis  
e-mail: [lewis@uta.edu](mailto:lewis@uta.edu)

Z. Lin  
University of Virginia, Charlottesville, USA  
e-mail: [z15y@virginia.edu](mailto:z15y@virginia.edu)

J. Xie  
San Diego State University, San Diego, USA  
e-mail: [jxie4@sdsu.edu](mailto:jxie4@sdsu.edu)

B. A. Jalaian  
U.S. Army Research Laboratory, Adelphi, USA  
e-mail: [brian.a.jalaian.civ@mail.mil](mailto:brian.a.jalaian.civ@mail.mil)

### 13.1 Introduction

Modern dynamical systems, e.g., air traffic systems, power grids, and multi-agent unmanned systems, often involve multi-dimensional uncertainties that modulate system dynamics in a complicated fashion. The uncertainties can arise either from the agents' uncertain intentions, or from uncertain environment conditions, such as probabilistic weather forecasts. They lead to challenges for real-time control, considering the significant computational load needed to evaluate these uncertainties in real-time decision processes. In this chapter, we study effective methods for adaptive optimal control under uncertainties.

Uncertainty quantification and evaluation for modern systems with complex and nonlinear dynamics typically requires sampling-based methods. Monte Carlo (MC) method and its variants, including the Markov Chain MC and Sequential MC, have been widely used for uncertainty evaluation [1–3]. A large number of simulation points are needed for the MC-based methods to achieve high accuracy, which makes them computationally unrealistic for real-time decisions. Recently, two computationally effective methods have been developed for uncertainty evaluation, the multivariate probabilistic collocation method (MPCM) [4] and its extension the MPCM-OFFD that integrates the MPCM with the orthogonal fractional factorial design (OFFD) [5–7] to break the curse of dimensionality [8]. These two methods estimate accurately the mean output of systems modulated by multi-dimensional uncertainties with very small computational load. In this chapter, we show that these effective uncertainty evaluation methods, rooted in the statistical experiment design, can effectively solve adaptive optimal control problems under multi-dimensional uncertainties, by integrating them with learning control and differential games.

Reinforcement learning (RL) has been widely used to solve optimal control problems online for a broad range of systems, including continuous-time linear systems [9], discrete-time linear systems [10, 11], continuous-time nonlinear systems [12, 13], and discrete-time nonlinear systems [14]. Reference [9] developed an integral reinforcement learning (IRL) approach for continuous-time systems, which finds the optimal control policy online without using the complete information of system dynamics. These aforementioned studies typically assume that the system dynamics are time-invariant and deterministic. In this chapter, we explore the use of RL to solve adaptive optimal control problems for systems that operate in uncertain environments or involve agents' uncertain intentions.

Game-theoretical methods recently gained significant attention in control theory to find optimal control policies that consider the evolution of agents' payoffs in multi-agent systems [15–17]. Widely studied differential games include two-player zero-sum and multi-player nonzero-sum games. To obtain Nash equilibrium solutions for these games, Hamiltonian–Jacobi–Bellman (HJB) equations need to be solved, which is extremely difficult or even impossible [17]. References [18–22] show that the RL method can be used to solve the HJB equations numerically, and further, to find Nash equilibrium solutions online. In particular, to solve the two-player zero-sum game, one- and two-loop iteration integral RL (IRL) have been developed, respectively [18,

[23], and a model-free IRL was then developed using Q-learning [24]. To solve the multi-player nonzero-sum game, an adaptive dynamic programming (ADP)-based IRL [21], and an off-policy IRL, which does not require any information of the system dynamics [22], have been investigated, respectively. In this chapter, we formulate and investigate stochastic differential games, where the system dynamics are modulated by multi-dimensional uncertainties. Nash equilibrium solutions for these games are found effectively using the MPCM-based on-policy/off-policy RL methods.

The remainder of this chapter is organized as follows. In Sect. 13.2, the optimal control problems under uncertainties are motivated and formulated. Two types of uncertain systems are introduced, systems with parameters modulated by multi-dimensional uncertainties, and random switching systems. In Sect. 13.3, the uncertainty evaluation problem is formulated, and simulation-based effective uncertainty evaluation methods, the MPCM and the MPCM-OFFD, are reviewed. In Sects. 13.4 and 13.5, the capabilities of MPCM are explored in facilitating optimal controls for the two types of uncertain systems, respectively. Section 13.6 further develops solutions for stochastic differential games, including two-player zero-sum and multi-player nonzero-sum games. Section 13.7 includes two real-world examples to demonstrate the practical use of MPCM-based learning control for the two types of uncertain systems.

## 13.2 Problem Formulation of Optimal Control for Uncertain Systems

This section formulates optimal control problems for two types of uncertain systems. The first represents systems with parameters modulated by multi-dimensional uncertainties to capture the impact of uncertain environments (Sect. 13.2.1). The second is random switching models to capture agents' uncertain intentions (Sect. 13.2.2).

### 13.2.1 *Optimal Control for Systems with Parameters Modulated by Multi-dimensional Uncertainties*

In this subsection, we first motivate and describe the systems with parameters modulated by multi-dimensional uncertainties, and then formulate the optimal control problem.

#### 13.2.1.1 **Systems with Parameters Modulated by Multi-dimensional Uncertainties**

Consider a system that operates in an uncertain environment, with environmental impact captured by an  $m$ -dimensional time-varying uncertain vector  $\mathbf{a}[k]$ . The system

dynamics are described as follows:

$$\mathbf{x}[k+1] = h(\mathbf{x}[k], \mathbf{u}[k], \mathbf{a}[k]), \quad (13.1)$$

where  $\mathbf{x}[k] \in S$  is the system state vector,  $\mathbf{u}[k] \in C$  is the control input, and  $S$  and  $C$  are, respectively, the known state and control spaces, and  $h(\cdot)$  is a general system function. Each element of  $\mathbf{a}[k]$ ,  $a_p[k]$  ( $p = 1, 2, \dots, m$ ), changes independently over time with pdf  $f_{A_p}(a_p[k])$ .

Equation (13.1) can describe uncertain systems of a wide range of realistic applications. One example is the aircraft dynamics described as

$$\mathbf{v}[k+1] = \mathbf{v}[k] - (A\mathbf{v}[k] + \mathbf{F}_u[k])\delta,$$

where  $\mathbf{v}$  is the velocity,  $\mathbf{F}_u$  is the acceleration caused by controlled thrust force,  $\delta$  is the sampling period, and  $A$ , the air resistance coefficient related to air density and air humidity, is a randomly time-varying parameter affected by uncertain weather condition. The statistics (e.g., pdfs) of such weather conditions can be obtained from probabilistic weather forecasts.

### 13.2.1.2 Optimal Control for Systems with Parameters Modulated by Multi-dimensional Uncertainties

Consider the following optimal control problem for a system with multi-dimensional uncertainties (13.1). The cost function to optimize is

$$J(\mathbf{x}[0], \mathbf{u}) = E \left[ \sum_{k=0}^{\infty} \alpha^k r(\mathbf{x}[k], \mathbf{u}[k]) \right], \quad (13.2)$$

where  $E[\cdot]$  is the expectation,  $\alpha \in (0, 1]$  is a discount factor, and  $r(\cdot)$  is the running cost function at each time step.

The value function  $V(\mathbf{x}[k])$  corresponding to the performance index is defined as

$$V(\mathbf{x}[k]) = E \left[ \sum_{k'=k}^{\infty} \alpha^{k'-k} r(\mathbf{x}[k'], \mathbf{u}[k']) \right]. \quad (13.3)$$

Consider the problem of finding an optimal control policy  $\mathbf{u}^*[k]$ , such that

$$\mathbf{u}^*[k] = \min_{\mathbf{u}} V(\mathbf{x}[k]). \quad (13.4)$$

### 13.2.2 Optimal Control for Random Switching Systems

To capture the uncertain intentions and random movement patterns of mobile agents, random switching models, including random direction (RD), random walk (RW), random waypoint (RWP), and smooth turn (ST), have been widely used in areas of communication, physics, and aerospace [25–29]. In this subsection, we first introduce the random switching models and then formulate the optimal control problems for such uncertain systems.

#### 13.2.2.1 Random Switching Models

Consider a system of  $N$  agents, where each agent moves randomly according to the following random switching model. At randomly selected time points  $T_0^i, T_1^i, T_2^i, \dots$ , where  $0 = T_0^i < T_1^i < \dots$ , agent  $i$  randomly chooses a maneuver  $\mathbf{a}_i[T_l^i]$  (e.g., velocity, heading direction, and turning center) until the next time point. The time duration to maintain the current maneuver for agent  $i$  is  $\tau_i[T_l^i]$ , i.e.,  $\tau_i[T_l^i] = T_{l+1}^i - T_l^i$ . Note that two types of random variables are involved in the random switching model. Type 1 random variable,  $\mathbf{a}_i[T_l^i]$ , describes the characteristics for each maneuver, and type 2 random variable,  $\tau_i[T_l^i]$ , describes how often the switching of type 1 random variable occurs [30]. The agent dynamics is described as

$$\mathbf{x}_i[k] = f(\mathbf{x}_i[k-1], \mathbf{a}_i[k], \tau_i[T_l^i]), \quad (13.5)$$

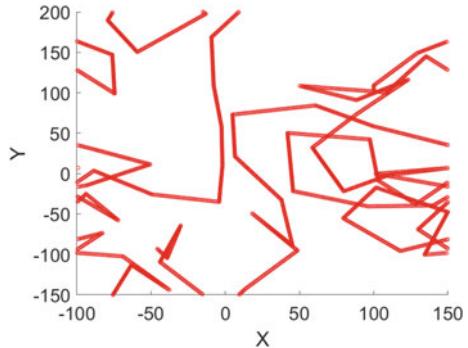
where  $\mathbf{x}_i[k] \in \mathbb{R}^n$  is the system state vector of agent  $i$  at time instant  $k$ ,  $\mathbf{a}_i[k] \in \mathbb{R}^m$  is the agent's maneuver at time  $k$ , and  $f(\cdot)$  captures the general agent dynamics. Each element of  $\mathbf{a}_i[k]$ ,  $a_{i,p}[k]$ , where  $p \in \{1, 2, \dots, m\}$ , follows the random switching rule,

$$a_{i,p}[k] = \begin{cases} a_{i,p}[T_l^i], & \text{if } k = T_l^i \text{ for some } l \in \{0, 1, \dots\}, \\ a_{i,p}[k-1], & \text{if } k \neq T_l^i \text{ for all } l \in \{0, 1, \dots\}, \end{cases} \quad (13.6)$$

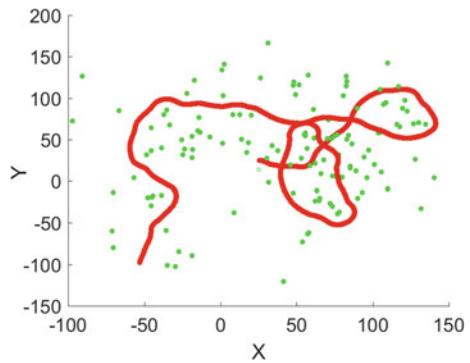
where  $a_{i,p}[T_l^i]$  ( $p = 1, 2, \dots, m$ ) is the element of the type 1 random variable  $\mathbf{a}_i[T_l^i]$ , and  $a_{i,p}[T_l^i]$  ( $p = 1, 2, \dots, m$ ) changes independently over time with pdf  $f_{A_p}(a_{i,p}[T_l^i])$ . The random variables  $(\mathbf{a}_i[T_l^i], \tau_i[T_l^i])$  are independent for each agent  $i$  to capture their independent movement patterns.

We use two simple but widely used random mobility models (RMMs), the random direction RMM, or RD RMM [25], and the smooth turn RMM, or ST RMM [28], to illustrate the random switching dynamics. Sample trajectories of the two RMMs are shown in Figs. 13.1 and 13.2, respectively. In the RD RMM, each agent  $i$  randomly selects a heading direction  $\theta_i[T_l^i]$  and velocity  $v_i[T_l^i]$  at randomly selected points  $T_0^i, T_1^i, T_2^i, \dots$ , and moves along this direction with the velocity  $v_i[T_l^i]$  until the next selected time point [25, 27]. The type 1 random variable in the RD RMM,  $\mathbf{a}_i[T_l^i] = [\theta_i[T_l^i], v_i[T_l^i]]^\top$ , is uniformly distributed, and the type 2 random variable,  $\tau_i[T_l^i] = T_{l+1}^i - T_l^i$ , is exponentially distributed. In the ST RMM, each agent selects a turning

**Fig. 13.1** Sample trajectory of RD RMM



**Fig. 13.2** Sample trajectory of ST RMM. Green spots are randomly chosen turning centers



center with a turning radius  $R_i [T_l^i]$  along the line perpendicular to its current heading direction, and then circles around it until it chooses another turning center [27, 28]. The type 1 random variable for the ST RMM,  $\mathbf{a}_i [T_l^i] = R_i [T_l^i]$ , is inversely Gaussian distributed, and the type 2 random variable,  $\tau_i [T_l^i]$ , is exponentially distributed.

### 13.2.2.2 Optimal Control for Random Switching Systems

Consider the following optimal control problem for a network of agents with random switching dynamics. Each agent  $i$  seeks its control policies  $\mathbf{u}_{i,j}$  to optimize a performance cost  $J_{i,j}$  with its neighbor  $j$ . The expected cost to optimize for agent  $i$  is

$$J_{i,j} = E \left[ \sum_{k=0}^{\infty} r_{i,j} (\mathbf{x}_i[k], \mathbf{x}_j[k], \mathbf{u}_{i,j}[k], \mathbf{u}_{j,i}[k]) \right], \quad (13.7)$$

where  $r_{i,j}[k]$ , ( $j = 1, 2, \dots, n_i$ ) is the cost between agent  $i$  and its neighbor  $j$  at time  $k$ ,  $n_i$  is the number of agent  $i$ 's neighbors.

The value function  $V_{i,j}(\mathbf{x}[k])$  corresponding to the performance index is defined as

$$V_{i,j}[k] = E \left[ \sum_{k'=k}^{\infty} r_{i,j} (\mathbf{x}_i[k'], \mathbf{x}_j[k'], \mathbf{u}_{i,j}[k'], \mathbf{u}_{j,i}[k']) \right]. \quad (13.8)$$

Consider the problem of finding the optimal control policy  $\mathbf{u}_{i,j}^*[k]$  such that

$$\mathbf{u}_{i,j}^*[k] = \operatorname{argmin}_{\mathbf{u}_{i,j}} V_{i,j}[k]. \quad (13.9)$$

### 13.3 Effective Uncertainty Evaluation Methods

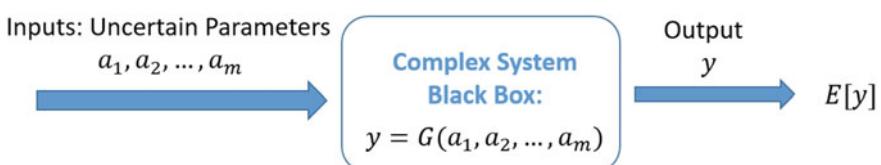
This section formulates the uncertainty evaluation problem and introduces the methods MPCM and MPCM-OFFD that evaluate the uncertainties effectively and accurately. These methods, rooted in the statistical experimental design in the testing and simulation fields [5–7], promise effective decision-making under uncertainty.

#### 13.3.1 Problem Formulation

We first formulate the general uncertainty evaluation problem as follows. Consider a system  $G(\cdot)$  modulated by  $m$ -dimensional uncertainties,  $a_1, a_2, \dots, a_m$ , which are considered as inputs to a black box (see Fig. 13.3). The output of the system,  $y$ , is the system performance of interest. The goal is to correctly estimate the mean output of the systems, i.e.,  $E[y]$ , given the statistical information of the uncertain inputs, e.g., pdfs of  $a_p$ ,  $f_{A_p}(a_p)$ .

#### 13.3.2 The MPCM

The MPCM is an effective uncertainty evaluation method, which estimates the mean output of a system modulated by multi-dimensional uncertainties accurately and effectively. The MPCM smartly selects a limited number of sample points according to the Gaussian Quadrature rules, and runs simulations at these samples to produce



**Fig. 13.3** Estimation of mean output for a system modulated by  $m$ -dimensional uncertainties

a reduced-order mapping, which has the same mean output as that of the original mapping [4]. The main properties of the MPCM are summarized as follows.

**Theorem 13.1** ([4, Theorem 2]) *Consider a system mapping modulated by  $m$  independent uncertain parameters:*

$$G(a_1, a_2, \dots, a_m) = \sum_{q_1=0}^{2n_1-1} \sum_{q_2=0}^{2n_2-1} \cdots \sum_{q_m=0}^{2n_m-1} \psi_{q_1, q_2, \dots, q_m} \prod_{p=1}^m a_p^{q_p}, \quad (13.10)$$

where  $a_p$  is an uncertain parameter with the degree up to  $2n_p - 1$ ,  $p \in 1, 2, \dots, m$ .  $n_p$  is a positive integer, and  $\psi_{q_1, q_2, \dots, q_m} \in \mathbb{R}$  are the coefficients. Each uncertain parameter  $a_p$  follows an independent pdf  $f_{A_p}(a_p)$ . The MPCM approximates  $G(a_1, a_2, \dots, a_m)$  with the following low-order mapping

$$G'(a_1, a_2, \dots, a_m) = \sum_{q_1=0}^{n_1-1} \sum_{q_2=0}^{n_2-1} \cdots \sum_{q_m=0}^{n_m-1} \Omega_{q_1, q_2, \dots, q_m} \prod_{p=1}^m a_p^{q_p}, \quad (13.11)$$

and  $E[G(a_1, a_2, \dots, a_m)] = E[G'(a_1, a_2, \dots, a_m)]$ , where  $\Omega_{q_1, q_2, \dots, q_m} \in \mathbb{R}$  are coefficients.

Theorem 13.1 shows that the MPCM reduces the number of simulations from  $2^m \prod_{p=1}^m n_p$  to  $\prod_{p=1}^m n_p$ , where  $m$  is the number of uncertain parameters. Despite the dramatic reduction of computation by a factor of  $2^m$ , MPCM evaluates the mean output of system (13.10) precisely. The MPCM design procedure is summarized as follows [4].

---

### Algorithm 13.1 MPCM

---

#### Step 1: Simulation point selection

- 1: Compute the orthonormal polynomials,  $h_p^{n_p}(a_p)$ , of degree  $n_p$  for the random variable  $a_p$ , for  $p = 1, 2, \dots, m$  according to Steps 2–7:
- 2: **For**  $p = 1$  to  $m$
- 3: Initialize  $H_p^{-1}(a_p) = h_p^{-1}(a_p) = 0$  and  $H_p^0(a_p) = h_p^0(a_p) = 1$ .
- 4: **For**  $q_p = 1$  to  $n_p$

$$\begin{aligned} H_p^{k_p}(a_p) &= a_p h_p^{q_p-1}(a_p) - \langle a_p h_p^{q_p-1}(a_p), h_p^{q_p-1}(a_p) \rangle h_p^{q_p-1}(a_p) \\ &\quad - \langle H_p^{q_p-1}(a_p), H_p^{q_p-1}(a_p) \rangle^{\frac{1}{2}} h_p^{q_p-2}(a_p), \end{aligned}$$

$$h_p^{q_p}(a_p) = H_p^{q_p}(a_p) / \langle H_p^{q_p}(a_p), H_p^{q_p}(a_p) \rangle^{\frac{1}{2}}.$$

- 5: **End**

- 6: **End**

- 7: Find the roots of  $h_p^{n_p}(a) = 0$  as the  $n_p$  PCM simulation points for  $a_p$ , denoted as  $a_{p(1)}, a_{p(2)}, \dots, a_{p(n_p)}$ .

#### Step 2: Evaluation of system outputs at selected simulation points

- 8: For each  $m$ -tuple simulation point  $(a_{1(r_1)}, a_{2(r_2)}, \dots, a_{m(r_m)})$  found in Step 1, where  $r_p \in \{1, 2, \dots, n_p\}$ , run simulation and find the associated output  $G(a_{1(r_1)}, a_{2(r_2)}, \dots, a_{m(r_m)})$ .

**Step 3: Mean output evaluation**

- 9: Find the coefficients  $b_{q_1, q_2, \dots, q_m}$  in the low-order PCM mapping  $G'(a_1, a_2, \dots, a_m) = \sum_{q_1=0}^{n_1-1} \sum_{q_2=0}^{n_2-1} \dots \sum_{q_m=0}^{n_m-1} b_{q_1, q_2, \dots, q_m} \prod_{p=1}^m h_p^{q_p}(a_p)$  following:

$$\begin{bmatrix} b_{0, \dots, 0} \\ b_{0, \dots, 1} \\ \vdots \\ b_{n_1-1, \dots, n_m-1} \end{bmatrix} = \Gamma^{-1} \begin{bmatrix} G(a_{1(1)}, \dots, a_{m(1)}) \\ G(a_{1(1)}, \dots, a_{m(2)}) \\ \vdots \\ G(a_{1(n_1)}, \dots, a_{m(n_m)}) \end{bmatrix},$$

where

$$\Gamma =$$

$$\begin{bmatrix} h_1^0(a_{1(1)}) \dots h_m^0(a_{m(1)}) & h_1^1(a_{1(1)}) \dots h_m^1(a_{m(1)}) & \dots & h_1^{n_1-1}(a_{1(1)}) \dots h_m^{n_1-1}(a_{m(1)}) \\ h_1^0(a_{1(1)}) \dots h_m^0(a_{m(2)}) & h_1^1(a_{1(1)}) \dots h_m^1(a_{m(2)}) & \dots & h_1^{n_1-1}(a_{1(1)}) \dots h_m^{n_1-1}(a_{m(2)}) \\ \vdots & \vdots & \vdots & \vdots \\ h_1^0(a_{1(n_1)}) \dots h_m^0(a_{m(n_m)}) & h_1^1(a_{1(n_1)}) \dots h_m^1(a_{m(n_m)}) & \dots & h_1^{n_1-1}(a_{1(n_1)}) \dots h_m^{n_1-1}(a_{m(n_m)}) \end{bmatrix}.$$

- 10: The predicted mean output is  $b_{0,0,\dots,0}$ .

Here  $H_p^{q_p}(a)$  represents the orthogonal polynomial of degree  $q_p$  for the uncertain parameter  $a_p$ ,  $h_p^{q_p}(a_p)$  is the normalized orthonormal polynomial, and  $\langle x_1(a_p), x_2(a_p) \rangle$  denotes the integration operation  $\int x_1(a_p)x_2(a_p)f_{A_p}(a_p)da_p$ .

Note that the MPCM can accurately predict not only the mean output of the system, but also the cross-statistics, i.e., statistics of cross input–output relationship, up to a certain degree. The MPCM can be applied to cases where the uncertain parameters are correlated, and where only moments, instead of pdfs, are available. Please see [4] for all detailed solutions and properties of the MPCM.

### 13.3.3 The MPCM-OFFD

The MPCM reduces the computational costs from  $2^m \prod_{p=1}^m n_p$  to  $\prod_{p=1}^m n_p$ ; however, it does not yet scale with the dimension  $m$  of uncertain input parameters. In real applications, high-order cross-terms in a large-scale complex system mapping have negligible effects on the output, which can then lead to further computational reduction. With this idea, the MPCM-OFFD method is developed to break the curse of dimensionality [8].

The MPCM-OFFD is an extension of the MPCM. It leverages the systematic procedures of 2-level OFFD [5, 6] to select a subset of the MPCM simulation points. The selected subset has special structural features, such as balance and orthogonality. With the MPCM-OFFD, the mean output of the original system can be esti-

mated accurately with the reduction of simulation points from  $2^{2m}$  to the range of  $\left[2^{\lceil \log_2^{(m+1)} \rceil}, 2^{m-1}\right]$ , and hence breaks the curse of dimensionality. In addition, the MPCM-OFFD solution is the most robust to numerical approximations. We refer our readers to paper [8] for a detailed development of the MPCM-OFFD method.

### 13.4 Optimal Control Solutions for Systems with Parameter Modulated by Multi-dimensional Uncertainties

In this section, we develop online learning solutions to the optimal control problems for systems with parameters modulated by multi-dimensional uncertainties as described in Sect. 13.2.1.

#### 13.4.1 Reinforcement Learning-Based Stochastic Optimal Control

Consider the stochastic optimal control problem formulated in (13.2)–(13.4). RL can be integrated with the MPCM or the MPCM-OFFD to learn the optimal control policy  $\mathbf{u}^*$  adaptively and forward-in-time. This is achieved by iterating two steps: *policy evaluation* and *policy improvement*. In the policy evaluation step, the value function (13.3) is updated for a set of admissible state  $\mathbf{x}[k] \in S'_j \subseteq S$ , given a current control policy  $\mathbf{u}^{(s-1)}[k]$ , by

$$V^{(s)}(\mathbf{x}[k]) = E[r(\mathbf{x}[k], \mathbf{u}^{(s-1)}[k]) + \alpha V^{(s-1)}(\mathbf{x}[k+1])], \quad (13.12)$$

where  $s$  is the iteration step index. The value function  $V^{(s)}(\mathbf{x}[k])$  in (13.12) is approximated using the MPCM or the MPCM-OFFD by regarding it as the mean output of the system mapping  $G(\cdot)$  subject to uncertain parameters  $\mathbf{a}[k]$  as described in (13.10),

$$G(\mathbf{x}[k], \mathbf{a}[k]) = r(\mathbf{x}[k], \mathbf{u}^{(s-1)}[k]) + \alpha V^{(s-1)}(\mathbf{x}[k+1]). \quad (13.13)$$

In the policy improvement step, the control policy is updated using the value function  $V^{(s)}(\mathbf{x}[k])$  by

$$\mathbf{u}^{(s)}[k] = \underset{\mathbf{u}[k]}{\operatorname{argmin}} E[r(\mathbf{x}[k], \mathbf{u}^{(s-1)}[k]) + \alpha V^{(s)}(\mathbf{x}[k+1])]. \quad (13.14)$$

Similarly,  $E[r(\mathbf{x}[k], \mathbf{u}^{(s-1)}[k]) + \alpha V^{(s)}(\mathbf{x}[k+1])]$ , the expected cost of a candidate control policy, can be approximated by the MPCM or the MPCM-OFFD. Algo-

rithm 13.2 summarizes the procedure of the stochastic optimal control that integrates RL with the MPCM or the MPCM-OFFD.

---

**Algorithm 13.2** RL-based Stochastic Optimal Control

---

Initialize the system with initial state  $\mathbf{x}[0]$ , and admissible control policy  $\mathbf{u}[0]$ .

Apply the MPCM (Steps 1–7 in Algorithm 13.1) or the MPCM-OFFD [8, Steps 1–9 in Algorithm 3] to select a set of samples for the uncertain vector  $\mathbf{a}[k]$ .

For each iteration  $s$ , evaluate  $G(\mathbf{x}[k], \mathbf{a}[k])$  shown in (13.13) at each the MPCM or the MPCM-OFFD sample.

Find the value function  $V^{(s)}(\mathbf{x}[k])$  using the MPCM (Steps 9–10 in Algorithm 13.1) or the MPCM-OFFD [8, Steps 11–16 in Algorithm 3], which is the mean output of mapping  $G(\mathbf{x}[k], \mathbf{a}[k])$  in (13.13).

Update the control policy  $\mathbf{u}^{(s)}[k]$  using (13.14).

Repeat Steps 3–5.

---

The optimality of Algorithm 13.2 is guaranteed by the policy iteration method [31] and the accuracy of the MPCM [4]. Theorem 13.2 describes the optimality.

**Theorem 13.2** ([32, Theorem 4]) *Consider a system with dynamics described by (13.1). Assume that the relation between the value function  $V^{(s)}(\mathbf{x}[k])$  and the uncertain parameters  $\mathbf{a}[k]$  can be captured by a polynomial system mapping (13.13) with the form shown in (13.10). Also assume that there exists a unique optimal solution and Algorithm 13.2 converges. Then the control policy derived from Algorithm 13.2 is the optimal control policy.*

For a system on an infinite state space, the value function can be approximated using neural networks or other types of approximators [32] by  $V(\mathbf{x}) = \mathbf{W}^T \Phi(\mathbf{x})$ , where  $\mathbf{W}$  is the weight vector and  $\Phi(\mathbf{x})$  is the basis vector. The value function in (13.12) then becomes

$$\mathbf{W}^{(s)T} \Phi(\mathbf{x}[k]) = E \left[ r(\mathbf{x}[k], \mathbf{u}^{(s-1)}[k]) + \alpha \mathbf{W}^{(s-1)T} \Phi(\mathbf{x}[k+1]) \right]. \quad (13.15)$$

Furthermore, consider a special case where the control space of the system is infinite, the system dynamics (13.1) takes the following linear form

$$\mathbf{x}[k+1] = \mathbf{A}\mathbf{x}[k] + \mathbf{B}\mathbf{u}[k], \quad (13.16)$$

and the running cost function  $r(\mathbf{x}[k], \mathbf{u}[k])$  in (13.2) has the quadratic form

$$r(\mathbf{x}[k], \mathbf{u}[k]) = \mathbf{x}^T[k] \mathbf{Q} \mathbf{x}[k] + \mathbf{u}^T[k] \mathbf{R} \mathbf{u}[k], \quad (13.17)$$

where  $\mathbf{Q}$  is a positive semi-definite matrix and  $\mathbf{R}$  is a positive definite matrix. Approximate the control policy  $\mathbf{u}$  using neural network by  $\mathbf{u} = \mathbf{W}_u^T \Phi_u(\mathbf{x})$ , where  $\mathbf{W}_u$  and  $\Phi_u(\mathbf{x})$  are the weight and basis vectors, respectively. Then the control policy in (13.14) can be approximated using a gradient descent method by combining (13.15)–(13.17) as follows (see [33] for a detailed analysis),

$$\mathbf{W}_u^{(s,j)} = \mathbf{W}_u^{(s,j-1)} - \beta \Phi_u(\mathbf{x}[k]) \left\{ 2\mathbf{R} \left( \mathbf{W}_u^{(s,j-1)} \right)^T \Phi_u(\mathbf{x}[k]) + E \left[ \alpha \mathbf{B}^T \nabla \Phi^T(\mathbf{x}[k+1]) W^{(s)} \right] \right\}^T, \quad (13.18)$$

where  $\nabla \Phi(\mathbf{x}) = \frac{\partial \Phi(\mathbf{x})}{\partial \mathbf{x}}$ ,  $\beta \in (0, 1)$  is a tuning parameter and  $j$  is the tuning index.

### 13.4.2 Q-Learning-Based Stochastic Optimal Control

For systems of finite state and control spaces, the policy iteration can be achieved using Q-learning, and storing and updating lookup tables [34]. Define the *Q-function* as

$$Q(\mathbf{x}[k], \mathbf{u}[k]) = E[r(\mathbf{x}[k], \mathbf{u}[k]) + \alpha V(\mathbf{x}[k+1])]. \quad (13.19)$$

Q-learning evaluates the Q-function in the policy evaluation step by

$$Q^{(s)}(\mathbf{x}[k], \mathbf{u}[k]) = E[r(\mathbf{x}[k], \mathbf{u}^{(s-1)}[k]) + \alpha Q^{(s-1)}(\mathbf{x}[k+1], \mathbf{u}^{(s-1)}[k+1])], \quad (13.20)$$

for each state  $\mathbf{x}[k] \in S$ . The control policy is then updated in the policy improvement step by

$$\mathbf{u}^{(s)}[k] = \underset{\mathbf{u}[k]}{\operatorname{argmin}} Q^{(s)}(\mathbf{x}[k], \mathbf{u}[k]). \quad (13.21)$$

Similarly, to address high-dimensional uncertainties, the MPCM or the MPCM-OFFD can be used to approximate the Q-function in (13.20), which is the mean output of system mapping  $G(\cdot)$  subject to uncertain parameters  $\mathbf{a}[k]$ ,

$$G(\mathbf{x}[k], \mathbf{u}[k], \mathbf{a}[k]) = r(\mathbf{x}[k], \mathbf{u}^{(s-1)}[k]) + \alpha Q^{(s-1)}(\mathbf{x}[k+1], \mathbf{u}^{(s-1)}[k+1]). \quad (13.22)$$

Algorithm 13.3 provides the pseudocode of the Q-learning based stochastic optimal control approach.

---

#### Algorithm 13.3 Q-Learning based Stochastic Optimal Control

---

- 1: Initialize the Q-function  $Q^{(0)}(\mathbf{x}[k], \mathbf{u}[k])$  for each  $\mathbf{x}[k] \in S$  and  $\mathbf{u}[k] \in C$ .
  - 2: Apply the MPCM (Steps 1–7 in Algorithm 13.1) or the MPCM-OFFD [8, Steps 1–9 in Algorithm 3] to select a set of samples of the uncertain vector  $\mathbf{a}[k]$ .
  - 3: For each iteration  $s$ , each  $\mathbf{x}[k] \in S$  and each  $\mathbf{u}[k] \in C$ , evaluate  $G(\mathbf{x}[k], \mathbf{u}[k], \mathbf{a}[k])$  shown in (13.22) at each MPCM or MPCM-OFFD sample.
  - 4: Update the Q-function  $Q^{(s)}(\mathbf{x}[k], \mathbf{u}[k])$  using the MPCM (lines 9–10 in Algorithm 13.1) or the MPCM-OFFD [8, Steps 11–16 in Algorithm 3], which is the mean output of mapping  $G(\mathbf{x}[k], \mathbf{u}[k], \mathbf{a}[k])$  in (13.22).
  - 5: Update the control policy  $\mathbf{u}^{(s)}[k]$  using (13.21).
  - 6: Repeat Steps 3–5.
-

## 13.5 Optimal Control Solutions for Random Switching Systems

This section develops optimal control solutions for random switching systems that capture agents' uncertain intentions. An optimal controller (Sect. 13.5.1) and an effective estimator (Sect. 13.5.2) are developed for multi-agent random switching systems defined in Sect. 13.2.2.

### 13.5.1 Optimal Controller for Random Switching Systems

We design a state feedback optimal controller for the random switching systems defined in (13.5)–(13.9). From the defined value function (13.8), we have the following Bellman equation,

$$V_{i,j}[k] = E \left[ r_{i,j}[k] + \sum_{k'=k+1}^{\infty} r_{i,j}[k'] \right] = E [r_{i,j}[k] + V_{i,j}[k+1]]. \quad (13.23)$$

This Bellman equation can be solved online using RL. In particular, assume that a neural network weight  $\mathbf{W}_{i,j}$  exists such that the value function can be approximated as

$$V_{i,j}(\mathbf{x}_i[k], \mathbf{x}_j[k]) = \mathbf{W}_{i,j}^T \phi_{i,j}(\mathbf{x}_i[k], \mathbf{x}_j[k]), \quad (13.24)$$

where  $\phi_{i,j}$  is the basis vector. Using the value function approximation (VFA) method, the optimal control policy can be found from the policy iteration (PI) algorithm [31]. Two iterative steps are involved in the PI algorithm: (1) policy evaluation, which evaluates the value function at each time step and (2) policy improvement, which finds the optimal policy based on the current value function [31].

To evaluate the value function  $V_{i,j}[k]$  at each time instant, one needs to calculate  $E [V_{i,j}[k+1]]$  according to the Bellman equation (13.23). The value function  $V_{i,j}[k+1]$ , determined uniquely by the system states  $\mathbf{x}_i[k+1]$  and  $\mathbf{x}_j[k+1]$ , can be found from the current states  $\mathbf{x}_i[k]$  and  $\mathbf{x}_j[k]$ , system dynamics  $f(\cdot)$ , and the random switching behaviors. In particular, given the current states  $\mathbf{x}_i[k]$  and  $\mathbf{x}_j[k]$ , agent  $i$  can predict its future state  $\mathbf{x}_i[k+1]$  according to its current maneuver  $\mathbf{a}_i[k+1]$  using the system dynamics  $f(\cdot)$  in (13.5). However, because agent  $i$  does not know agent  $j$ 's current maneuver  $\mathbf{a}_j[k+1]$ ,  $\mathbf{x}_j[k+1]$  needs to be estimated by agent  $i$  considering the switching behaviors.

Denote the switching behavior of agent  $j$  at time  $k$  as  $s_j[k]$ , with  $s_j[k] = 1$  or 0 indicating if the current maneuver switches at time  $k$  or not. Define  $P$  as the probability of agent  $j$  to switch its maneuver at time  $k$ , then the value function  $V_{i,j}[k]$  in a random switching system can be calculated as

$$\begin{aligned} V_{i,j}[k] &= E[V_{i,j}[k]|s_j[k]=0] P(s_j[k]=0) + E[V_{i,j}[k]|s_j[k]=1] P(s_j[k]=1) \\ &= PV_{i,j}^0[k] + (1-P)V_{i,j}^1[k], \end{aligned} \quad (13.25)$$

where  $V_{i,j}^0[k]$  and  $V_{i,j}^1[k]$  are the value functions when  $s_j[k] = 0$  and 1, respectively.

When  $s_j[k] = 0$ , agent  $j$  keeps its previous maneuver  $\mathbf{a}_j[k]$  and the value function  $V_{i,j}^0[k]$  can be obtained using  $\mathbf{a}_j[k]$  as

$$V_{i,j}^0[k] = r_{i,j}[k] + V_{i,j}[k+1](\mathbf{x}_i[k+1], \mathbf{x}_j[k], \mathbf{a}_j[k]). \quad (13.26)$$

When  $s_j[k] = 1$ , agent  $j$  chooses a new random maneuver from  $\mathbf{a}_j[T_l^j]$  at time  $k$ . In this case, the value function  $V_{i,j}^1[k]$  can be approximated using the MPCM by regarding it as the mean output of system mapping  $G_{V_{i,j}}(\cdot)$  subject to uncertain parameters  $\mathbf{a}_j[T_l^j]$ ,

$$G_{V_{i,j}}(\mathbf{x}_i[k+1], \mathbf{x}_j[k], \mathbf{a}_j[T_l^j]) = r_{i,j}[k] + V_{i,j}[k+1](\mathbf{x}_i[k+1], \mathbf{x}_j[k], \mathbf{a}_j[T_l^j]). \quad (13.27)$$

**Theorem 13.3** ([30]) *The value function described in (13.23) can be estimated as*

$$V_{i,j}[k] = PV_{i,j}^0[k] + (1-P)V_{i,j}^1[k], \quad (13.28)$$

where  $V_{i,j}^0[k]$  is described in (13.26), and  $V_{i,j}^1[k]$  is obtained from the system mapping in (13.27) as  $V_{i,j}^1[k] = E[G_{V_{i,j}}(\mathbf{x}_i[k+1], \mathbf{x}_j[k], \mathbf{a}_j[T_l^j])]$ .

Algorithm 13.4 summarizes the steps of the optimal control algorithm that integrates RL and the MPCM for agents of random switching dynamics defined in Sect. 13.2.2 [30].

---

#### Algorithm 13.4 RL-based Optimal Control for Random Switching Systems

---

- 1: Initialize the system with initial states  $\mathbf{x}_i[0]$  and  $\mathbf{x}_j[0]$ , and admissible control policies  $\mathbf{u}_{i,j}[0]$  and  $\mathbf{u}_{j,i}[0]$ .
- 2: Apply the MPCM (Steps 1–7 in Algorithm 13.1) to select a set of samples for the uncertain vector  $\mathbf{a}_j[T_l^j]$ .
- 3: For each iteration  $s$ , find the value function  $V_{i,j}^{0,(s)}$  according to (13.26) as

$$V_{i,j}^{0,(s)}[k] = r_{i,j}[k] + V_{i,j}^{(s-1)}[k+1](\mathbf{x}_i[k+1], \mathbf{x}_j[k], \mathbf{a}_j[k]).$$

- 4: Find the value of  $r_{i,j}[k] + V_{i,j}^{(s-1)}[k+1](\mathbf{x}_i[k+1], \mathbf{x}_j[k+1])$  at each selected MPCM sample.
- 5: Find the value function  $V_{i,j}^{1,(s)}[k]$  from the MPCM (Steps 9–10 in Algorithm 13.1), which, according to (13.27), is the mean output of mapping

$$G_{V_{i,j}}^{(s)} \left( \mathbf{x}_i[k+1], \mathbf{x}_j[k], \mathbf{a}_j \left[ T_l^j \right] \right) = r_{i,j}[k] + V_{i,j}^{(s-1)}[k+1] (\mathbf{x}_i[k+1], \mathbf{x}_j[k+1]).$$

- 
- 6: Find the value function  $V_{i,j}^{(s)}[k]$  by combining  $V_{i,j}^{0,(s)}[k]$ ,  $V_{i,j}^{1,(s)}[k]$  according to (13.28).  
 7: Update the control policy  $\mathbf{u}_{i,j}^{(s)}$  as  $\mathbf{u}_{i,j}^{(s)} = \operatorname{argmin}_{\mathbf{u}_{i,j}} V_{i,j}^{(s)}[k]$ .  
 8: Repeat Steps 3–7.

Theorem 13.4 describes the optimality property. The detailed performance analysis and discussions can be found in [30].

**Theorem 13.4** ([30]) Consider the optimal control problem defined on a group of agents subject to random switching dynamics shown in Sect. 13.2.2. Assume (1) VFA in (13.24) holds, (2) the relation between the value function  $V_{i,j}^1[k]$  and the uncertain parameters  $\mathbf{a}_j \left[ T_l^j \right]$  can be approximated by a polynomial system mapping (13.27) with the form shown in (13.10), and (13.3) there exists a unique optimal solution and Algorithm 13.4 converges, then the solution found by Algorithm 13.4 is the optimal control policy.

### 13.5.2 Effective Estimator for Random Switching Systems

In many practical applications, state information  $\mathbf{x}_i[k]$  and  $\mathbf{x}_j[k]$  may not be directly available for the controller design. In this subsection, we develop an online state estimation algorithm for random switching systems.

Given the previous state  $\mathbf{x}_i[k-1]$ , the expected current state  $E[\mathbf{x}_i[k]|\mathbf{x}_i[k-1]]$  can be estimated according to the random switching behaviors described in (13.6) as

$$\begin{aligned} & E[\mathbf{x}_i[k]|\mathbf{x}_i[k-1]] \\ &= E[\mathbf{x}_i[k]|\mathbf{x}_i[k-1], s_i[k-1] = 0] P(s_i[k-1] = 0) \\ &\quad + E[\mathbf{x}_i[k]|\mathbf{x}_i[k-1], s_i[k-1] = 1] P(s_i[k-1] = 1). \end{aligned} \tag{13.29}$$

When  $s_i[k-1] = 0$ , agent  $i$  keeps its previous maneuver  $\mathbf{a}_i[k-1]$ , and the expected current state  $E[\mathbf{x}_i[k]|\mathbf{x}_i[k-1], s_i[k-1] = 0]$  can be obtained from  $\mathbf{a}_i[k-1]$  as

$$E[\mathbf{x}_i[k]|\mathbf{x}_i[k-1], s_i[k-1] = 0] = f(\mathbf{x}_i[k-1], \mathbf{a}_i[k-1]). \tag{13.30}$$

When  $s_i[k-1] = 1$ , agent  $i$  chooses a new random maneuver from  $\mathbf{a}_i[T_l^i]$  at time  $k-1$ . In this case, the expected system state  $E[\mathbf{x}_i[k]|\mathbf{x}_i[k-1], s_i[k-1] = 1]$  can be approximated using the MPCM by regarding it as the mean output of system mapping  $G_i(\cdot)$  subject to uncertain parameters  $\mathbf{a}_i[T_l^i]$ ,

$$G_i(\mathbf{x}_i[k-1], \mathbf{a}_i[T_l^i]) = f(\mathbf{x}_i[k-1], \mathbf{a}_i[T_l^i]). \tag{13.31}$$

**Theorem 13.5** ([30]) Given the previous system state  $\mathbf{x}_i[k - 1]$ , the expected current state for agent  $i$  is estimated as

$$E[\mathbf{x}_i[k]|\mathbf{x}_i[k - 1]] = PE\left[G_i\left(\mathbf{x}_i[k - 1], \mathbf{a}_i[T_l^i]\right)\right] + (1 - P)f(\mathbf{x}_i[k - 1], \mathbf{a}_i[k - 1]). \quad (13.32)$$

Theorem 13.5 provides an accurate and computationally efficient approach to estimating the expected system state for random switching systems, given the previous state. Next we integrate it with the unscented Kalman filter (UKF) to provide the state estimation solution from the measurement signals  $\mathbf{z}_{i|j}[k]$  [30].

**Step 1: Initialize.** Initialize  $\hat{\mathbf{x}}_i[0]$  and  $\hat{\mathbf{P}}_i[0]$ .

**Step 2: Select MPCM points.** Apply the MPCM (Steps 1–7 in Algorithm 13.1) to select a set of samples for the uncertain vector  $\mathbf{a}_i[T_l^j]$ .

**Step 3: Estimate the system state when  $s_i[k - 1] = 0$ .** Estimate the expected system state  $E[\mathbf{x}_i[k]|\hat{\mathbf{x}}_i[k - 1], \mathbf{z}_{i|j}[k], s_i[k - 1] = 0]$  using the UKF according to the system dynamics  $f(\cdot)$  in (13.5) and the previous maneuver  $\mathbf{a}_i[k - 1]$  [35].

**Step 4: Estimate the system state when  $s_i[k - 1] = 1$ .** Estimate the expected system state  $E[\mathbf{x}_i[k]|\hat{\mathbf{x}}_i[k - 1], \mathbf{z}_{i|j}[k], s_i[k - 1] = 1]$  using the UKF and the MPCM, by conducting the following two sub-steps (a)–(b). Here we define the system mappings  $G_{\mathbf{x}_i}(\hat{\mathbf{x}}_i[k - 1], \mathbf{a}_i[T_l^i])$  and  $G_{\mathbf{P}_i}(\hat{\mathbf{x}}_i[k - 1], \mathbf{a}_i[T_l^i])$  as the relationships between the expected system state and covariance with the uncertain vector  $\mathbf{a}_i[T_l^i]$ , respectively.

(a) Estimate the system state and covariance from UKF at each selected MPCM sample.

(b) Find the expected system state  $E[\mathbf{x}_i[k]|\hat{\mathbf{x}}_i[k - 1], \mathbf{z}_{i|j}[k], s_i[k - 1] = 1]$  and covariance  $E[\mathbf{P}_i[k]|\hat{\mathbf{P}}_i[k - 1], \mathbf{z}_{i|j}[k], s_i[k - 1] = 1]$  using the MPCM (Steps 9–10 in Algorithm 13.1), which are the mean outputs of mappings  $G_{\mathbf{x}_i}(\hat{\mathbf{x}}_i[k - 1], \mathbf{a}_i[T_l^i])$  and  $G_{\mathbf{P}_i}(\hat{\mathbf{x}}_i[k - 1], \mathbf{a}_i[T_l^i])$ , respectively.

**Step 5: Find  $\hat{\mathbf{x}}_i[k]$  and  $\hat{\mathbf{P}}_i[k]$ .** Find the estimator of  $\mathbf{x}_i[k]$ , i.e.,  $\hat{\mathbf{x}}_i[k]$ , and the expected error covariance  $\hat{\mathbf{P}}_i[k]$  as

$$\begin{aligned} \hat{\mathbf{x}}_i[k] &= E[\mathbf{x}_i[k]|\hat{\mathbf{x}}_i[k - 1], \mathbf{z}_{i|j}[k]] \\ &= PE\left[\mathbf{x}_i[k]|\hat{\mathbf{x}}_i[k - 1], \mathbf{z}_{i|j}[k], s_i[k - 1] = 1\right] \\ &\quad + (1 - P)E\left[\mathbf{x}_i[k]|\hat{\mathbf{x}}_i[k - 1], \mathbf{z}_{i|j}[k], s_i[k - 1] = 0\right], \end{aligned}$$

$$\begin{aligned} \hat{\mathbf{P}}_i[k] &= E\left[\mathbf{P}_i[k]|\hat{\mathbf{P}}_i[k - 1], \mathbf{z}_{i|j}[k]\right] \\ &= PE\left[\mathbf{P}_i[k]|\hat{\mathbf{P}}_i[k - 1], \mathbf{z}_{i|j}[k], s_i[k - 1] = 1\right] \\ &\quad + (1 - P)E\left[\mathbf{P}_i[k]|\hat{\mathbf{P}}_i[k - 1], \mathbf{z}_{i|j}[k], s_i[k - 1] = 0\right]. \end{aligned}$$

The above algorithm provides a state estimation solution for agents of random switching dynamics by integrating the UKF and the MPCM. The UKF addresses the nonlinearity of system dynamics and measurement models, and the MPCM effectively samples the random switching behaviors.

## 13.6 Differential Games for Systems with Parameters Modulated by Multi-dimensional Uncertainties

Differential games have been widely studied to solve optimal control problems in multi-agent systems [15, 17, 36]. In this section, we study stochastic two-player zero-sum and multi-player nonzero-sum games, where the system dynamics are modulated by multi-dimensional uncertainties. Nash equilibrium solutions for these stochastic differential games are found analytically. Numerical algorithms that solve these games online are provided by integrating RL and the MPCM.

### 13.6.1 Stochastic Two-Player Zero-Sum Game

Consider the following two-player system dynamics with a randomly time-varying vector  $\mathbf{a}(t)$  of dimension  $m$ ,

$$\dot{\mathbf{x}} = \mathbf{A}(\mathbf{a})\mathbf{x} + \mathbf{B}\mathbf{u} + \mathbf{C}\mathbf{d}, \quad (13.33)$$

where  $\mathbf{x}(t) \in \mathbb{R}^n$  is the system state vector,  $\mathbf{u}(t) \in \mathbb{R}^p$  is the control input,  $\mathbf{d}(t) \in \mathbb{R}^q$  is the adversarial control input, and  $\mathbf{A}(\mathbf{a})$ ,  $\mathbf{B}$ , and  $\mathbf{C}$  are the drift dynamics, input dynamics, and adversarial input dynamics, respectively. Each element of  $\mathbf{a}(t)$ ,  $a_p(t)$  ( $p = 1, 2, \dots, m$ ), changes independently over time with pdf  $f_{A_p}(a_p(t))$ , and the sample functions of  $a_p(t)$  are well behaved so that the sample equations for (13.33) are ordinary differential equations [37, 38].

The expected cost to optimize is

$$J(\mathbf{x}(0), \mathbf{u}, \mathbf{d}) = E \left[ \int_0^\infty r(\mathbf{x}, \mathbf{u}, \mathbf{d}) dt \right] = E \left[ \int_0^\infty \left( \mathbf{x}^\top \mathbf{Q} \mathbf{x} + \mathbf{u}^\top \mathbf{R} \mathbf{u} - \gamma^2 \|\mathbf{d}\|^2 \right) dt \right], \quad (13.34)$$

where  $\mathbf{Q}$  is a positive semi-definite matrix,  $\mathbf{R}$  is a symmetric positive definite matrix, and  $\gamma$  is the amount of attenuation from the disturbance input to the defined performance. It is assumed that  $\gamma > \gamma^*$ , where  $\gamma^*$  is the smallest  $\gamma$  that satisfies the disturbance attenuation condition for all possible  $\mathbf{A}(\mathbf{a})$ , to guarantee that the system is always stabilizable.

The value function  $V(\mathbf{x}(t))$  corresponding to the performance index is defined as

$$V(\mathbf{x}(t)) = E \left[ \int_t^\infty \left( \mathbf{x}^T \mathbf{Q} \mathbf{x} + \mathbf{u}^T \mathbf{R} \mathbf{u} - \gamma^2 \|\mathbf{d}\|^2 \right) d\tau \right]. \quad (13.35)$$

Define the two-player zero-sum differential game as

$$V^*(\mathbf{x}(0)) = \min_{\mathbf{u}} \max_{\mathbf{d}} J(\mathbf{x}(0), \mathbf{u}, \mathbf{d}), \quad (13.36)$$

where  $V^*(\mathbf{x}(0))$  is the optimal value function. In a two-player zero-sum game, one player  $\mathbf{u}$  seeks to minimize the value function, and the other player  $\mathbf{d}$  seeks to maximize it.

With the value function in (13.35), the following Bellman equation can be obtained by taking derivative of  $V(\mathbf{x}(t))$  with respect to time  $t$ ,

$$r(\mathbf{x}, \mathbf{u}, \mathbf{d}) + E \left[ \frac{\partial V^T}{\partial \mathbf{x}} (A(\mathbf{a})\mathbf{x} + \mathbf{B}\mathbf{u} + \mathbf{C}\mathbf{d}) \right] = 0, \quad (13.37)$$

with the Hamiltonian function

$$H \left( \mathbf{x}, \mathbf{u}, \mathbf{d}, \frac{\partial V}{\partial \mathbf{x}} \right) = r(\mathbf{x}, \mathbf{u}, \mathbf{d}) + E \left[ \frac{\partial V^T}{\partial \mathbf{x}} (A(\mathbf{a})\mathbf{x} + \mathbf{B}\mathbf{u} + \mathbf{C}\mathbf{d}) \right] = 0. \quad (13.38)$$

The optimal control policies  $\mathbf{u}^*$  and  $\mathbf{d}^*$  are then derived by employing the stationary conditions in the Hamiltonian function [15, Page 447],

$$\begin{aligned} \frac{\partial H}{\partial \mathbf{u}} = 0 &\rightarrow \mathbf{u}^* = -\frac{1}{2} \mathbf{R}^{-1} \mathbf{B}^T \frac{\partial V^*}{\partial \mathbf{x}}, \\ \frac{\partial H}{\partial \mathbf{d}} = 0 &\rightarrow \mathbf{d}^* = \frac{1}{2\gamma^2} \mathbf{C}^T \frac{\partial V^*}{\partial \mathbf{x}}. \end{aligned} \quad (13.39)$$

By substituting (13.39) into the Bellman Equation (13.37) and noticing  $R^{-T} = R^{-1}$ , the following Hamilton–Jacobi–Bellman (HJB) equation is obtained.

$$H(\mathbf{x}, \mathbf{u}^*, \mathbf{d}^*, V_X^*) = \mathbf{x}^T \mathbf{Q} \mathbf{x} + E \left[ V_{X^*}^T A(\mathbf{a}) - \frac{1}{4} V_{X^*}^T \mathbf{B} \mathbf{R}^{-1} \mathbf{B}^T V_X^* + \frac{1}{4\gamma^2} V_{X^*}^T \mathbf{C} \mathbf{C}^T V_X^* \right] = 0, \quad (13.40)$$

where  $V_X^* = \frac{\partial V^*}{\partial \mathbf{x}}$ .

**Theorem 13.6** ([39]) Let  $V(\mathbf{x}(t)) > 0$  be a smooth function satisfying the HJB equation (13.40), then the following statements hold.

(1) System (13.33) is asymptotically stable in the mean with the policies  $\mathbf{u}^*$  and  $\mathbf{d}^*$  described in (13.39).

(2) The solution (i.e., policies  $\mathbf{u}^*$  and  $\mathbf{d}^*$ ) derived in (13.39) provides a saddle point solution to the game, and the system is at the Nash equilibrium with this solution.

Solving the HJB Equation (13.40) analytically is extremely difficult or even impossible [17]. Here we integrate IRL and the MPCM to provide effective online learning algorithms to solve the HJB equation numerically.

The IRL Bellman equation is

$$V(\mathbf{x}(t)) = E \left[ \int_t^{t+T} r(\mathbf{x}(\tau), \mathbf{u}(\tau), \mathbf{d}(\tau)) d\tau + V(\mathbf{x}(t+T)) \right], \quad (13.41)$$

where  $T$  is the time interval.

Assume that there exists a neural network weight  $\mathbf{W}$  such that the value function is approximated as

$$V(\mathbf{x}) = \mathbf{W}^T \phi(\mathbf{x}), \quad (13.42)$$

where  $\phi(\mathbf{x})$  is the polynomial basis function vector. Now we develop on-policy and off-policy IRL methods.

### 13.6.1.1 On-Policy IRL

An on-policy IRL algorithm is first provided to solve the stochastic two-player zero-sum game in Algorithm 13.5.

---

**Algorithm 13.5** Policy iteration algorithm for stochastic two-player zero-sum game

- 1: Initialize players with initial state  $\mathbf{x}(0)$  and admissible control and disturbance policies  $\mathbf{u}(0)$  and  $\mathbf{d}(0)$ .
- 2: Apply the MPCM (Steps 1–7 in Algorithm 13.1) to select a set of samples for the uncertain vector  $\mathbf{a}(t)$ .
- 3: For each iteration  $s$ , find the value of

$$\int_t^{t+T} r(\mathbf{x}(\tau), \mathbf{u}^{(s)}(\tau), \mathbf{d}^{(s)}(\tau)) d\tau + \mathbf{W}^{(s-1)^T} \phi(\mathbf{x}(t+T))$$

at each MPCM sample.

- 4: Find the value function  $V^{(s)}(\mathbf{x}(t))$  using the MPCM (Steps 9–10 in Algorithm 13.1), which is the mean output of the mapping  $G_{V^{(s)}}(\cdot)$  subject to uncertain parameters  $\mathbf{a}(t)$ ,

$$G_{V^{(s)}}(\mathbf{x}, \mathbf{u}^{(s)}, \mathbf{d}^{(s)}, \mathbf{a}) = \mathbf{W}^{(s-1)^T} \phi(\mathbf{x}(t+T)) + \int_t^{t+T} r(\mathbf{x}(\tau), \mathbf{u}^{(s)}(\tau), \mathbf{d}^{(s)}(\tau)) d\tau. \quad (13.43)$$

- 5: Update the value function weight  $\mathbf{W}^{(s)}$  from the value function  $V^{(s)}(\mathbf{x}(t))$ .

$$\mathbf{W}^{(s)^T} \phi(\mathbf{x}(t)) = V^{(s)}(\mathbf{x}(t)).$$

- 6: Update the control and disturbance policies  $\mathbf{u}^{(s+1)}$  and  $\mathbf{d}^{(s+1)}$  as

$$\mathbf{u}^{(s+1)} = -\frac{1}{2}\mathbf{R}^{-1}\mathbf{B}^T \frac{\partial V^{(s)}}{\partial \mathbf{x}}; \quad \mathbf{d}^{(s+1)} = \frac{1}{2\gamma^2} \mathbf{C}^T \frac{\partial V^{(s)}}{\partial \mathbf{x}}. \quad (13.44)$$

7: Repeat Steps 3–6.

---

The following theorem describes the optimality of Algorithm 13.5.

**Theorem 13.7** ([39]) Consider the stochastic two-player zero-sum game described in (13.33)–(13.36). The uncertainties in the system dynamics  $a_p$  follow time-invariant pdfs  $f_{A_p}(a_p)$ . Assume (1) VFA in (13.42) holds, (2) the relation between the value function  $V(\mathbf{x}(t))$  and the uncertain parameters  $\mathbf{a}(t)$  can be approximated by a polynomial system mapping (13.43) with the form shown in (13.10), and (3) Algorithm 13.5 converges. Then, the policies  $\mathbf{u}$  and  $\mathbf{d}$  derived from Algorithm 13.5 are optimal.

### 13.6.1.2 Off-Policy IRL

An off-policy IRL algorithm is developed in this subsection for the stochastic two-player zero-sum game. Compared to the on-policy IRL, the off-policy algorithm has the following advantages: (1) It does not require the knowledge of the system dynamics, i.e., matrix  $\mathbf{B}$  and  $\mathbf{C}$ , and (2) It does not require the disturbance policies to be manipulated. In particular, three neural networks (NNs), the critic NN, the actor NN, and the disturbance NN, are utilized in this algorithm, to approximate not only the value function, but also the control and disturbance policies [40, 41].

The detailed algorithm is shown in Algorithm 13.6 [39]. Note that here  $\mathbf{u}$  and  $\mathbf{d}$  represent the behavior policies applied to the system, and  $\mathbf{u}^{(s)}$  and  $\mathbf{d}^{(s)}$  are the desired policies to be updated.

---

#### Algorithm 13.6 Off-policy IRL for stochastic two-player zero-sum game

---

- 1: Initialize the players with initial state  $\mathbf{x}(0)$  and admissible control and disturbance policies  $\mathbf{u}(0)$  and  $\mathbf{d}(0)$ .
- 2: Apply the MPCM (Steps 1–7 in Algorithm 13.1) to select a set of samples for the uncertain vector  $\mathbf{a}(t)$ .
- 3: For each iteration  $s$ , find the value of

$$V^{(s)}(\mathbf{x}(t+T)) + \int_t^{t+T} \left( \mathbf{x}^T \mathbf{Q} \mathbf{x} + \mathbf{u}^{(s)T} \mathbf{R} \mathbf{u}^{(s)} - \gamma^2 \|\mathbf{d}^{(s)}\|^2 \right) d\tau$$

at each MPCM sample.

- 4: Find the mean output of mapping  $G_{V^{(s)}}^o(\cdot)$  subject to uncertain parameters  $\mathbf{a}(t)$  using the MPCM (Steps 9–10 in Algorithm 13.1),

$$G_{V^{(s)}}^o \left( \mathbf{x}, \mathbf{u}^{(s)}, \mathbf{d}^{(s)}, \mathbf{a} \right) = V^{(s)}(\mathbf{x}(t+T)) + \int_t^{t+T} \left( \mathbf{x}^T \mathbf{Q} \mathbf{x} + \mathbf{u}^{(s)T} \mathbf{R} \mathbf{u}^{(s)} - \gamma^2 \|\mathbf{d}^{(s)}\|^2 \right) d\tau. \quad (13.45)$$

- 5: Solve the following equation for  $V^{(s)}(\mathbf{x})$ ,  $\mathbf{u}^{(s+1)}$ , and  $\mathbf{d}^{(s+1)}$  simultaneously.

$$\begin{aligned} V^{(s)}(\mathbf{x}(t)) &= \int_t^{t+T} \left( 2\mathbf{u}^{(s+1)^\top} \mathbf{R} (\mathbf{u} - \mathbf{u}^{(s)}) - 2\gamma^2 \mathbf{d}^{(s+1)^\top} (\mathbf{d} - \mathbf{d}^{(s)}) \right) d\tau \\ &= E \left[ G_{V^{(s)}}^o(\mathbf{x}, \mathbf{u}^{(s)}, \mathbf{d}^{(s)}, \mathbf{a}) \right]. \end{aligned} \quad (13.46)$$

6: Repeat Steps 3–5.

---

The following theorem describes the optimality of Algorithm 13.6.

**Theorem 13.8** ([39]) Consider the stochastic two-player zero-sum game described in (13.33)–(13.36). The uncertainties in the system dynamics  $a_p$  follow time-invariant pdfs  $f_{A_p}(a_p)$ . Assume (1) VFA in (13.42) holds, (2) the relation between the value function  $V(\mathbf{x}(t))$  and the uncertain parameters  $\mathbf{a}(t)$  can be approximated by a polynomial system mapping (13.45) with the form shown in (13.10), and (3) Algorithm 13.6 converges. Then, the policies derived from off-policy IRL described in Algorithm 13.6 are optimal policies.

### 13.6.2 Multi-player Nonzero-Sum Game

Consider a generic  $N$ -player linear system with a time-varying uncertain vector  $\mathbf{a}(t)$  of dimension  $m$ . The system dynamics are described as

$$\dot{\mathbf{x}} = \mathbf{A}(\mathbf{a})\mathbf{x} + \sum_{j=1}^N \mathbf{B}\mathbf{u}_j, \quad (13.47)$$

where  $\mathbf{x}(t) \in \mathbb{R}^n$  is the system state vector,  $\mathbf{u}_j(t) \in \mathbb{R}^p$  is the control input of player  $j$ , and  $\mathbf{A}(\mathbf{a})$  and  $\mathbf{B}$  are the drift dynamics and input dynamics, respectively. Each element of  $\mathbf{a}(t)$ ,  $a_p(t)$  ( $p = 1, 2, \dots, m$ ), changes independently over time with pdf  $f_{A_p}(a_p(t))$ , and the sample functions of  $a_p(t)$  are well behaved so that the sample Eqs. (13.47) are ordinary differential equations [37, 38]. It is assumed that the system is stabilizable for every possible  $\mathbf{a}(t)$ .

The expected cost to optimize for player  $i$  is

$$J_i(\mathbf{x}(0), \mathbf{u}_i, \mathbf{u}_{-i}) = E \left[ \int_0^\infty r_i(\mathbf{x}, \mathbf{u}_i, \mathbf{u}_{-i}) dt \right] = E \left[ \int_0^\infty \left( \mathbf{x}^\top \mathbf{Q}_i \mathbf{x} + \sum_{j=1}^N \mathbf{u}_j^\top \mathbf{R}_{ij} \mathbf{u}_j \right) dt \right], \quad (13.48)$$

where  $\mathbf{u}_{-i}$  is the supplementary set of  $\mathbf{u}_i$ :  $\mathbf{u}_{-i} = \{\mathbf{u}_j, j \in \{1, 2, \dots, i-1, i+1, \dots, N\}\}$ .  $\mathbf{Q}_i$  and  $\mathbf{R}_{ij}$  ( $i \neq j$ ) are positive semi-definite matrices, and  $\mathbf{R}_{ii}$  is symmetric positive definite matrix.

The value function for player  $i$  is defined as

$$V_i(\mathbf{x}(t)) = E \left[ \int_t^\infty \left( \mathbf{x}^\top \mathbf{Q}_i \mathbf{x} + \sum_{j=1}^N \mathbf{u}_j^\top \mathbf{R}_{ij} \mathbf{u}_j \right) d\tau \right]. \quad (13.49)$$

Define the multi-player differential game

$$V_i^*(\mathbf{x}(0)) = \min_{\mathbf{u}_i} J_i(\mathbf{x}(0), \mathbf{u}_i, \mathbf{u}_{-i}), \quad (13.50)$$

where  $V_i^*(\mathbf{x}(0))$  is the optimal value function for player  $i$ . In the multi-player game, each player tries to minimize its cost by choosing its control policy  $\mathbf{u}_i$  based on the full state information of the system.

Consider the value function described in (13.49), the differential Bellman equation can be found by taking derivative of  $V_i(\mathbf{x}(t))$  with respect to time  $t$ ,

$$r_i(\mathbf{x}, \mathbf{u}_i, \mathbf{u}_{-i}) + E \left[ \frac{\partial V_i^T}{\partial \mathbf{x}} \left( \mathbf{A}(\mathbf{a})\mathbf{x} + \sum_{j=1}^N \mathbf{B}\mathbf{u}_j \right) \right] = 0. \quad (13.51)$$

The Hamiltonian function is

$$H_i \left( \mathbf{x}, \mathbf{u}_i, \mathbf{u}_{-i}, \frac{\partial V_i^T}{\partial \mathbf{x}} \right) = r_i(\mathbf{x}, \mathbf{u}_i, \mathbf{u}_{-i}) + E \left[ \frac{\partial V_i^T}{\partial \mathbf{x}} \left( \mathbf{A}(\mathbf{a})\mathbf{x} + \sum_{j=1}^N \mathbf{B}\mathbf{u}_j \right) \right]. \quad (13.52)$$

The optimal control policy  $\mathbf{u}_i^*$  is derived by employing the stationary condition in the Hamiltonian function,

$$\frac{\partial H_i}{\partial \mathbf{u}_i} = 0 \rightarrow \mathbf{u}_i^* = -\frac{1}{2} \mathbf{R}_{ii}^{-1} \mathbf{B}^T \frac{\partial V_i^*}{\partial \mathbf{x}}. \quad (13.53)$$

Substituting (13.53) into the Bellman Equation (13.51), the following Hamilton–Jacobi–Bellman (HJB) equation is obtained

$$\mathbf{x}^T \mathbf{Q}_i \mathbf{x} + E \left[ \frac{1}{4} \sum_{j=1}^N \frac{\partial V_j^{*T}}{\partial \mathbf{x}} \mathbf{B} \mathbf{R}_{jj}^{-1} \mathbf{R}_{ij} \mathbf{R}_{jj}^{-1} \mathbf{B}^T \frac{\partial V_j^*}{\partial \mathbf{x}} + \frac{\partial V_i^{*T}}{\partial \mathbf{x}} \left( \mathbf{A}(\mathbf{a})\mathbf{x} - \frac{1}{2} \sum_{j=1}^N \mathbf{B} \mathbf{R}_{jj}^{-1} \mathbf{B}^T \frac{\partial V_j^*}{\partial \mathbf{x}} \right) \right] = 0. \quad (13.54)$$

To solve this HJB equation, we introduce two online learning algorithms based on the modules of IRL and the MPCM. In particular, we define the following IRL Bellman equation

$$V_i(\mathbf{x}(t)) = E \left[ \int_t^{t+T} r_i(\mathbf{x}(\tau), \mathbf{u}_i(\tau), \mathbf{u}_{-i}(\tau)) d\tau + V_i(\mathbf{x}(t+T)) \right]. \quad (13.55)$$

Assume that there exists a neural network weight  $\mathbf{W}_i$  for each player  $i$  such that the value function  $V_i(\mathbf{x})$  can be approximated as

$$V_i(\mathbf{x}) = \mathbf{W}_i^T \phi_i(\mathbf{x}), \quad (13.56)$$

where  $\phi_i(\mathbf{x})$  is the polynomial basis function vector for player  $i$ . Based on this VFA and the IRL Bellman Equation (13.55), the optimal control policy for each player can be learned from on-policy IRL or off-policy IRL.

### 13.6.2.1 On-Policy IRL

An on-policy IRL algorithm is first provided to solve the stochastic multi-player nonzero-sum game [39].

---

#### Algorithm 13.7 Policy iteration for stochastic multi-player nonzero-sum game

---

- 1: Initialize each player with initial state  $\mathbf{x}(0)$  and admissible control policy  $\mathbf{u}_i(0)$ ,  $i = 1, 2, \dots, N$ .
- 2: Apply the MPCM (Steps 1–7 in Algorithm 13.1) to select a set of samples for the uncertain vector  $\mathbf{a}(t)$ .
- 3: For each iteration  $s$ , find the value of

$$\int_t^{t+T} r_i \left( \mathbf{x}(\tau), \mathbf{u}_i^{(s)}(\tau), \mathbf{u}_{-i}^{(s)}(\tau) \right) d\tau + \mathbf{W}_i^{(s-1)\top} \phi_i(\mathbf{x}(t+T))$$

at each MPCM sample.

- 4: Find the value function  $V_i^{(s)}(\mathbf{x}(t))$  using the MPCM (Steps 9–10 in Algorithm 13.1), which is the mean output of mapping  $G_{V_i^{(s)}}(\cdot)$  subject to uncertain parameters  $\mathbf{a}(t)$ ,

$$G_{V_i^{(s)}} \left( \mathbf{x}, \mathbf{u}_i^{(s)}, \mathbf{u}_{-i}^{(s)}, \mathbf{a} \right) = \int_t^{t+T} r_i \left( \mathbf{x}(\tau), \mathbf{u}_i^{(s)}(\tau), \mathbf{u}_{-i}^{(s)}(\tau) \right) d\tau + \mathbf{W}_i^{(s-1)\top} \phi_i(\mathbf{x}(t+T)). \quad (13.57)$$

- 5: Update the value function weight vector  $W_i^{(s)}$  from  $V_i^{(s)}(\mathbf{x}(t))$ .

$$\mathbf{W}_i^{(s)\top} \phi_i(\mathbf{x}(t)) = V_i^{(s)}(\mathbf{x}(t)).$$

- 6: Update the control policy  $\mathbf{u}_i$  using

$$\mathbf{u}_i^{(s+1)} = -\frac{1}{2} \mathbf{R}_{ii}^{-1} \mathbf{B}^\top \frac{\partial V_i^{(s)}}{\partial \mathbf{x}}. \quad (13.58)$$

- 7: Repeat Steps 3–6.
- 

Theorem 13.9 describes the optimality of Algorithm 13.7.

**Theorem 13.9** ([39]) *Consider the stochastic multi-player nonzero-sum game described in (13.47)–(13.50). The uncertainties in the system dynamics  $a_p$  follow time-invariant pdfs  $f_{A_p}(a_p)$ . Assume (1) VFA in (13.56) holds, (2) the relation between the value function  $V_i(\mathbf{x}(t))$  and the uncertain parameters  $\mathbf{a}(t)$  can be approximated by a polynomial system mapping (13.57) with the form shown in (13.10), and*

(3) Algorithm 13.7 converges. Then, the solution found from Algorithm 13.7 is the optimal control.

### 13.6.2.2 Off-Policy IRL

An off-policy IRL integrated with the MPCM is described in this subsection, to solve the stochastic multi-player nonzero-sum game. In this off-policy algorithm,  $\mathbf{u}_i$  is behavior policy applied to the system and  $\mathbf{u}_i^{(s)}$  is the desired policy to be updated. The system matrix  $\mathbf{B}$  is not required to be known in this algorithm.

---

**Algorithm 13.8** Off-Policy IRL for multi-player nonzero-sum game with uncertain system dynamics

---

- 1: Initialize the players with initial state  $\mathbf{x}(0)$  and admissible control policies  $\mathbf{u}_i(0)$ .
- 2: Apply the MPCM (Steps 1–7 in Algorithm 13.1) to select a set of samples for the uncertain vector  $\mathbf{a}(t)$ .
- 3: For each iteration  $s$ , find the value of

$$\int_t^{t+T} \left( \mathbf{x}^T \mathbf{Q}_i \mathbf{x} + \sum_{j=1}^N \mathbf{u}_j^{(s)T} \mathbf{R}_{ij} \mathbf{u}_j \right) d\tau + V_i^{(s)}(\mathbf{x}(t+T))$$

at each MPCM sample.

- 4: Find the mean output of mapping  $G_{V_i^{(s)}}^o(\cdot)$  subject to uncertain parameters  $\mathbf{a}(t)$  using the MPCM (Steps 9–10 in Algorithm 13.1),

$$G_{V_i^{(s)}}^o \left( \mathbf{x}, \mathbf{u}_i^{(s)}, \mathbf{u}_{-i}^{(s)}, \mathbf{a} \right) = V_i^{(s)}(\mathbf{x}(t+T)) + \int_t^{t+T} \left( \mathbf{x}^T \mathbf{Q}_i \mathbf{x} + \sum_{j=1}^N \mathbf{u}_j^{(s)T} \mathbf{R}_{ij} \mathbf{u}_j \right) d\tau. \quad (13.59)$$

- 5: Solve the following equation for  $V_i^{(s)}(\mathbf{x})$  and  $\mathbf{u}_i^{(s+1)}$ , respectively.

$$\begin{aligned} V_i^{(s)}(\mathbf{x}(t)) - \int_t^{t+T} \left( \sum_{j=1}^N 2\mathbf{u}_j^{(s+1)T} \mathbf{R}_{ii} (\mathbf{u}_j - \mathbf{u}_j^{(s)}) \right) d\tau \\ = E \left[ G_{V_i^{(s)}}^o \left( \mathbf{x}, \mathbf{u}_i^{(s)}, \mathbf{u}_{-i}^{(s)}, \mathbf{a} \right) \right]. \end{aligned} \quad (13.60)$$

- 6: Repeat Steps 3–5.
- 

The following theorem describes the optimality of Algorithm 13.8.

**Theorem 13.10** ([39]) Consider the stochastic multi-player nonzero-sum game described in (13.47)–(13.50). The uncertainties in the system dynamics  $a_p$  follow time-invariant pdfs  $f_{A_p}(a_p)$ . Assume (1) VFA in (13.56) holds, (2) the relation between the value function  $V_i(\mathbf{x}(t))$  and the uncertain parameters  $\mathbf{a}(t)$  can be approx-

*imated by a polynomial system mapping (13.59) with the form shown in (13.10), and (3) Algorithm 13.8 converges. Then, the solution found from off-policy IRL described in Algorithm 13.8 is optimal.*

## 13.7 Applications

In this section, we use two real-world examples to demonstrate the practical use of learning control solutions developed in Sects. 13.4 and 13.5. Section 13.7.1 applies the Q-learning based stochastic optimal control method to derive strategic air traffic flow management solutions under uncertain weather. Section 13.7.2 develops automatic antenna control solutions to build robust long-distance and broad-band air-to-air communication.

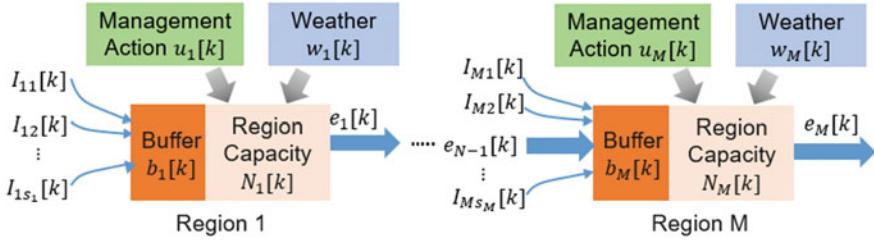
### 13.7.1 Traffic Flow Management Under Uncertain Weather

Increasing traffic demand and uncertain convective weather lead to demand-capacity imbalance in the national airspace system, and hence growing traffic delays. Strategic air traffic flow management aims to mitigate traffic congestion at a long look-ahead time window. In this section, we apply the Q-learning based stochastic optimal control method described in Sect. 13.4.2 to derive strategic air traffic flow management solutions robust to uncertain weather and traffic demands [34].

Consider the scenario where air traffic flows from different airports enter a sequence of  $M$  regions (see Fig. 13.4). Let  $I_{ij}[k]$  be the number of aircraft arriving at the boundary of region  $i \in \{1, 2, \dots, M\}$  from airport  $j \in s_i$  at time  $k$ , where  $s_i$  denotes the set of airports connecting to region  $i$ . Suppose a maximum of  $N_i[k]$  aircraft are allowed to pass region  $i$  at time  $k$  and the remaining aircraft have to wait in a queue with length captured by  $b_i[k]$ . The air traffic flow dynamics can then be described by

$$\begin{aligned} e_i[k] &= \min\{b_i[k - 1], N_i[k]\}, \\ b_i[k] &= \begin{cases} b_i[k - 1] + \sum_{j \in s_i} I_{ij}[k] - e_i[k], & \text{if } i = 1, \\ b_i[k - 1] + \sum_{j \in s_i} I_{ij}[k] + e_{i-1}[k] - e_i[k], & \text{else,} \end{cases} \\ B_i[k] &= b_i[k - 1] - e_i[k], \end{aligned} \quad (13.61)$$

where  $e_i[k]$  is the number of aircraft passing region  $i$  at time  $k$ , and  $B_i[k]$  is the backlog, reflecting the severity of traffic congestion. The impact of convective weather is modeled as weather reduction  $w_i[k]$ . To mitigate traffic congestion, we adopt en route flow restriction strategies at each region, captured by a scaling factor  $u_i[k] \in [0, 1]$ . The capacity of each region can then be computed by



**Fig. 13.4** Air traffic flows from different airports passing through  $M$  sequential weather zones.  
© 2017 IEEE. Reprinted, with permission, from [34]

$$N_i[k] = \min\{N_{ni} - w_i[k], u_i[k]N_{ni}\}, \quad (13.62)$$

where  $N_{ni}$  is the capacity of region  $i$  at nominal conditions.

To minimize traffic congestion, a stochastic optimal control problem can be formulated, with a cost function capturing the total expected backlogs over the planning horizon as follows:

$$J(\mathbf{b}[0], \mathbf{u}) = E \left[ \sum_{k=0}^{\mathcal{N}-1} \alpha^k r(\mathbf{b}[k], \mathbf{u}[k]) \right], \quad (13.63)$$

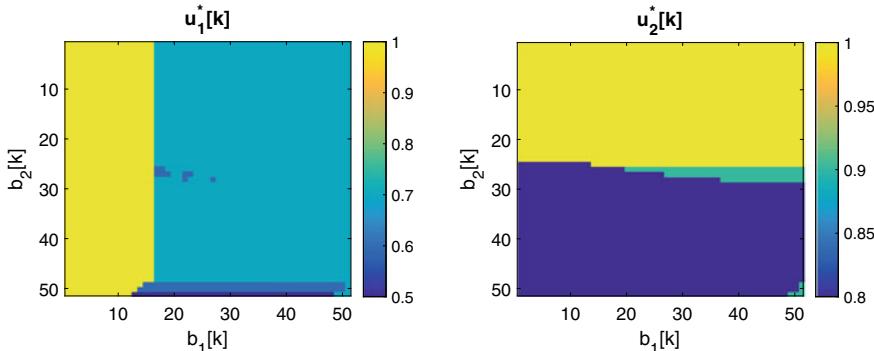
where  $\mathcal{N}$  describes the planning horizon and  $\mathbf{b}[k] = [b_1[k], b_2[k], \dots, b_M[k]]^\top$  and  $\mathbf{u}[k] = [u_1[k], u_2[k], \dots, u_M[k]]^\top$  are state and control vectors, respectively. The expected cost is evaluated over the whole uncertainty space, where the uncertain vector  $\mathbf{a}[k] = [I_{11}[k], \dots, I_{1s_1}[k], \dots, I_{M1}[k], \dots, I_{Ms_M}[k], w_1[k], \dots, w_M[k]]^\top$  includes all random variables that describe probabilistic weather and traffic demand, and  $\alpha \in (0, 1]$  is the discount factor. A running cost function  $r(\cdot)$  is designed to penalize weather-induced backlogs and high-transient backlogs, and also to differentiate weather- and management-induced capacity reductions as follows:

$$r(\mathbf{b}[k], \mathbf{u}[k]) = \sum_{i=1}^M r_i(B_i[k], u_i[k]), \quad (13.64)$$

$$\begin{aligned} r_i(B_i[k], u_i[k]) &= C_1 B_{mi}[k] + H_1 \mathbf{1}(B_{mi}[k] - \Gamma) \\ &\quad + C_2 B_{wi}[k] + H_2 \mathbf{1}(B_{wi}[k] - \Gamma) + C_3(1 - u_i[k]), \end{aligned} \quad (13.65)$$

where  $B_{mi}[k]$  and  $B_{wi}[k]$  are the management- and weather-induced backlogs at time  $k$ , respectively,  $\Gamma$  is a threshold,  $\mathbf{1}(x)$  is a function which equals  $x$  if  $x \geq 0$  and 0 otherwise, and  $C_1, C_2, C_3, H_1$  and  $H_2$  are unit costs. With (13.63), the optimal control policy  $\mathbf{u}^*[k]$  is the one that minimizes  $J(\mathbf{b}[0], \mathbf{u})$ .

Assume that each uncertain variable follows an independent probability distribution. We can then apply the Q-learning based stochastic optimal control method in Algorithm 13.3 to approximate the optimal control policy, where  $\mathcal{N}$  is set to  $\infty$ .



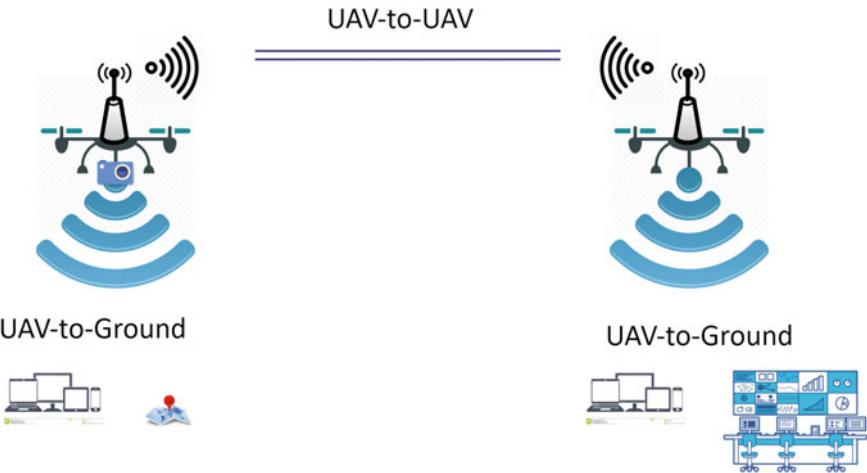
**Fig. 13.5** Optimal control policy estimated by the MPCM-OFFD and Q-learning based stochastic optimal control method. © 2017 IEEE. Reprinted, with permission, from [34]

In the simulation, consider  $M = 2$ ,  $0 \leq b_i[k] \leq 50$  and  $u_i[k] \in \{0.1, 0.2, \dots, 1\}$ , for each  $i = 1, 2$ . The uncertain traffic demand  $I_{ij}[k]$  is assumed to follow a uniform or truncated exponential distribution, and the uncertain weather impact  $w_i[k]$  is assumed to follow a truncated normal distribution (see [34] for more details). The other parameters are set as  $\alpha = 0.8$ ,  $C_1 = H_1 = C_3 = 1$ ,  $C_2 = 5$ ,  $H_2 = 2$ , and  $\Gamma = 30$ . Figure 13.5 shows the optimal control policy estimated by the Q-learning based method with the value function approximated using the MPCM-OFFD. Fast convergence is achieved in about 15 iterations.

### 13.7.2 Learning Control for Aerial Communication Using Directional Antennas (ACDA) Systems

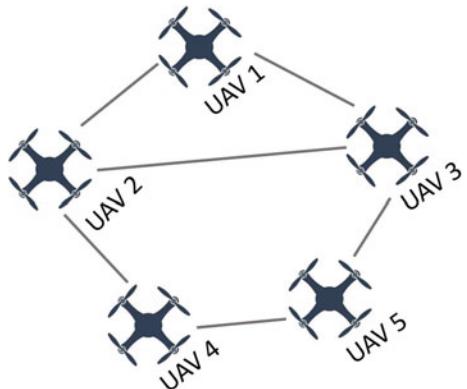
Aerial communication using directional antennas (ACDA) has shown its value in establishing long-distance and broad-band unmanned aerial vehicle (UAV)-to-UAV communication (Fig. 13.6) [42–44]. The self-alignment of directional antennas allows wireless energy to focus on certain direction, and significantly extends the communication distance. With ACDA, on-demand UAV-carried communication infrastructure can be delivered without ground infrastructure support. High-rate data such as monitoring streams from remote locations can be transmitted in real time through the air-to-air communication. The detailed design protocol and practical applications of the ACDA system can be found in [43, 44].

The automatic alignment of directional antennas is not easy to achieve considering practical issues such as the limited sensing devices, uncertain and varying UAV movement patterns, and unknown communication environments. Here we apply the optimal control solutions developed for random switching systems (Sect. 13.5) to solve the automatic alignment problem for UAV-carried directional antennas.



**Fig. 13.6** Illustration of an ACDA system. © 2020 IEEE. Reprinted, with permission, from [44]

**Fig. 13.7** Communication topology of the five-UAV network. © 2020 IEEE. Reprinted, with permission, from [30]

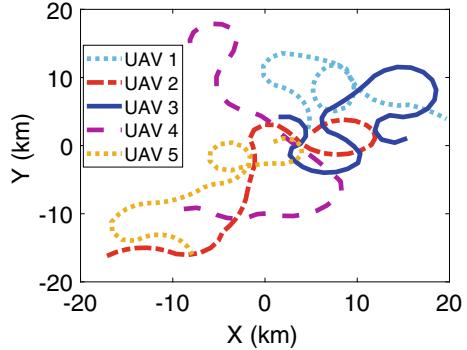


Consider a UAV network of five UAVs connected by a communication graph shown in Fig. 13.7. Each UAV moves independently according to the smooth turn RMM to support a surveillance-like mission (see Sect. 13.2.2.1). The randomly generated trajectories of the five UAVs are shown in Fig. 13.8.

Each UAV is installed with a three-sector directional antenna to communicate with its neighbor UAVs over long distances. The goal of each UAV is to maximize the communication performances with its neighbors by controlling the heading directions of its antennas. The heading angle dynamics of the directional antennas for UAV  $i$  is described as

$$\theta_i[k+1] = \theta_i[k] + (\omega'_i[k] + \omega_i[k]) \delta, \quad (13.66)$$

**Fig. 13.8** Sample trajectories of five UAVs.  
 © 2020 IEEE. Reprinted, with permission, from [30]



where  $\theta_i$  is the heading angle vector of the directional antenna mounted on UAV  $i$ ,  $\omega'_i$  is the angular velocity of the directional antenna due to the heading control,  $\delta$  is the sampling period, and  $\omega_i[k]$  is the angular velocity of UAV  $i$ . Note that the change of  $\theta_i$  is caused by both the control of directional antenna,  $\omega'_i$ , and the movement of UAV  $i$ ,  $\omega_i$ .

We use the received signal strength indicators (RSSI) to measure the communication performance. A larger RSSI indicates a better communication performance. We define the cost function to maximize as the summation of the predicted RSSI signals over a look-ahead window, i.e.,

$$J_{i,j} = E \left[ \sum_{k=0}^N R_{i,j}[k] \right], \quad (13.67)$$

where  $N$  is the experimental time, and  $R_{i,j}$  is the RSSI that UAV  $i$  receives from its neighbor  $j$ ,

$$R_{i,j}[k] = P_{t|dBm} + 20 \log_{10} \left( \frac{\lambda}{4\pi d[k]} \right) + G_{l|dBi}[k]. \quad (13.68)$$

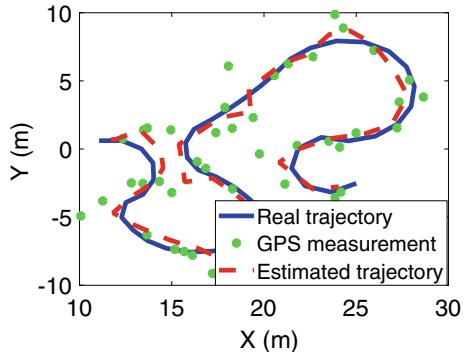
Here  $P_{t|dBm}$  is the transmitted signal power,  $\lambda$  is the wavelength,  $d[k]$  is the distance between neighboring UAVs, and  $G_{l|dBi}[k]$  is the sum of gains of the two directional antennas, which depends on their heading angles [45]:

$$\begin{aligned} G_{l|dBi}[k] = & \left( G_{t|dBi}^{max} - G_{t|dBi}^{min} \right) \sin \frac{\pi}{2n} \frac{\sin \left( \left( \frac{n}{2} (k_a d_a (\cos(\gamma_t[k] - \theta_t[k])) - 1) - \frac{\pi}{n} \right) \right)}{\sin \left( \frac{1}{2} (k_a d_a (\cos(\gamma_t[k] - \theta_t[k])) - 1) - \frac{\pi}{n} \right)} \\ & + \left( G_{r|dBi}^{max} - G_{r|dBi}^{min} \right) \sin \frac{\pi}{2n} \frac{\sin \left( \left( \frac{n}{2} (k_a d_a (\cos(\gamma_r[k] - \theta_r[k])) - 1) - \frac{\pi}{n} \right) \right)}{\sin \left( \frac{1}{2} (k_a d_a (\cos(\gamma_r[k] - \theta_r[k])) - 1) - \frac{\pi}{n} \right)} \\ & + G_{t|dBi}^{min} + G_{r|dBi}^{min}. \end{aligned} \quad (13.69)$$

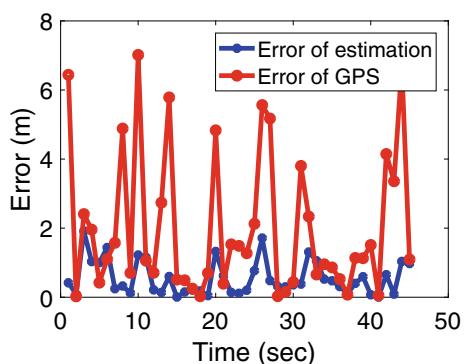
Here  $G_{t|dBi}^{\max}$ ,  $G_{t|dBi}^{\min}$ ,  $G_{r|dBi}^{\max}$ , and  $G_{r|dBi}^{\min}$  are the maximum and minimum gains of transmitting and receiving antennas,  $k_a = \frac{2\pi}{\lambda}$  is the wave number,  $n$  and  $d_a$  are design parameters of the directional antenna,  $\theta_t[k]$  and  $\theta_r[k]$  are the heading angles of the transmitting and receiving antennas at time  $k$ , respectively, and  $\gamma_t[k]$  and  $\gamma_r[k]$  are the heading angles of the transmitting and receiving antennas corresponding to the maximal  $G_l$  at time  $k$ , respectively.

GPS signals corrupted with Gaussian white noise are used as measurement signals to facilitate state estimations and optimal controls. The estimation and control algorithms developed in Sect. 13.5 are simulated for all five UAVs. Here we show the performance of UAV 3 with the communication link to UAV 2 in Figs. 13.9 and 13.10. The estimated trajectory matches well with the real UAV trajectory, and the estimation errors are much smaller than the errors of GPS signals, which validates the effectiveness of the developed estimation solution described in Sect. 13.5.2. The control performances are shown in Figs. 13.11 and 13.12. In particular, Fig. 13.11 shows the controlled heading directions of the directional antenna mounted on UAV 3 to communicate with UAV 2, and Fig. 13.12 shows the errors between the controlled and real optimal heading directions. The controlled directional antenna heading direction is very close to the optimal solution, and the errors are within  $(-0.2, 0.15)$  rad, which

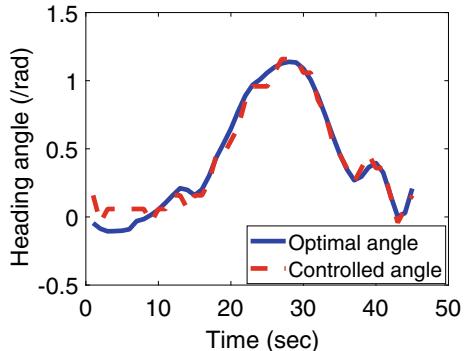
**Fig. 13.9** Trajectory of UAV 3. The blue solid and red dotted curves are real and estimated trajectories, respectively, and green dots are GPS measurements.  
© 2020 IEEE. Reprinted, with permission, from [30]



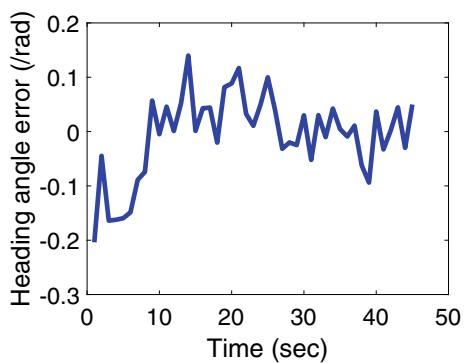
**Fig. 13.10** Comparison of errors between estimated trajectories and GPS signals.  
© 2020 IEEE. Reprinted, with permission, from [30]



**Fig. 13.11** Comparison between the controlled heading angles and the optimal headings of directional antenna on UAV 3 to communicate with UAV 2. © 2020 IEEE. Reprinted, with permission, from [30]



**Fig. 13.12** Errors between the optimal and the controlled heading angles of the directional antenna. © 2020 IEEE. Reprinted, with permission, from [30]



validates effectiveness of the proposed adaptive optimal control solution described in Algorithm 13.4.

## 13.8 Summary

This chapter explored computationally effective learning control and differential game solutions for multi-agent uncertain systems. Two types of uncertain systems of broad applicability were investigated. The first one is the class of systems with parameters modulated by multi-dimensional uncertainties to capture the impact of uncertain environments. The second one is the class of random switching systems to capture agents' uncertain intentions. Due to the significant computational load required to evaluate uncertainty, online stochastic optimal control for these uncertain systems is challenging to achieve in real-world applications. To address this challenge, two effective uncertainty evaluation methods, the MPCM and its extension the MPCM-OFFD, were leveraged and integrated with RL and the UKF to develop practical stochastic optimal control solutions. These uncertainty evaluation methods, originated in the statistical experimental design, were proved to be very valuable to

facilitate decision-making under uncertainty. To consider the interactions of players in multi-player systems, stochastic differential games, including two-player zero-sum and multi-player nonzero-sum games, were also formulated and investigated, where agent dynamics are modulated by multi-dimensional uncertainties to capture the impact of uncertain environments. To find the Nash equilibrium solutions online, the MPCM was further integrated with on-policy and off-policy IRL for these games. Two real-world applications, the air traffic management and broad-band long-distance aerial networking, verified the practical use of the developed MPCM- and the MPCM-OFFD-based learning control solutions. In the future work, we will study stochastic graphical games where players communicate on a graph topology. We will also pursue additional applications of the computationally effective optimal decision solutions, such as autonomous driving and UAV traffic management.

**Acknowledgements** The authors would like to thank the ONR Grant N00014-18-1-2221 and NSF grants 1714519, 1839804, and 1953049 for the support of this work.

## References

1. Kantas, N., Lecchini-Visintini, A., Maciejowski, J.: Simulation-based Bayesian optimal design of aircraft trajectories for air traffic management. *Int. J. Adapt. Control Signal Process.* **24**(10), 882–899 (2010)
2. Prandini, M., Hu, J., Lygeros, J., Sastry, S.: A probabilistic approach to aircraft conflict detection. *IEEE Trans. Intell. Transp. Syst.* **1**(4), 199–220 (2000)
3. Visintini, A.L., Glover, W., Lygeros, J., Maciejowski, J.: Monte Carlo optimization for conflict resolution in air traffic control. *IEEE Trans. Intell. Transp. Syst.* **7**(4), 470–482 (2006)
4. Zhou, Y., Wan, Y., Roy, S., Taylor, C., Wanke, C., Ramamurthy, D., Xie, J.: Multivariate probabilistic collocation method for effective uncertainty evaluation with application to air traffic flow management. *IEEE Trans. Syst. Man Cybern.: Syst.* **44**(10), 1347–1363 (2014)
5. Prinz, J., Tobias, P., Guthrie, W.F., Hembree, B., Croarkin, M., Filliben, J.J., Heckert, N.: NIST/SEMATECH e-Handbook of Statistical Methods. NIST Handbook, vol. 151 (2013). <http://www.itl.nist.gov/div898/handbook/>
6. Mee, R.: A Comprehensive Guide to Factorial Two-Level Experimentation. Springer Science & Business Media, Berlin (2009)
7. Mills, K.L., Filliben, J.J.: Comparison of two dimension-reduction methods for network simulation models. *J. Res. Natl. Inst. Stand. Technol.* **116**(5), 771 (2011)
8. Xie, J., Wan, Y., Mills, K., Filliben, J.J., Lei, Y., Lin, Z.: M-PCM-OFFD: an effective output statistics estimation method for systems of high dimensional uncertainties subject to low-order parameter interactions. *Math. Comput. Simul.* **159**, 93–118 (2019)
9. Vrabie, D., Pastravanu, O., Abu-Khalaf, M., Lewis, F.L.: Adaptive optimal control for continuous-time linear systems based on policy iteration. *Automatica* **45**(2), 477–484 (2009)
10. Lewis, F.L., Vrabie, D.: Reinforcement learning and adaptive dynamic programming for feedback control. *IEEE Circuits Syst. Mag.* **9**(3) (2009)
11. Kiumarsi, B., Lewis, F.L., Modares, H., Karimpour, A., Naghibi-Sistani, M.-B.: Reinforcement Q-learning for optimal tracking control of linear discrete-time systems with unknown dynamics. *Automatica* **50**(4), 1167–1175 (2014)
12. Vamvoudakis, K.G., Lewis, F.L.: Online actor-critic algorithm to solve the continuous-time infinite horizon optimal control problem. *Automatica* **46**(5), 878–888 (2010)
13. Vrabie, D., Lewis, F.: Neural network approach to continuous-time direct adaptive optimal control for partially unknown nonlinear systems. *Neural Netw.* **22**(3), 237–246 (2009)

14. Kiumarsi, B., Lewis, F.L.: Actor-critic-based optimal tracking for partially unknown nonlinear discrete-time systems. *IEEE Trans. Neural Netw. Learn. Syst.* **26**(1), 140–151 (2015)
15. Lewis, F.L., Vrabie, D., Syrmos, V.L.: *Optimal Control*. Wiley, Hoboken (2012)
16. Başar, T., Bernhard, P.:  *$H_\infty$  Optimal Control and Related Minimax Design Problems: A Dynamic Game Approach*. Springer Science & Business Media, Berlin (2008)
17. Vamvoudakis, K.G., Modares, H., Kiumarsi, B., Lewis, F.L.: Game theory-based control system algorithms with real-time reinforcement learning: how to solve multiplayer games online. *IEEE Control Syst.* **37**(1), 33–52 (2017)
18. Vrabie, D., Lewis, F.: Adaptive dynamic programming for online solution of a zero-sum differential game. *J. Control Theory Appl.* **9**(3), 353–360 (2011)
19. Al-Tamimi, A., Lewis, F.L., Abu-Khalaf, M.: Model-free Q-learning designs for linear discrete-time zero-sum games with application to H-infinity control. *Automatica* **43**(3), 473–481 (2007)
20. Kim, J.-H., Lewis, F.L.: Model-free  $H_\infty$  control design for unknown linear discrete-time systems via Q-learning with LMI. *Automatica* **46**(8), 1320–1326 (2010)
21. Vrabie, D., Lewis, F.: Integral reinforcement learning for online computation of feedback nash strategies of nonzero-sum differential games. In: Proceedings of IEEE Conference on Decision and Control (CDC), Atlanta, GA (2010)
22. Song, R., Lewis, F.L., Wei, Q.: Off-policy integral reinforcement learning method to solve nonlinear continuous-time multiplayer nonzero-sum games. *IEEE Trans. Neural Netw. Learn. Syst.* **28**(3), 704–713 (2017)
23. Wu, H.-N., Luo, B.: Simultaneous policy update algorithms for learning the solution of linear continuous-time  $H_\infty$  state feedback control. *Inf. Sci.* **222**, 472–485 (2013)
24. Li, H., Liu, D., Wang, D., Yang, X.: Integral reinforcement learning for linear continuous-time zero-sum games with completely unknown dynamics. *IEEE Trans. Autom. Sci. Eng.* **11**(3), 706–714 (2014)
25. Nain, P., Towsley, D., Liu, B., Liu, Z.: Properties of random direction models. In: Proceedings of IEEE INFOCOM, vol. 3 (2005)
26. Howse, J.R., Jones, R.A., Ryan, A.J., Gough, T., Vafabakhsh, R., Golestanian, R.: Self-motile colloidal particles: from directed propulsion to random walk. *Phys. Rev. Lett.* **99**(4), 048102 (2007)
27. Xie, J., Wan, Y., Kim, J.H., Fu, S., Namuduri, K.: A survey and analysis of mobility models for airborne networks. *IEEE Commun. Surv. Tutor.* **16**(3), 1221–1238 (2014)
28. Wan, Y., Namuduri, K., Zhou, Y., Fu, S.: A smooth-turn mobility model for airborne networks. *IEEE Trans. Veh. Technol.* **62**(7), 3359–3370 (2013)
29. Liu, M., Wan, Y.: Analysis of random mobility model with sense and avoid protocols for UAV traffic management. In: Proceedings of AIAA Information Systems-AIAA Infotech Aerospace, Kissimmee, FL (2018)
30. Liu, M., Wan, Y., Lewis, F.L.: Adaptive optimal decision in multi-agent random switching systems. *IEEE Control Syst. Lett.* **4**(2), 265–270 (2020)
31. Bertsekas, D.P.: Value and policy iterations in optimal control and adaptive dynamic programming. *IEEE Trans. Neural Netw. Learn. Syst.* **28**(3), 500–509 (2017)
32. Xie, J., Wan, Y., Mills, K., Filliben, J.J., Lewis, F.L.: A scalable sampling method to high-dimensional uncertainties for optimal and reinforcement learning-based controls. *IEEE Control Syst. Lett.* **1**(1), 98–103 (2017)
33. Lewis, F.L., Vrabie, D., Vamvoudakis, K.G.: Reinforcement learning and feedback control: using natural decision methods to design optimal adaptive controllers. *IEEE Control Syst.* **32**(6), 76–105 (2012)
34. Xie, J., Wan, Y., Lewis, F.L.: Strategic air traffic flow management under uncertainties using scalable sampling-based dynamic programming and Q-learning approaches. In: Proceedings of IEEE Asian Control Conference (ASCC), Gold Coast, QLD, Australia (2017)
35. Julier, S.J., Uhlmann, J.K., Durrant-Whyte, H.F.: A new approach for filtering nonlinear systems. In: Proceedings of IEEE American Control Conference, Seattle, WA (1995)
36. Modares, H., Lewis, F.L., Jiang, Z.-P.:  $H_\infty$  tracking control of completely unknown continuous-time systems via off-policy reinforcement learning. *IEEE Trans. Neural Netw. Learn. Syst.* **26**(10), 2550–2562 (2015)

37. Bertram, J., Sarachik, P.: Stability of circuits with randomly time-varying parameters. *IEEE Trans. Circuit Theory* **6**(5), 260–270 (1959)
38. Kozin, F.: A survey of stability of stochastic systems. *Automatica* **5**(1), 95–112 (1969)
39. Liu, M., Wan, Y., Lewis, L.F., Victor, L.: Adaptive optimal control for stochastic multi-agent differential games using on-policy and off-policy reinforcement learning. *IEEE Trans. Neural Netw. Learn. Syst.* (2020)
40. Li, J., Modares, H., Chai, T., Lewis, F.L., Xie, L.: Off-policy reinforcement learning for synchronization in multiagent graphical games. *IEEE Trans. Neural Netw. Learn. Syst.* **28**, 2434–2445 (2017)
41. Luo, B., Wu, H.-N., Huang, T.: Off-policy reinforcement learning for  $H_\infty$  control design. *IEEE Trans. Cybern.* **45**(1), 65–76 (2015)
42. Chen, J., Xie, J., Gu, Y., Li, S., Fu, S., Wan, Y., Lu, K.: Long-range and broadband aerial communication using directional antennas (ACDA): design and implementation. *IEEE Trans. Veh. Technol.* **66**(12), 10793–10805 (2017)
43. Li, S., He, C., Liu, M., Wan, Y., Gu, Y., Xie, J., Fu, S., Lu, K.: The design and implementation of aerial communication using directional antennas: learning control in unknown communication environments. *IET Control Theory Appl.* **13**(17), 2906–2916 (2019)
44. Liu, M., Wan, Y., Li, S., Lewis, L.F., Fu, S.: Learning and uncertainty-exploited directional antenna control for robust long-distance and broad-band aerial communication. *IEEE Trans. Veh. Technol.* **69**(1), 593–606 (2019)
45. Rappaport, T.S., et al.: *Wireless Communications: Principles and Practice*, vol. 2. PTR, New Jersey (1996)

# Chapter 14

## A Top-Down Approach to Attain Decentralized Multi-agents



Alex Tong Lin, Guido Montúfar, and Stanley J. Osher

**Abstract** In the modern age with the enormous amounts of data being collected, machine learning has blossomed into a necessary tool in nearly all academic and industrial fields. In many cases, machine learning deals with a static dataset, e.g., a fixed training data, for its analysis, such as in recognition of hand-written digits. Yet there is also a need to apply machine learning to streams of data, i.e., online learning. One approach is that of reinforcement learning, the setting being an agent seeking to maximize a cumulative reward signal it receives from the environment. An extension of this notion is multi-agent reinforcement learning, where now the setting is that we have many agents that seek to maximize their cumulative reward signal; the stipulation being that each agent can only observe its own local observation and communication from other agents which may have limited bandwidth, i.e., the agents must act in a decentralized manner. In this chapter, we demonstrate a method, centralized expert supervises multi-agents (CESMA), to obtain decentralized multi-agents through a top-down approach; we first obtain a solution with a centralized controller, and then decentralize this using imitation learning.

---

A. T. Lin (✉) · G. Montúfar · S. J. Osher

Department of Mathematics, University of California, Los Angeles, CA 90095, USA  
e-mail: [atlin@math.ucla.edu](mailto:atlin@math.ucla.edu)

G. Montúfar  
e-mail: [montufar@math.ucla.edu](mailto:montufar@math.ucla.edu)

S. J. Osher  
e-mail: [sjo@math.ucla.edu](mailto:sjo@math.ucla.edu)

G. Montúfar  
Department of Statistics, University of California, Los Angeles, CA 90095, USA  
Max Planck Institute for Mathematics in the Sciences, 04103 Leipzig, Germany

## 14.1 Introduction

In the present age, with increasing computational power, we are collecting vast amounts of data driven by the need to model and predict. Whether it be from finding the best recommendation for a customer, or diagnosing a patient with the correct disease given their medical history, or determining the best investment strategy given sales data, the list goes on. And in order to analyze this data, the field of machine learning has exploded in popularity, both in the number of researchers and the number of users, and this has recently resulted in advances in many fields, such as biology, computer vision, and robotics. Indeed, we have only recently obtained machines that outperform humans in such tasks as classifying images or playing the game Go [22, 23]. And advances are still being made, for reasons such as building self-driving cars or bettering speech recognition systems.

Machine learning has an enormous amount of subfields, and many of them deal with data that is “offline,” in the sense that the training data is fixed, and the goal is to obtain an algorithm or method that uses this training data to obtain a model that performs well on test data. Yet, there is also a need to deal with data that is yet-to-come, i.e., data that is coming as a stream. And we seek to learn from past data in order to perform well on incoming future data, i.e., “online learning.” A popular field within this online learning framework is the field of reinforcement learning. In this setting there is an agent interacting with its environment, and it is also receiving a reward signal from the environment based on its states, and the actions it takes. The goal of this agent (and reinforcement learning) is to obtain a policy that maximizes its cumulative reward. We note this is distinct from planning problems, where most of the time a model of the environment is provided to the agent; in reinforcement learning the agent does not have a model.

Many algorithms have been proposed for Reinforcement Learning [19, 24, 28, 29], and it has been paired up with using neural networks [6, 7, 12, 13, 27] in order to achieve superhuman performance on tasks, such as the games of Go [3]. There is still a flurry of research in Reinforcement Learning, especially in real-world applications such as in robotics [4]. Going beyond the single-agent setting, we have multi-agent reinforcement learning.

In the multi-agent reinforcement learning setting [1, 15, 21], we now have many agents interacting with the environment, and each agent seeks to maximize its own cumulative reward. The setting can be both collaborative or competitive. And the end goal of a MARL algorithm is to find a policy for each agent that will maximize the agent’s reward. But a problem that is inherent in the multi-agent setting is that there are other agents in the environment, and so each agent must try to predict other agents’ behavior (directly or indirectly) if it wants to maximize its cumulative reward. This can cause problems, as assumptions made in the reinforcement learning framework are inherently violated in the multi-agent setting, i.e., the Markovian property is violated; this is elaborated in Sect. 14.2.2.

In this chapter, we will provide an overview of reinforcement learning and multi-agent reinforcement learning. We first provide background on the theory in both

these settings, and also discuss the problem of moving from a single-agent problem to a multi-agent problem. We then discuss the approach of centralized learning, but decentralized execution [5], and a couple algorithm under this paradigm. We then discuss an algorithm called CESMA which seeks to obtain decentralized agents by decentralizing a centralized expert with imitation learning.

## 14.2 Background

Here we provide background on reinforcement learning and multi-agent reinforcement learning, noting also the issues one faces when moving from a single-agent problem to a multi-agent one.

### 14.2.1 Reinforcement Learning

Reinforcement learning (RL) is concerned with finding a policy for an agent interacting with an environment such that it maximizes its cumulative reward. Namely, we have an agent interacting with an environment which provides a state to the agent, to which the agent then provides an action. Upon receiving the action, the environment then produces a reward along with the next state. This repeats until termination condition is met, or it may continue forever, depending on the problem. The goal of the agent is to maximize *cumulative* reward. The most considered setting of the RL problem is the Markov decision process (MDP), in which case the environment has the Markovian property, which states that the current state and action contains all information required to produce the next state. Non-Markovian models will be discussed later, and are inherent in multi-agent reinforcement learning.

In mathematical terms (here we borrow from [25]), an MDP is a tuple  $(\mathcal{S}, \mathcal{A}, P, R)$ , where

- $\mathcal{S}$  is the set of possible states,
- $\mathcal{A}$  is the set of possible actions, which we assume is the same no matter the state (one may also consider different action sets for each state, i.e.,  $\mathcal{A} = \mathcal{A}_s$  for each  $s \in \mathcal{S}$ ),
- $R = R(s, a) = r$  is the instantaneous reward function (one can also consider reward functions that depend on the next state  $s'$  as well, i.e.,  $R = R(s', s, a)$ ). We denote a value of  $R$  as  $r$ , and
- $P = P(s', r|s, a)$  is the dynamics of the MDP, which describes how the system behaves.

We will mainly be dealing with *finite* MDPs, where the set of states and actions is finite. Then the goal is to find a policy  $\pi : \mathcal{S} \rightarrow \mathcal{P}(\mathcal{A})$  (where  $\mathcal{P}(\mathcal{A})$  is the space of probability distributions over  $\mathcal{A}$  that maximizes the *cumulative* reward, i.e., for every state  $s \in \mathcal{S}$ ,  $\pi(s)$  is a probability over all actions)

$$\pi^* = \arg \max_{\pi} \mathbb{E}_{\pi} \left[ \sum_t \gamma^t R(s_t, a_t) \right],$$

where  $\gamma \in (0, 1)$  is a discounting term, so that the sum is sure to converge in cases when the task is never ending, and  $s_t$  and  $a_t \sim \pi(s_t)$  can be thought of as coming from the distribution of states and actions at time  $t$  as a result of following policy  $\pi$ . We also implicitly have a distribution over the starting state. For some start state  $s$ , the term inside the argmax is called the *state-value function for policy  $\pi$* :

$$v_{\pi}(s) = \mathbb{E}_{\pi} \left[ \sum_t \gamma^t R(s_t, a_t) \mid s_0 = s \right].$$

Now one can reformulate the problem of RL as finding a policy  $\pi^*$  that maximizes  $v_{\pi^*}(s)$  for every possible state  $s \in \mathcal{S}$ . It turns out that for *finite* MDPs (where the set of states and actions are finite), one can define a partial ordering over the space of policies via the state-value function:  $\pi' \geq \pi$  if and only if  $v_{\pi'}(s) \geq v_{\pi}(s)$  for all  $s \in \mathcal{S}$ . Then one can define an *optimal policy*  $\pi^*$  as one which is maximal under this partial ordering. And it further turns out that (for finite MDPs) all maximal policies have the same state-value function, which we denote  $v_* = \max_{\pi} v_{\pi}(s)$ .

If we wanted to include the action, then we have the *action-value function for policy  $\pi$* :

$$q_{\pi}(s, a) = \mathbb{E}_{\pi} \left[ \sum_t \gamma^t R(s_t, a_t) \mid s_0 = s, a_0 = a \right],$$

and the two value functions are related by  $v_{\pi}(s) = \mathbb{E}_{a \sim \pi(s)} [q_{\pi}(s, a)]$ . We can denote the optimal action-value function by  $q_*(s, a) = \max_{\pi} q_{\pi}(s, a)$ , and thus we have  $v_*(s) = \max_a q_*(s, a)$ .

The optimal state-value function enjoys the *Bellman optimality condition*:

$$v_*(s) = \max_a \sum_{s', r} P(s', r | s, a) (r + \gamma v_*(s')),$$

or if one considers a deterministic environment (i.e., the dynamics function  $P$  is deterministic), then the above simplifies into

$$v_*(s) = R(s, a) + \gamma v_*(s'),$$

The optimal action-value function  $q_*(s, a)$ , being related to  $v_*$ , also enjoys the Bellman optimality condition,

$$q_*(s, a) = \sum_{s', r} P(s', r | s, a) \left( r + \gamma \max_{a'} q_*(s', a') \right),$$

and under the deterministic setting the above simplifies into

$$q_*(s, a) = R(s, a) + \gamma \max_{a'} q_*(s', a').$$

Many RL algorithms attempt to find policies that maximize the value functions, or have them satisfy the Bellman optimality conditions.

### 14.2.2 Multi-agent Reinforcement Learning

In multi-agent reinforcement learning (MARL), we move from problems concerning a single agent, to problems concerning multiple agents. In this setting, we have multiple agents that interact with an environment, and the goal is to find a policy for each agent that maximizes its own reward. In some cases, we have a collaborative multi-agent problem and here we have a global reward function that is shared among all agents and thus the goal is to maximize this shared reward; otherwise each agent will have its own reward function and the goal is to find an equilibrium (e.g., a Nash equilibrium).

There are two main issues when dealing with multiple agents: (1) the exponential growth in state and action space as the number of agents increases and (2) the violation of the Markovian assumption from the perspective of each agent.

The exponential growth in state and action space occurs because we now have to consider the joint state space, enumerating all possible combinations of states the multi-agents together can hold, as well as the joint action space, enumerating all possible combinations of actions the multi-agents together can do. The violation of the Markovian property is due to the fact that during the learning phase, from the perspective of one agent, other agents comprise the environment and are also learning. Thus the behavior of these other agents are not Markovian, as the one agent cannot assume that the current state is enough information to predict the next state, even if it observes the whole environment (excluding the “brain” of each agent). This means each agent is in a partially observable Markov decision process (POMDP).

In mathematical terms (we borrow from [14] where they describe decentralized POMDPs (Dec-POMDPs), and [9] where they describe Markov games, similar to the single-agent RL setting, the MARL setting can be described by a tuple: If we have  $M$  agents, and the agents are indexed by  $i$ , then we have the tuple  $(\mathcal{S}, (\mathcal{O}_i), (\mathcal{A}_i), P, (R_i))$ , where

- $\mathcal{S}$  is the set of all joint states possible by the multi-agents. If we denote  $\mathcal{S}_i$  as the possible states for agent  $i$ , then  $\mathcal{S} = \mathcal{S}_1 \times \dots \times \mathcal{S}_M$ ,
- $\mathcal{O}_i = \mathcal{O}_i(s)$  for  $s \in \mathcal{S}$  is the observation seen by agent  $i$  when the multi-agents are in state  $s \in \mathcal{S}$ ,
- $\mathcal{A}_i$  is the set of actions for agent  $i$ ,
- $P = P(s', (r_i) \mid s, (a_i))$  is the dynamics of the environment, where  $r_i$  is the reward given to agent  $i$ , and  $a_i$  is the action taken by agent  $i$ , and
- $R_i = R(s, (a_i)) = r_i$  is the reward function given to agent  $i$ .

We can immediately see the explosion in the joint state and action spaces, as the number of states is  $|\mathcal{S}| = \prod_i |\mathcal{S}_i|$ , and the number of actions is  $|\mathcal{A}_1 \times \cdots \times \mathcal{A}_M| = \prod_i |\mathcal{A}_i|$ . A multi-agent policy will be denoted  $\pi(s) = (\pi_1, \dots, \pi_M) : S \rightarrow A_1 \times \cdots \times A_M$ , where

$$\pi(s) = (\pi_1(\mathcal{O}_1(s)), \dots, \pi_M(\mathcal{O}_M(s))), \quad s \in \mathcal{S}.$$

(And we can consider these to be stochastic policies.)

We can also define the state-value and action-value functions for each agent: Each agent  $i$  wants to maximize its cumulative reward, so we have the state-value function for agent  $i$ ,

$$v_{\pi_i}(s) = \mathbb{E}_{\pi=(\pi_1, \dots, \pi_M)} \left[ \sum_t \gamma^t R(s_t, \{a_{j,t}\}) \mid s_0 = s \right],$$

and the action-value function for agent  $i$ ,

$$q_{\pi_i}(s, a) = \mathbb{E}_{\pi=(\pi_1, \dots, \pi_M)} \left[ \sum_t \gamma^t R(s_t, \{a_{j,t}\}) \mid s_0 = s, a_{i,0} = a, \text{ and } a_{j,0} = \pi_j(s) \ (j \neq i) \right].$$

### 14.3 Centralized Learning, But Decentralized Execution

The end goal of MARL is to obtain *decentralized* policies for the multi-agents, namely, a policy for each agent that only takes as input the local observation of the agent:

$$(\pi_1, \dots, \pi_M)(s) = (\pi_1(\mathcal{O}_1(s)), \dots, \pi_M(\mathcal{O}_M(s))).$$

But as mentioned above, due to the violation of the Markovian property, this presents convergence problems during training as each agent cannot assume that the environment (which includes the other agents) is Markovian, as other agents are also learning and updating their own policies. (We note this is distinct from non-stationarity of Markov chains, where the agent needs to take into account the time variable—here that is not enough.)

To tackle the instability of the learning phase, an approach called *centralized learning, but decentralized execution* [5] has become popular, where during training the agents are able to use any available information, but after training is complete, the agents are only allowed to use their local observation when executing their policies. We describe two popular approaches under this framework: a bottom-up approach, where the solution to the multi-agent problem is constructed simultaneously along with learning, and the top-down approach where a solution is first found, which we call the expert, and then the multi-agents learn in a decentralized fashion from this expert.

### 14.3.1 A Bottom-Up Approach

Many popular methods today take a “bottom-up” approach to obtaining decentralized multi-agents [10, 11, 26]. One popular approach is called multi-agent deep deterministic policy gradient (MADDPG) [10], which uses a modified actor–critic method to train the multi-agents. During the learning phase, the critic takes in the joint observations and actions of the multi-agents, but the actor only takes in its local observation. An analogy for this approach is that the critic is like a coach that can observe the whole environment, and feeds this information to the actor, and the actor adjusts its behavior accordingly. At the end of training, we only need the actors, and they can act in a decentralized manner as they did in the learning phase.

### 14.3.2 A Top-Down Approach

A top-down approach to MARL is in terms of decentralizing a centralized expert [2, 8, 16]. An approach we discuss here is centralized expert supervises multi-agents (CESMA) [8], which first obtains a centralized expert policy either through training, or obtained analytically (or any other way), and then it uses an imitation learning algorithm to decentralize the multi-agents. They also obtain bounds on the size of the dataset of trajectories needed. This top-down view separates finding a solution (centralized or not) to the multi-agent problem from the learning phase of the multi-agents, unlike the bottom-up approach. CESMA also allows the expert to take in any available information, and its output will be an action for each agent; thus this centralized agent is unrestricted in its ability to synthesize information from all agents in order to produce the best solution to the multi-agent problem. And the method does not impose any structural conditions on the centralized expert—it can even be an analytical solution.

## 14.4 Centralized Expert Supervises Multi-agents

### 14.4.1 Imitation Learning

Imitation learning [17, 18, 20] attempts to train an agent in order to learn from expert demonstrations. In this setting, the agent will have a dataset of observations with action labels that come from the expert, and the goal of the agent is to match the expert as well as possibly. This is usually done by performing supervised learning on the learner with the dataset being observation and action-label pairs. And the agent must minimize a loss function (sometimes called the surrogate loss), in order to obtain good cumulative rewards from the environment. Imitation learning has been used successfully, such as in self-driving cars.

A simple method in imitation learning is to collect a dataset of trajectories from the expert, and then use this dataset for supervised learning of the agent. The advantage of this method is in its simplicity, but it does suffer from the fact that there will be a mismatch in the distribution of states seen by the expert (who by assumption acts perfectly) and the learning agent (who by assumption will indeed make mistakes). So indeed the learning agent can train on expert trajectories, but it will see a different distribution of states while it is acting and the learning agent may not generalize well enough in these situations.

In order to alleviate this issue, one can collect a dataset of trajectories from the learner agent, and have the expert label the observations with its own expert actions. Then there is no mismatch between the learner's trajectories and its dataset. This is the method of dataset aggregation (DAgger) [18], which at iteration  $i$  collects trajectories from agent  $\pi^{(i)}$ , labels it with expert actions, and then appends these to an aggregated dataset  $\mathcal{D}$ . It then trains a policy  $\pi^{(i+1)}$  on this new dataset, and then repeats the process. We combine this with multi-agent learning in order to obtain the top-down approach for attaining decentralized multi-agents.

#### 14.4.2 CESMA

CESMA takes the top-down approach to MARL by first obtaining a centralized expert that solves the multi-agent environment—i.e., finding the optimal action for each agent given a (joint) state—although not, presumably, in a decentralized fashion (or else we would not need a decentralization step). This solution can be found through training a neural network (as we do in the experiments), or analytically if one has a model of the environment and it is simple enough, or any other way the imagination can procure. Then we do imitation learning on the multi-agents, but in this learning phase the multi-agents can only observe their own local observation. The imitation learning in our experiments is achieved through supervised learning with an aggregated dataset, like DAgger. We provide a pseudo-algorithm in Algorithm 14.1.

---

**Algorithm 14.1** CESMA: centralized expert supervises multi-agents

---

**Require:** A centralized policy  $\pi^*$  that sufficiently solves the environment.  
**Require:**  $M$  agents  $\pi_{\theta_1}, \dots, \pi_{\theta_M}$  (with parameters  $\theta_1, \dots, \theta_M$ ), observation buffer  $\mathcal{D}$  for multi-agent observations, batch size  $B$

- 1: **while**  $\pi_{\theta_1}, \dots, \pi_{\theta_M}$  not converged **do**
- 2:   Obtain observations  $o_1, \dots, o_M$  from the environment
- 3:   Obtain agents' actions,  $a_1 = \pi_{\theta_1}(o_1), \dots, a_M = \pi_{\theta_M}(o_M)$
- 4:   Obtain expert action labels  $a_i^* = \pi^*(o_1, \dots, o_M)_i$ , for  $i = 1, \dots, M$
- 5:   Store the joint observation with expert action labels  $((o_1, a_1^*), \dots, (o_M, a_M^*))$  in  $\mathcal{D}$
- 6:   **if**  $|\mathcal{D}|$  sufficiently large **then**
- 7:     Sample a batch of  $B$  multi-agent observations  $\{(o_1^{(\beta)}, a_1^{*(\beta)}), \dots, (o_M^{(\beta)}, a_M^{*(\beta)})\}_{\beta=1}^B$
- 8:     Perform supervised learning for  $\pi_{\theta_i}$  using the observation-label pairs  $\{(o_i^{(\beta)}, a_i^{*(\beta)})\}_{\beta=1}^B$ .
- 9:   **end if**
- 10: **end while**

---

CESMA has also been used to train agents that can communicate—the main idea is to backpropagate the learning loss of a receiving agent through the communication channel and to the parameters of the broadcasting agent.

## 14.5 Experiments

Here we apply our aforementioned techniques to a multi-agent gridworld where agents must cover landmarks. At the start of each episode, the agents and landmarks spawn randomly in a  $10 \times 10$  grid. The reward is shared among all agents, and is calculated by how far an agent is from each landmark, namely, the reward function (or punishment) at each timestep is,

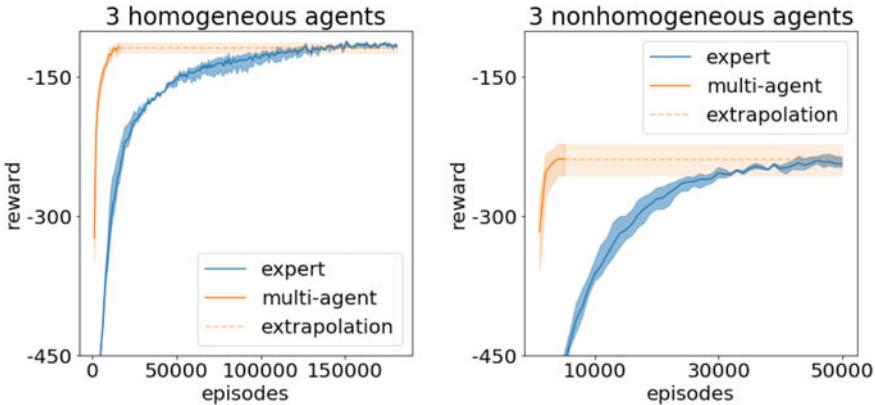
$$\text{reward at each timestep} = (-1) \cdot \sum_{j=1}^M \min_{i=1,\dots,M} \{\|\text{agent}_i - \text{landmark}_j\|\},$$

where  $\text{agent}_i$  is the position of agent  $i$ , and similarly for  $\text{landmark}_j$ . We also give a reward (or punishment) of  $-10$  whenever the agents collide (whenever two agents have the same position).

We also have a variation of an environment with lava in the shape of a square, with corners at positions  $(2, 2)$  and  $(7, 7)$ . The agents are given a reward of  $-100$  for each agent in the lava. In Fig. 14.1, we provide pictorial examples of the environments.



**Fig. 14.1** Examples of the environment used. On the **left** we see a multi-agent gridworld where the agents (orange) must move to the landmarks (blue). On the **right**, we modified the environment to have lava (red), and the agents are heavily punished if they are on these lava grid positions



**Fig. 14.2** On the **left**, we show reward curves plotted for homogeneous multi-agents. As we can see, the multi-agents are able to achieve the same reward as the centralized expert. The dashed line is a visual aid that represents extrapolation of the multi-agent reward. The experiments were averaged on training three centralized experts and decentralizing each of them. The envelopes represent max and min values. On the **right**, we again find the same result for nonhomogeneous agents, where now some agents skip over gridpoints as they move, representing faster agents. And we again find that the multi-agents are able to achieve the same reward as the centralized expert

### 14.5.1 Decentralization Can Achieve Centralized Optimality

Here we provide experiments demonstrating the effectiveness of the method in obtaining decentralized multi-agents; namely, we find that in the setting of our experiment, the agents are able to obtain the same reward as the centralized experts. We provide reward curves in Fig. 14.2, which are averaged over decentralizing three centralized experts, which were trained; the envelopes represent max and min values. In one environment we experiment on homogeneous agents (i.e., with the same properties, but not sharing the same policy necessarily) and find that indeed the multi-agents are able to obtain the same reward as the centralized expert. We also tried an example with nonhomogeneous agents, where now some agents skip over gridpoints when they move. These agents in a sense represent more agile agents, although due to the nature of gridworld, skipping means some agents are not able to exactly cover a landmark. In Fig. 14.2, we again see that the nonhomogeneous agents are able to obtain the same reward as the centralized expert. The dashed line is a visual aid that extrapolates the reward of the multi-agents.

### 14.5.2 Expert Trajectories Versus Multi-agent Trajectories

In this experiment, we demonstrate that it is better to use a dataset of trajectories that comes from the multi-agents' own distribution, rather than using a dataset of trajectories that come from the centralized expert. We now use the multi-agent grid-

**Table 14.1** Here we show a table comparing multi-agents trained on expert trajectories versus a database of their own trajectories with expert action labels. We find that training on the multi-agents' own trajectories provides almost double the sample efficiency of the number of episodes

	Number of episodes to reach expert reward
Expert trajectories	58,000
Multi-agent trajectories	34,000

world environment *with lava*, and each agent that is in the lava receives a reward (or punishment) of  $-100$ . The idea is that the centralized expert will learn to avoid the extremely costly lava positions, and its trajectories will have no lava. So if the multi-agents only train on expert trajectories, since they make mistakes during training, they will not know what to do on lava positions, and thus this slows down training. This is in contrast to when the multi-agents train on their own state distributions with expert labels, and in this case the multi-agents will now have guidance on lava positions.

In Table 14.1, we plot tables for multi-agents trained on their own distribution of states, versus trained on a dataset of expert trajectories. We stop training once the same reward as the centralized expert is obtained. We see that training on the multi-agents' own dataset of trajectories provides almost double the sample efficiency versus training on the expert trajectories.

## 14.6 Conclusion

In this age, enormous amounts of data are being collected, and therefore machine learning has become tremendously important in nearly all academic and industrial fields. Reinforcement learning has become especially popular in constructing methods to solve problems in a variety of areas, including robots and computer vision. And multi-agent reinforcement learning is an important subfield, and here we provide a description of how to decentralize a centralized expert/controller by aggregating trajectories of multi-agents in a dataset, which is then used for supervised learning. We also gave experimental evidence that this decentralization procedure is able to obtain multi-agents that achieve the same optimum as the expert, and is much faster than training on expert trajectories. Going forward, there are many extensions possible to this framework, such as seeking faster methods to find optimum, or extending to an enormous amount of multi-agents.

## References

- Bu, L., Babu, R., De Schutter, B., et al.: A comprehensive survey of multiagent reinforcement learning. *IEEE Trans. Syst. Man Cybern. Part C (Appl. Rev.)* **38**(2), 156–172 (2008)
- Dobbe, R., Fridovich-Keil, D., Tomlin, C.: Fully decentralized policies for multi-agent systems: an information theoretic approach. In: Guyon, I., Luxburg, U.V., Bengio, S., Wallach, H., Fergus, R., Vishwanathan, S., Garnett, R. (eds.) *Advances in Neural Information Processing Systems*, vol. 30, pp. 2941–2950. Curran Associates, Inc., New York (2017)
- Evans, R., Gao, J.: Deepmind AI reduces Google data centre cooling bill by 40 (2017)
- Kober, J., Bagnell, J.A., Peters, J.: Reinforcement learning in robotics: a survey. *Int. J. Robot. Res.* **32**(11), 1238–1274 (2013)
- Kraemer, L., Banerjee, B.: Multi-agent reinforcement learning as a rehearsal for decentralized planning. *Neurocomputing* **190**, 82–94 (2016)
- Levine, S., Finn, C., Darrell, T., Abbeel, P.: End-to-end training of deep visuomotor policies. *J. Mach. Learn. Res.* **17**(1), 1334–1373 (2016)
- Lillicrap, T.P., Hunt, J.J., Pritzel, A., Heess, N., Erez, T., Tassa, Y., Silver, D., Wierstra, D.: Continuous control with deep reinforcement learning. *CoRR* (2015). [arXiv:1509.02971](https://arxiv.org/abs/1509.02971)
- Lin, A.T., Debord, M.J., Estabridis, K., Hewer, G.A., Osher, S.J.: CESMA: centralized expert supervises multi-agents. *CoRR* (2019). [arXiv:1902.02311](https://arxiv.org/abs/1902.02311)
- Littman, M.L.: Markov games as a framework for multi-agent reinforcement learning. In: Cohen, W.W., Hirsh, H. (eds.) *Machine Learning Proceedings 1994*, pp. 157–163. Morgan Kaufmann, San Francisco (1994)
- Lowe, R., Wu, Y., Tamar, A., Harb, J., Abbeel, P., Mordatch, I.: Multi-agent actor-critic for mixed cooperative-competitive environments. *CoRR* (2017). [arXiv:1706.02275](https://arxiv.org/abs/1706.02275)
- Matignon, L., Laurent, G.J., Le Fort Piat, N.: Review: independent reinforcement learners in cooperative Markov games: a survey regarding coordination problems. *Knowl. Eng. Rev.* **27**(1), 1–31 (2012)
- Mnih, V., Kavukcuoglu, K., Silver, D., Graves, A., Antonoglou, I., Wierstra, D., Riedmiller, M.A.: Playing Atari with deep reinforcement learning. *CoRR* (2013). [arXiv:1312.5602](https://arxiv.org/abs/1312.5602)
- Mnih, V., Kavukcuoglu, K., Silver, D., Rusu, A.A., Veness, J., Bellemare, M.G., Graves, A., Riedmiller, M., Fidjeland, A.K., Ostrovski, G., et al.: Human-level control through deep reinforcement learning. *Nature* **518**(7540), 529 (2015)
- Oliehoek, F.A.: Decentralized POMDPs. *Reinforcement Learning*, pp. 471–503. Springer, Berlin (2012)
- Panait, L., Luke, S.: Cooperative multi-agent learning: the state of the art. *Auton. Agents Multi-agent Syst.* **11**(3), 387–434 (2005)
- Paulos, J., Chen, S.W., Shishika, D., Kumar, V.: Decentralization of multiagent policies by learning what to communicate (2018)
- Ross, S., Bagnell, D.: Efficient reductions for imitation learning. In: *Proceedings of the Thirteenth International Conference on Artificial Intelligence and Statistics*, pp. 661–668 (2010)
- Ross, S., Gordon, G.J., Bagnell, J.A.: No-regret reductions for imitation learning and structured prediction. *CoRR* (2010). [arXiv:1011.0686](https://arxiv.org/abs/1011.0686)
- Rummery, G.A., Niranjan, M.: On-line Q-learning using connectionist systems, vol. 37. University of Cambridge, Department of Engineering Cambridge, England (1994)
- Schaal, S., Ijspeert, A., Billard, A.: Computational approaches to motor learning by imitation. *Philos. Trans. R. Soc. Lond. Ser. B: Biol. Sci.* **358**(1431), 537–547 (2003)
- Shoham, Y., Leyton-Brown, K.: *Multiagent Systems: Algorithmic, Game-Theoretic, and Logical Foundations*. Cambridge University Press, New York (2008)
- Silver, D., Huang, A., Maddison, C.J., Guez, A., Sifre, L., Van Den Driessche, G., Schrittwieser, J., Antonoglou, I., Panneershelvam, V., Lanctot, M., et al.: Mastering the game of go with deep neural networks and tree search. *Nature* **529**(7587), 484 (2016)
- Silver, D., Schrittwieser, J., Simonyan, K., Antonoglou, I., Huang, A., Guez, A., Hubert, T., Baker, L., Lai, M., Bolton, A., et al.: Mastering the game of go without human knowledge. *Nature* **550**(7676), 354 (2017)

24. Sutton, R.S.: Learning to predict by the methods of temporal differences. *Mach. Learn.* **3**(1), 9–44 (1988)
25. Sutton, R.S., Barto, A.G.: Reinforcement Learning: An Introduction. MIT Press, Cambridge (2018)
26. Tan, M.: Readings in agents. In: Huhns, M.N., Singh, M.P. (eds.) *Multi-agent Reinforcement Learning: Independent Versus Cooperative Agents*, pp. 487–494. Morgan Kaufmann Publishers Inc., San Francisco (1998)
27. Wang, Z., Schaul, T., Hessel, M., Van Hasselt, H., Lanctot, M., De Freitas, N.: Dueling network architectures for deep reinforcement learning (2015). [arXiv:1511.06581](https://arxiv.org/abs/1511.06581)
28. Watkins, C.J.C.H., Dayan, P.: Q-learning. *Mach. Learn.* **8**(3), 279–292 (1992)
29. Williams, R.J.: Simple statistical gradient-following algorithms for connectionist reinforcement learning. *Mach. Learn.* **8**, 09 (1998)

# Chapter 15

## Modeling and Mitigating Link-Flooding Distributed Denial-of-Service Attacks via Learning in Stackelberg Games



Guosong Yang and João P. Hespanha

**Abstract** This work formulates the mitigation of link-flooding distributed denial-of-service attacks as a routing problem among parallel links. In order to address the challenge that the adversary can observe the routing strategy before assigning attack traffic, we model the conflict between routing and attack as a Stackelberg game. For a general class of adversaries, we establish a characterization of an optimal attack strategy that reduces the search space to a finite set, and construct explicit formulae for Stackelberg equilibria and costs for a special class of networks. When the attack objective and capacity are unknown, we propose a learning-based approach that predicts the routing cost using a neural network and minimizes the predicted cost via projected gradient descent. A simulation study is provided to demonstrate the effectiveness of our approach.

### 15.1 Introduction

A major threat to Internet security today is the distributed denial-of-service (DDoS) attack, in which an adversary attempts to interrupt legitimate users' access to certain network resources by sending superfluous traffic from a vast number of subverted machines (bots). Since the first incident of DDoS attack reported by the Computer Incident Advisory Capability in 2000 [1], many large-scale DDoS attacks have been launched against crucial infrastructures and services [2, 3]. Moreover, there has been a drastic and persistent increase in the size and frequency of DDoS attacks every

---

G. Yang (✉) · J. P. Hespanha

Center for Control, Dynamical Systems, and Computation, University of California,  
Santa Barbara, CA 93106, USA  
e-mail: [guosongyang@ucsb.edu](mailto:guosongyang@ucsb.edu)

J. P. Hespanha  
e-mail: [hespanha@ucsb.edu](mailto:hespanha@ucsb.edu)

year. The 14th annual Worldwide Infrastructure Security Report from NETSCOUT Systems, Inc. showed that the global max attack size reached 1.7 Tbps in 2018, a 273% increase from 2017 [4]. The same report also found that 94% of enterprises observed DDoS attacks on their encrypted traffic in 2018, nearly twice the percentage as 2017.

As pointed out in [3], most DDoS attacks exploit one or both of the following methods: (1) disrupting legitimate users' service by exhausting server resources and (2) disrupting legitimate users' connectivity by exhausting link bandwidth. The second method (called link-flooding DDoS) proves to be especially effective and stealthy as it can be executed without sending attack traffic to the victims. Novel link-flooding DDoS attacks such as the Coremelt attack [5] and the Crossfire attack [6] have attracted substantial research interest, since they allow individual bots participating in the coordinated attack to keep their transmission rates below detection thresholds, yet effectively exhausts the bandwidth of target links. Existing results on mitigating link-flooding DDoS attacks mainly focus on developing techniques to distinguish attack traffic from legitimate one [7–12]. However, the adversary can often rotate attack traffic among different sets of bots and links to maintain congestion [6], which motivates us to consider scenarios where the detection of attack traffic is impossible. In [13], the authors studied the modeling and mitigation of the Coremelt attack [5] in such scenarios.

Game theory provides a systematic framework for modeling the conflict between a router and an adversary orchestrating a DDoS attack [14–17]. Existing game-theoretical results on mitigating DDoS attacks mainly focus on Nash equilibrium—a tuple of actions for which no player has a unilateral incentive to change, see, e.g., [18–20] and references therein. However, most DDoS attacks are characterized by asymmetric information: the router is unaware of the attack objective and capacity *a priori*, whereas the adversary is able to observe the routing action and assign attack traffic accordingly [6]. Since asymmetric information often leads to scenarios with no Nash equilibrium, we consider instead a hierarchical game model proposed by Stackelberg [21]. In our two-player Stackelberg game, the router (called the leader) selects its action first, and then the adversary (called the follower), informed of the router's choice, selects its own action. Standard conditions for existence of a Stackelberg equilibrium are weaker than those of a Nash equilibrium [16, p. 181].

Stackelberg games have been applied to network problems with asymmetric information in applications such as routing [22], scheduling [23], and channel allocation [24]. They have also been applied to various real-world security domains and have lead to practical implementations including the ARMOR program at the Los Angeles International Airport [25], the IRIS program used by the US Federal Air Marshals [26], and counterterrorism programs for crucial infrastructures such as power grid and oil reserves [27, 28]. A recent work [29] analyzed Nash and Stackelberg equilibria for a routing game that is similar but more restrictive than the one considered in this work.

Since the router only has incomplete information of the attack objective and capacity, a fundamental question is whether an iterative data-driven routing algorithm can converge to a Stackelberg equilibrium as it adjusts routing based on historical out-

comes. For our Stackelberg game, standard game-theoretical learning processes such as fictitious play [30, 31] and gradient response [32, 33] cannot be applied. Most existing results on learning in Stackelberg games are limited to linear and quadratic costs and finite action sets [34–36], which are too restrictive for modeling the DDoS attacks of interest. In [37], the authors proposed a learning-based approach for Stackelberg games that could simultaneously estimate the attack strategy and minimize the routing cost, based on adaptive control techniques and hysteresis switching. A motivation for the current work is to extend the results from [37] for more complicated problems such as mitigating link-flooding DDoS attacks based on neural network techniques.

In this work, we formulate the mitigation against link-flooding DDoS attacks as a routing problem among parallel links. Our goal is to construct an optimal routing strategy that minimizes the damage caused by the attack, with the caveat that the adversary is able to observe the routing strategy before assigning attack traffic. The network concepts and the routing and attack optimization problems are described in Sect. 15.2. Our formulation provides a high level of generality by allowing scenarios where the “attack” is actually benign network traffic or a combination of benign and malicious traffic.

In Sect. 15.3, we model the conflict between routing and attack as a Stackelberg game. In Stackelberg games, it is common for the follower to have multiple optimal actions against a leader’s action, leading to different equilibria for “optimistic” or “pessimistic” leaders that assume the follower will play the optimal actions that are best or worst for the leader, respectively. Therefore, we adopt the notions of strong and weak Stackelberg equilibria, in which the former corresponds to an optimistic router and the latter a pessimistic one.

In Sect. 15.4, we consider a general class of adversaries that may have different priorities for different links, and establish a characterization of an optimal attack strategy by showing that it belongs to a finite set. Moreover, we construct explicit formulae for strong and weak Stackelberg equilibria and costs for a special class of networks. Our results show that there is no weak Stackelberg equilibrium unless the adversary has the same priority for all links, in which case the strong Stackelberg equilibrium is also a weak one. Nevertheless, a pessimistic router can always achieve a cost arbitrarily close to the strong Stackelberg cost.

In Sect. 15.5, we considered the more general scenario where the router does not know the attack objective or capacity and thus cannot predict the attack action or the cost associated with a specific routing action. We propose a learning-based approach that predicts the routing cost using a neural network trained based on historical data, and then minimizes the predicted routing cost via projected gradient descent. A simulation study is provided in Sect. 15.6 to demonstrate the effectiveness of our learning-based approach. Section 15.7 concludes the chapter with a brief summary and an outlook for future research directions.

*Notations:* Let  $\mathbb{R}_{\geq 0}$ ,  $\mathbb{R}_{> 0}$ , and  $\mathbb{Z}_{> 0}$  be the sets of non-negative real numbers, positive real numbers, and positive integers, respectively. Denote by  $\mathbf{1}_n$  the vector  $(1, \dots, 1) \in \mathbb{R}^n$ . Denote by  $\|\cdot\|$  the Euclidean norm for vectors. For a vector  $v \in \mathbb{R}^n$ ,

denote by  $v_i$  its  $i$ -th scalar component and write  $v = (v_1, \dots, v_n)$ . For a set  $\mathcal{S} \subset \mathbb{R}^n$ , denote by  $|\mathcal{S}|$  its cardinality, and by  $\partial\mathcal{S}$  and  $\overline{\mathcal{S}}$  its boundary and closure, respectively.

## 15.2 Routing and Attack in Communication Network

We are interested in modeling and mitigating link-flooding DDoS attacks. Many such attacks focus on flooding parallel links that form a so-called bottleneck of the network to maximize congestion [38]. Therefore, we abstract the communication network as a directed graph of  $L$  parallel bottleneck links connecting a source to a destination, and focus our discussion on routing traffic among them. The set of links is denoted by  $\mathcal{L} := \{1, \dots, L\}$ .

Let  $R > 0$  be the total *desired transmission rate* of legitimate traffic that a router needs to distribute among the  $L$  parallel links. The traffic distribution is represented by a *routing assignment* vector  $r \in \mathbb{R}_{\geq 0}^L$  that satisfies

$$\sum_{l \in \mathcal{L}} r_l = 1.$$

If there was no congestion on a link  $l \in \mathcal{L}$ , then the transmission rate of legitimate traffic from the router on link  $l$  would be the *desired user rate*  $r_l R$ .

Motived by link-flooding DDoS attacks such as the crossfire attack [6], we assume that an adversary disrupts communication by injecting superfluous traffic on the links according to an *attack assignment* vector  $a \in \mathbb{R}_{\geq 0}^L$ , and call  $a_l$  the *attack rate* on link  $l \in \mathcal{L}$ . The attack is constrained by a total budget  $A > 0$ , that is,

$$\sum_{l \in \mathcal{L}} a_l \leq A.$$

For each link  $l \in \mathcal{L}$ , there is a constant capacity  $c_l > 0$  that upper-bounds its total transmission rate. If the sum of the desired user rate  $r_l R$  and the attack rate  $a_l$  is larger than the capacity  $c_l$ , then there is congestion on link  $l$  which results in packet drops and retransmissions. In many widely used transmission protocols such as the transmission control protocol (TCP) [39, 40], legitimate users will decrease their transmission rates until the total rate on link  $l$  is no longer larger than its capacity  $c_l$ . By contrast, a malicious adversary aims at sustaining congestion and thus does not decrease the attack rates. Therefore, we model the effect of congestion by a decrease in user rates and define the *actual user rates* by

$$u_l := \min\{r_l R, \max\{c_l - a_l, 0\}\}, \quad l \in \mathcal{L}. \quad (15.1)$$

The goal of the router is to maximize the sum of actual user rates. Therefore, it minimizes the *routing cost* defined by

$$J(r, a) := - \sum_{l \in \mathcal{L}} u_l. \quad (15.2)$$

Motivated by link-flooding DDoS attacks, we call  $r_l R$  and  $a_l$  the transmission rates of *legitimate* and of *attack* traffic, respectively, but in practice the key distinction is that the former represents traffic that the router aims to protect (in terms of minimizing the routing cost (15.2) above), whereas the router's objective does not care for the latter. If  $a$  was indeed generated by a malicious adversary, it would likely attempt to maximize the routing cost (15.2), or equivalently, to minimize the *attack cost* defined by

$$H(a, r) := -J(r, a) = \sum_{l \in \mathcal{L}} u_l. \quad (15.3)$$

However, in general one should consider more general attack costs  $H(a, r)$  that may not be precisely the symmetry of  $J(r, a)$  and could even be altogether independent of the routing assignment  $r$ . Not knowing the attack cost  $H(r, a)$  is in fact the key motivation for the learning-based approach proposed in this paper. In view of this, we only make the mild assumption that  $H(a, r)$  is continuous in  $a$  for each fixed  $r$ , except in Sect. 15.4 where we will consider an attack cost for a malicious adversary that generalizes (15.3).

In summary, the routing optimization problem is given by

$$\min_r J(r, a), \quad (15.4a)$$

$$\text{s.t. } \sum_{l \in \mathcal{L}} r_l = 1 \quad (\text{rate conservation}), \quad (15.4b)$$

$$r_l \geq 0 \quad \forall l \in \mathcal{L} \quad (\text{non-negative rate}), \quad (15.4c)$$

and the attack optimization problem is given by

$$\min_a H(a, r), \quad (15.5a)$$

$$\text{s.t. } \sum_{l \in \mathcal{L}} a_l \leq A \quad (\text{attack budget}), \quad (15.5b)$$

$$a_l \geq 0 \quad \forall l \in \mathcal{L} \quad (\text{non-negative rate}). \quad (15.5c)$$

In particular, we consider the general case where the router does not even know the total desired transmission rate  $R$ , and thus do not assume  $r_l R \leq c_0$  on any link  $l \in \mathcal{L}$ . Both the routing cost  $J(r, a)$  and the attack cost  $H(a, r)$  are functions of the routing assignment  $r$  and the attack assignment  $a$ . Therefore, we model the conflict between the router and the adversary using a game-theoretical model defined in the next section.

### 15.3 Stackelberg Game Model

We are interested in scenarios where there is asymmetric information between the router and the adversary. Specifically, the adversary is able to observe the routing assignment  $r$  before selecting the attack assignment  $a$ . To model such scenarios, we consider a hierarchical game model called Stackelberg games [21], in which the router (called the leader) selects the routing action  $r \in \mathcal{R}$  first, and then the adversary (called the follower), informed of the router's choice, selects the attack action  $a \in \mathcal{A}$ . The routing action set  $\mathcal{R}$  and the attack action set  $\mathcal{A}$  are defined by

$$\begin{aligned}\mathcal{R} &:= \{r \in \mathbb{R}_{\geq 0}^L : (4b) \text{ and } (4c) \text{ hold}\}, \\ \mathcal{A} &:= \{a \in \mathbb{R}_{\geq 0}^L : (5b) \text{ and } (5c) \text{ hold}\}.\end{aligned}$$

The game between the router and the adversary is fully defined by the tuple  $(\mathcal{R}, \mathcal{A}, J, H)$ . Formally, Stackelberg equilibria are defined as follows, see, e.g., [14, Sect. 3.1], [16, Definition 4.6, p. 179], and [41, 42].

**Definition 15.1** (Stackelberg) Given a game defined by the tuple  $(\mathcal{R}, \mathcal{A}, J, H)$ , a routing action  $r_s^* \in \mathcal{R}$  is called a *strong Stackelberg equilibrium routing action* if

$$J_s^* := \inf_{r \in \mathcal{R}} \min_{a \in \beta_a(r)} J(r, a) = \min_{a \in \beta_a(r_s^*)} J(r_s^*, a), \quad (15.6)$$

where

$$\beta_a(r) := \arg \min_{a \in \mathcal{A}} H(a, r)$$

denotes the set of optimal attack actions against a routing action  $r \in \mathcal{R}$ , and  $J_s^*$  is known as the *strong Stackelberg routing cost*; a routing action  $r_w^* \in \mathcal{R}$  is called a *weak Stackelberg equilibrium routing action* if

$$J_w^* := \inf_{r \in \mathcal{R}} \max_{a \in \beta_a(r)} J(r, a) = \max_{a \in \beta_a(r_w^*)} J(r_w^*, a), \quad (15.7)$$

where  $J_w^*$  is known as the *weak Stackelberg routing cost*; and for an  $\varepsilon > 0$ , a routing action  $r_\varepsilon^* \in \mathcal{R}$  is called a *weak  $\varepsilon$  Stackelberg routing action* for  $(\mathcal{R}, \mathcal{A}, J, H)$  if

$$\max_{a \in \beta_a(r_\varepsilon^*)} J(r_\varepsilon^*, a) \leq J_w^* + \varepsilon. \quad (15.8)$$

For each fixed  $r \in \mathcal{R}$ , as the function  $H(a, r)$  is continuous in  $a$  and the set  $\mathcal{A}$  is compact, the set  $\beta_a(r)$  is nonempty and compact; thus the minima in (15.6) and the maxima in (15.7) and (15.8) can be attained. However, the functions  $\min_{a \in \beta_a(r)} J(r, a)$  and  $\max_{a \in \beta_a(r)} J(r, a)$  are not necessarily continuous, and thus may only have infima over the set  $\mathcal{R}$ .

The difference between strong and weak Stackelberg equilibria stems from scenarios where the optimal attack action is non-unique, that is, where  $|\beta_a(r)| > 1$ .

Such scenarios are common under Stackelberg equilibrium routing actions, a phenomenon similar to the fact that in a mixed-strategy Nash equilibrium, every pure strategy in the support of a player's mixed strategy is a best response to the opponents mixed strategy [15, Lemma 33.2, p. 33]. If  $|\beta_a(r)| > 1$ , which optimal attack action the adversary plays will affect the resulting routing cost. The strong Stackelberg equilibrium represents the "optimistic" view that the adversary will select the one that is best for the router, whereas the weak Stackelberg equilibrium represents the "pessimistic" view that the worst one for the router will be selected. The conditions for existence of a strong Stackelberg equilibrium are weaker than those of a weak Stackelberg equilibrium [43], as we shall see in Sect. 15.4.1. However, a weak Stackelberg equilibrium is usually more desirable as it provides a guaranteed upper bound for the routing cost, regardless of the adversary's tie-breaking rule. The terminology "strong" and "weak" originates from [42] and is widely used in *Stackelberg Security Games*, see, e.g., [44] and references therein.

In this work, we consider games with incomplete information where the router does not know the attack cost function  $H$  or even the attack action set  $\mathcal{A}$  and thus cannot predict the set of optimal attack actions  $\beta_a(r)$ . This level of generality allows us to capture scenarios where the "adversary" is in fact benign and the attack action  $a$  may not even depend on the routing action  $r$ . However, in the following section we do focus our attention on malicious adversaries with a cost function that generalizes (15.3).

## 15.4 Optimal Attack and Stackelberg Equilibria for Malicious Adversaries

As mentioned in Sect. 15.2, a natural candidate for the attack cost  $H(a, r)$  for a malicious adversary is given by (15.3), which makes  $(\mathcal{R}, \mathcal{A}, J, H)$  a zero-sum game. In this section, we consider a more general scenario that deviates from the zero-sum case, as the adversary may have different priorities for different links, and the attack cost is defined by

$$H(a, r) := \sum_{l \in \mathcal{L}} \gamma_l u_l, \quad (15.9)$$

where  $\gamma = (\gamma_1, \dots, \gamma_L) \in (0, 1]^L$  can be viewed as an *attack priority* vector.

From the definition of the actual user rates (15.1), we see that a malicious adversary with the attack cost (15.9) has no incentive to assign less attack traffic than the total budget  $A$  or to assign more attack traffic to a link than its capacity. Therefore, for each routing action  $r \in \mathcal{R}$ , there is an optimal attack action  $a \in \beta_a(r)$  such that

$$\sum_{l \in \mathcal{L}} a_l = A \quad (15.10)$$

with

$$a_l \leq c_l \quad \forall l \in \mathcal{L}. \quad (15.11)$$

For attack actions  $a \in \mathcal{A}$  such that (15.11) holds, we can rewrite the definition of the actual user rates (15.1) by

$$u_l := \min\{r_l R, c_l - a_l\}, \quad l \in \mathcal{L}.$$

Also, we assume

$$A < \sum_{l \in \mathcal{L}} c_l,$$

since otherwise the adversary can simply fill all the bottleneck links and the attack optimization problem becomes trivial.

In the following result, we establish that there is always an optimal attack action with the attack rates equal to 0 or to the corresponding link capacities for all but one link.

**Theorem 15.1** *For each routing action  $r \in \mathcal{R}$ , there is an optimal attack action  $a^* \in \beta_a(r)$  such that (15.10) and (15.11) hold, and there is at most one link  $l_0 \in \mathcal{L}$  where  $a_{l_0}^* \in (0, c_{l_0})$ , that is,*

$$a^* \in \mathcal{A}^* := \{a \in \mathcal{A} : (10) \text{ and } (11) \text{ hold, and } |\{l \in \mathcal{L} : a_l \in (0, c_l)\}| \leq 1\}. \quad (15.12)$$

**Proof** Based on the analysis before Theorem 15.1, there is an optimal attack action  $\bar{a}^* \in \beta_a(r)$  such that (15.10) and (15.11) hold. We construct an optimal attack action  $a^* \in \beta_a(r) \cap \mathcal{A}^*$  as follows.

1. Let  $a = \bar{a}^*$ . Then  $a \in \beta_a(r)$  and satisfies (15.10) and (15.11).
2. If there are two links  $l_1, l_2 \in \mathcal{L}$  such that

$$a_{l_1} \in (0, c_{l_1}), \quad a_{l_2} \in (0, c_{l_2}),$$

then  $a \notin \mathcal{A}^*$  and we go to step 3; otherwise  $a \in \mathcal{A}^*$  and we go to step 4.

3. Consider the following two possibilities.

- (a) If there is a link  $\bar{l}_1 \in \{l_1, l_2\}$  such that  $a_{\bar{l}_1} \leq c_{\bar{l}_1} - r_{\bar{l}_1} R$ , then the corresponding actual user rate satisfies

$$u_{\bar{l}_1} = \min\{r_{\bar{l}_1} R, c_{\bar{l}_1} - a_{\bar{l}_1}\} = r_{\bar{l}_1} R.$$

We define an attack action  $\bar{a} \in \mathcal{A}$  by moving as much attack traffic on  $\bar{l}_1$  as possible to the other link  $\bar{l}_2 \in \{l_1, l_2\} \setminus \{\bar{l}_1\}$ , that is,

$$\bar{a}_l := \begin{cases} \max\{0, a_{l_1} + a_{l_2} - c_{\bar{l}_2}\}, & l = \bar{l}_1, \\ \min\{a_{l_1} + a_{l_2}, c_{\bar{l}_2}\}, & l = \bar{l}_2, \\ a_l, & l \in \mathcal{L} \setminus \{l_1, l_2\}. \end{cases} \quad (15.13)$$

Then clearly  $\bar{a} \in \mathcal{A}$  and satisfies (15.10) and (15.11) with  $\bar{a}$  in place of  $a$ , and at least one of  $\bar{a}_{\bar{l}_1} = 0$  and  $\bar{a}_{\bar{l}_2} = c_{\bar{l}_2}$  holds. Moreover, we have

$$\bar{a}_{\bar{l}_1} \leq a_{l_1} \leq c_{\bar{l}_1} - r_{\bar{l}_1} R, \quad \bar{a}_{\bar{l}_2} \geq a_{l_2};$$

thus the corresponding actual user rates satisfy  $\bar{u}_l = u_l$  for all  $l \in \mathcal{L} \setminus \{l_1, l_2\}$  and

$$\begin{aligned} \bar{u}_{\bar{l}_1} &= \min\{r_{\bar{l}_1} R, c_{\bar{l}_1} - \bar{a}_{\bar{l}_1}\} = r_{\bar{l}_1} R = u_{\bar{l}_1}, \\ \bar{u}_{\bar{l}_2} &= \min\{r_{\bar{l}_2} R, c_{\bar{l}_2} - \bar{a}_{\bar{l}_2}\} \leq \min\{r_{\bar{l}_2} R, c_{\bar{l}_2} - a_{l_2}\} = u_{\bar{l}_2}. \end{aligned}$$

Therefore, the attack costs for  $\bar{a}$  and  $a$  satisfy

$$H(\bar{a}, r) - H(a, r) = \gamma_{\bar{l}_1} \bar{u}_{\bar{l}_1} + \gamma_{\bar{l}_2} \bar{u}_{\bar{l}_2} - \gamma_{\bar{l}_1} u_{\bar{l}_1} - \gamma_{\bar{l}_2} u_{\bar{l}_2} \leq 0;$$

thus  $a \in \beta_a(r)$  implies  $\bar{a} \in \beta_a(r)$ .

- (b) Otherwise  $a_{l_1} \in (c_{l_1} - r_{l_1} R, c_{l_1})$  and  $a_{l_2} \in (c_{l_2} - r_{l_2} R, c_{l_2})$ . Let  $\bar{l}_1$  be the link between  $l_1$  and  $l_2$  with a lower priority for the attack, that is,

$$\bar{l}_1 \in \arg \min_{l \in \{l_1, l_2\}} \gamma_l$$

(if  $\gamma_1 = \gamma_2$ , pick an arbitrary one). Again, we define an attack action  $\bar{a} \in \mathcal{A}$  by moving as much attack traffic on  $l_1$  and  $l_2$  as possible to the other link  $\bar{l}_2 \in \{l_1, l_2\} \setminus \{\bar{l}_1\}$  according to (15.13). Then clearly  $\bar{a} \in \mathcal{A}$  and satisfies (15.10) and (15.11) with  $\bar{a}$  in place of  $a$ , and at least one of  $\bar{a}_{\bar{l}_1} = 0$  and  $\bar{a}_{\bar{l}_2} = c_{\bar{l}_2}$  holds. Moreover, we have

$$\bar{a}_{\bar{l}_1} + \bar{a}_{\bar{l}_2} = a_{l_1} + a_{l_2}, \quad \bar{a}_{\bar{l}_2} \geq a_{l_2} \geq c_{\bar{l}_2} - r_{\bar{l}_2} R;$$

thus the corresponding actual user rates satisfy

$$\begin{aligned} \bar{u}_{\bar{l}_1} &= \min\{r_{\bar{l}_1} R, c_{\bar{l}_1} - \bar{a}_{\bar{l}_1}\} \leq c_{\bar{l}_1} - \bar{a}_{\bar{l}_1}, \\ \bar{u}_{\bar{l}_2} &= \min\{r_{\bar{l}_2} R, c_{\bar{l}_2} - \bar{a}_{\bar{l}_2}\} = c_{\bar{l}_2} - \bar{a}_{\bar{l}_2}. \end{aligned}$$

Therefore, the attack costs for  $\bar{a}$  and  $a$  satisfy

$$\begin{aligned}
H(\bar{a}, r) - H(a, r) &= \gamma_{\bar{l}_1} \bar{u}_{\bar{l}_1} + \gamma_{\bar{l}_2} \bar{u}_{\bar{l}_2} - \gamma_{\bar{l}_1} u_{\bar{l}_1} - \gamma_{\bar{l}_2} u_{\bar{l}_2} \\
&\leq \gamma_{\bar{l}_1} (c_{\bar{l}_1} - \bar{a}_{\bar{l}_1}) + \gamma_{\bar{l}_2} (c_{\bar{l}_2} - \bar{a}_{\bar{l}_2}) \\
&\quad - \gamma_{\bar{l}_1} (c_{\bar{l}_1} - a_{\bar{l}_1}) - \gamma_{\bar{l}_2} (c_{\bar{l}_2} - a_{\bar{l}_2}) \\
&= \gamma_{\bar{l}_1} (a_{\bar{l}_1} - \bar{a}_{\bar{l}_1}) + \gamma_{\bar{l}_2} (a_{\bar{l}_2} - \bar{a}_{\bar{l}_2}) \\
&= (\gamma_{\bar{l}_1} - \gamma_{\bar{l}_2})(\bar{a}_{\bar{l}_2} - a_{\bar{l}_2}) \leq 0;
\end{aligned}$$

thus  $a \in \beta_a(r)$  implies  $\bar{a} \in \beta_a(r)$ .

In summary, we have constructed an optimal attack action  $\bar{a} \in \beta_a(r)$  so that (15.10) and (15.11) hold with  $\bar{a}$  in place of  $a$ , and there is only one link  $l \in \{l_1, l_2\}$  where  $a_l \in (0, c_l)$ . Let  $a = \bar{a}$  and return to Step 2.

4. Let  $a^* = a$ . Then  $a^* \in \beta_a(r) \cap \mathcal{A}^*$ .

As the number of links  $L$  is constant, the above algorithm is guaranteed to terminate before running Step 3 for  $L$  times. Therefore, an optimal attack action  $a^* \in \beta_a(r) \cap \mathcal{A}^*$  exists.  $\square$

Based on Theorem 15.1, it suffices to search for an optimal attack action in the subset  $\mathcal{A}^*$  defined by (15.12) of the attack action set  $\mathcal{A}$ . Therefore, from now on we restrict our attention to attack actions from  $\mathcal{A}^*$  and the game  $(\mathcal{R}, \mathcal{A}^*, J, H)$ , and define the corresponding set of optimal attack actions against a routing action  $r \in \mathcal{R}$  by

$$\beta_a^*(r) := \arg \min_{a \in \mathcal{A}^*} H(a, r) = \beta_a(r) \cap \mathcal{A}^*.$$

**Remark 15.1** Consider the case where the attack action  $a = a^{\text{mal}} + a^{\text{ben}}$ , where  $a^{\text{mal}}$  represents traffic from a malicious adversary and  $a^{\text{ben}}$  represents traffic from benign users that do not respond to the router, that is, the corresponding best response set  $\beta_{a^{\text{ben}}}(r)$  is constant. Then it is straightforward to see that the result in Theorem 15.1 still holds with  $c_l - a_l^{\text{ben}}$  in place of  $c_l$  in (15.12).

### 15.4.1 Optimal Attack and Stackelberg Equilibria for Networks with Identical Links

One can show that, in general, finding an optimal attack action in the set  $\mathcal{A}^*$  defined by (15.12) is at least as hard as solving the NP-hard *knapsack problem* [45]; cf. [29]. However, the problem is simpler when all the parallel links have the same capacity  $c_0$ . In this case, the set  $\mathcal{A}^*$  defined in (15.12) is the set of attack actions with attack rate  $c_0$  on  $\lfloor A/c_0 \rfloor$  links and 0 on  $L - \lceil A/c_0 \rceil$  links, that is,

$$\begin{aligned}
\mathcal{A}^* &= \{a \in \mathcal{A} : (10) \text{ and } (11) \text{ hold,} \\
&\text{and } |\{l \in \mathcal{L} : a_l = c_0\}| = \lfloor A/c_0 \rfloor \text{ and } |\{l \in \mathcal{L} : a_l = 0\}| = L - \lceil A/c_0 \rceil\}.
\end{aligned}$$

For an attack action  $a \in \mathcal{A}^*$ , we denote by  $l_0(a)$  the link on which  $a_{l_0(a)} \in (0, c_0)$ . The attack rate on the link  $l_0(a)$  is a constant given by

$$a_{l_0(a)} = a_{\text{rem}} := A - \lfloor A/c_0 \rfloor c_0.$$

For an attack budget  $A$  such that  $A/c_0 \in \mathbb{Z}$ , we have  $\{l_0(a)\} = \emptyset$  and  $a_{\text{rem}} = 0$ . For a routing action  $r \in \mathcal{R}$  and an attack action  $a \in \mathcal{A}^*$ , the routing cost  $J(r, a)$  and attack cost  $H(a, r)$  are given by

$$\begin{aligned} J(r, a) &= -u_{l_0(a)} - \sum_{l \in \mathcal{L}: a_l=0} u_l \\ &= -\min\{r_{l_0(a)} R, c_0 - a_{\text{rem}}\} - \sum_{l \in \mathcal{L}: a_l=0} \min\{r_l R, c_0\} \\ H(a, r) &= \gamma_{l_0(a)} u_{l_0(a)} + \sum_{l \in \mathcal{L}: a_l=0} \gamma_l u_l \\ &= \gamma_{l_0(a)} \min\{r_{l_0(a)} R, c_0 - a_{\text{rem}}\} + \sum_{l \in \mathcal{L}: a_l=0} \gamma_l \min\{r_l R, c_0\}. \end{aligned} \tag{15.14}$$

If additionally

$$\min\{r_l R, c_0\} \leq c_0 - a_{\text{rem}} \quad \forall l \in \mathcal{L}, \tag{15.15}$$

then an attack  $a \in \mathcal{A}^*$  will not cause congestion on  $l_0(a)$ , and thus

$$J(r, a) = - \sum_{l \in \mathcal{L}: a_l < c_0} \min\{r_l R, c_0\}, \quad H(a, r) = \sum_{l \in \mathcal{L}: a_l < c_0} \gamma_l \min\{r_l R, c_0\}. \tag{15.16}$$

Note that (15.15) always holds for an attack budget  $A$  such that  $A/c_0 \in \mathbb{Z}$ . A necessary condition for (15.15) is  $A/c_0 - \lfloor A/c_0 \rfloor \leq 1 - R/(Lc_0)$ .

For the case where all the parallel links have the same capacity  $c_0$ , Theorem 15.1 can be extend to the following result, which shows that an optimal attack action can be found after at most  $L$  trials.

**Corollary 15.1** *Consider the case of equal link capacities  $c_l = c_0$  for all  $l \in \mathcal{L}$ . For a routing action  $r \in \mathcal{R}$ , the set of optimal attack actions in  $\mathcal{A}^*$  satisfies that*

$$\begin{aligned} \beta_a^*(r) \subset \mathcal{A}_0^*(r) &:= \{a \in \mathcal{A}^* : \min\{\gamma_l \min\{r_l R, c_0\} : a_l = c_0\} \\ &\geq \max\{\gamma_l \min\{r_l R, c_0\} : a_l = 0\}\}, \end{aligned} \tag{15.17}$$

or equivalently, an attack action  $a \in \mathcal{A}^*$  is optimal against  $r$  only if there exists a permutation  $(l_1^a, \dots, l_L^a)$  of  $\mathcal{L}$  such that

$$\gamma_{l_1^a} \min\{r_{l_1^a} R, c_0\} \geq \dots \geq \gamma_{l_L^a} \min\{r_{l_L^a} R, c_0\} \tag{15.18}$$

and

$$\{l \in \mathcal{L} : a_l = c_0\} \subset \mathcal{L}_1^a \cup \{l_{\lceil A/c_0 \rceil}^a\}, \quad \{l \in \mathcal{L} : a_l = 0\} \subset \mathcal{L}_2^a \cup \{l_{\lfloor A/c_0 \rfloor + 1}^a\}$$

with

$$\mathcal{L}_1^a := \{l_1^a, \dots, l_{\lceil A/c_0 \rceil}^a\}, \quad \mathcal{L}_2^a := \{l_{\lceil A/c_0 \rceil + 1}^a, \dots, l_L^a\}.$$

If additionally (15.15) holds, then

$$\begin{aligned} \beta_a^*(r) = \mathcal{A}_1^*(r) &:= \{a \in \mathcal{A}^* : \min\{\gamma_l \min\{r_l R, c_0\} : a_l = c_0\} \\ &\geq \max\{\gamma_l \min\{r_l R, c_0\} : a_l < c_0\}\}, \end{aligned} \quad (15.19)$$

or equivalently, an attack action  $a \in \mathcal{A}^*$  is optimal against  $r$  if and only if there exists a permutation  $(l_1^a, \dots, l_L^a)$  of  $\mathcal{L}$  such that (15.18) holds and

$$\{l \in \mathcal{L} : a_l = c_0\} = \mathcal{L}_1^a.$$

**Proof** Suppose there is an optimal attack action  $a^* \in \beta_a^*(r) \setminus \mathcal{A}_0^*(r)$ . Then there are two links  $l_1, l_2 \in \mathcal{L}$  such that  $a_{l_1}^* = c_0$ ,  $a_{l_2}^* = 0$  and  $\gamma_{l_1} \min\{r_{l_1} R, c_0\} < \gamma_{l_2} \min\{r_{l_2} R, c_0\}$ . We define an attack action  $\bar{a}$  by switching the attack rates on  $l_1$  and  $l_2$ , that is,

$$\bar{a}_l := \begin{cases} a_{l_2}^*, & l = l_1, \\ a_{l_1}^*, & l = l_2, \\ a_l^*, & l \in \mathcal{L} \setminus \{l_1, l_2\}. \end{cases}$$

Then the formula for attack cost in (15.14) implies

$$H(\bar{a}, r) - H(a^*, r) = \gamma_{l_1} \min\{r_{l_1} R, c_0\} - \gamma_{l_2} \min\{r_{l_2} R, c_0\} < 0,$$

which contradicts the assumption that  $a^*$  is optimal against  $r$ . Hence  $\beta_a^*(r) \subset \mathcal{A}_0^*(r)$ , that is, (15.17) holds.

If additionally (15.15) holds, the same analysis with  $a_{l_2}^* < c_0$  shows that  $\beta_a^*(r) \subset \mathcal{A}_1^*(r)$ . Meanwhile, the formula for attack cost in (15.16) implies that all attack actions from  $\mathcal{A}_1^*(r)$  yield the same attack cost which is the sum of the  $L - \lfloor A/c_0 \rfloor$  smallest  $\gamma_l \min\{r_l R, c_0\}$ . Hence  $\beta_a^*(r) = \mathcal{A}_1^*(r)$ , that is, (15.19) holds.  $\square$

In the remainder of this section, we investigate Stackelberg equilibrium routing actions and Stackelberg routing costs for the nonzero-sum game  $(\mathcal{R}, \mathcal{A}^*, J, H)$ , assuming that all links have same capacity  $c_0$  and the attack priorities satisfy a condition. We construct explicit formulae for a strong Stackelberg equilibrium routing action and for weak  $\varepsilon$  Stackelberg routing actions with arbitrarily small  $\varepsilon > 0$ . Our results shows that there is no weak Stackelberg equilibrium routing action unless  $\gamma_l$  are the same for all links, in which case the strong Stackelberg equilibrium routing action is also a weak one. In practice, a pessimistic router would either play the weak Stackelberg equilibrium routing action when all  $\gamma_l$  are the same, or a weak  $\varepsilon$

Stackelberg routing action with a small  $\varepsilon$  when they are not, with the understanding that there will be an  $\varepsilon$  penalty in the latter case.

First, we construct the strong Stackelberg equilibrium routing action and the strong Stackelberg routing cost.

**Theorem 15.2** *Consider the case of equal link capacities  $c_l = c_0$  for all  $l \in \mathcal{L}$  and attack priorities  $\gamma_l$  such that*

$$\frac{1/\gamma_l}{\sum_{\bar{l} \in \mathcal{L}} 1/\gamma_{\bar{l}}} < \frac{c_0 - a_{\text{rem}}}{R} \quad \forall l \in \mathcal{L}. \quad (15.20)$$

For the game  $(\mathcal{R}, \mathcal{A}^*, J, H)$ , there is a strong Stackelberg equilibrium routing action  $r^* \in \mathcal{R}$  defined by

$$r_l^* := \frac{1/\gamma_l}{\sum_{\bar{l} \in \mathcal{L}} 1/\gamma_{\bar{l}}}, \quad l \in \mathcal{L}, \quad (15.21)$$

and the strong Stackelberg routing cost is given by

$$J^* := \min_{a \in \beta_a^*(r^*)} J(r^*, a) = - \sum_{l \in \mathcal{L}_1^r} r_l^* R, \quad (15.22)$$

where  $\mathcal{L}_1^r$  is a subset of  $L - \lfloor A/c_0 \rfloor$  links  $l \in \mathcal{L}$  with the largest  $r_l^*$ , that is, there exists a permutation  $(l_1^r, \dots, l_L^r)$  of  $\mathcal{L}$  such that

$$r_{l_1^r}^* \geq \dots \geq r_{l_L^r}^*,$$

and

$$\mathcal{L}_1^r = \{l_1^r, \dots, l_{\lfloor A/c_0 \rfloor}^r\}.$$

If additionally the attack budget  $A \geq c_0$ , then  $r^*$  defined by (15.21) is the unique strong Stackelberg equilibrium routing action.

Before proving Theorem 15.2, we observe that the routing action  $r^*$  defined by (15.21) satisfies (15.15) with  $r^*$  in place of  $r$  due to (15.20). Moreover, we have

$$\gamma_l \min\{r_l^* R, c_0\} = \frac{R}{\sum_{\bar{l} \in \mathcal{L}} 1/\gamma_{\bar{l}}} \quad \forall l \in \mathcal{L},$$

which combined with (15.19), implies

$$\beta_a^*(r^*) = \mathcal{A}^*. \quad (15.23)$$

Therefore, if the router plays  $r^*$ , then all attack actions  $a \in \mathcal{A}^*$  will yield the same attack cost

$$H^* := H(a, r^*) = \frac{(L - \lfloor A/c_0 \rfloor)R}{\sum_{l \in \mathcal{L}} 1/\gamma_l}$$

given by the formula for attack cost in (15.16).

**Proof of Theorem 3** Clearly,  $r^*$  defined by (15.21) satisfies  $r^* \in \mathcal{R}$ . Then the equality in (15.22) follows from (15.23) and the formula for routing cost in (15.16).

Consider an arbitrary  $r \in \mathcal{R}$  such that  $r \neq r^*$ , and let

$$\mathcal{L}_1 := \{l \in \mathcal{L} : r_l < r_l^*\}, \quad \mathcal{L}_2 := \{l \in \mathcal{L} : r_l \geq r_l^*\}.$$

Then

$$r_l R < r_l^* R < c_0 - a_{\text{rem}} \quad \forall l \in \mathcal{L}_1 \quad (15.24)$$

and

$$\gamma_{l_1} r_{l_1} < \frac{1}{\sum_{l \in \mathcal{L}} 1/\gamma_l} \leq \gamma_{l_2} r_{l_2} \quad \forall l_1 \in \mathcal{L}_1, \forall l_2 \in \mathcal{L}_2. \quad (15.25)$$

Note that (15.15) may not hold for  $r$  since  $\min\{r_l R, c_0\} \leq c_0 - a_{\text{rem}}$  may not hold for  $l \in \mathcal{L}_2$ . Consider an arbitrary optimal attack action  $a \in \beta_a^*(r)$  against  $r$ . Then there are two possibilities.

1. If  $|\mathcal{L}_1| \geq L - \lfloor A/c_0 \rfloor$ , then from (15.17) and (15.25), we have  $\{l \in \mathcal{L} : a_l = 0\} \subset \mathcal{L}_1$ . If (15.15) holds, then from (15.19) and (15.25) we have  $\{l \in \mathcal{L} : a_l < c_0\} \subset \mathcal{L}_1$ . Next, we show that even if (15.15) does not hold, the link  $l_0(a)$  on which  $a_{l_0(a)} = a_{\text{rem}} \in (0, c_0)$  still satisfies  $l_0(a) \in \mathcal{L}_1$ . Indeed, suppose  $l_0(a) \in \mathcal{L}_2$ . Then the formula for attack cost in (15.14) implies

$$H(a, r) = \gamma_{l_0(a)} \min\{r_{l_0(a)} R, c_0 - a_{\text{rem}}\} + \sum_{l \in \mathcal{L}: a_l=0} \gamma_l r_l R.$$

Meanwhile, as  $|\mathcal{L}_1| \geq L - \lfloor A/c_0 \rfloor$ , there is at least one link  $\bar{l}_1 \in \mathcal{L}_1$  on which  $a_{\bar{l}_1} = c_0$ . We define an attack action  $\bar{a}$  by switching the attack rates on  $l_0(a)$  and  $\bar{l}_1$ , that is,

$$\bar{a}_l := \begin{cases} a_{\text{rem}}, & l = \bar{l}_1, \\ c_0, & l = l_0(a), \\ a_l, & l \in \mathcal{L} \setminus \{\bar{l}_1, l_0(a)\}. \end{cases}$$

Then

$$H(\bar{a}, r) = \gamma_{\bar{l}_1} r_{\bar{l}_1} R + \sum_{l \in \mathcal{L}: a_l=0} \gamma_l r_l R < H(a, r),$$

where the inequality follows from (15.24) and (15.25), which contradicts the assumption that  $a$  is optimal against  $r$ . Hence  $\{l \in \mathcal{L} : a_l < c_0\} \subset \mathcal{L}_1$  holds regardless of whether (15.15) holds. Let  $\tilde{\mathcal{L}}_1$  be a subset of  $L - \lfloor A/c_0 \rfloor$  links  $l \in \mathcal{L}_1$  with the largest  $r_l$ . Then the formula for routing cost in (15.14), together with (15.24), implies

$$J(r, a) \geq - \sum_{l \in \tilde{\mathcal{L}}_1} r_l R > - \sum_{l \in \tilde{\mathcal{L}}_1} r_l^* R \geq - \sum_{l \in \mathcal{L}_1^r} r_l^* R = J^*.$$

2. If  $|\mathcal{L}_1| < L - \lfloor A/c_0 \rfloor$ , then from (15.17) and (15.25) we have  $\mathcal{L}_1 \subset \{l \in \mathcal{L} : a_l < c_0\}$ . In the following, we assume that there is a link  $l_0(a) \in \mathcal{L}_2$  on which  $a_{l_0(a)} = a_{\text{rem}} \in (0, c_0)$ , and let  $\tilde{\mathcal{L}}_2$  be a subset of  $L - \lceil A/c_0 \rceil - |\mathcal{L}_1|$  links  $l \in \mathcal{L}_2$  with the largest  $r_l$ . (The cases where  $l_0(a) \in \mathcal{L}_1$  or  $l_0(a)$  does not exist can be proved along the same lines while removing the terms related to  $l_0(a)$  and letting  $\tilde{\mathcal{L}}_2$  be a subset of  $L - \lfloor A/c_0 \rfloor - |\mathcal{L}_1|$  links  $l \in \mathcal{L}_2$  with the largest  $r_l$ . In particular, if  $l_0(a) \in \mathcal{L}_1$  then (15.24) implies  $r_{l_0(a)}R < c_0 - a_{\text{rem}}$ .) From (15.4b), we have

$$\begin{aligned} - \sum_{l \in \mathcal{L}_1} (r_l - r_l^*)R &= \sum_{l \in \mathcal{L}_2} (r_l - r_l^*)R \geq (r_{l_0(a)} - r_{l_0(a)}^*)R + \sum_{l \in \tilde{\mathcal{L}}_2} (r_l - r_l^*)R \\ &\geq (\min\{r_{l_0(a)}R, c_0 - a_{\text{rem}}\} - r_{l_0(a)}^*R) + \sum_{l \in \tilde{\mathcal{L}}_2} (\min\{r_lR, c_0\} - r_l^*R), \end{aligned}$$

where the last inequality is strict if  $r_{l_0(a)}R > c_0 - a_{\text{rem}}$ . Hence the formula for routing cost in (15.14) implies

$$\begin{aligned} J(r, a) &\geq -\min\{r_{l_0(a)}R, c_0 - a_{\text{rem}}\} - \sum_{l \in \mathcal{L}_1} r_l R - \sum_{l \in \tilde{\mathcal{L}}_2} \min\{r_l R, c_0\} \\ &= -(\min\{r_{l_0(a)}R, c_0 - a_{\text{rem}}\} - r_{l_0(a)}^*R) - \sum_{l \in \mathcal{L}_1} (r_l - r_l^*)R \\ &\quad - \sum_{l \in \tilde{\mathcal{L}}_2} (\min\{r_l R, c_0\} - r_l^*R) - r_{l_0(a)}^*R - \sum_{l \in \mathcal{L}_1} r_l^*R - \sum_{l \in \tilde{\mathcal{L}}_2} r_l^*R \\ &\geq -r_{l_0(a)}^*R - \sum_{l \in \mathcal{L}_1} r_l^*R - \sum_{l \in \tilde{\mathcal{L}}_2} r_l^*R \\ &\geq -\sum_{l \in \mathcal{L}_1'} r_l^*R = J^*, \end{aligned}$$

where the second inequality is strict if  $r_{l_0(a)}R > c_0 - a_{\text{rem}}$ .

Next, we show that if the attack budget  $A \geq c_0$ , then we have  $J(r, a) > J^*$  even if  $r_{l_0(a)}R \leq c_0 - a_{\text{rem}}$ . Indeed, as  $r_{l_0(a)}R \leq c_0 - a_{\text{rem}}$ , there is no congestion on  $l_0(a)$ . Hence the routing and attack costs are given by the corresponding formulae in (15.16), and thus similar analysis to the proof of Corollary 15.1 shows that  $a \in \mathcal{A}_1^*(r)$  defined in (15.19). Then as  $A \geq c_0$ , there is a link

$$\bar{l}_2 \in \arg \max_{l \in \mathcal{L}} \gamma_l \min\{r_l R, c_0\}$$

on which  $a_{\bar{l}_2} = c_0$ . Moreover, we have  $r_{\bar{l}_2} > r_{\bar{l}_2}^*$  as  $r \neq r^*$ ; thus  $\bar{l}_2 \in \mathcal{L}_2$ . Let  $\tilde{\mathcal{L}}_2$  be a subset of  $L - \lfloor A/c_0 \rfloor - |\mathcal{L}_1|$  links  $l \in \mathcal{L}_2 \setminus \{\bar{l}_2\}$  with the largest  $r_l$ . Then (15.4b) implies

$$\begin{aligned}
-\sum_{l \in \mathcal{L}_1} (r_l - r_l^*)R &> \sum_{l \in \mathcal{L}_2 \setminus \{\tilde{l}_2\}} (r_l - r_l^*)R \\
&\geq \sum_{l \in \tilde{\mathcal{L}}_2} (r_l - r_l^*)R \geq \sum_{l \in \tilde{\mathcal{L}}_2} (\min\{r_l R, c_0\} - r_l^* R).
\end{aligned}$$

Hence the formula for routing cost in (15.16) implies

$$\begin{aligned}
J(r, a) &\geq -\sum_{l \in \mathcal{L}_1} r_l R - \sum_{l \in \tilde{\mathcal{L}}_2} \min\{r_l R, c_0\} \\
&= -\sum_{l \in \mathcal{L}_1} (r_l - r_l^*)R - \sum_{l \in \tilde{\mathcal{L}}_2} (\min\{r_l R, c_0\} - r_l^* R) \\
&\quad - \sum_{l \in \mathcal{L}_1} r_l^* R - \sum_{l \in \tilde{\mathcal{L}}_2} r_l^* R \\
&> -\sum_{l \in \mathcal{L}_1} r_l^* R - \sum_{l \in \tilde{\mathcal{L}}_2} r_l^* R \\
&\geq -\sum_{l \in \mathcal{L}_1} r_l^* R = J^*.
\end{aligned}$$

In summary, we have established that

$$J^* = \min_{a \in \beta_a^*(r^*)} J(r^*, a) \leq \min_{a \in \beta_a^*(r)} J(r, a) \quad \forall r \neq r^*, \quad (15.26)$$

thus  $r^*$  is a strong Stackelberg equilibrium routing action and  $J^*$  is the strong Stackelberg routing cost for  $(\mathcal{R}, \mathcal{A}^*, J, H)$ . If additionally the attack budget  $A \geq c_0$ , then the inequality in (15.26) is strict; thus  $r^*$  is the unique strong Stackelberg equilibrium routing action.

Next, we establish that  $J^*$  defined in (15.22) is also the weak Stackelberg routing cost for  $(\mathcal{R}, \mathcal{A}^*, J, H)$ , by constructing a weak  $\varepsilon$  Stackelberg routing action for an arbitrarily small  $\varepsilon > 0$ ; see also [43, Sect. 6] for a related result for finite games with mixed strategies.

**Theorem 15.3** *Consider the case of equal link capacities  $c_l = c_0$  for all  $l \in \mathcal{L}$  and attack priorities  $\gamma_l$  such that (15.20) holds. For the game  $(\mathcal{R}, \mathcal{A}^*, J, H)$ , we have*

1. *for an arbitrary  $\varepsilon > 0$  such that*

$$\varepsilon < \min\{(L - \lfloor A/c_0 \rfloor) r_l^*, \lfloor A/c_0 \rfloor ((c_0 - a_{\text{rem}})/R - r_l^*)\} \quad \forall l \in \mathcal{L}, \quad (15.27)$$

*where  $r^*$  is the routing action defined by (15.21), there is a weak  $\varepsilon$  Stackelberg routing action  $r^\varepsilon \in \mathcal{R}$  defined by*

$$r_l^\varepsilon := \begin{cases} r_l^* - \varepsilon / (L - \lfloor A/c_0 \rfloor), & l \in \mathcal{L}_1^r, \\ r_l^* + \varepsilon / \lfloor A/c_0 \rfloor, & l \in \mathcal{L}_2^r, \end{cases} \quad (15.28)$$

where  $\mathcal{L}_1^r$  is a subset of  $L - \lfloor A/c_0 \rfloor$  links  $l \in \mathcal{L}$  with the largest  $r_l^*$  as in (15.22) and  $\mathcal{L}_2^r := \mathcal{L} \setminus \mathcal{L}_1^r$ , and

2. the weak Stackelberg routing cost is equal to the strong Stackelberg routing cost  $J^*$  given by (15.22).

**Proof** The condition (15.27) ensures that  $r^\varepsilon$  defined by (15.28) satisfies  $r^\varepsilon \in \mathcal{R}$  and

$$r_l^\varepsilon < \frac{c_0 - a_{\text{rem}}}{R} \quad \forall l \in \mathcal{L},$$

and thus (15.15) holds with  $r^\varepsilon$  in place of  $r$ . Moreover, combining (15.21) and (15.28) yields

$$\gamma_{l_1} r_{l_1}^\varepsilon < \frac{1}{\sum_{l \in \mathcal{L}} 1/\gamma_l} < \gamma_{l_2} r_{l_2}^\varepsilon \quad \forall l_1 \in \mathcal{L}_1^r, \forall l_2 \in \mathcal{L}_2^r.$$

Then (15.19) implies that the set of optimal attack actions is given by

$$\beta_a^*(r^\varepsilon) = \{a \in \mathcal{A}^* : \{l \in \mathcal{L} : a_l = c_0\} = \mathcal{L}_2^r\}.$$

Hence the formula for routing cost in (15.16) implies

$$J(r^\varepsilon, a) = - \sum_{l \in \mathcal{L}_1^r} r_l^\varepsilon R = \varepsilon - \sum_{l \in \mathcal{L}_1^r} r_l^* R = J^* + \varepsilon \quad \forall a \in \beta_a^*(r^\varepsilon).$$

As  $\varepsilon > 0$  can be arbitrarily small, we have

$$J^* \geq \inf_{r \in \mathcal{R}} \max_{a \in \beta_a^*(r)} J(r, a).$$

Additionally, Theorem 15.2 implies

$$J^* = \min_{r \in \mathcal{R}} \min_{a \in \beta_a^*(r)} J(r, a) \leq \inf_{r \in \mathcal{R}} \max_{a \in \beta_a^*(r)} J(r, a).$$

Hence

$$J^* = \inf_{r \in \mathcal{R}} \max_{a \in \beta_a^*(r)} J(r, a),$$

that is,  $J^*$  is the weak Stackelberg routing cost and  $r^\varepsilon$  is a weak  $\varepsilon$  Stackelberg routing action for  $(\mathcal{R}, \mathcal{A}^*, J, H)$ .  $\square$

Based on (15.23), all attack actions from  $\mathcal{A}^*$  are optimal against the strong Stackelberg equilibrium routing action  $r^*$  defined by (15.21). Hence  $r^*$  cannot be a weak Stackelberg equilibrium routing action unless all attack actions from  $\mathcal{A}^*$  also yield

the same routing cost. Combining (15.21) and the formula for routing cost in (15.16), we see that is the case if and only if the adversary has the same priority for all links. Moreover, if the adversary has the same priority for all links, then (15.20) can be replaced by a less restrictive condition; the proof is along the lines of that of Theorem 15.2 and thus omitted here.

**Corollary 15.2** *Consider the case of equal link capacities  $c_l = c_0$  for all  $l \in \mathcal{L}$  and attack priorities such that (15.20) holds. The routing action  $r^*$  defined by (15.21) is a weak Stackelberg equilibrium routing action for the game  $(\mathcal{R}, \mathcal{A}^*, J, H)$  if and only if there is a constant  $\gamma_0 \in (0, 1]$  such that*

$$\gamma_l = \gamma_0 \quad \forall l \in \mathcal{L}. \quad (15.29)$$

Moreover, if (15.29) holds, then (15.20) can be replaced by

$$R < Lc_0$$

and  $r^*$  defined by (15.21), that is,

$$r_l^* = 1/L, \quad l \in \mathcal{L},$$

is still both a strong Stackelberg equilibrium routing action and a weak one.

## 15.5 Mitigating Attacks via Learning

We now focus our attention on scenarios where the “adversary” may be driven by a cost more general than either (15.3) or (15.9) and, in fact, the router does not know the attack cost function  $H$  or even the attack action set  $\mathcal{A}$ , leading to a game with incomplete information. In this scenario, the attack strategy is a *best response function*  $f^* : \mathcal{R} \rightarrow \mathcal{A}$  that satisfies

$$f^*(r) \in \beta_a(r) \quad \forall r \in \mathcal{R},$$

but the function  $f^*$  (and the set-valued function  $\beta_a$  as well) is unknown to the router. Since the router cannot predict the attack action  $f^*(r)$  or the routing cost  $J(r, f^*(r))$  associated with a specific routing action  $r$ , it constructs a prediction of  $J(r, f^*(r))$  via a learning-based approach that consists of two components:

1. a neural network trained to construct a prediction  $\hat{J}(r)$  of the actual routing cost  $J(r, f^*(r))$  that results from a routing action  $r \in \mathcal{R}$ , and
2. a gradient descent algorithm used to minimize the predicted routing cost  $\hat{J}(r)$ .

### 15.5.1 Predicting the Routing Cost

We train a neural network

$$y = \text{NN}(\theta, r)$$

to predict the routing cost  $J(r, f^*(r))$  that results from a routing action  $r \in \mathcal{R}$ , where  $\theta$  denotes the parameters of the neural network. The input to the neural network is the routing action  $r$ ; thus there are  $L$  neurons in the input layer. Based on this input, the neural network predicts the actual user rates  $u$  as defined by (15.1) and the resulting routing cost  $J(r, f^*(r))$  as defined by (15.2); thus there are  $L + 1$  neurons in the output layer. We use  $n$  fully connected hidden layers, each of which consists of  $m$  *rectified linear units (ReLU)*. Therefore, the input  $x^i \in \mathbb{R}^m$  to the  $i$ -th hidden layer is given by

$$x^i = \text{ReLU}(\theta_{\text{weight}}^i x^{i-1} + \theta_{\text{bias}}^i), \quad i \in \{1, \dots, n\},$$

where  $x^0 = r \in \mathcal{R}$  is the input and ReLU is the *rectifier* function

$$\text{ReLU}(x) = \max\{x, 0\},$$

in which the maximum is taken in each scalar component. The output of the neural network  $y \in \mathbb{R}^{L+1}$  is given by

$$y = \theta_{\text{weight}}^{n+1} x^n + \theta_{\text{bias}}^{n+1}.$$

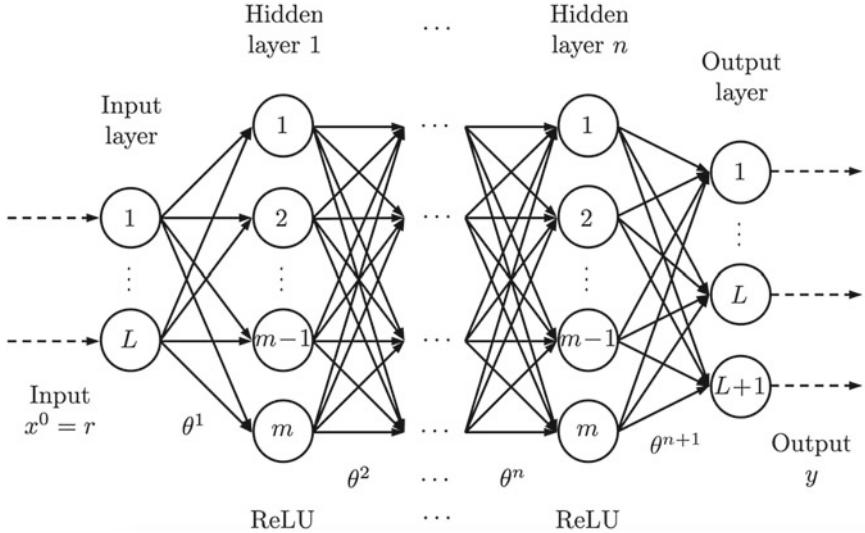
The neural network is visualized in Fig. 15.1. It is trained using a history of past routing actions and the corresponding values of actual user rates and routing costs. It is important to highlight that this neural network can be trained without observing the attack actions  $a$  and with no input of the total desired transmission rate of legitimate traffic  $R$ .

### 15.5.2 Minimizing the Predicted Routing Cost

After training, the router adjusts the routing action  $r \in \mathcal{R}$  based on a gradient descent algorithm to minimize the *predicted routing cost*  $\hat{J}(r) = y_{L+1}$ , which is the last component of the output  $y = \text{NN}(\theta, r)$  of the neural network. To specify the gradient descent algorithm, we recall the following notions and basic properties from convex analysis; for more details, see, e.g., [46, Sect. 6] or [47, Sect. 5.1].

For a closed convex set  $\mathcal{C} \subset \mathbb{R}^n$  and a point  $v \in \mathbb{R}^n$ , we denote by  $[v]_{\mathcal{C}}$  the *projection* of  $v$  onto  $\mathcal{C}$ , that is,

$$[v]_{\mathcal{C}} := \arg \min_{w \in \mathcal{C}} \|w - v\|.$$



**Fig. 15.1** The neural network used to predict the routing cost. There are  $L$  neurons in the input layer,  $L + 1$  neurons in the output layer, and  $n$  fully connected hidden layers each with  $m$  ReLU neurons

The projection  $[v]_{\mathcal{C}}$  satisfies  $[v]_{\mathcal{C}} = v$  if  $v \in \mathcal{C}$ . For a convex set  $\mathcal{S} \subset \mathbb{R}^n$  and a point  $x \in \mathcal{S}$ , we denote by  $T_{\mathcal{S}}(x)$  the *tangent cone* to  $\mathcal{S}$  at  $x$ , that is,

$$T_{\mathcal{S}}(x) := \overline{\{h(z - x) : z \in \mathcal{S}, h > 0\}}.$$

The set  $T_{\mathcal{S}}(x)$  is closed and convex, and satisfy  $T_{\mathcal{S}}(x) = \mathbb{R}^n$  if  $x \in \mathcal{S} \setminus \partial \mathcal{S}$ .

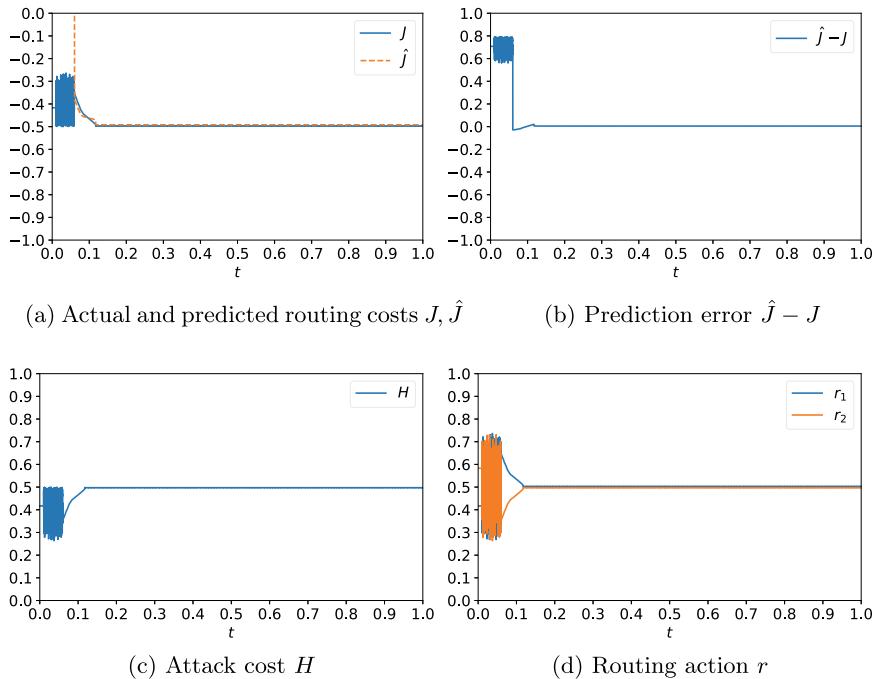
The gradient descent algorithm used to minimize the predict routing cost  $\hat{J}(r) = y_{L+1}$  is

$$\dot{r} = [-\lambda \nabla_r \hat{J}(r)]_{T_{\mathcal{R}}(r)} = [-\lambda \nabla_{x^n} y_{L+1} \nabla_{x^{n-1}} x^n \cdots \nabla_{x^0} x^1]_{T_{\mathcal{R}}(r)}, \quad (15.30)$$

where  $\lambda > 0$  is a preselected constant, and the projection  $[\cdot]_{T_{\mathcal{R}}(r)}$  onto the tangent cone  $T_{\mathcal{R}}(r)$  is used to guarantee that the routing action  $r$  remains inside the routing action set  $\mathcal{R}$ . Note that it is particularly convenient to use a gradient descent method in our approach, as the gradients  $\nabla_{x^n} y_{L+1}$ ,  $\nabla_{x^{n-1}} x^n$ , ..., and  $\nabla_{x^0} x^1$  are readily available from backpropagation during training.

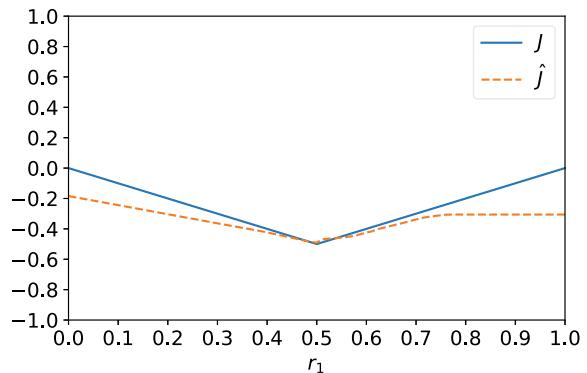
## 15.6 Simulation Study

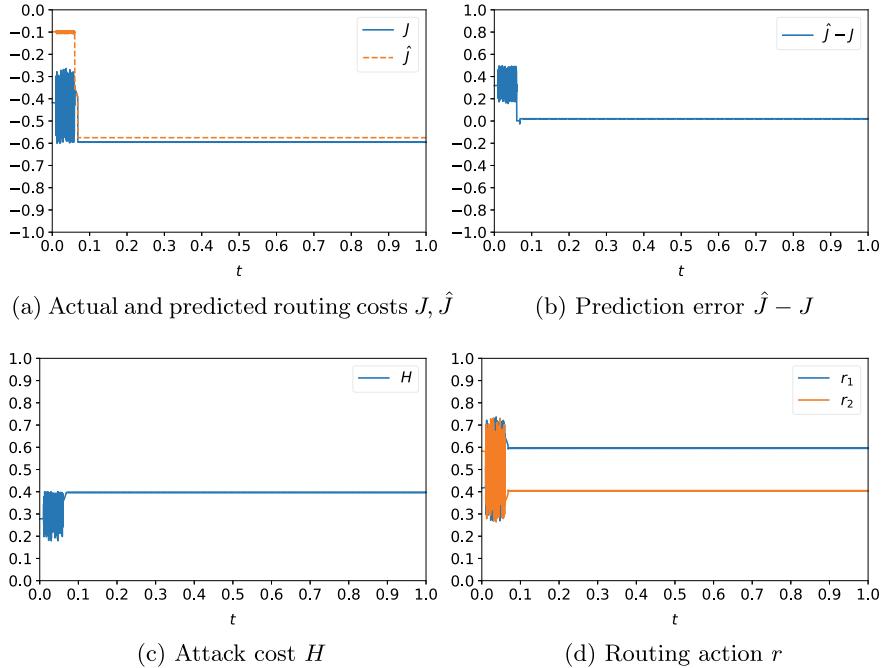
In this section, we present simulation results for the learning-based approach described in Sect. 15.5 against the malicious adversary described in Sect. 15.4.1, for networks with up to 10 parallel links.



**Fig. 15.2** Simulation results for a network of  $L = 2$  links and an attack priority vector  $\gamma = \mathbf{1}_2$ , using  $m = 16$  ReLU neurons in each hidden layer. The actual and predicted routing costs  $J$  and  $\hat{J}$  converge to  $-0.50$  and  $-0.49$ , respectively, where the former is within  $0.60\%$  from the weak Stackelberg routing cost  $J^* = -0.5$ . The prediction error  $\hat{J} - J$  converges to  $0.01$ , which is within  $1.04\%$  from  $J(T)$ . The attack cost  $H$  converges to  $0.50$ . The routing action  $r$  converges to  $(0.50, 0.50)$ , which is close to the weak Stackelberg equilibrium routing action  $r^* = \mathbf{1}_2/2$

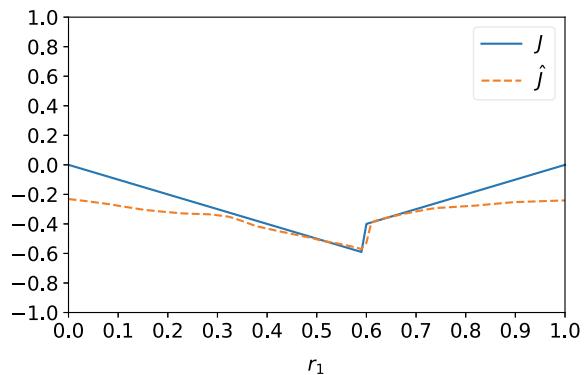
**Fig. 15.3** Actual and predicted routing cost functions  $J(r, f^*(r))$  and  $\hat{J}(r)$  for the simulation in Fig. 15.2

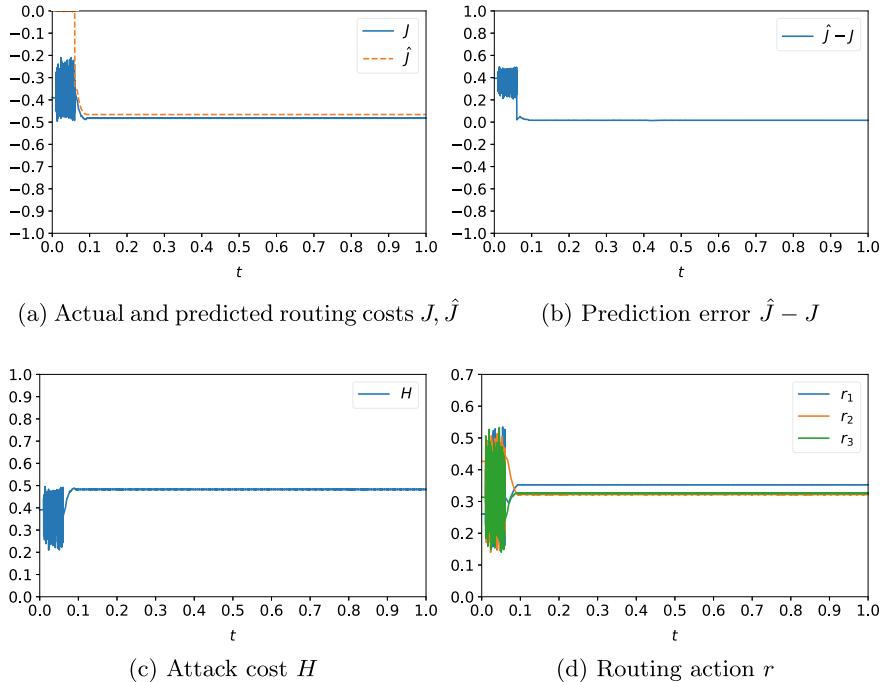




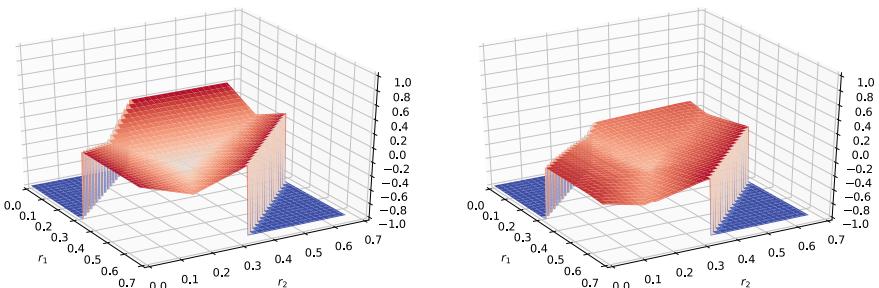
**Fig. 15.4** Simulation results for a network with  $L = 2$  links and the attack priority vector  $\gamma = (2/3, 1)$ , using  $m = 32$  ReLU neurons in each hidden layer. The actual and predicted routing costs  $J$  and  $\hat{J}$  converge to  $-0.60$  and  $-0.58$ , respectively, where the former is within  $0.77\%$  from the weak Stackelberg routing cost  $J^* = -3/5$ . The prediction error  $\hat{J} - J$  converges to  $0.02$ , which is within  $3.40\%$  from  $J(T)$ . The attack cost  $H$  converges to  $0.40$ . The routing action  $r$  converges to  $(0.60, 0.40)$ , which is close to the strong Stackelberg equilibrium routing action  $r^* = (3/5, 2/5)$

**Fig. 15.5** Actual and predicted routing cost functions  $J(r, f^*(r))$  and  $\hat{J}(r)$  for the simulation in Fig. 15.4

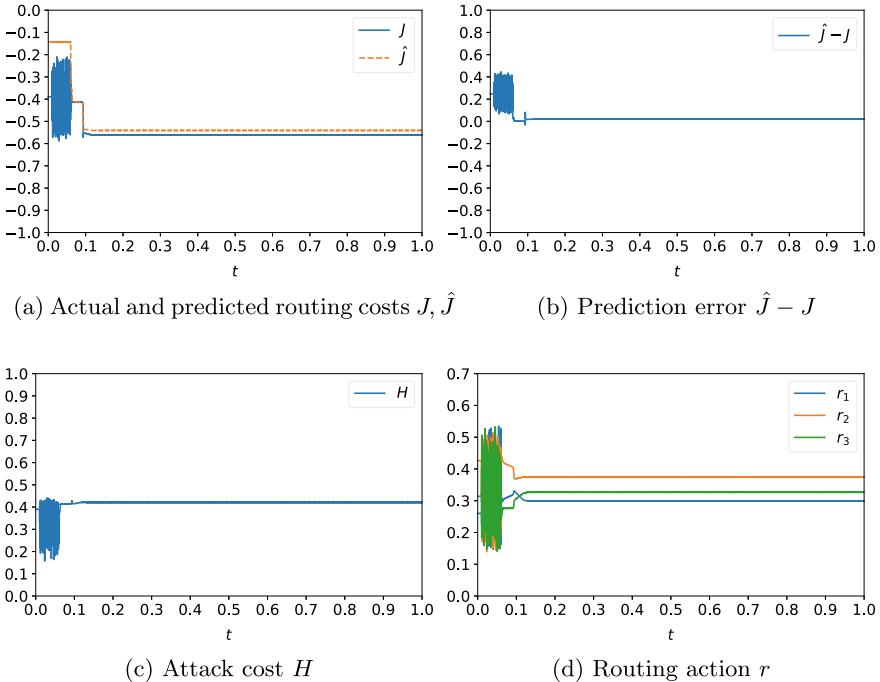




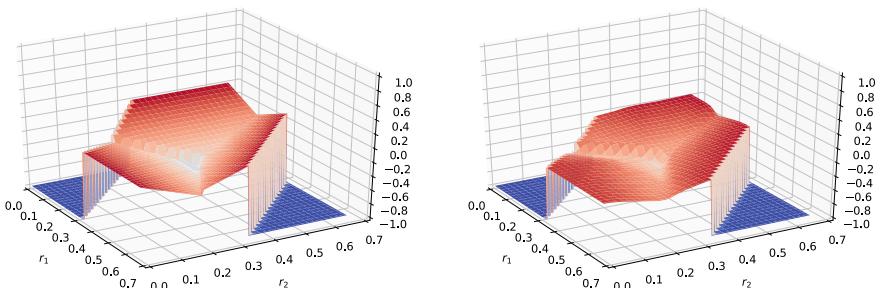
**Fig. 15.6** Simulation results for a network of  $L = 3$  links and an attack priority vector  $\gamma = \mathbf{1}_3$ , using  $m = 16$  ReLU neurons in each hidden layer. The actual and predicted routing costs  $J$  and  $\hat{J}$  converge to  $-0.48$  and  $-0.47$ , respectively, where the former is within  $3.52\%$  from the weak Stackelberg routing cost  $J^* = -1/2$ . The prediction error  $\hat{J} - J$  converges to  $0.02$ , which is within  $3.39\%$  from  $J(T)$ . The attack cost  $H$  converges to  $0.48$ . The routing action  $r$  converges to  $(0.35, 0.32, 0.33)$ , which is close to the weak Stackelberg equilibrium routing action  $r^* = \mathbf{1}_3/3$



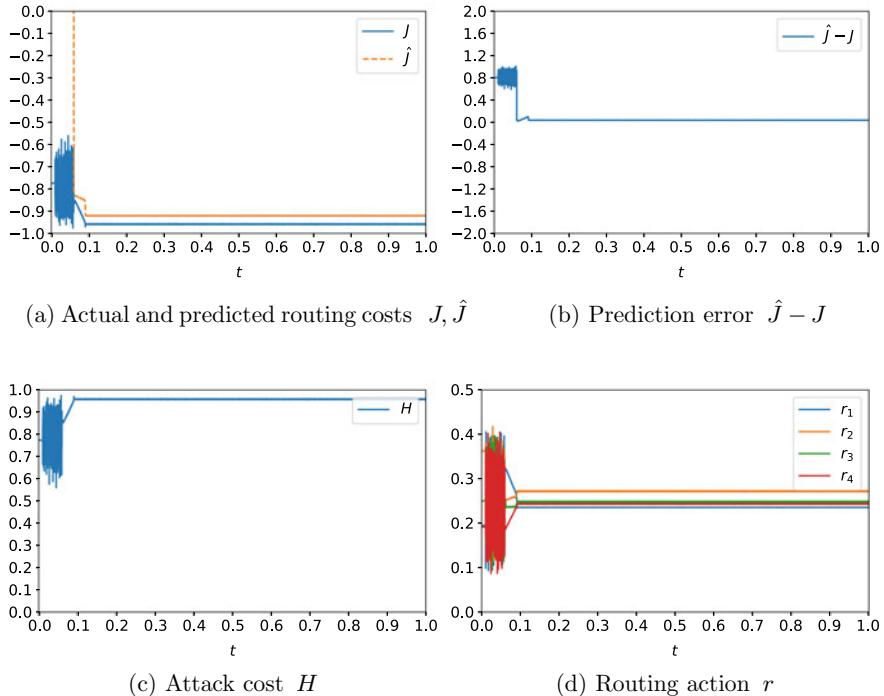
**Fig. 15.7** Actual and predicted routing cost functions  $J(r, f^*(r))$  and  $\hat{J}(r)$  for the simulation in Fig. 15.6



**Fig. 15.8** Simulation results for a network of  $L = 3$  links and an attack priority vector  $\gamma = (1, 0.5, 0.5)$ , using  $m = 32$  ReLU neurons in each hidden layer. The actual and predicted routing costs  $J$  and  $\hat{J}$  converge to  $-0.56$  and  $-0.54$ , respectively, where the former is within  $6.26\%$  from the weak Stackelberg routing cost  $J^* = -3/5$ . The prediction error  $\hat{J} - J$  converges to  $0.02$ , which is within  $3.85\%$  from  $J(T)$ . The attack cost  $H$  converges to  $0.42$ . The routing action  $r$  converges to  $(0.30, 0.37, 0.33)$ , which is close to the strong Stackelberg equilibrium routing action  $r^* = (3/10, 2/5, 3/10)$



**Fig. 15.9** Actual and predicted routing cost functions  $J(r, f^*(r))$  and  $\hat{J}(r)$  for the simulation in Fig. 15.8

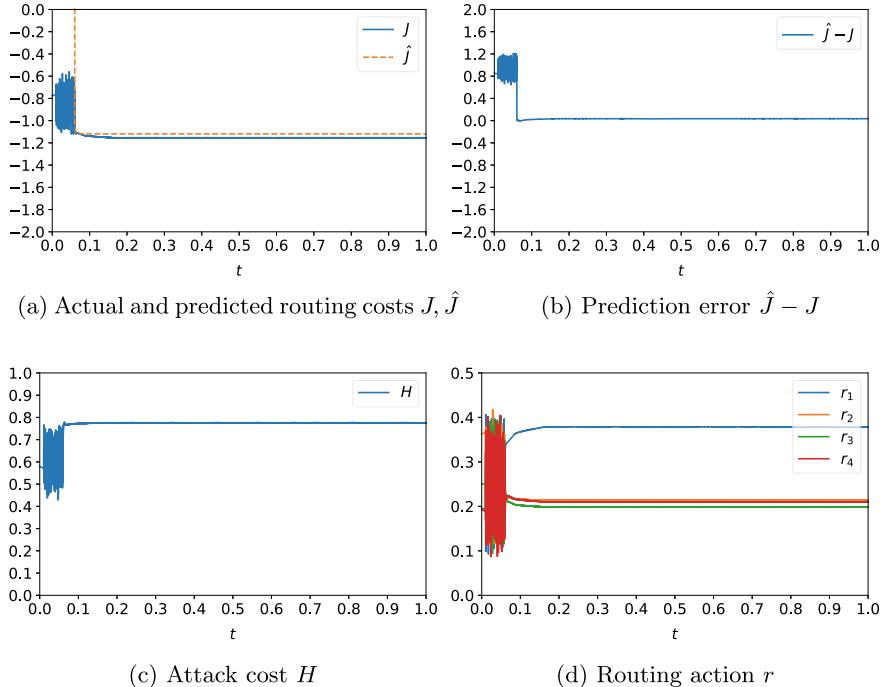


**Fig. 15.10** Simulation results for a network of  $L = 4$  links and an attack priority vector  $\gamma = \mathbf{1}_4$ , using  $m = 32$  ReLU neurons in each hidden layer. The actual and predicted routing costs  $J$  and  $\hat{J}$  converge to  $-0.96$  and  $-0.92$ , respectively, where the former is within  $4.27\%$  from the weak Stackelberg routing cost  $J^* = -1$ . The prediction error  $\hat{J} - J$  converges to  $0.04$ , which is within  $3.85\%$  from  $J(T)$ . The attack cost  $H$  converges to  $0.96$ . The routing action  $r$  converges to  $(0.24, 0.27, 0.25, 0.24)$ , which is close to the weak Stackelberg equilibrium routing action  $r^* = \mathbf{1}_4/4$

In the simulation, all links have the same capacity  $c_0 = 1$ , the total desired user rate  $R = L/2$ , and the attack budget  $A = \lfloor L/2 \rfloor$ . For cases where the adversary has the same priority for all links, we use the attack priority vector  $\gamma = \mathbf{1}_L$ ; otherwise  $\gamma \in (0, 1]^L$  will be specified.

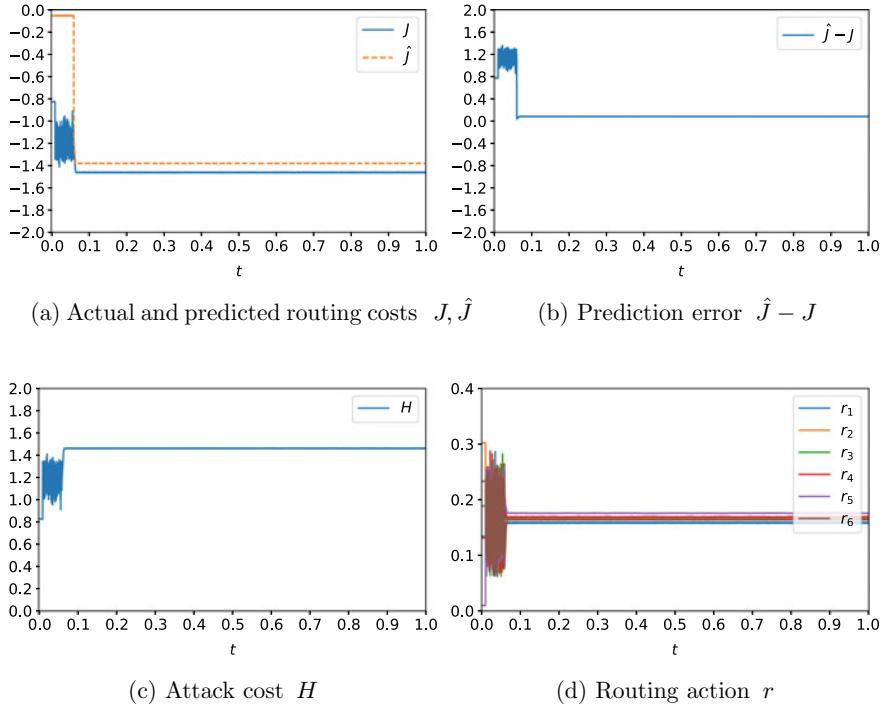
In this problem, a router that mistakenly believes the adversary is playing a constant attack action will regret its choice no matter which routing action is played; thus there is no Nash equilibrium for the game  $(\mathcal{R}, \mathcal{A}^*, J, H)$  [17, p. 106]. The results in Sect. 15.4.1 show that there is a strong Stackelberg equilibrium routing action  $r^*$  given by (15.21) but not necessarily a weak Stackelberg equilibrium routing action, while the strong and weak Stackelberg routing costs are both  $J^*$  given by (15.22). However, even if the adversary always plays an optimal attack action that is worst for the router, the router is able to approach the weak Stackelberg routing cost  $J^*$  using a weak  $\varepsilon$  Stackelberg routing action  $r^\varepsilon$  closed to  $r^*$  given by (15.28).

In each simulation, a neural network is constructed using Python 3.7.7 and PyTorch 1.3.1. It has  $n = 3$  fully connected hidden layers, each with  $m$  ReLU neu-



**Fig. 15.11** Simulation results for a network of  $L = 4$  links and an attack priority vector  $\gamma = (0.5, 1, 1, 1)$ , using  $m = 64$  ReLU neurons in each hidden layer. The actual and predicted routing costs  $J$  and  $\hat{J}$  converge to  $-1.15$  and  $-1.12$ , respectively, where the former is within  $3.79\%$  from the weak Stackelberg routing cost  $J^* = -6/5$ . The prediction error  $\hat{J} - J$  converges to  $0.04$ , which is within  $3.05\%$  from  $J(T)$ . The attack cost  $H$  converges to  $0.78$ . The routing action  $r$  converges to  $(0.38, 0.21, 0.20, 0.21)$ , which is close to the strong Stackelberg equilibrium routing action  $r^* = (2/5, 1/5, 1/5, 1/5)$

rons. One of the focus for this simulation study is to construct the predicted routing cost  $\hat{J}(r)$  using a small set of training data. Therefore, we train the neural network using only 500 samples with randomly generated routing actions (sampled from the 101-st to the 600-th unit of time). The neural network is trained for 2000 epochs for each simulation, with the batch size 80 and the learning rate 0.001. After training, the gradient algorithm (15.30) is applied to minimize the predicted routing cost  $\hat{J}(r)$  generated by the neural network, until the  $T = 10000$ -th unit of time. The constant in (15.30) is set by  $\lambda = 0.001$ . The simulation results are shown in Figs. 15.2, 15.3, 15.4, 15.5, 15.6, 15.7, 15.8, 15.9, 15.10, 15.11, 15.12, and 15.13. In Figs. 15.2, 15.4, 15.6, 15.8, 15.10, 15.11, 15.12, and 15.13, the horizontal axis is in  $\times 10^4$  units of time.



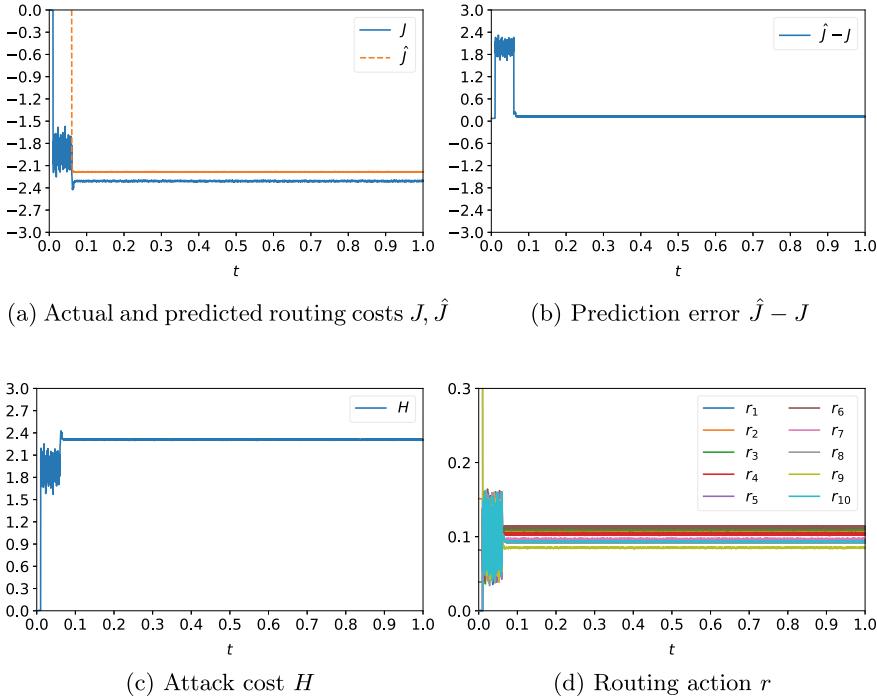
**Fig. 15.12** Simulation results for a network of  $L = 6$  links and an attack priority vector  $\gamma = \mathbf{1}_6$ , using  $m = 64$  ReLU neurons in each hidden layer. The actual and predicted routing costs  $J$  and  $\hat{J}$  converge to  $-1.46$  and  $-1.38$ , respectively, where the former is within  $2.58\%$  from the weak Stackelberg routing cost  $J^* = -3/2$ . The prediction error  $\hat{J} - J$  converges to  $0.08$ , which is within  $5.57\%$  from  $J(T)$ . The attack cost  $H$  converges to  $1.46$ . The routing action  $r$  converges to  $(0.16, 0.17, 0.16, 0.17, 0.18, 0.16)$ , which is close to the weak Stackelberg equilibrium routing action  $r^* = \mathbf{1}_6/6$

### 15.6.1 Discussion

In Table 15.1, we summarize the simulation parameters and final results at time  $T$ , including the number of links  $L$ , the attack priority vector  $\gamma$ , the number of neurons in hidden layers  $m$ , the strong Stackelberg equilibrium routing action  $r^*$  defined by (15.21), the weak Stackelberg routing cost  $J^*$  defined by (15.22), the final actual and predicted routing costs  $J(T) := J(r(T), f^*(r(T)))$  and  $\hat{J}(T) := \hat{J}(r(T))$ , the relative difference  $|J(T)/J^* - 1|$ , the percentage prediction error  $|\hat{J}(T)/J(T) - 1|$ , and the corresponding figures.

Based on Table 15.1 and Figs. 15.2–15.13, we make the following observations:

- Even with a small set of training data (500 samples), our learning-based approach is able to provide a routing cost  $J(T)$  that is reasonably close to the weak Stackelberg routing cost  $J^*$ .



**Fig. 15.13** Simulation results for a network of  $L = 10$  links and an attack priority vector  $\gamma = \mathbf{1}_{10}$ , using  $m = 128$  ReLU neurons in each hidden layer. The actual and predicted routing costs  $J$  and  $\hat{J}$  converge to  $-2.30$  and  $-2.19$ , respectively, where the former is within 7.94% from the weak Stackelberg routing cost  $J^* = -5/2$ . The prediction error  $\hat{J} - J$  converges to  $0.11$ , which is within 4.96% from  $J(T)$ . The attack cost  $H$  converges to  $2.30$ . The routing action  $r$  converges to  $(0.10, 0.11, 0.11, 0.10, 0.09, 0.11, 0.10, 0.09, 0.09, 0.09)$ , which is close to the weak Stackelberg equilibrium routing action  $r^* = \mathbf{1}_{10}/10$

2. Figures 15.5 and 15.9 demonstrate that our neural networks succeed at approximating the routing cost in spite of strong discontinuities in the function  $J(r, f^*(r))$  that result from non-unique optimal attack actions against a strong Stackelberg equilibrium routing action. However, this is clearly a challenge for high-dimensional problems and motivates the need for further research.
3. In practice, the selection of the routing action  $r$  only cares about minimizing the last component of the neural network's output (the predicted routing cost  $\hat{J}(r)$  above). However, we found that the performance of the neural network improved significantly when it was trained to also predict the actual user rates  $u$ . Our conjecture is that the additional dimensions in training data force the hidden layers to “respond” to the actual user rates and thus provides a more persistent structure for the neural network. This phenomenon will also be a topic for future research.

**Table 15.1** Simulation parameters and results

$L$	$\gamma$	$m$	$r^*$	$J^*$	$J(T)$	$\hat{J}(T)$	$  \frac{J(T)}{J^*} - 1  $	$  \frac{\hat{J}(T)}{J^*} - 1  $	Figs.
2	$\mathbf{1}_2$	16	$\frac{1}{2}\mathbf{1}_2$	$-\frac{1}{2}$	-0.50	-0.49	0.60%	1.04%	<a href="#">15.2, 15.3</a>
2	$(\frac{2}{3}, 1)$	32	$(\frac{3}{5}, \frac{2}{5})$	$-\frac{3}{5}$	-0.60	-0.58	0.77%	3.40%	<a href="#">15.4, 15.5</a>
3	$\mathbf{1}_3$	16	$\frac{1}{3}\mathbf{1}_3$	$-\frac{1}{2}$	-0.48	-0.47	3.52%	3.39%	<a href="#">15.6, 15.7</a>
3	$(1, \frac{3}{4}, 1)$	32	$(\frac{3}{10}, \frac{2}{5}, \frac{3}{10})$	$-\frac{3}{5}$	-0.56	-0.54	6.26%	3.85%	<a href="#">15.8, 15.9</a>
4	$\mathbf{1}_4$	32	$\frac{1}{4}\mathbf{1}_4$	-1	-0.96	-0.92	4.27%	3.85%	<a href="#">15.10</a>
4	$(\frac{1}{2}, 1, 1, 1)$	64	$(\frac{2}{5}, \frac{1}{5}, \frac{1}{5}, \frac{1}{5})$	$-\frac{6}{5}$	-1.15	-1.12	3.79%	3.05%	<a href="#">15.11</a>
6	$\mathbf{1}_6$	64	$\frac{1}{6}\mathbf{1}_6$	$-\frac{3}{2}$	-1.46	-1.38	2.58%	5.57%	<a href="#">15.12</a>
10	$\mathbf{1}_{10}$	128	$\frac{1}{10}\mathbf{1}_{10}$	$-\frac{5}{2}$	-2.30	-2.19	7.94%	4.96%	<a href="#">15.13</a>

## 15.7 Conclusion

We formulated the mitigation of link-flooding DDoS attacks as a routing problem among parallel links. A Stackelberg game model was constructed to address the challenge that the adversary can observe the routing strategy before assigning attack traffic. For a general class of adversaries, we characterized an optimal attack that belongs to a finite set, and constructed explicit formulae for Stackelberg equilibria and costs for a special class of networks. For the more general case of unknown attack cost and capacity, we proposed a learning-based approach that predicts the routing cost using a neural network and then minimizes the predicted routing cost via projected gradient descent. The effectiveness of our approach was demonstrated through a simulation study.

A future research direction is to extend our results to scenarios where the attack objective is partially known and incorporates the additional information about attack to our learning-based approach. For example, if the attack was known to belong to the finite set  $\mathcal{A}^*$  defined in (15.12), how should the router adjust the neural network to improve its efficiency? Other future research topics include to test our approach in more complex networks and to generalize our results to the case of multiple routers and/or adversaries.

## References

1. Criscuolo, P.J.: Distributed denial of service: Trin00, tribe flood network, tribe flood network 2000, and stacheldraht. Technical report, Lawrence Livermore National Laboratory, University of California (2000)
2. Mirkovic, J., Reiher, P.: A taxonomy of DDoS attack and DDoS defense mechanisms. ACM SIGCOMM Comput. Commun. Rev. **34**(2), 39–53 (2004)
3. Zargar, S.T., Joshi, J., Tipper, D.: A survey of defense mechanisms against distributed denial of service (DDoS) flooding attacks. IEEE Commun. Surv. Tutor. **15**(4), 2046–2069 (2013)
4. NETSCOUT’s 14th Annual Worldwide Infrastructure Security Report, NETSCOUT Systems, Inc, Technical report (2019)

5. Studer, A., Perrig, A.: The coremelt attack. In: 16th European Symposium on Research in Computer Security, pp. 37–52 (2011)
6. Kang, M.S., Lee, S.B., Gligor, V.D.: The crossfire attack. In: IEEE Symposium on Security and Privacy 2013, pp. 127–141 (2013)
7. Lee, S.B., Kang, M.S., Gligor, V.D.: CoDef: collaborative defense against large-scale link-flooding attacks. In: 9th ACM Conference on Emerging Networking Experiments and Technologies, pp. 417–428 (2013)
8. Kang, M.S., Gligor, V.D., Sekar, V.: SPIFFY: inducing cost-detectability tradeoffs for persistent link-flooding attacks. In: Network and Distributed System Security Symposium, pp. 1–15 (2016)
9. Liaskos, C., Kotronis, V., Dimitropoulos, X.: A novel framework for modelling and mitigating distributed link flooding attacks. In: 35th Annual IEEE International Conference on Computer Communications, pp. 1–9 (2016)
10. Aydeger, A., Saputro, N., Akkaya, K., Rahman, M.: Mitigating Crossfire attacks using SDN-based moving target defense. In: 2016 IEEE 41st Conference on Local Computer Networks, pp. 627–630 (2016)
11. Gkounis, D., Kotronis, V., Liaskos, C., Dimitropoulos, X.: On the interplay of link-flooding attacks and traffic engineering. ACM SIGCOMM Comput. Commun. Rev. **46**(2), 5–11 (2016)
12. Xue, L., Ma, X., Luo, X., Chan, E.W.W., Miu, T.T.N., Gu, G.: LinkScope: toward detecting target link flooding attacks. IEEE Trans. Inf. Forensics Secur. **13**(10), 2423–2438 (2018)
13. Yang, G., Hosseini, H., Sahabandu, D., Clark, A., Hespanha, J.P., Poovendran, R.: Modeling and mitigating the Coremelt attack. In: American Control Conference, pp. 3410–3416 (2018)
14. Fudenberg, D., Tirole, J.: Game Theory. MIT Press, Cambridge (1991)
15. Osborne, M.J., Rubinstein, A.: A Course in Game Theory. MIT Press, Cambridge (1994)
16. Basar, T., Olsder, G.J.: Dynamic Noncooperative Game Theory, 2nd edn. SIAM (1999)
17. Hespanha, J.P.: Noncooperative Game Theory: An Introduction for Engineers and Computer Scientists. Princeton University Press, Princeton (2017)
18. Alpcan, T., Basar, T.: Network Security: A Decision and Game-Theoretic Approach. Cambridge University Press, Cambridge (2010)
19. Liang, X., Xiao, Y.: Game theory for network security. IEEE Commun. Surv. Tutor. **15**(1), 472–486 (2013)
20. Do, C.T., Tran, N.H., Hong, C., Kamhoua, C.A., Kwiat, K.A., Blasch, E., Ren, S., Pissinou, N., Iyengar, S.S.: Game theory for cyber security and privacy. ACM Comput. Surv. **50**(2), 1–37 (2017)
21. von Stackelberg, H.: Market Structure and Equilibrium. Springer, Berlin (2011), transl. from German
22. Korilis, Y.A., Lazar, A.A., Orda, A.: Achieving network optima using Stackelberg routing strategies. IEEE/ACM Trans. Netw. **5**(1), 161–173 (1997)
23. Roughgarden, T.: Stackelberg scheduling strategies. SIAM J. Comput. **33**(2), 332–350 (2004)
24. Bloem, M., Alpcan, T., Basar, T.: A Stackelberg game for power control and channel allocation in cognitive radio networks. In: 2nd International Conference on Performance Evaluation Methodologies and Tools, pp. 1–9 (2007)
25. Pita, J., Jain, M., Marecki, J., Ordóñez, F., Portway, C., Tambe, M., Western, C., Paruchuri, P., Kraus, S.: Deployed ARMOR protection: the application of a game-theoretic model for security at the Los Angeles International Airport. In: 7th International Conference on Autonomous Agents and Multiagent Systems, pp. 125–132 (2008)
26. Tsai, J., Rathi, S., Kiekintveld, C., Ordóñez, F., Tambe, M.: IRIS - A tool for strategic security allocation in transportation networks. In: 8th International Conference on Autonomous Agents and Multiagent Systems, pp. 37–44 (2009)
27. Brown, G.G., Carlyle, W.M., Salmerón, J., Wood, K.: Analyzing the vulnerability of critical infrastructure to attack and planning defenses. In: Smith, J.C. (eds.) Emerging Theory, Methods, and Applications, pp. 102–123. INFORMS (2005)
28. Brown, G.G., Carlyle, M., Salmerón, J., Wood, K.: Defending critical infrastructure. Interfaces **36**(6), 530–544 (2006)

29. Grimsman, D., Hespanha, J.P., Marden, J.R.: Stackelberg equilibria for two-player network routing games on parallel networks. In: 2020 American Control Conference, 2020, to be published
30. Brown, G.W.: Iterative solution of games by fictitious play. In: Koopmans, T.C. (ed.) *Activity Analysis of Production and Allocation*, pp. 374–376. Wiley, Hoboken (1951)
31. Robinson, J.: An iterative method of solving a game. *Ann. Math.* **54**(2), 296–301 (1951)
32. Brown, G.W., von Neumann, J.: Solutions of games by differential equations. In: Kuhn, H.W., Tucker, A.W. (eds.) *Contributions to the Theory of Games*, vol. I, ch. 6, pp. 73–80. Princeton University Press, Princeton (1952)
33. Rosen, J.B.: Existence and uniqueness of equilibrium points for concave n-person games. *Econometrica* **33**(3), 520–534 (1965)
34. Brückner, M., Scheffer, T.: Stackelberg games for adversarial prediction problems. In: 17th ACM International Conference on Knowledge Discovery and Data Mining, pp. 547–555 (2011)
35. Marecki, J., Tesauro, G., Segal, R.: Playing repeated Stackelberg games with unknown opponents. In: 11th International Conference on Autonomous Agents and Multiagent Systems, vol. 2, pp. 821–828 (2012)
36. Blum, A., Haghtalab, N., Procaccia, A.D.: Learning optimal commitment to overcome insecurity. *Neural Inf. Process. Syst.* **2014**, 1826–1834 (2014)
37. Yang, G., Poovendran, R., Hespanha, J.P.: Adaptive learning in two-player Stackelberg games with continuous action sets. In: 58th IEEE Conference on Decision and Control, pp. 6905–6911 (2019)
38. Kang, M.S., Gligor, V.D.: Routing bottlenecks in the Internet: Causes, exploits, and countermeasures. In: ACM SIGSAC Conference on Computer and Communications Security, pp. 321–333 (2014)
39. Postel, J.: User Datagram Protocol. Internet Engineering Task Force, Internet Standard (1980)
40. Henderson, T., Floyd, S., Gurtov, A., Nishida, Y.: The NewReno modification to TCP's fast recovery algorithm. Internet Engineering Task Force, Internet Standard (2012)
41. Leitmann, G.: On generalized Stackelberg strategies. *J. Optim. Theory Appl.* **26**(4), 637–643 (1978)
42. Breton, M., Alj, A., Haurie, A.: Sequential Stackelberg equilibria in two-person games. *J. Optim. Theory Appl.* **59**(1), 71–97 (1988)
43. von Stengel, B., Zamir, S.: Leadership with commitment to mixed strategies. Technical report, Centre for Discrete and Applicable Mathematics, London School of Economics (2004)
44. Kiekintveld, C., Jain, M., Tsai, J., Pita, J., Ordóñez, F., Tambe, M.: Computing optimal randomized resource allocations for massive security games. In: 8th International Conference on Autonomous Agents and Multiagent Systems, pp. 689–696 (2009)
45. Kellerer, H., Pferschy, U., Pisinger, D.: *Knapsack Problems*. Springer, Berlin (2004)
46. Rockafellar, R.T., Wets, R.J.B.: *Variational Analysis*. Springer, Berlin (1998)
47. J.-P. Aubin, *Viability Theory*. Birkhäuser, Basel (1991)

**Part IV**

**Bounded Rationality and Value  
of Information in RL and Games**

# Chapter 16

## Bounded Rationality in Differential Games: A Reinforcement Learning-Based Approach



Nick-Marios T. Kokolakis, Aris Kanellopoulos, and Kyriakos G. Vamvoudakis

**Abstract** This chapter presents a unified framework of bounded rationality for control systems, as this can be employed in a coordinated unmanned aerial vehicle tracking problem. By considering limitations in the decision-making mechanisms of the vehicles and utilizing learning algorithms that capture their level of cognition, we are able to design appropriate countermeasures. Via game-theoretic concepts, we derive the Nash equilibrium of the interaction between the evader and the pursuing team, with the first being the maximizing player and the team being the minimizing ones. We derive optimal pursuing and evading policies while taking into account the physical constraints imposed by Dubins vehicles. Subsequently, the infinite rationality assumption underlying the equilibrium is relaxed, and level- $k$  thinking policies are computed through reinforcement-learning architectures. Convergence to the Nash equilibrium is shown as the levels of intelligence increase. Finally, simulation results verify the efficacy of our approach.

### 16.1 Introduction

Game theory is a powerful tool that captures the behavior of self-interested, competing agents [1]. When those agents interact upon a dynamically changing environment, notions of control theory can be introduced in order to define complex differential games [2]. The theory of games, under various solution concepts, has been utilized in a plethora of security applications, both in the cyber and the cyber-physical domains. In its majority, the research that has been conducted on security applications has focused on the equilibrium notions of games introduced by Nash [3].

---

N.-M. T. Kokolakis (✉) · A. Kanellopoulos · K. G. Vamvoudakis

Georgia Institute of Technology, Atlanta, GA 30332, USA

e-mail: [nmkokolakis@gatech.edu](mailto:nmkokolakis@gatech.edu)

A. Kanellopoulos

e-mail: [ariskan@gatech.edu](mailto:ariskan@gatech.edu)

K. G. Vamvoudakis

e-mail: [kyriakos@gatech.edu](mailto:kyriakos@gatech.edu)

Even though the Nash equilibrium is one of the most important concepts in game theory, and experiments conducted on humans are known to fail to abide by the predictions derived via such models. Consequently, there has been extensive research on alternative solution concepts that capture the nonequilibrium behavior exhibited by the experiments' participants. Those frameworks aim to model the bounded rationality that forces competing agents to make sub-optimal, out-of-equilibrium decisions in various games. The two underlying assumptions that permeate Nash equilibria in both static and dynamic games are the assumption of optimality—the fact that every decision-maker deterministically optimizes with respect to their objective—and the assumption of mutual consistency—the fact that each agent's beliefs over their opponents' strategies are correct. In our work, we propose a bounded rationality framework for differential games based on level- $k$  and cognitive hierarchy theories; frameworks that maintain the optimality assumption but relax the need for mutual consistency.

Our ultimate goal is to formulate a concrete framework that is able to be implemented in the cyber-physical security domain. One of the most sensitive civilian cyber-physical markets is that of unmanned aerial vehicles (UAVs), whose decrease in manufacturing cost, increase in availability, and of ease of use have enabled their penetration in various applications. In turn, the need to protect the public against accidents or malicious operators has become all the more important as well. As such, a priority in the aerospace community is to develop algorithms that will enable autonomous swarms of UAVs to apprehend vehicles that enter protected airspace—a phenomenon that has already been observed—in order to enforce “geofencing” protocols.

One of the greatest decision-theoretic issues that we have to solve to allow the autonomous agents to employ those protocols is the problem of strategy prediction given observations of the opponents' behavior. This will allow the autonomous UAVs to estimate the future position of a moving target; a problem that becomes even more complicated in human-centric environments. Given the potentially adversarial nature of the invading agent, this problem can be investigated in game-theoretic terms. Furthermore, the existence of human attackers makes the consideration of boundedly rational frameworks mandatory.

### 16.1.1 Related Work

One of the first works on nonequilibrium game-theoretic behavior in static environments has been reported in [4]. The work of [5, 6] discusses the development of a low-rationality game-theoretic framework, namely, behavioral game theory. Quantal response models [7] take into account stochastic mistakes perturbing the optimal policies. Structural nonequilibrium models were considered in [8] and applied for system security in [9] and autonomous vehicle behavioral training in [10]. Finally, the authors in [11] introduced nonequilibrium concepts for differential games.

Research in the area of coordinated tracking of moving target(s) using cooperative UAVs has made remarkable progress. Specifically, the coordinated standoff tracking has been covered enough providing useful results [12–16] where the UAVs are loitering around the target with the desired phase separation. In addition, optimal strategies have been developed in which the fixed wings UAVs are equipped with cameras, and they collaborate in order to attain a multitude of goals; namely, to reduce the geolocation error [17, 18] or to track an unpredictable moving ground vehicle [19]. Finally, the work of [20] formulates the coordinated target tracking problem as a nonequilibrium differential game achieving the coordination of a team of bounded rational UAVs for tracking successfully a highly maneuvering target.

## 16.2 Problem Formulation

In this section, we describe the mathematical representation that captures the behavior of the competing agents in its general setting, as well as in simple—and easier to tackle—scenarios.

### 16.2.1 Nash Equilibrium Solutions for Differential Games

Initially, we describe the differential game structure along with the Nash equilibrium solutions. Let  $\mathcal{N}$  denote the set of all players, and  $\mathcal{N}_{-i} = \mathcal{N} \setminus \{i\}$  the set containing all players except player  $i$ . In order to investigate the interactions between non-cooperative players in dynamically changing environments, we define an underlying nonlinear time-invariant dynamical system upon which the agents act,

$$\dot{x} = f(x) + \sum_{j \in \mathcal{N}} g_j(x)u_j, \quad x(0) = x_0, \quad t \geq 0,$$

where  $x(t) \in \mathbb{R}^n$  is the state of the environment,  $u_i \in \mathbb{R}^m$ ,  $\forall i \in \mathcal{N}$  the decision variables,  $f(x) : \mathbb{R}^n \rightarrow \mathbb{R}^n$  the drift dynamics, and  $g_i(x) : \mathbb{R}^n \rightarrow \mathbb{R}^m$ ,  $\forall i \in \mathcal{N}$  the input dynamics.

Each player aims to minimize a cost functional of the form,

$$\begin{aligned} J_i(u_1, \dots, u_N) &= \int_0^\infty r_i(x(\tau), u_1, \dots, u_N) d\tau \\ &:= \int_0^\infty \left( Q_i(x) + u_i^T R_{ii} u_i + \sum_{j \in \mathcal{N}_{-i}} u_j^T R_{ij} u_j \right) d\tau, \quad \forall i \in \mathcal{N}. \end{aligned}$$

Due to the coupled nature of the cost functionals and the environment dynamics, there is a plethora of different approaches to the “solution” of a differential game. The most common one, which constitutes a rational equilibrium concept, is the Nash equilibrium solution [1].

**Definition 16.1** A tuple of decision policies  $(u_1^*, \dots, u_i^*, \dots, u_N^*)$  constitute a Nash equilibrium if it holds that

$$J_i(u_1^*, \dots, u_i^*, \dots, u_N^*) \leq J_i(u_1^*, \dots, u_i, \dots, u_N^*), \quad \forall i \in \{1, \dots, N\}.$$

The derivation of the Nash policies  $u_i^*$ ,  $i \in \{1, \dots, N\}$  corresponds to solving the coupled minimization problems,

$$J_i(u_i^*, u_{-i}^*) = \min_{u_i} J_i(u_i, u_{-i}^*), \quad i \in \{1, \dots, N\},$$

where for brevity we define  $u_{-i} = (u_1, \dots, u_{i-1}, u_{i+1}, \dots, u_N)$ .

Following methods from the theory of optimal control [21], the goal of the decision-makers is to evaluate the optimal value functions  $V_i(x)$ ,  $i \in \{1, \dots, N\}$  such that

$$V_i(x) = \min_{u_i} \int_t^\infty r_i(x, u_i, u_{-i}^*) d\tau.$$

This is achieved with the help of the Hamiltonian function, defined  $\forall i \in \{1, \dots, N\}$  as

$$H_i(x, u_i, u_{-i}) = r_i(x(t), u_1, \dots, u_N) + \nabla V_i^T \left( f(x) + \sum_{j \in \mathcal{N}} g_j(x) u_j \right). \quad (16.1)$$

It is known that minimizing inputs on the system minimize the Hamiltonian. Consequently, this allows for the use of stationarity conditions to derive the Nash policies,

$$\frac{\partial H}{\partial u_i} = 0 \Rightarrow u_i^*(x) = -R_{ii}^{-1} g_i^T(x) \frac{\partial V_i^*}{\partial x}. \quad (16.2)$$

Substituting (16.2) into (16.1) yields the following coupled Hamilton–Jacobi system of  $N$  equations,

$$\begin{aligned} Q_i(x) + \frac{1}{4} \sum_{j \in \mathcal{N}} \nabla V_j^T g_j(x) R_{jj}^{-T} R_{ij} R_{jj} g_j^T(x) \nabla V_j + \\ (\nabla V_i)^T \left( f(x) - \frac{1}{2} \sum_{j \in \mathcal{N}} g_j(x) R_{jj}^{-1} g_j^T(x) \nabla V_j \right) = 0. \end{aligned} \quad (16.3)$$

A specific case of the general framework presented above that allows us to exhibit clearer results while capturing a large class of applications is the linear quadratic

zero-sum case. In this scenario, we consider that  $\mathcal{N} = \{1, 2\}$ , i.e., there are two players, whose cost functionals are such that  $J_1 = -J_2 = J$ . This formulation is extremely useful in the field of cyber-physical security, in which the competing players correspond to a defending agent (or team of defenders), whose decision vector is  $u(t) \in \mathbb{R}^m$  and an attacking agent whose decision is denoted by  $d(t) \in \mathbb{R}^l$ . Furthermore, the environment evolves according to the simplified dynamics,

$$\dot{x}(t) = Ax(t) + Bu(t) + Kd(t), \quad x(0) = x_0, \quad (16.4)$$

and the cost is quadratic in its arguments,

$$J(u, d) = \frac{1}{2} \int_0^\infty (x^T M x + u^T R u - \gamma^2 \|d\|) dt.$$

Due to the zero-sum property of the players' costs, we can redefine the problem of deriving the Nash equilibrium, as one of solving for the minimax policies, i.e.,

$$V^*(x) = \min_u \max_d \int_0^\infty (x^T M x + u^T R u - \gamma^2 \|d\|) dt.$$

The value function for the linear quadratic case is known to also be a quadratic in the states function, i.e.,  $V^*(x) = x^T P x$ . The optimal kernel  $P$  satisfies a simplified version of (16.3),

$$A^T P + PA - PBR^{-1}B^T P + \frac{1}{\gamma^2} P K K^T P + M = 0. \quad (16.5)$$

Finally, the Nash policies can be found after solving (16.5) for  $P$  and substituting in,

$$\begin{aligned} u^*(x) &= -R^{-1} B^T P x, \\ d^*(x) &= \frac{1}{\gamma} K^T P x. \end{aligned}$$

Computational and cognitive limitations have been shown to prevent players from reaching the Nash equilibrium solution even in simple scenarios [22]. This behavior, which has been observed in most experimental setups, has lead to the formulation of various nonequilibrium solution concepts that aim to better approximate realistic agents. In our work, we introduce bounded rationality concepts via the level- $k$  thinking and cognitive hierarchy frameworks. Toward this, we will define strategies of lower "level- $k$ " thinkers that perform a certain number of steps of strategic thinking and show methods to derive the distribution of beliefs for their abilities.

### 16.3 Boundedly Rational Game Solution Concepts

In this section, we construct a framework that introduces level- $k$  thinking strategies in differential games. Initially, we will propose an iterative approach to deriving boundedly rational decision policies by leveraging structural connections between cognitive hierarchy theory and reinforcement-learning techniques. Subsequently, we will investigate how a high-level thinker can obtain beliefs on the cognitive abilities of their opponents through repeated interaction.

Consider a player  $i \in \mathcal{N}$  belonging to a cognitive type  $k \in \mathbb{Z}^+$ . This player is able to perform  $k$ -steps of iterative strategic thinking while her beliefs about the behavior of the rest of the players are constructed based on a fixed, full-support distribution, e.g., according to the work of [8], a Poisson distribution.

**Level-0 policies:** The approach of iterative best responses to model bounded rationality—such as the one we employ in this work—requires the existence of a common *anchor strategy*; a policy that a level-0 player will adopt. It is clear that the assumption that the structure of the anchor policy is common knowledge to all the players is a strict one. However, in various scenarios those policies can be derived as completely random responses or as focal policies for the game without loss of generality of the framework [23].

**Level-k policies:** All high-level players follow a model inspired by policy iteration schemes in order to compute their strategies. Specifically, given the cognitive level- $k$  of an agent, their thinking process is initialized by assigning level-0 intelligence to the opponents and computing the optimal response to this environment. Subsequently, they increase the level of the opponents via a similar process until a predefined distribution of lower levels has been embedded on the rest of the players. This process involves the solution of a number of single-agent optimization problems that allow for the determination of the lower-level policies while avoiding the difficulties associated with game solutions.

The single-agent optimization that a level- $k$  player performs corresponds to solving the following Hamilton–Jacobi–Bellman (HJB) equation with respect to a nonequilibrium value function  $V^k(x) : \mathbb{R}^n \rightarrow \mathbb{R}$ , which is conditioned by the belief probabilities of the agent of level- $k$ , regarding the intelligence level of player  $h \in \mathcal{N} \setminus \{i\}$ , denoted as  $b^k(h)$ ,

$$\begin{aligned} Q(x) + u_i^{k\top} R_{ii} u_i^k + \sum_{j \in \mathcal{N}_{-i}} \sum_{h=0}^{k-1} (b^k(h) u_j)^{h\top} R_{ij} b^k(h) u_j^h \\ + (\nabla V_i^k)^{\top} \left( f(x) + g(x) u_i^k + \sum_{j \in \mathcal{N}_{-i}} \sum_{h=0}^{k-1} (g(x) b^k(h) u_j) \right) = 0, \end{aligned}$$

which yields the level- $k$  policy given by

$$u_i^k(x) = -R_{ii}^{-1} g_i^{\top} \nabla V_i^k, \quad \forall x.$$

A simplified version of this bounded rationality principle can be extracted for linear quadratic zero-sum games. For ease of exposition, we will focus on the iterative scheme that corresponds to the minimizing player of system (16.4). Initially, for cyber-physical security applications, a focal strategy for a minimizing (defending) player, is taken as one that assumes attacker-free environments. This leads to a level-0 policy derived by the one-sided optimal control problem,

$$V_u^0(x_0) = \min_u \int_0^\infty (x^T M x + u^T R u) d\tau, \quad \forall x. \quad (16.6)$$

The optimal decision for (16.6) given (16.4) with  $d_i = 0$  is

$$u^0(x) = -R^{-1} B^T \frac{\partial V_u^0(x)}{\partial x} = -R^{-1} B^T P_u^0 x, \quad \forall x,$$

where the value function is taken as quadratic in the states  $V_u^0(\cdot) = x^T P_u^0 x$  and the kernel  $P_u^0$  solves the Riccati equation,  $A^T P_u^0 + P_u^0 A + M - P_u^0 B R^{-1} B^T P_u^0 = 0$ .

The iterative process described above, leads to the formulation of high-level optimal control problems described by

$$V_u^k(x_0) = \min_u \int_0^\infty (x^T M x + u^T R u - \gamma^2 \|d^{k-1}\|^2) d\tau,$$

where the  $k - 1$ th attack  $d^{k-1}$  corresponds to an adversary of limited cognitive ability. Solving this problem through classical optimal control tools yields the level- $k$  Riccati equation,

$$\begin{aligned} & \left( A + \frac{1}{\gamma^2} K K^T P_d^{k-1} \right)^T P_u^k + P_u^k \left( A + \frac{1}{\gamma^2} K K^T P_d^{k-1} \right) \\ & + \left( M - \frac{1}{\gamma^2} P_d^{k-1} K K^T P_d^{k-1} \right) - P_u^k B R^{-1} B^T P_u^k = 0 \end{aligned} \quad (16.7)$$

with its associated level- $k$  defense strategy given by

$$u^k(x) = -R^{-1} B^T P_u^k x. \quad (16.8)$$

Similarly, the attacking agent can calculate their level- $k$  response through the solution of the Riccati equation,

$$\begin{aligned} & (A - B R^{-1} B^T P_u^k)^T P_d^k + P_d^k (A - B R^{-1} B^T P_u^k) \\ & + (M + P_u^k B R^{-1} B^T P_u^k) + \frac{1}{\gamma^2} P_d^k K K^T P_d^k = 0, \end{aligned} \quad (16.9)$$

and the optimal attack policy given by

$$d^k(x) = \frac{1}{\gamma^2} K^T P_d^k x, \quad \forall x. \quad (16.10)$$

With this iterative procedure, a defending agent is able to compute the strategies of the adversaries with finite cognitive abilities, for a given number of levels. For the linear quadratic zero-sum case, proper conditions that guarantee convergence to the Nash equilibrium of the game as the levels of cognition of the players increase have been derived and are summarized in the theorem that follows.

**Theorem 16.1** ([11]) *Consider the pairs of strategies at a specific cognitive level- $k$ , given by (16.8) and (16.7) for the defender, and (16.10) and (16.9) for the adversary. The policies converge to a Nash equilibrium for higher levels if the following conditions hold as the levels increase,*

$$\begin{aligned} (P_d^{k-1} + P_d^k - P_u^{k+1}) K K^T (P_d^k - P_d^{k-1}) &\succ 0, \\ (3P_u^{k-1} - P_u^k + P_d^k) B R^{-1} B^T (P_u^k - P_u^{k-1}) &\succ 0. \end{aligned}$$

## 16.4 Cognitive Hierarchy for Adversarial Target Tracking

This section addresses the problem of tracking an actively evading target by employing a team of coordinating unmanned aerial vehicles while also learning the level of intelligence for appropriate countermeasures. Initially, under infinite cognitive resources, we formulate a game between the evader and the pursuing team, with an evader being the maximizing player and the pursuing team being the minimizing one. We derive optimal pursuing and evading policies while taking into account the physical constraints imposed by Dubins vehicles. Subsequently, we relax the infinite rationality assumption via the use of level- $k$  thinking. Such rationality policies are computed by using a reinforcement learning-based architecture and are proven to converge to the Nash policies as the thinking levels increase. Finally, simulation results verify the efficacy of the approach.

### 16.4.1 Problem Formulation

Consider  $N$  camera-equipped UAVs tasked with estimating the state of a target vehicle moving evasively in the ground plane. The UAVs fly at a fixed airspeed and constant altitude and are subject to a minimum heading rate. The target vehicle moves in the ground plane and is subject to a maximum turning rate and maximum speed that is less than the UAVs' ground speed, which is the same as its airspeed in the ideal case of no wind. Each UAV takes measurements of the target's position using a gimbaled video camera, and we assume that the target can be detected at all times and kept in the center of the camera's field of view by onboard software. We shall

first discuss the dynamical models for each of the two types ( $N$  UAVs and 1 target) of vehicles and then proceed to derive the relative kinematics of each UAV with respect to the ground moving target.

### 16.4.1.1 Vehicle Dynamics

In our approach, we consider that the kinematics of each moving vehicle, namely, both the UAV and the target, can be described via Dubins. The Dubins vehicle is a planar vehicle that moves forward at a fixed speed and has a bounded turning rate.

Consider that each UAV  $i \in \mathcal{N} := \{1, 2, \dots, N\}$  flies at a constant speed  $s_i$ , at a fixed altitude, and has a bounded turning rate  $u_i \in \mathcal{U}$  in the sense that  $\mathcal{U} := \{u_i \in \mathbb{R} : |u_i| \leq \bar{u}_i\}$  with  $\bar{u}_i \in \mathbb{R}^+$ .

Denote the state of each vehicle by  $\xi^i := [\xi_1^i \ \xi_2^i \ \xi_3^i]^T \in \mathbb{R}^3$ ,  $i \in \mathcal{N}$ , which comprises the planar position of each UAV in  $p_i := [\xi_1^i \ \xi_2^i]^T$  and its heading  $\psi_i := \xi_3^i$ , all of which are measured in a local East-North-Up coordinate frame. Hence the kinematics of each UAV are given  $\forall i \in \mathcal{N}$  by

$$\dot{\xi}^i = f_v^i(\xi^i, u_i) := [s_i \cos \xi_3^i \ s_i \sin \xi_3^i \ u_i]^T, \ t \geq 0.$$

On the basis of the above, the target is also modeled as a Dubins vehicle with a bounded turning rate  $d$  in the sense that  $\mathcal{D} := \{d \in \mathbb{R} : |d| \leq \bar{d}\}$  where  $\bar{d}$  is the maximum turning rate. Denote the state of the target by  $\eta := [\eta_1 \ \eta_2 \ \eta_3]^T \in \mathbb{R}^3$ , where  $p_T := [\eta_1 \ \eta_2]^T$  is the planar position of the target in the same local East-North-Up coordinate frame as the UAVs and  $\eta_3$  is its heading.

To proceed, we shall make the following assumption.

**Assumption 16.1** The following are needed to make the problem feasible.

- At  $t = 0$ , the tracking vehicle is observing the target and is not dealing with the problem of initially locating the target.
- Since the UAVs fly at a constant altitude, there is no need to consider the 3-D distance, but only the projection of each UAV's position on the flat-Earth plane where the target is moving.
- The airspeed and the heading rate of the target satisfy,  $s_t < \min\{s_1, s_2, \dots, s_N\}$ , and  $\bar{d} < \min\{\bar{u}_1, \bar{u}_2, \dots, \bar{u}_N\}$ .  $\square$

### 16.4.1.2 Relative Kinematics

In a target tracking problem, it is necessary to determine the relative motion of the target with respect to the UAV. Thus, we will work in the polar coordinates, i.e.,  $(r_i, \theta_i)$ , where  $r_i$  is the relative distance of each UAV to the target  $r_i = \|p_i - p_T\|_2 := \sqrt{(\xi_1^i - \eta_1)^2 + (\xi_2^i - \eta_2)^2}$  and  $\theta_i$  the azimuth angle defined as

$\theta_i = \arctan \frac{\xi_2^i - \eta_2}{\xi_1^i - \eta_1} \forall i \in \mathcal{N}$ . Also, we define the “relative heading” angle [12], as  $\phi_i = \arctan \frac{r_i \dot{\theta}_i}{\dot{r}_i}$  and by taking  $\phi_i = \psi_i - \theta_i$  into account, we can derive the tracking dynamics for each UAV as follows

$$\begin{aligned}\dot{r}_i &= s_i \cos \phi_i - \dot{\eta}_1^i \cos \theta_i - \dot{\eta}_2^i \sin \theta_i, \quad \forall i \in \mathcal{N}, \\ \dot{\xi}_3^i &= u_i, \quad \forall i \in \mathcal{N}, \\ \dot{\eta}_3 &= d.\end{aligned}$$

We can now write the augmented state  $\mathbf{r} := [r_1 \ \xi_3^1 \dots r_N \ \xi_3^N \ \eta_3]^T \in \mathbb{R}^{2N+1}$  to yield the following dynamics,

$$\begin{aligned}\dot{\mathbf{r}} &= \begin{bmatrix} \dot{r}_1 \\ 0 \\ \dot{r}_2 \\ 0 \\ \vdots \\ \dot{r}_N \\ 0 \\ 0 \end{bmatrix} + \begin{bmatrix} 0 & \dots & \dots & 0 \\ 1 & 0 & \dots & \dots \\ 0 & \dots & \dots & \vdots \\ \vdots & & & \ddots \\ 1 & 0 & \dots & \vdots \\ \vdots & \ddots & \ddots & \vdots \\ \dots & \dots & 0 & \vdots \\ 0 & \dots & 0 & 1 \\ 0 & \dots & 0 & 0 \end{bmatrix} u + \begin{bmatrix} 0 \\ \vdots \\ \vdots \\ 1 \end{bmatrix} d \\ &\equiv F(\mathbf{r}) + Gu + Kd, \quad \mathbf{r}(0) = \mathbf{r}_0, \quad t \geq 0,\end{aligned}\tag{16.11}$$

where  $u := [u_1 \ u_2 \ \dots \ u_N]^T$  is the vector of the turning rates of the UAVs.

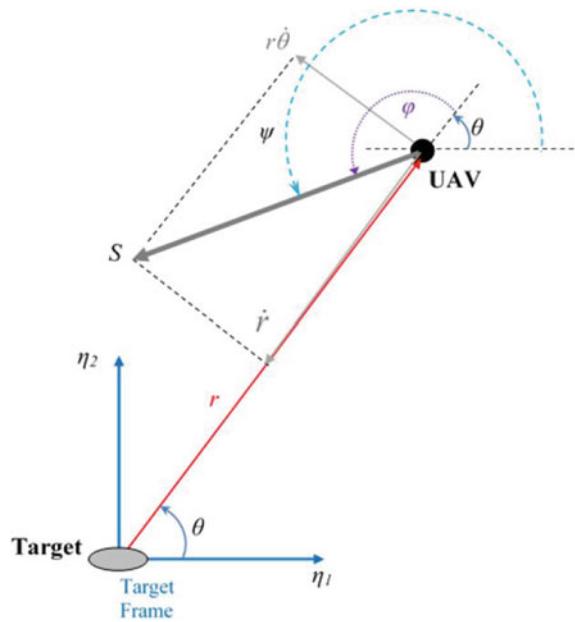
On the basis of the above analysis, Fig. 16.1 depicts the target tracking geometry, where we are interested in the position of the UAV with respect to the target frame. For more details regarding the “relative heading” angle  $\phi$  as well as the relative motion between the target and the UAV, the interested reader is directed to [12].

#### 16.4.1.3 Differential Game Formulation

The target tracking problem can be regarded as a two-player zero-sum game in which the team of UAVs tries to minimize the distance from the target  $r_i$ , and the target tries to maximize it. Note that the UAVs coordinate their movements in order to ensure that at least one UAV is close to the target. Additionally, the UAVs should keep their individual distances to the target sufficiently small to maintain the adequate resolution of the target in the camera’s image plane for effective visual detection. The above motivates us to choose the following cost functional,

$$J = \int_0^\infty (R_u(u) - R_d(d) + R_r(\mathbf{r})) dt,$$

**Fig. 16.1** Target tracking geometry. We can observe that the angle  $\phi$  expresses the orientation of the UAV with respect to the relative distance vector  $r$ , namely, it is the angle between  $s$  and  $r$ . Moreover, in polar coordinates, the velocity of the UAV can be split into two vertical components, namely, the contraction component  $\dot{r}$  and the circulation component  $r\dot{\theta}$



where  $R_r(r) := \beta_1 \frac{1}{\sum_{i=1}^N r_i^2} + \beta_2 \sum_{i=1}^N r_i^2$ , with  $\beta_1, \beta_2 \in \mathbb{R}^+$  weighting constants.

Specifically, the term being weighted by  $\beta_1$  enforces distance coordination so that one UAV is always close to the target to improve measurement quality and the term being weighted by  $\beta_2$  penalizes the individual UAV distances to the target to ensure that the size of the target in each UAV's image plane is sufficiently large for reliable detection by image processing software.

To enforce *bounded UAV inputs* and *bounded target input* we shall use a non-quadratic penalty function of the form,

$$R_u(u) = 2 \int_0^u (\theta_1^{-1}(v))^T R dv, \quad \forall u, \quad (16.12)$$

and

$$R_d(d) = 2 \int_0^d (\theta_2^{-1}(v))^T \Gamma dv, \quad \forall d, \quad (16.13)$$

where  $R > 0$ ,  $\Gamma \in \mathbb{R}^+$  and  $\theta_i(\cdot)$ ,  $i \in \{1, 2\}$  are continuous, one-to-one, real-analytic integrable functions of class  $C^\mu$ ,  $\mu \geq 1$ , used to map  $\mathbb{R}$  onto the intervals  $[-\bar{u}, \bar{u}]$  and  $[-\bar{d}, \bar{d}]$ , respectively, satisfying  $\theta_i(0) = 0$ ,  $i \in \{1, 2\}$ . Also note that  $R_u(u)$  and  $R_d(d)$  are positive definite because  $\theta_i^{-1}(\cdot)$ ,  $i \in \{1, 2\}$  are monotonic odd.

First, by assuming infinite rationality (the players in the game are familiar with the decision-making mechanism), we are interested in finding the following optimal

value function,  $\forall r, t \geq 0$ ,

$$V^*(r(t)) = \min_{u \in U} \max_{d \in \mathcal{D}} \int_t^\infty (R_u(u) - R_d(d) + R_r(r)) d\tau,$$

subject to (16.11).

### 16.4.2 Zero-Sum Game

According to the analysis in Sect. 16.2.1, the value function will satisfy the following Hamilton–Jacobi equation,

$$H\left(r, \frac{\partial V^*}{\partial r}, u^*, d^*\right) = 0, \quad (16.14)$$

with Nash policies are given by

$$u^*(r) = \arg \min_u H\left(r, \frac{\partial V^*}{\partial r}, u, d^*\right) = -\theta_1\left(\frac{1}{2}R^{-1}G^T \frac{\partial V^*}{\partial r}\right), \quad \forall r, \quad (16.15)$$

for the UAV, and,

$$d^*(r) = \arg \max_d H\left(r, \frac{\partial V^*}{\partial r}, u^*, d\right) = \theta_2\left(\frac{1}{2}\Gamma^{-1}K^T \frac{\partial V^*}{\partial r}\right), \quad \forall r \quad (16.16)$$

for the target.

The closed-loop dynamics can be found by substituting (16.15) and (16.16) in (16.11), to write

$$\dot{r} = F(r) + Gu^* + Kd^*, \quad r(0) = r_0, \quad t \geq 0. \quad (16.17)$$

For completeness purposes, we characterize the stability of the equilibrium point of the closed-loop system.

**Theorem 16.2** ([20]) Consider the closed-loop system given by (16.17). Assume that the equilibrium point is  $r_e = 0$ . Then,  $s_i = s_t, \forall i \in \mathcal{N}$ .

The next theorem provides a sufficient condition for the existence of a saddle-point based on (16.14).

**Theorem 16.3** ([20]) Suppose that there exists a continuously differentiable radially unbounded positive definite function  $V^* \in C^1$  such that for the optimal policies given by (16.15) and (16.16), the following is satisfied,

$$R_u(u^*(r)) - R_d(d^*(r)) + R_r(r) \geq 0, \quad \forall r,$$

with  $V^*(0) = 0$ . Then, the closed-loop system given by (16.17) has a globally asymptotically stable equilibrium point. Moreover, the policies (16.15)–(16.16) form a saddle point and the value of the game is  $J^*(\cdot; u^*, d^*) = V^*(r(0))$ .

### 16.4.3 Cognitive Hierarchy

In this section, we apply the proposed bounded rationality framework to the target tracking problem.

#### 16.4.3.1 Level-0 (Anchor) Policy

We need to introduce an anchor policy for the level-0 player. First, we will define the level-0 UAV strategy as the policy based on the assumption that the target is not maneuvering and moves in a horizontal line which arises by solving an optimal control problem described by

$$V_u^0(r_0) = \min_{u \in U} \int_0^\infty (R_u(u) + R_r(r)) d\tau. \quad (16.18)$$

The optimal control input for the optimization problem (16.18) given (16.4) with  $d = 0$  is

$$u^0(r) = -\theta_1 \left( \frac{1}{2} R^{-1} G^T \frac{\partial V_u^0}{\partial r} \right), \quad \forall r,$$

where the value function  $V_u^0(\cdot)$  satisfies the Hamilton–Jacobi–Bellman (HJB) equation, namely  $H(r, \frac{\partial V_u^0}{\partial r}, u^0) = 0$ .

Subsequently, the intuitive response of a level-1 adversary target is an optimal policy under the belief that the UAV assumes that the target is not able to perform evasive maneuvers. To this end, we define the optimization problem from the point of view of the target for the anchor input  $u = u^0(r)$ ,

$$V_d^1(r_0) = \max_{d \in \mathcal{D}} \int_0^\infty (R_u(u^0) - R_d(d) + R_r(r)) d\tau, \quad \forall r,$$

subject to  $\dot{r} = F(r) + Gu^0 + Kd$ ,  $r(0) = r_0$ ,  $t \geq 0$ .

The level-1 target's input is computed as  $d^1(r) = \theta_2 \left( \frac{1}{2} \Gamma^{-1} K^T \frac{\partial V_d^1}{\partial r} \right)$ , where the value function  $V_d^1(\cdot)$  satisfies the HJB equation, i.e.,  $H(r, \frac{\partial V_d^1}{\partial r}, u^0, d^1) = 0$ .

### 16.4.3.2 Level- $k$ Policies

To derive the policies for the agents of higher levels of rationality, we will follow an iterative procedure, wherein the UAV and the adversary target optimize their respective strategies under the belief that their opponent is using a lower level of thinking. The UAV performing an arbitrary number of  $k$  strategic thinking interactions solves the following minimization problem,

$$V_u^k(r_0) = \min_{u \in U} \int_0^\infty (R_u(u) - R_d(d^{k-1}) + R_r(r)) d\tau,$$

subject to the constraint,  $\dot{r} = F(r) + Gu + Kd^{k-1}$ ,  $r(0) = r_0$ ,  $t \geq 0$ .

The corresponding Hamiltonian is

$$\begin{aligned} H_u^k \left( r, \frac{\partial V_u^k}{\partial r}, u, d^{k-1} \right) &= R_u(u) - R_d(d^{k-1}) + R_r(r) \\ &+ \left( \frac{\partial V_u^k}{\partial r} \right)^T (F(r) + Gu + Kd^{k-1}), \quad \forall r, u. \end{aligned}$$

Substituting the target's input with the policy of the previous level  $d^{k-1} = \theta_2 \left( \frac{1}{2} \Gamma^{-1} K^T \frac{\partial V_d^{k-1}}{\partial r} \right)$  yields

$$u^k(r) = -\theta_1 \left( \frac{1}{2} R^{-1} G^T \frac{\partial V_u^k}{\partial r} \right), \quad \forall r, \quad (16.19)$$

where the level- $k$  UAV value function  $V_u^k(\cdot)$  satisfies the HJB equation, namely,

$$H_u^k \left( r, \frac{\partial V_u^k}{\partial r}, u^k, d^{k-1} \right) = 0, \quad \forall r. \quad (16.20)$$

Similarly, the target of an arbitrary  $k + 1$  level of thinking, maximizes her response to the input of a UAV of level- $k$ ,

$$V_d^{k+1}(r_0) = \max_{d \in \mathcal{D}} \int_0^\infty (R_u(u^k) - R_d(d) + R_r(r)) d\tau,$$

subject to  $\dot{r} = F(r) + Gu^k + Kd$ ,  $r(0) = r_0$ ,  $t \geq 0$ .

The corresponding Hamiltonian is

$$\begin{aligned} H_d^{k+1} \left( r, \frac{\partial V_d^{k+1}}{\partial r}, u^k, d \right) &= R_u(u^k) - R_d(d) + R_r(r) \\ &+ \frac{\partial V_d^{k+1}}{\partial r}^T (F(r) + Gu^k + Kd), \quad \forall r, d. \end{aligned} \quad (16.21)$$

Substituting (16.19) in (16.21) yields the following response,

$$d^{k+1}(r) = \theta_2 \left( \frac{1}{2} \Gamma^{-1} K^T \frac{\partial V_d^{k+1}}{\partial r} \right), \quad \forall r, \quad (16.22)$$

where the level- $k + 1$  target value function  $V_d^{k+1}(\cdot)$  satisfies the HJB equation, namely,

$$H_d^{k+1} \left( r, \frac{\partial V_d^{k+1}}{\partial r}, u^k, d^{k+1} \right) = 0, \quad \forall r. \quad (16.23)$$

With this iterative procedure, the UAV is able to compute the strategies of the target with finite cognitive abilities for a given number of levels.

**Theorem 16.4** ([20]) Consider the pairs of strategies at a specific cognitive level- $k$ , given by (16.19) and (16.20) for the UAV, and (16.22) and (16.23) for the level- $k + 1$  adversarial target. The policies converge to a Nash equilibrium for higher levels if the following conditions hold as the levels increase,

$$R_u(u^{k-1}) - R_u(u^{k+1}) > 0, \quad (16.24)$$

$$R_d(d^{k+2}) - R_d(d^k) > 0. \quad (16.25)$$

**Remark 16.1** It is worth noting that the inequalities (16.24), (16.25) have a meaningful interpretation. The input penalties (16.12), (16.13) are strictly increasing and decreasing functions, respectively, and as the level- $k$  of rationality tends to infinity, the players follow a policy such that the corresponding penalty functions become sufficiently small and large, respectively.  $\square$

Now, the following theorem provides a sufficient condition which establishes the global asymptotic stability of the equilibrium point  $r_e = 0$  of the closed-loop system at each level of rationality  $k$ .

**Theorem 16.5** ([20]) Consider the system (16.11) under the effect of agents with bounded rationality whose policies are defined by (16.19) for the UAV and (16.22) for the adversarial target. Assuming that the pursuer and evader have the same speed, the game can be terminated at any cognitive level- $k$  as long as the following relationship holds

$$R_u(u) - R_d(d) + R_r(r) \geq 0, \quad \forall r, u, d.$$

#### 16.4.4 Coordination with Nonequilibrium Game-Theoretic Learning

Due to the inherent difficulties of solving the HJI equation (16.14), we will employ an actor/critic structure. Toward this, initially, we will construct a critic approximator

to learn the optimal value function that solves (16.14). Specifically, let  $\Omega \subseteq \mathbb{R}^{2N+1}$  be a simply connected set, such that  $0 \in \Omega$ . It is known [24] that we can rewrite the optimal value function as

$$V^*(r) = W^T \phi(r) + \epsilon_c(r), \quad \forall r,$$

where  $\phi := [\phi_1 \ \phi_2 \ \dots \ \phi_h]^T : \mathbb{R}^{2N+1} \rightarrow \mathbb{R}^h$  are activation functions,  $W \in \mathbb{R}^h$  are unknown ideal weights, and  $\epsilon_c : \mathbb{R}^{2N+1} \rightarrow \mathbb{R}$  is the approximation error. Specific choices of activation functions can guarantee that  $\|\epsilon_c(r)\| \leq \bar{\epsilon}_c, \forall r \in \Omega$ , with  $\bar{\epsilon}_c \in \mathbb{R}^+$  being a positive constant [25–27].

Since the ideal weights  $W$  are unknown, we define an approximation of the value function as

$$\hat{V}(r) = \hat{W}_c^T \phi(r), \quad \forall r, \quad (16.26)$$

where  $\hat{W}_c \in \mathbb{R}^h$  are the estimated weights. Now, we can write the Hamiltonian utilizing the estimated value function (16.26) as

$$\begin{aligned} \hat{H} \left( r, \hat{W}_c^T \frac{\partial \phi}{\partial r}, u, d \right) &\equiv R_u(u) - R_d(d) + R_r(r) \\ &+ \hat{W}_c^T \frac{\partial \phi}{\partial r} (F(r) + Gu + Kd), \quad \forall u, d. \end{aligned}$$

The approximate Bellman error due to the bounded approximation error and the use of estimated weights is defined as

$$\begin{aligned} e_c &= R_u(u) - R_d(d) + R_r(r) \\ &+ \hat{W}_c^T \frac{\partial \phi}{\partial r} (F(r) + Gu + Kd). \end{aligned}$$

An update law for  $\hat{W}_c$  must be designed, such that the estimated values of the weights converge to the ideal ones. To this end, we define the squared residual error  $K_c = \frac{1}{2}e_c^2$ , which we want to minimize. Tuning the critic weights according to a modified Levenberg–Marquardt [28] gradient descent algorithm yields

$$\dot{\hat{W}}_c = -\alpha \frac{\omega(t)e_c(t)}{(\omega(t)^T \omega(t) + 1)^2}, \quad (16.27)$$

where  $\alpha \in \mathbb{R}^+$  is a constant gain that determines the speed of convergence and  $\omega = \nabla \phi(F(r(t)) + Gu(t) + Kd(t))$ .

We use similar ideas to learn the best response policy. For compactness, we denote  $a_j(\mathbf{r})$ ,  $j \in \{u, d\}$ , which will allow us to develop a common framework for the pursuers and the evaders. Similar to the value function, the feedback policy  $a_j(\mathbf{r})$  can be rewritten as

$$a_j^*(\mathbf{r}) = W_{a_j}^* \phi_{a_j}(\mathbf{r}) + \epsilon_{a_j}, \quad \forall \mathbf{r}, \quad j \in \{u, d\},$$

where  $W_{a_j}^* \in \mathbb{R}^{h_{a_j} \times N_{a_j}}$  is an ideal weight matrix with  $N_{a_u} := N$ , and  $N_{a_d} := 1$ ,  $\phi_{a_j}(\mathbf{r})$  are the activation functions defined similar to the critic approximator, and  $\epsilon_{a_j}$  is the actor approximation error. Similar assumptions with the critic approximator are needed to guarantee boundedness of the approximation error  $\epsilon_{a_j}$ .

Since the ideal weighs  $W_{a_j}^*$  are not known, we introduce  $\hat{W}_{a_j} \in \mathbb{R}^{h_{a_j} \times N}$  to approximate the optimal control in (16.15), and (16.16) as

$$\hat{a}_j(\mathbf{r}) = \hat{W}_{a_j}^T \phi_{a_j}(\mathbf{r}), \quad \forall \mathbf{r}, \quad j \in \{u, d\}. \quad (16.28)$$

Our goal is then to tune  $\hat{W}_{a_j}$  such that the following error is minimized,

$$K_{a_j} = \frac{1}{2} e_{a_j}^T e_{a_j}, \quad j \in \{u, d\},$$

where the reinforcement signal for the actor network is

$$e_{a_j} := \hat{W}_{a_j}^T \phi_{a_j} - \hat{a}_j^V, \quad j \in \{u, d\},$$

where  $\hat{a}_j^V$  is a version of the optimal policy in which  $V^*$  is approximated by the critic's estimate (16.26),

$$\hat{a}_j^V = \begin{cases} -\theta_1 \left( \frac{1}{2} R^{-1} G^T \nabla \phi^T \hat{W}_c \right), & j = u, \\ \theta_2 \left( \frac{1}{2} \Gamma^{-1} K^T \nabla \phi^T \hat{W}_c \right), & j = d. \end{cases}$$

We note that the error considered is the difference between the estimate (16.28) and versions of (16.15) and (16.16). The tuning for the UAV actor approximator is obtained by a modified Levenberg–Marquardt gradient descent rule,

$$\dot{\hat{W}}_{a_j} = -\alpha_{a_j} \phi_{a_j} e_{a_j}, \quad j \in \{u, d\}, \quad (16.29)$$

where  $\alpha_{a_j} \in \mathbb{R}^+$  is a constant gain that determines the speed of convergence. The issue of guaranteeing convergence of the learning algorithms on nonlinear systems has been investigated in the literature. For the proposed approach, rigorous proofs and sufficient conditions of convergence have been presented in [29].

We will now propose an algorithmic framework that allows the UAV to estimate the thinking level of an evader that changes her behavior unpredictably by sequentially interacting over time windows of length  $T_{\text{int}} \in \mathbb{R}^+$ . In essence, we will allow for arbitrary evading policies to be mapped to the level- $k$  policy database.

Let  $\mathcal{S} := \{1, 3, 5, \dots, \mathcal{K}\}$  be the index set including the computed estimated adversarial levels of rationality, and  $\mathcal{K}$  is the largest number of the set. Assuming that the UAV is able to directly measure the target's heading rate, we define the error between the actual measured turning rate, denoted as  $d(t)$ , and the estimated one of a level- $k$  adversarial target,

$$r^k(t) := \int_t^{t+T_{\text{int}}} (d - \hat{a}_d)^2 d\tau, \quad \forall t \geq 0, k \in \mathcal{S}. \quad (16.30)$$

However, the  $i$ -th sample shows the estimated target level of intelligence and the sampling period is  $T_{\text{int}}$ . The classification of the  $i$ -th sample is found according to the minimum distance classifier, namely,

$$x_i = \arg \min_k r^k, \quad \forall k \in \mathcal{S}, \quad \forall i \in \{1, \dots, \mathcal{L}\}, \quad (16.31)$$

where  $\mathcal{L}$  is the total number of samples. Note that the notions of “thinking steps” and “rationality levels” do not coincide as in [8]. Let  $k_i = \frac{x_i+1}{2}$  be the random variable counting the target thinking steps per game that follows the Poisson distribution [8] with the following probability mass function,  $p(k_i; \lambda) = \frac{\lambda^{k_i} e^{-\lambda}}{k_i!}$ , where  $\lambda \in \mathbb{R}^+$  is both the mean and variance.

Our goal is to estimate the parameter  $\lambda$  from the observed data by using the sample mean of the observations which forms an unbiased maximum likelihood estimator,

$$\hat{\lambda}(n_S) = \frac{\sum_{i=1}^{n_S} k_i}{n_S}, \quad \forall n_S \in \{1, \dots, \mathcal{L}\}. \quad (16.32)$$

However, in order to ensure the validity of our estimation we need to make the following assumption.

**Assumption 16.2** The target is at most at the  $\mathcal{K}$ -th level of thinking and does not change policy over the time interval  $((i-1)T_{\text{int}}, iT_{\text{int}})$ ,  $\forall i \in \{1, \dots, \mathcal{L}\}$ .  $\square$

On the basis of the above, we write the following learning algorithm that enables the UAV to estimate the intelligence level distribution of a target.

**Algorithm 16.1** Intelligence Level Learning

---

```

1: procedure
2:   Given initial state  $r_0$ , cost weights  $\Gamma$ ,  $R$ ,  $\beta_1$ ,  $\beta_2$  and highest allowable target
   level defined to be  $\mathcal{K}$ .
3:   for  $k = 0, \dots, \mathcal{K} - 1$  do
4:     Set  $j := u$  to learn the level- $k$  UAV policy.
5:     Start with  $\hat{W}_u^k(0)$ ,  $\hat{W}_{a_u}^k(0)$ .
6:     Propagate the augmented system with states  $\chi = [r^T \ (\hat{W}_u^k)^T \ (\hat{W}_{a_u}^k)^T]^T$ , accord-
      ing to (16.11), (16.27), and (16.29) until convergence.
7:     Set  $j := d$  to learn the level- $k + 1$  adversarial target policy.
8:     Start with  $\hat{W}_d^{k+1}(0)$ ,  $\hat{W}_{a_d}^{k+1}(0)$ .
9:     Propagate the augmented system with states  $\chi = [r^T \ (\hat{W}_d^{k+1})^T \ (\hat{W}_{a_d}^{k+1})^T]^T$ , according
      to (16.11), (16.27), and (16.29) until convergence. Go to 3.
10:    end for
11:    Define the interaction time with the target as  $T_{\text{int}}$ , the number of total samples  $\mathcal{L}$ .
12:    for  $i = 1, \dots, \mathcal{L}$  do
13:      for  $k = 1, \dots, \mathcal{K}$  do
14:        Given  $t \in [t_i, t_i + T_{\text{int}}]$ , measure the value of (16.30).
15:      end for
16:      Estimate the level of rationality of target according to (16.31).
17:      Update  $\lambda$  based on (16.32). Go to 12 to take the next sample.
18:    end for
19: end procedure

```

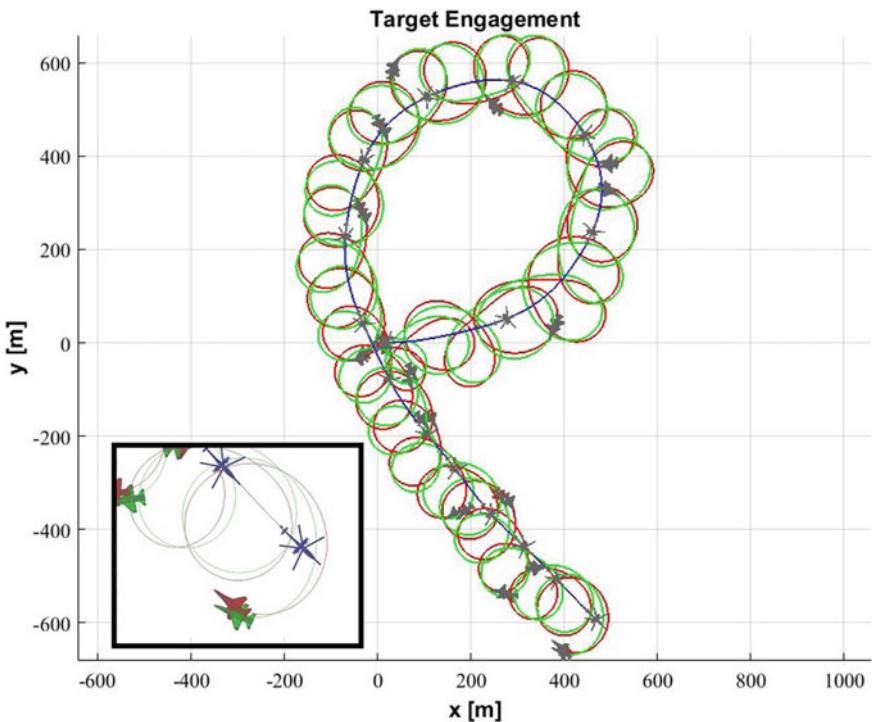
---

### 16.4.5 Simulation

Consider a team of two cooperative UAVs with the same capabilities, namely, the constant airspeed is  $s_i = 20$  m/s where  $i \in \mathcal{N} := \{1, 2\}$ , and the maximum turning rate is  $\bar{u} = 0.5$  rad/s. The speed of the target is  $s_t = 10$  m/s, and the maximum turning rate is  $\bar{d} = 0.2$  rad/s.

First, we examine the case of infinite rationality, and from Fig. 16.2, one can see that the UAVs are engaging the target. The relative distance trajectories of each UAV with respect to the target are shown in Fig. 16.3. Note that each UAV can attain a minimum relative distance of 1.5 m.

We will now examine the case where the UAVs and the target have bounded cognitive abilities. In particular, we consider the scenario where one UAV is assigned to pursue a target operating in a level of thinking included in the set  $\mathcal{K} := \{1, 3, 5, 7\}$ , i.e., performing at most 4 thinking steps. Figure 16.5 shows the UAV's beliefs over the levels of intelligence of the target. From the latter, we can observe that the pursuer believes that the target has a probabilistic belief state of 8% of being level-1, 15% of being level-3, 20% of being level-5 and 18% of being level-7. From Fig. 16.4, we observe the evolution of the Poisson parameter  $\lambda$  in terms of the number of samples. It is evident that it converges as long as enough data have been gathered by observing

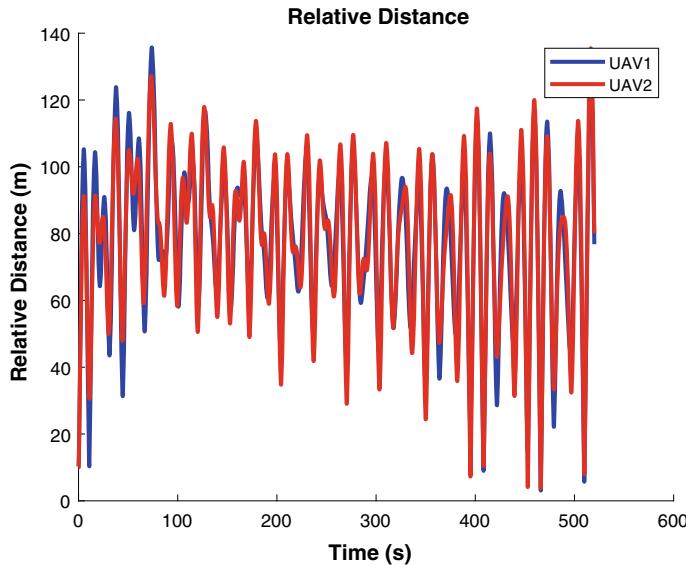


**Fig. 16.2** Trajectories of the pursuing (red and green) and evading (blue) vehicles on the 2D plane. The coordination taking place between the UAVs can be seen

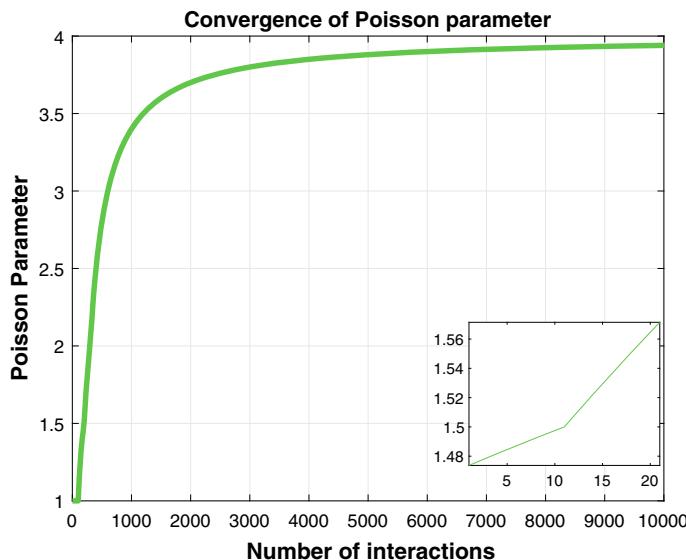
the motion of the target. Moreover, note that since it is a piece-wise smooth function, a spike appears when the target performs one thinking step and moves up to a higher level of intelligence. The latter observations let us build a profile of “intelligence” for the target and follow appropriate countermeasures. The visualization of the target engagement game was conducted via the “flighthpath3d” MATLAB package [30].

## 16.5 Conclusion and Future Work

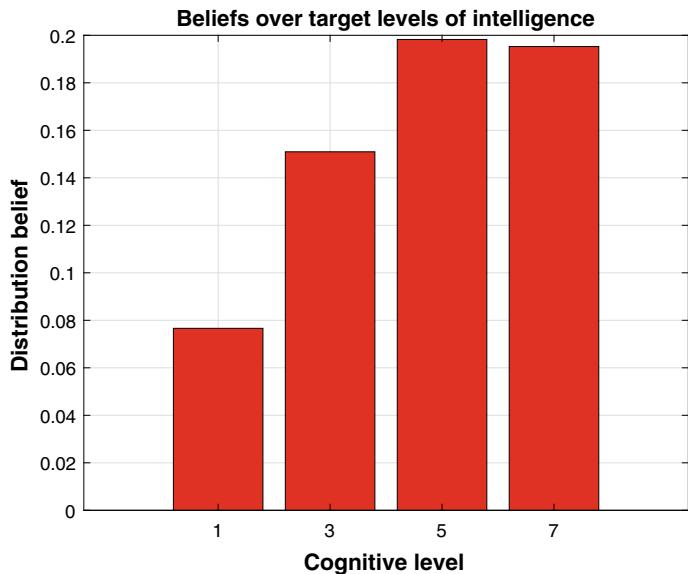
This chapter developed a coordinated target tracking framework through a nonequilibrium game-theoretic approach. We introduced a cognitive hierarchy formulation where agents both with bounded and unbounded rationality are considered, with the capabilities of learning intentions of evading UAVs. In the case of infinite rationality, we derived the saddle-point policies of the agents, bounded within the enforced input limits and with guaranteed global asymptotic stability of the equilibrium point. We then considered bounded rationality, and we showed the conditions for convergence to the Nash equilibrium as the levels of thinking increase. Moreover, we formulated



**Fig. 16.3** Relative distance of each pursuer with respect to the evader. We can see that the distances remain bounded as learning takes place



**Fig. 16.4** Evolution of the Poisson parameter  $\lambda$ . As the UAV observes the target's behavior, the distribution converges to the actual one



**Fig. 16.5** Distribution of the beliefs over different thinking levels after convergence of the estimation algorithm

a framework which enables the UAVs to estimate the level of intelligence of the target provided that enough information has been collected regarding its cognitive abilities. Finally, we showed the efficacy of the proposed approach with a simulation example.

Future work will extend the framework to probabilistic game protocols so that the coordinated team of UAVs to explicitly adapt to a boundedly rational evader.

**Acknowledgements** This work was supported in part, by ARO under grant No. W911NF-19-1-0270, by ONR Minerva under grant No. N00014-18-1-2160, and by NSF under grant Nos. CPS-1851588, and SaTC-1801611.

## References

1. Başar, T., Olsder, G.J.: *Dynamic Noncooperative Game Theory*. SIAM, Philadelphia (1998)
2. Isaacs, R.: *Differential Games: a Mathematical Theory with Applications to Warfare and Pursuit, Control and Optimization*. Courier Corporation, Chelmsford (1999)
3. Alpcan, T., Başar, T.: *Network Security: a Decision and Game-Theoretic Approach*. Cambridge University Press, Cambridge (2010)
4. Fudenberg, D., Levine, D.K.: *The Theory of Learning in Games*, vol. 2. MIT Press, Cambridge (1998)
5. Roth, A.E., Erev, I.: Learning in extensive-form games: experimental data and simple dynamic models in the intermediate term. *Games Econ. Behav.* **8**(1), 164–212 (1995)

6. Erev, I., Roth, A.E.: Predicting how people play games: reinforcement learning in experimental games with unique, mixed strategy equilibria. *Am. Econ. Rev.* **848–881** (1998)
7. McKelvey, R.D., Palfrey, T.R.: Quantal response equilibria for normal form games. *Games Econ. Behav.* **10**(1), 6–38 (1995)
8. Camerer, C.F., Ho, T.-H., Chong, J.-K.: A cognitive hierarchy model of games. *Q. J. Econ.* **119**(3), 861–898 (2004)
9. Abuzainab, N., Saad, W., Poor, H.V.: Cognitive hierarchy theory for heterogeneous uplink multiple access in the internet of things. In: 2016 IEEE International Symposium on Information Theory (ISIT), pp. 1252–1256. IEEE (2016)
10. Li, N., Oyler, D., Zhang, M., Yildiz, Y., Girard, A., Kolmanovsky, I.: Hierarchical reasoning game theory based approach for evaluation and testing of autonomous vehicle control systems. In: 2016 IEEE 55th Conference on Decision and Control (CDC), pp. 727–733. IEEE (2016)
11. Kanellopoulos, A., Vamvoudakis, K.G.: Non-equilibrium dynamic games and cyber-physical security: a cognitive hierarchy approach. *Syst. Control Lett.* **125**, 59–66 (2019)
12. Kokolakis, N.-M.T., Koussoulas, N.T.: Coordinated standoff tracking of a ground moving target and the phase separation problem. In: 2018 International Conference on Unmanned Aircraft Systems (ICUAS), pp. 473–482. IEEE (2018)
13. Kim, S., Oh, H., Tsourdos, A.: Nonlinear model predictive coordinated standoff tracking of a moving ground vehicle. *J. Guid. Control Dyn.* **36**(2), 557–566 (2013)
14. Oh, H., Kim, S., Shin, H.-S., Tsourdos, A.: Coordinated standoff tracking of moving target groups using multiple UAVs. *IEEE Trans. Aerosp. Electron. Syst.* **51**(2), 1501–1514 (2015)
15. Frew, E.W., Lawrence, D.A., Morris, S.: Coordinated standoff tracking of moving targets using Lyapunov guidance vector fields. *J. Guid. Control Dyn.* **31**(2), 290–306 (2008)
16. Summers, T.H., Akella, M.R., Mears, M.J.: Coordinated standoff tracking of moving targets: control laws and information architectures. *J. Guid. Control Dyn.* **32**(1), 56–69 (2009)
17. Quintero, S.A., Papi, F., Klein, D.J., Chisci, L., Hespanha, J.P.: Optimal UAV coordination for target tracking using dynamic programming. In: 49th IEEE Conference on Decision and Control (CDC), pp. 4541–4546. IEEE (2010)
18. Quintero, S.A., Ludkovski, M., Hespanha, J.P.: Stochastic optimal coordination of small UAVs for target tracking using regression-based dynamic programming. *J. Intell. Robot. Syst.* **82**(1), 135–162 (2016)
19. Quintero, S.A., Hespanha, J.P.: Vision-based target tracking with a small UAV: optimization-based control strategies. *Control Eng. Pract.* **32**, 28–42 (2014)
20. Kokolakis, N.-M.T., Kanellopoulos, A., Vamvoudakis, K.G.: Bounded rational unmanned aerial vehicle coordination for adversarial target tracking. In: 2020 American Control Conference (ACC), pp. 2508–2513. IEEE (2020)
21. Lewis, F.L., Vrabie, D., Syrmos, V.L.: *Optimal Control*. Wiley, New York (2012)
22. Daskalakis, C., Goldberg, P.W., Papadimitriou, C.H.: The complexity of computing a Nash equilibrium. *SIAM J. Comput.* **39**(1), 195–259 (2009)
23. Strzalecki, T.: Depth of reasoning and higher order beliefs. *J. Econ. Behav. Organ.* **108**, 108–122 (2014)
24. Haykin, S.S.: *Neural Networks and Learning Machines*, vol. 3. Pearson, Upper Saddle River (2009)
25. Hornik, K., Stinchcombe, M., White, H.: Universal approximation of an unknown mapping and its derivatives using multilayer feedforward networks. *Neural Netw.* **3**(5), 551–560 (1990)
26. Ioannou, P., Fidan, B.: *Adaptive Control Tutorial*, vol. 11. SIAM, Philadelphia (2006)
27. Lewis, F., Jagannathan, S., Yesildirak, A.: *Neural Network Control of Robot Manipulators and Non-linear Systems*. CRC Press, Boca Raton (1998)
28. Ioannou, P.A., Sun, J.: *Robust Adaptive Control*. Courier Corporation, Chelmsford (2012)
29. Vamvoudakis, K.G., Miranda, M.F., Hespanha, J.P.: Asymptotically stable adaptive-optimal control algorithm with saturating actuators and relaxed persistence of excitation. *IEEE Trans. Neural Netw. Learn. Syst.* **27**(11), 2386–2398 (2015)
30. Bużantowicz, W.: Matlab script for 3D visualization of missile and air target trajectories. *Int. J. Comput. Inf. Technol.* **5**, 419–422 (2016)

# Chapter 17

## Bounded Rationality in Learning, Perception, Decision-Making, and Stochastic Games



Panagiotis Tsiotras

**Abstract** Current algorithms that try to emulate “human-like” decision-making and perception capabilities for autonomous systems are computationally very demanding. We propose to utilize bounded rationality ideas for generating suitable hierarchical abstractions to handle demanding tasks under time and other resource constraints, when exact optimality/rationality may be elusive. These abstractions can be utilized in order to focus the available computational resources on the relevant part of the problem in a top-down, task-specific manner. The bounded rational paradigm for decision-making can also be extended to the case of stochastic games formulated in terms of competitive Markov decision processes in order to account for the presence of competing/adversarial agents.

### 17.1 The Autonomy Challenge

The science of autonomy (broadly interpreted here as a mix of control, artificial intelligence, machine learning, and robotics) has shown great promise in recent years, mainly owing to the success of statistical learning methods in achieving impressive results in a variety of pattern recognition and classification tasks. Nonetheless, most current algorithms for autonomous systems fail to address critical aspects of the problem that are ever present in many real-life applications: (a) paucity of training data; (b) limited computational resources; (c) presence of adversarial agents. There is, therefore, a great need to better understand the way autonomous systems make decisions under the above constraints, especially if they are used to close perception-action (PA) feedback loops for fast “human-like” decision-making.

---

P. Tsiotras (✉)

School Aerospace Engineering, Institute for Robotics and Intelligent Machines,  
Georgia Institute of Technology, Atlanta, GA 30332, USA  
e-mail: [tsiotras@gatech.edu](mailto:tsiotras@gatech.edu)

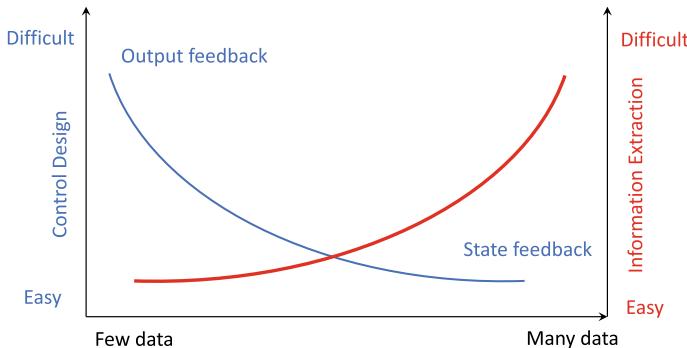
Future control and decision-making systems will have to strike a balance between *guaranteed performance* in the presence of uncertainty (a topic that is central to control theory), useful and *relevant information extraction* from incomplete or scarce data (a topic of estimation and information theory), dealing with *adversarial agents* (a topic of game theory), while, at the same time, ensure reduced *algorithmic complexity* so as to enable real-time (re)planning and learning (which is better addressed by computer science). This confluence of control, information theory, and computer/computational science and engineering is necessary for developing scalable computational methods for joint perception and planning in order to enable an autonomous agent extract task-relevant information from a multitude of data sources in a timely manner.

### 17.1.1 *The Case of Actionable Data*

Autonomous systems supported by recent advances in artificial intelligence (AI) and machine learning (ML)—as well as the proliferation of new powerful computer hardware architectures—are poised to transform the current scientific paradigm for problem-solving by replacing many traditional model-based methodologies with data-intensive alternatives. Having access to a huge amount of data to sieve through, these computationally intensive algorithms are capable of finding patterns that may elude human observers. Despite their great success, however, these algorithms still suffer from unpredictability and often exhibit “child-like” behavior. They may confuse correlation for causality and may find “patterns” when none exist, becoming confused and perplexed, unable to come up with the correct answer even in very simple scenarios where the answer would have been clear to a human observer [52, 64]. This tendency of “theoretical statistical accuracy” but “demonstrated fragility in practice” makes these algorithms currently unsuitable for many safety-critical systems.

The main observation that may explain the current predicament with these data-driven algorithms is the realization that collected data is not necessarily information or knowledge. Extracting *actionable* information from data—and similarly, deciding how to best represent this information—is the crux of the problem. So the key issue is not the absence of enough data, but rather the quality of the data and how to make sense out of it, especially if an adversary is capable of actively manipulating the data.

Not surprisingly, the answer to this fundamental question, namely, extracting the relevant information from the collected measurements, has been a long-standing open problem in the control, estimation, and information theory communities. A large part of control theory is devoted to constructing observers and estimators that recover the state from scarce measurements (e.g., output feedback), essentially reconstructing the whole history of the process, regardless of whether this history is needed for the control task. Today, we find ourselves in a different landscape. Sensor technology has progressed so rapidly that we have ended up at the other extreme: there is too much available data from a myriad of sources at various resolutions. And while the answer



**Fig. 17.1** The difficulty of high-confidence, robust control design and decision-making remains the same as it was many years ago, but for quite the opposite reasons. While before much of the effort was focused on finding the state of the system from few measurements (e.g., output feedback), today most of the difficulty lies in trying to extract meaningful information from a deluge of data

of finding the “relevant information” in the measurements was considered difficult many years ago because of lack of sufficient data (as in the output feedback case), it still remains difficult today, but for exactly the opposite reason: the availability of too much data (“data deluge”), see Fig. 17.1.

### 17.1.2 The Curse of Optimality

Current models of decision-making follow the von Neumann–Morgenstern maximum expected utility (MEU) dictum [51], which assumes the existence of a utility function, the maximization of which is solely responsible for the ensuing actions by the agent. The main issue with optimal decision-making is the fact that finding *optimal* actions is very difficult, since it typically requires to search over *all* available alternatives, a problem that requires a prohibitive amount of computational resources, which have not been taken into account *during problem formulation*.

An elegant theory (namely, dynamic programming [9, 11]) exists to solve optimal decision-making, but its application in practice is stymied by the computational explosion resulting from the increased dimensionality of the search space. Most often than not, agents have limited attention, time and other resources (or they are just lazy) to come up with what is best for them. Furthermore, in many contexts what is “optimal” is not even known, so attempting to carefully optimize the (wrong) objective would be counter-productive. For instance, in planning motion through (or exploring) an unknown environment, one can try to maximize the expected reward, the computation of which requires a prior of the (yet to be seen) scene. Thus, the current plan is as good as the prior, which is unfalsifiable, and the expected reward is just that, i.e., “expected.” In other words, we claim that the often stated “curse of dimensionality” bottleneck in decision-making is somewhat misleading, as it is not

necessarily the dimensionality of the problem that defies trackable answers, but rather the fact that one tends to seek *the* optimal solution among a myriad of alternatives. By seeking approximate or sub-optimal solutions, the problem dimensionality becomes secondary and the attention is turned to remedying the *curse of optimality*, instead.

## 17.2 How to Move Forward

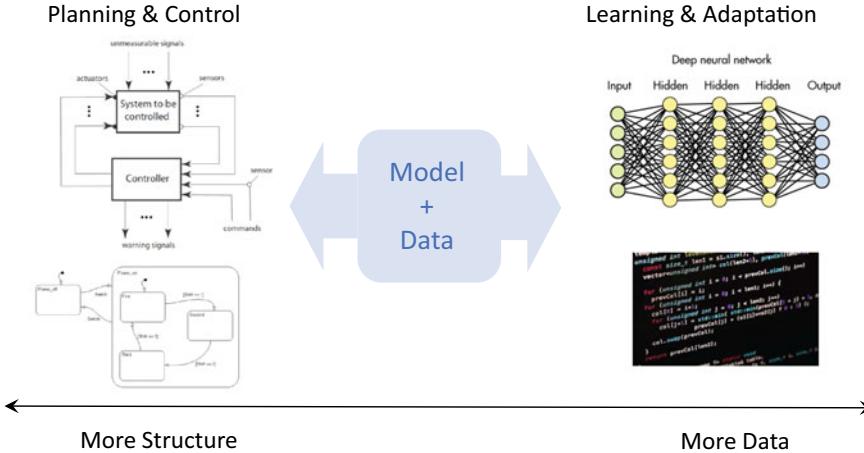
We argue that the current data-intensive paradigm of perception and decision-making in robotics (and autonomous systems in general) that tends to operate on all collected data is neither relevant nor necessary for problems involving high-performance robots operating in real time and at high speed, in uncertain and complex environments, demonstrating “human-like” intelligence and agility. This point of view stems from the following key observations.

First, current, purely “computational” models of perception—especially those based solely on vision cues—essentially serve as surrogates of full-state feedback estimators in order to achieve specific control objectives, thus enforcing an artificial separation of perception and control. This separation—deeply rooted in the community by its wide applicability for the stabilization of linear systems (“separation principle”)—is questionable for most real-life problems involving autonomous agents operating in complex and unstructured environments, where information gathering (perception/sensing) is tightly coupled with motion (planning/control).

Second, human cognitive models provide strong evidence that the manner by which human (especially expert) agents perceive and (re)act to environmental stimuli, and process information, is quite different than the way most current perception techniques deal with this problem. For instance, because of experience, and prior familiarity of how the world “works”, a human has already a preconceived notion of “what lies ahead” and has a good compact representation of the environment. She certainly does not “reconstruct” the whole environment at every instant of time, and her approach to closing the perception/action loop has a significant anticipatory (non-causal) component [12].

Despite the great progress witnessed recently by data-driven approaches to learning (fueled manly by convolutional/deep neural networks (CNNs)), these approaches are of limited applicability in capturing such priors or non-causal characteristics that are crucial for safety-critical control tasks. Traditional machine learning approaches need a huge amount of (typically labeled) data which are often not available in practice. In addition, these learning algorithms are based on “similarity” measures of historically collected data; as a result, they are primarily suitable for interpolation but not for extrapolation. This reliance solely on data (while by-passing the guidelines of physical laws) often leads to inefficient and “brittle” learning [52].

Successful operation of autonomous agents in complex (and even adversarial) environments will require a novel approach to sensing, perception, and communication (and their interaction) that merge data-based with model-based methods in



**Fig. 17.2** The development of computationally efficient and resilient communication/learning/action loops to enable autonomous decision-making and control depends on the ability to devise suitable methodologies to merge data with models, depending on the amount of prior knowledge

order to enable continuous and reliable operation with limited computation resources and/or bandwidth (Fig. 17.2).

Drawing on the cognitive hierarchies of the human brain, we seek to develop algorithms that are aware of the (limited) available computational resources of the agent/decision-maker. The proposed algorithms are able to utilize prior information (e.g., from physics-based models) to operate on *actionable data* in a *timely* manner. The limitations of the available resources give rise to hierarchical abstractions on the action space which can be utilized to solve the problem at the “right” level of granularity, and thus avoid wasting valuable resources by reasoning and acting at levels of fidelity that are beyond what is required for the problem at hand.

The approach borrows from bounded rationality theory, according to which human beings tend to weight the benefits of reaching the “best” or “most informed” decision, versus the pitfalls associated with the time, energy, and effort needed to collect all the information to reach that decision.

### 17.2.1 *Bounded Rationality for Human-Like Decision-Making*

Many surveys and laboratory experiments indicate that people often fail to conform to some of the basic tenets of rational (i.e., MEU) decision theory [54]. This is especially true when the environment is uncertain and/or complex, the task is challenging and is dominated by nonlinear phenomena, when the time is constrained, etc. As a result, decision-theory researchers, especially in economics [4, 49, 57, 59], have proposed an alternative paradigm for rational behavior, one that is commensurate with the

available knowledge and resources. According to this bounded rationality paradigm, human beings tend to weight the benefits of reaching the “best” or “most informed” decision, versus the pitfalls associated with the time, energy, and effort needed to collect all the information to reach that decision. Bounded rationality thus aims at formalizing the limited (*e.g., bounded*) rationality decisions and actions an intelligent agent is expected to take in those cases [30]. A bounded rational agent takes actions that meet the minimum requirements necessary to achieve a particular goal (i.e., those that meet the agent’s *aspiration level* [70]).

Another reason one may want to follow a bounded rational paradigm for decision-making stems from the desire to design autonomous robotic agents that can be trusted by humans. This implies that the robotic agent generates actions that are predictable/legible (and not necessarily optimal!) by the human. This point of view predicates that the intelligent agent needs to model the bounded-rational decision-making of the human (in essence inferring what constitutes predictable behavior for the human). Bounded rationality theory offers a convenient framework to formulate human–machine interaction.

Instead of focusing on maximizing expected utility (as in standard MDP formulations) the bounded rational approach suggests *maximizing expected free utility (or free energy)*, a quantity that encapsulates in a precise manner the effect of information and computational constraints of the decision-making process. This framework also intrinsically generates, *for free*, a *hierarchy of abstractions* for decision-making brought about by the severity of information-processing costs, with each abstraction tailored to the “right” level of granularity.

### 17.2.2 Hierarchical Abstractions for Scalability

A hallmark of human intelligence is arguably the ability to abstract, organize acquired knowledge, and process collected information in a hierarchical manner [21, 31, 45, 65]. The ability to organize collected data in hierarchies is so ingrained in the human brain that we take it for granted. Such hierarchies make it possible for us humans to make quick decisions even in complex situations we have never encountered before. Hierarchies can be utilized to focus the available resources in a top-down, task-specific manner [27, 44, 49]. Granted, these decisions may not be optimal but are made fast, respond to the most urgent or consequential stimuli, and are “good-enough” for the current situation. By moving up and down the hierarchy allows one to depart from pure optimality formulations and investigate (sub)optimal bounded rational solutions.

Hierarchies are also a powerful source of heuristics [73] that can be used to shortcut exact computation of complex quantities. Most importantly, hierarchies are enablers of *attention* mechanisms. One often life-critical aspect of operation under strong time pressure is to be able to rapidly and efficiently focus computational resources onto the most relevant or important information in the environment, in the context of the current situation. In primates, *attention acts as a fast heuristic* to guide the

organism toward the most important sensory stimuli (e.g., obstacles in one's path, landmarks, or other guiding signs, other objects/individuals). Unexpected events (or surprises) are dealt with quickly by focusing attention to the data and events at the right level of (temporal and spatial) granularity that is most relevant to the task(s) at hand [14, 23, 42]. In either case, attention provides a mechanism to shape or select incoming information under strict timing constraints. It is, therefore, clear that, at least in humans, attention does play a key role in identifying the “actionable information” in order to reduce computational complexity.

The idea of using hierarchical abstractions has been previously used by the author to develop efficient solutions to path-planning problems subject to computational constraints [18, 19, 68]. In those references hierarchies of multi-resolution abstractions were created to speed-up re-planning. Here, we wish to generate abstractions in a more principled manner, as the result of a variational principle that captures the (limited) available computational resources of the agent. While in those references these abstractions were imposed externally (and perhaps rather axiomatically) by the designer, here they arise intrinsically and naturally as part of the problem formulation and the agent's capabilities.

### 17.3 Sequential Decision-Making Subject to Resource Constraints

Over the past several years considerable effort has been devoted to developing a framework that accounts for the available computational constraints during decision-making [27, 44, 49, 55, 56]. The idea is to account for the cost of computation not after the fact, as is traditionally the case, but from the very beginning, as part of the problem formulation. This stream of research has led to the development of *information-constrained MDPs* [27, 32, 44] that utilize concepts from information theory in order to model the computational limitations of the agent [53, 56]. The approach has also strong connections with the thermodynamic point of view of the theory of computation [10, 24, 55]. The key idea is to look at the information flow  $U \rightarrow X \rightarrow Y$  from the input to the output (random) variables as a suitable communication channel [67], and formulate the problem in the space of state (or output) distributions over actions.

The introduction of the added constraints (computational and otherwise) to the decision process is meant to model an agent that does not have access to unlimited resources, and hence will select actions that are not optimal in the traditional sense of maximizing value or utility, but rather in the sense of maximizing (expected) value or utility *for a given processing cost* [49, 56, 67]. Such a decision-maker, who alters his behavior based on its resources, is said to be a *bounded rational* agent [15, 40, 49, 59].

In this work, we apply this framework to develop decision-making algorithms for planning problems while also accounting for the information processing costs

of computing the optimal actions for stochastic environments modeled in terms of Markov decision processes (MDPs). This resource-limited MDP problem is formally posed as a constrained optimization problem, where the objective is to maximize the value function subject to a constraint that penalizes the difference between the resulting policy and a known (computationally cheap) prior policy. As it will be shown later on, this optimization problem involves the mutual information between states and control actions and leads to a direct analogy of the amount of information an action carries about the state. A perfectly rational decision-maker will solve the optimization problem producing a policy that is *state specific* since it has the resources to choose specific actions for each state. On the other hand, a bounded rational, resource-constrained agent does not have this luxury, and hence prefers policies that are relatively generic and *state agnostic*. This classification of more informative versus less informative states leads in a natural way to a series of state space *abstractions*. Such abstractions can be utilized in order to focus the available computational resources in a top-down, task-specific manner. The details of the approach are discussed in greater detail next. Before proceeding with the details, however, it is worth reviewing some of the basic theory of MDPs. The next section also help us establish the notation used in the rest of this work.

### 17.3.1 Standard Markov Decision Processes

We consider sequential decision-making problems in the presence of uncertainty. Markov decision processes (MDPs) are the standard framework for optimal sequential decision-making under uncertainty [63]. We will use  $x_t$  to denote the system state and  $u_t$  the control input at time  $t$ , where  $t = 0, 1, 2, \dots$ . The system state is an element of the state set  $\mathcal{X}$  (i.e.,  $x_t \in \mathcal{X}$ ) and the control input is selected from the set of admissible inputs  $\mathcal{U}$  which, in general, depends on  $x_t$  so that  $u_t \in \mathcal{U} = \mathcal{U}(x_t)$ . The system dynamics is modeled via a collection of transition probabilities  $P(x_{t+1}|x_t, u_t)$ , which represent the probability that the agent, being at state  $x_t$  and selecting input  $u_t$  at time step  $t$ , will find itself at state  $x_{t+1}$  at the next instance of time. Each action results in a reward  $R(x_{t+1}, x_t, u_t)$  that the agent receives for selecting input  $u_t$  while at state  $x_t$  and at time  $t$  and transitioning to state  $x_{t+1}$  at time  $t + 1$ .

The objective of the infinite-time MDP is to find a policy which maximizes the future discounted expected reward. By policy we mean a map that, for each state  $x_t$ , provides a probability distribution over actions,  $\pi(u_t|x_t)$ . Since we will restrict the discussion to infinite horizon problems, we assume that our policy is stationary, that is,  $\pi$  is independent of  $t$  [11].

More formally, we seek to maximize the objective function

$$V^\pi(x_0) \triangleq \lim_{T \rightarrow \infty} \mathbb{E}_\pi \left[ \sum_{t=0}^{T-1} \gamma^t R(x_{t+1}, x_t, u_t) \right], \quad (17.1)$$

where  $\gamma \in [0, 1]$  is the *discount factor*,  $x_0$  is the initial state, and  $u_t$  is selected according to the provided policy  $\pi$  at each time step. Equation (17.1) can be interpreted as the cost-to-go provided the system starts at state  $x_0$  and executes the stationary policy  $\pi$ . The expectation in Eq.(17.1) is over all future trajectories  $\tau = (x_0, u_0, x_1, u_1, \dots, x_T)$  of length  $T$  starting at  $x_0$ . Under the Markov property assumption, the probability of trajectory  $\tau$  is given by

$$\Pr(u_0, x_1, u_1, x_2, \dots, x_T | x_0) = \prod_{t=0}^T \pi(u_t | x_t) P(x_{t+1} | x_t, u_t). \quad (17.2)$$

The objective is then to find the optimal distribution over admissible inputs at each state, in order to maximize the objective function in (17.1). Introducing the shorthand notation  $x = x_t$ ,  $u = u_t$ , and  $x' = x_{t+1}$ , expanding (17.1) and making use of (17.2), we obtain

$$V^\pi(x) = \sum_{u \in \mathcal{U}} \pi(u|x) \sum_{x' \in \mathcal{X}} P(x'|x, u) \left[ R(x', x, u) + \gamma V^\pi(x') \right]. \quad (17.3)$$

The optimal value function is

$$V^*(x) = \max_{\pi} V^\pi(x), \quad (17.4)$$

which can be computed by solving the *Bellman equation*

$$V^*(x) = \max_{u \in \mathcal{U}} \sum_{x' \in \mathcal{X}} P(x'|x, u) \left[ R(x', x, u) + \gamma V^*(x') \right], \quad (17.5)$$

with the optimal policy given by  $\pi^* = \text{argmax}_{\pi} V^\pi(x)$ .

We may also define the state-action value function (or Q-function) for any policy  $\pi$  as [63]

$$Q^\pi(x, u) \triangleq \sum_{x' \in \mathcal{X}} P(x'|x, u) \left[ R(x', x, u) + \gamma V^\pi(x') \right]. \quad (17.6)$$

Similarly, the optimal state-action value function is defined as

$$Q^*(x, u) \triangleq \sum_{x' \in \mathcal{X}} P(x'|x, u) \left[ R(x', x, u) + \gamma V^*(x') \right]. \quad (17.7)$$

In the case of a perfectly rational decision-maker, the agent will deterministically select the optimal action that maximizes Eq. (17.7). That is, the agent will act greedily with respect to  $Q^*(x, u)$  by selecting  $u^*(x) = \arg \max_u Q^*(x, u)$ . In this work, we distinguish between a general stochastic policy and one that is deterministic, by denoting optimal deterministic policies as  $\Gamma^*(u|x)$  and all other optimal policies as  $\pi^*(u|x)$ . Then, a perfectly rational agent will choose the policy  $\Gamma^*(u|x) = \delta(u - u^*(x))$ , where  $u^*(x) = \arg \max_u Q^*(x, u)$ .

### 17.3.2 Information-Limited Markov Decision Processes

The problem formulation presented above does not consider the “effort” required to find the optimal policy  $\Gamma^*(u|x)$ , which maximizes (17.1) [56, 67]. In order to model the effort required to obtain the (global) maximum, we provide the agent with a prior choice distribution  $\rho(u|x)$  and then limit the amount by which the posterior policy  $\pi(u|x)$  is permitted to differ from  $\rho(u|x)$  [27, 56, 67]. The idea of course being that changing the prior policy to the posterior policy requires effort and resources, and the more one wants to shape the prior to the posterior, the more resources one needs to consume. One way to capture a resource-constrained agent is, therefore, to penalize the acceptable difference between the prior and posterior distributions.

The difference between the prior and posterior distributions is measured using the Kullback–Leibler (KL) divergence [17]

$$D_{\text{KL}}(\pi(u|x)\|\rho(u|x)) \triangleq \sum_{u \in \mathcal{U}} \pi(u|x) \log \frac{\pi(u|x)}{\rho(u|x)}. \quad (17.8)$$

The total discounted information cost for a decision-maker starting from any initial state  $x$  and following the policy  $\pi(u|x)$  is then defined as [56]

$$D^\pi(x) \triangleq \lim_{T \rightarrow \infty} \mathbb{E}_\tau \left[ \sum_{t=0}^{T-1} \gamma^t \log \frac{\pi(u_t|x_t)}{\rho(u_t|x_t)} \right], \quad (17.9)$$

where the expectation in Eq. (17.9) is with respect to the resulting trajectory, given by Eq. (17.2). The goal is now to not only maximize the value alone as in (17.4), but rather maximize the trade-off between the value and the information cost in (17.9) [15, 56, 67]. This is a constrained optimization problem, which can be solved using Lagrange multipliers. The corresponding cost is the free energy for a given policy  $\pi(u|x)$ , defined as

$$F^\pi(x; \beta) \triangleq V^\pi(x) - \frac{1}{\beta} D^\pi(x), \quad (17.10)$$

where  $\beta > 0$ . The objective is now to find the policy  $\pi^*(u|x)$  which maximizes Eq. (17.10) instead of (17.4). Substituting known quantities and expanding terms, we obtain the mapping  $\mathcal{B} : \Re^{|X|} \rightarrow \Re^{|X|}$  defined as

$$\begin{aligned} \mathcal{B}[F](x; \beta) &\triangleq \max_{\pi(u|x)} \sum_{u \in \mathcal{U}} \pi(u|x) \left[ \sum_{x' \in \mathcal{X}} P(x'|x, u) R(x', x, u) - \frac{1}{\beta} \log \frac{\pi(u|x)}{\rho(u|x)} \right. \\ &\quad \left. + \gamma \sum_{x' \in \mathcal{X}} P(x'|x, u) F(x'; \beta) \right]. \end{aligned} \quad (17.11)$$

The proof of how this mapping results from Eq. (17.10) can be found in [56]. The optimal value of the free energy  $F^*(x; \beta)$  is the fixed point of the equation  $F^* =$

$\mathcal{B}[F^*]$ . Similarly to (17.6), we define

$$Q_F(x, u; \beta) \triangleq \sum_{x' \in \mathcal{X}} P(x'|x, u) \left[ R(x', x, u) + \gamma F(x'; \beta) \right] \quad (17.12)$$

and  $Z(x; \beta) \triangleq \sum_{u \in \mathcal{U}} \rho(u|x) e^{\beta Q_F(x, u; \beta)}$ . Equation (17.11) can be shown to be equivalent to

$$\mathcal{B}[F](x; \beta) = \frac{1}{\beta} \log Z(x; \beta). \quad (17.13)$$

This is iteratively applied until convergence to obtain the optimal free energy  $F^*(x; \beta)$ . For a given value of  $\beta$ , the optimal policy for the infinite horizon problem is then

$$\pi^*(u|x) = \frac{\rho(u|x) e^{\beta Q_F^*(x, u; \beta)}}{Z^*(x; \beta)}, \quad (17.14)$$

where  $Q_F^*(x, u; \beta)$  and  $Z^*(x; \beta)$  are given by

$$\begin{aligned} Q_F^*(x, u; \beta) &= \sum_{x' \in \mathcal{X}} P(x'|x, u) \left[ R(x', x, u) + \gamma F^*(x'; \beta) \right], \\ Z^*(x; \beta) &= \sum_{u \in \mathcal{U}} \rho(u|x) e^{\beta Q_F^*(x, u; \beta)}. \end{aligned} \quad (17.15)$$

Using this framework, we are able to model a wide variety of agents with limited resources by varying the resource parameter  $\beta$ . Hence  $\beta$  can be viewed as a parameter which reflects the agent's computational abilities [32]. When  $\beta \rightarrow 0$  the information term in Eqs. (17.10) and (17.11) becomes dominant and the agent becomes primarily concerned with reducing the information cost  $D^\pi(x)$ . This leads to  $\pi^* = \rho$  and the prior distribution is directly "copied" to the posterior policy, thus representing an agent who is not able to deviate from its initial control policy—indicative of a decision-maker with no computational resources [32]. When, on the other hand,  $\beta \rightarrow \infty$  the information cost in Eqs. (17.10) and (17.11) is effectively removed. Consequently, the agent focuses on finding a policy that maximizes  $V^\pi(x)$ . This means that the agent will behave more like its non-information limited counterpart. In this case, we therefore expect to recover the non-information limited policy in the limit (i.e.,  $\pi^* \rightarrow \Gamma^*$ ), which is indicative of an agent that has unlimited computational resources [32].

In the previous analysis, it was assumed that the prior distribution over actions  $\rho(u|x)$  was given. Next, we address the question of whether we can find an optimal a priori action distribution  $\rho(u|x)$  (or  $\rho(u)$ ) so as to minimize the information cost across all states *on average*. To this end, note that the information-limited MDP does not modify  $\rho(u|x)$  when searching for the optimal policy  $\pi^*$ . Instead, it will find the policy that best trades the value and information for a given  $\rho(u|x)$ . The problem now becomes one of finding both the posterior policy and prior action distribution so as to maximize Eq. (17.10) on average across all system states.

Averaging Eq.(17.10) over system states, and assuming for simplicity a state-agnostic prior distribution over actions ( $\rho(u|x) = \rho(u)$ ), we obtain

$$\max_{\rho} \sum_{x \in \mathcal{X}} p(x) \left( \max_{\pi} \sum_{u \in \mathcal{U}} \pi(u|x) \left[ Q_F(x, u; \beta) - \frac{1}{\beta} \log \frac{\pi(u|x)}{\rho(u)} \right] \right). \quad (17.16)$$

Optimizing Eq.(17.16) with respect to  $\rho(u)$  is a convex optimization problem [17, 27], and is numerically solved by recursively iterating between the following relations until convergence [27, 66, 67]

$$\pi^*(u|x) = \frac{\rho^*(u) e^{\beta Q_F^*(x, u; \beta)}}{\sum_u \rho^*(u) e^{\beta Q_F^*(x, u; \beta)}}, \quad (17.17)$$

$$\rho^*(u) = \sum_x p(x) \pi^*(u|x). \quad (17.18)$$

Here  $p(x)$  is the (fixed) probability distribution over states and  $\pi^*$  is found in the same manner described in the previous section. The proof of this result can be found in [27, 67]. It is worth mentioning that the iteration (17.17)–(17.18) is analogous to the Blahut–Arimoto (BA) algorithm in rate-distortion theory where the KL divergence plays the role of the “distortion function” [27, 56, 66, 67].

An agent utilizing the BA algorithm for path planning will find a policy that deviates in various amounts from the provided prior action distribution depending on the value of  $\beta$ .

The optimization problem (17.16) has an interesting interpretation in terms of the mutual information between states and actions. Recall that the mutual information between two random variables  $X$  and  $U$  is given by

$$I(X; U) \triangleq \sum_{x,u} p(x, u) \log \frac{p(x, u)}{p(x)p(u)}. \quad (17.19)$$

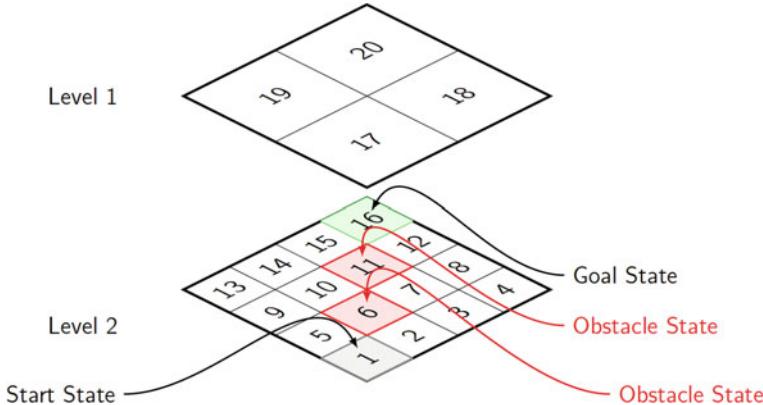
Rewrite now (17.16) as

$$\max_{\rho, \pi} \sum_{x,u} p(x) \pi(u|x) Q_F(x, u; \beta) - \frac{1}{\beta} \sum_x p(x) D_{\text{KL}}(\pi(u|x) \| \rho(u)) \quad (17.20)$$

or

$$\mathbb{E}_{\pi(u|x)} p(x) Q_F(x, u; \beta) - \frac{1}{\beta} I(X; U), \quad (17.21)$$

where we have used the fact that



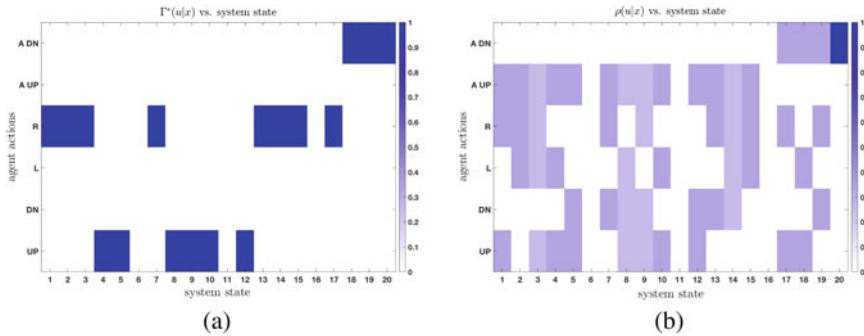
**Fig. 17.3** Path-planning example with two levels of abstraction. Level 2 is the actual (“fine resolution”) level, whereas Level 1 is a coarser resolution level and has lower cardinality, thus planning at this level is much easier. © 2017 IEEE. Reprinted, with permission, from [44]

$$\begin{aligned}
 \sum_x p(x) D_{\text{KL}}(\pi(u|x) \| \rho(u)) &= \sum_x p(x) \sum_u \pi(u|x) \log \frac{p(u|x)}{\rho(u)} \\
 &= \sum_{x,u} p(x)\pi(u|x) \log \frac{\pi(u|x)p(x)}{\rho(u)p(x)} \\
 &= \sum_{x,u} p(x,u) \log \frac{p(x,u)}{p(u)p(x)} = I(X, U).
 \end{aligned} \tag{17.22}$$

From (17.21), it can be immediately seen that for low values of  $\beta$ , the agent becomes mainly concerned with minimizing the mutual information between states and actions. This results in a policy that is generic with respect to the state, leading to state abstractions as shown in [27]. That is, the agent will focus its efforts to find a single action that yields a high reward, on average, across the state space.

We demonstrate the previous algorithm by applying it to a path-planning problem in the  $4 \times 4$  grid world shown in Fig. 17.3.

The agent has at its disposal the action set  $\mathcal{U}(x_t) \subseteq \{\text{UP}, \text{DN}, \text{L}, \text{R}, \text{A-UP}, \text{A-DN}\}$ . The first four actions (UP,DN,L,R) correspond to the standard actions “up-down-left-right” in a grid world with 4-cell connectivity. In addition to these actions, however, the agent has at its disposal the actions A-UP and A-DN. These do not correspond to physical movements of the agent, but rather encode changes in the “perceptual representation” of the environment that are motivated by the sensing and/or computational resources of the agent. By executing A-UP, the agent reasons in the higher (coarser) abstraction level, while by executing A-DN the agent moves one level down (less coarse, more accurate) abstraction level (see Fig. 17.3). For better accuracy, the agent would prefer, of course, to reason at the bottom (“finer”) resolution level but this may be expensive, and as a result the agent may be forced to reason at a coarse



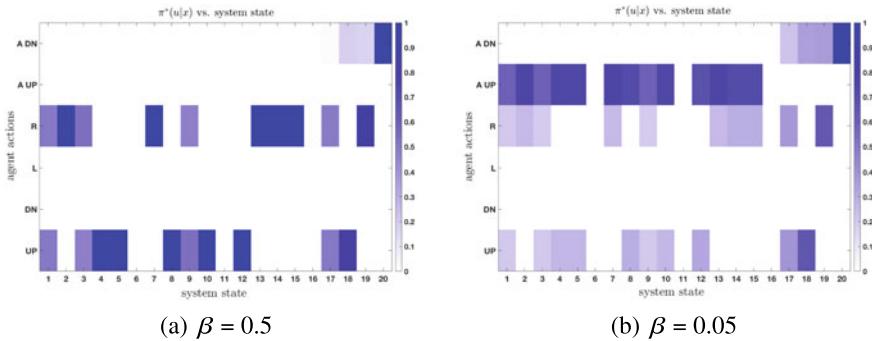
**Fig. 17.4** Deterministic policy,  $\Gamma^*(u|x)$  (left) and a priori action distribution,  $\rho(u|x)$  (right). © 2017 IEEE. Reprinted, with permission, from [44]

resolution level. These special actions can also be viewed, for example, as the agent's ability to toggle a sensor suite consisting of two variable resolution sensors on and off. The agent must pay an increased price for navigating the grid world at the finest resolution level, due to the demand to process an increased amount of information. The actions A-UP and A-DN can thus also be viewed as actions that are taken when the agent has limited computational resources, in which case it will favor higher level (coarser) abstractions since these correspond to search spaces of lower cardinality and hence are presumably easier to navigate in.

Figures 17.4 and 17.5 display the policies obtained for various values of  $\beta$  for the Blahut–Arimoto information-limited (BA-IL) algorithm. Specifically, the figures show the distribution over actions for a given state along with the value of  $\beta$  and the distribution over states,  $p(x)$ . In each of these figures, the y-axis corresponds to the action space whereas the x-axis displays the possible system states, with state numbering consistent with Fig. 17.3. Furthermore, the shading of a given state-action pair represents the probability that the action will be selected when the system finds itself in that state. Column  $n$  of these plots can, therefore, be considered a probability distribution over actions for the  $n^{\text{th}}$  state, where  $n \in \{1, 2, \dots, 20\}$ . Further details regarding the problem formulation and the reward structure for this paper can be found in [44].

It is important to note that the policy depicted in Fig. 17.4a by  $\Gamma^*$  represents a rational agent. That is, the policy  $\Gamma^*$  would be followed by an agent who has sufficient processing capabilities to find the optimal actions and does so at the finest grid level, electing to not abstract to a coarser resolution. As the agent's computational abilities are reduced, it begins to favor planning at the coarser grid level while its resulting policy becomes increasingly sub-optimal with respect to  $\Gamma^*$ .

Shown in Fig. 17.4b is the prior action distribution,  $\rho(u|x)$ . The prior distribution,  $\rho$ , can be viewed as the policy the agent would follow in case it had no resources allowing it to find another, possibly better, policy. Thus, if the agent has insufficient resources to deliberate (limit as  $\beta \rightarrow 0$ ),  $\rho$  becomes the posterior policy  $\pi$ , as seen when comparing Figs. 17.4b and 17.5b.



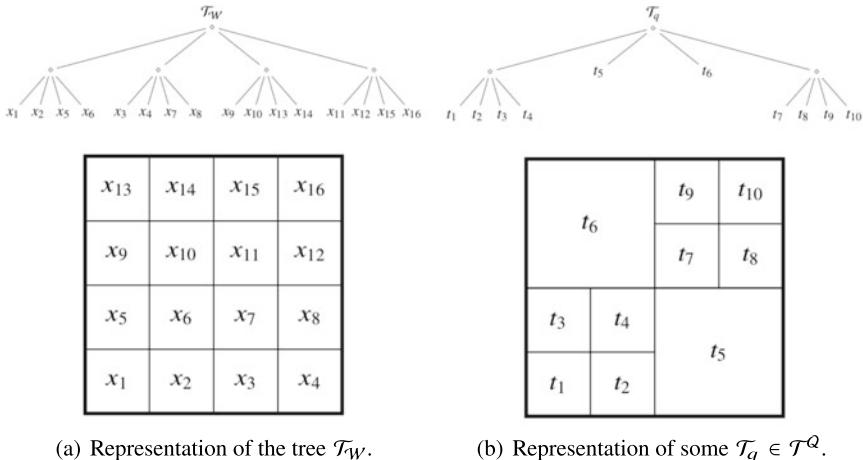
**Fig. 17.5**  $\pi^*(u|x)$  from the BA-IL algorithm for  $\beta = 0.5$  and  $\beta = 0.05$ . © 2017 IEEE. Reprinted, with permission, from [44]

This is also observed in Fig. 17.5 as the resulting policies ( $\pi^*$ ) approach  $\Gamma^*$  as  $\beta$  is increased. We also note that the policy becomes less deterministic as  $\beta \rightarrow 0$ . Furthermore, as  $\beta$  is reduced, abstractions emerge from the solution of the BA-IL algorithm. This can be seen in Fig. 17.5b where we see that the agent elects to select the same action for many states. That is, the resulting optimal policy  $\pi^*$  from the BA-IL algorithm is relatively *generic* with respect to the state and thus the state and action spaces have low mutual information. Finally, it should be noted from Fig. 17.5 that, as  $\beta$  is reduced, the agent prefers to plan paths at the coarser grid level, indicative of the fact that less computationally complex solutions are sought at the cost of perfect optimality.

## 17.4 An Information-Theoretic Approach for Hierarchical Decision-Making

In the previous section, we developed an information-theoretic approach to account for the computational resources in sequential decision-making problems modeled by MDPs using a hierarchy of abstractions of the state space. However, it was assumed that the hierarchy of abstractions was given *a priori*. In this section, we develop a framework to obtain abstractions for decision-making where the abstractions emerge as a function of the agent's limited computational resources.

We consider the emergence of graph abstractions in the form of multi-resolution quadtree representations. Quadtrees are a common tool utilized in the robotics community to reduce the complexity of environments in order to speed-up path planning and ease internal storage requirements [8, 22, 35, 36, 50]. To this end, we assume that the environment  $\mathcal{W} \subset \mathbb{R}^2$  (generalizable to  $\mathbb{R}^d$ ) is given by a two-dimensional grid world where each grid element is a unit square (hypercube). We assume that there exists an integer  $\ell > 0$  such that  $\mathcal{W}$  is contained within a square (hypercube) of side length  $2^\ell$ . A tree representation  $\mathcal{T} = (\mathcal{N}, \mathcal{E})$  of  $\mathcal{W}$  consists of a set of nodes  $\mathcal{N} = \mathcal{N}(\mathcal{T})$  and edges  $\mathcal{E} = \mathcal{E}(\mathcal{T})$  describing the interconnections between the nodes



**Fig. 17.6** Two different quadtree representations and the corresponding grid for a  $4 \times 4$  environment

in the tree [36]. We denote the set of all possible quadtree representations of maximum depth  $\ell$  of  $\mathcal{W}$  by  $\mathcal{T}^Q$  and we let  $\mathcal{T}_W \in \mathcal{T}^Q$  denote the finest quadtree representation of  $\mathcal{W}$ ; an example is shown in Fig. 17.6.

Varying the abstraction granularity of  $\mathcal{W}$  can be equivalently viewed as selecting various trees  $\mathcal{T}_q$  in the space  $\mathcal{T}^Q$ . Our problem is then one of selecting a tree  $\mathcal{T}_q \in \mathcal{T}^Q$  as a function of the agent's computational capabilities.

### 17.4.1 Agglomerative Information Bottleneck for Quadtree Compression

Our objective is to find compressed representations in a given hierarchy of quadtrees such that the compressed quadtree maintains as much as possible of the relevant information from the original quadtree. The task of obtaining optimal compressed representations of signals is addressed within the realm of information theory [17, 34, 60–62, 66]. Let  $(\Omega, \mathcal{F}, \mathbb{P})$  be a probability space with finite sample space  $\Omega$ ,  $\sigma$ -algebra  $\mathcal{F}$ , and probability measure  $\mathbb{P} : \mathcal{F} \rightarrow [0, 1]$ . Denote the set of real and positive real numbers as  $\mathfrak{N}$  and  $\mathfrak{N}_+$ , respectively. Let  $X : \Omega \rightarrow \mathfrak{N}$  denote the random variable corresponding to the original, uncompressed, signal, where  $X$  takes values in the set  $\Omega_X = \{x \in \mathfrak{N} : X(\omega) = x, \omega \in \Omega\}$  and, for any  $x \in \mathfrak{N}$ , let  $p(x) = \mathbb{P}(\{\omega \in \Omega : X(\omega) = x\})$ . Furthermore, let the random variable  $T : \Omega \rightarrow \mathfrak{N}$  denote the compressed representation of  $X$ , where  $T$  takes values in the set  $\Omega_T = \{t \in \mathfrak{N} : T(\omega) = t, \omega \in \Omega\}$ . The level of compression between the random variables  $X$  and  $T$  is measured by the mutual information [17, 66]

$$i(T; X) \triangleq \sum_{t,x} p(t, x) \log \frac{p(t, x)}{p(t)p(x)}. \quad (17.23)$$

The goal is then to find a stochastic mapping (encoder), denoted by  $p(t|x)$ , which maps outcomes in the uncompressed space  $x \in \Omega_X$ , to outcomes in the compressed representation  $t \in \Omega_T$  so as to minimize  $I(T; X)$ , while, at the same time, keep as much information encoded in another random variable  $Y$ . The variable  $Y$  represents information we are interested in preserving when forming the compressed representation  $T$  [61, 66].

In this work, we will use the agglomerative information bottleneck (AIB) method [61] to form compressed representations of  $X$ , which is useful when deterministic clusters that retain predictive information regarding the relevant variable  $Y$  are desired. The AIB method is based on the information bottleneck (IB) approach [66] to solve for deterministic, or hard, encoders (i.e.,  $p(t|x) \in \{0, 1\}$  for all  $t, x$ ). Specifically, we wish to use the AIB idea to find compressed versions of  $\mathcal{T}_W$ . Concepts from AIB will prove useful in our formulation, since each tree  $\mathcal{T}_q \in \mathcal{T}^Q$  encodes a hard (deterministic) abstraction of  $\mathcal{W}$ , where each leaf node of  $\mathcal{T}_W$  is aggregated to a specific leaf node of  $\mathcal{T}_q$ . That is, by viewing the uncompressed space ( $\Omega_X$ ) as the leaf nodes of  $\mathcal{N}(\mathcal{T}_W)$  and the abstracted (compressed) space ( $\Omega_T$ ) as the leaf nodes of  $\mathcal{N}(\mathcal{T}_q)$ , the abstraction operation can be specified in terms of an encoder  $p(t|x)$  where  $p(t|x) \in \{0, 1\}$  for all  $t$  and  $x$ , where  $p(t|x) = 1$  if a leaf node  $x$  of  $\mathcal{N}(\mathcal{T}_W)$  is aggregated to a leaf node  $t$  of  $\mathcal{N}(\mathcal{T}_q)$ , and zero otherwise (see Fig. 17.6). To better understand these connections, below, we briefly review the AIB before presenting how it is applied to our problem.

The AIB considers the optimization problem

$$\max_{p(t|x)} \mathcal{L}_Y(p(t|x); \beta), \quad (17.24)$$

where the Lagrangian is defined as

$$\mathcal{L}_Y(p(t|x); \beta) \triangleq i(T; Y) - \frac{1}{\beta} i(T; X), \quad (17.25)$$

and the maximization is performed over deterministic distributions  $p(t|x)$  for given  $\beta > 0$  and  $p(x, y)$  [60, 61]. The AIB works from bottom-up, starting with  $T = X$  and with each consecutive iteration reduces the cardinality of  $T$  until  $|\Omega_T| = 1$ . Specifically, let  $T_m$  represent the abstracted space with  $m$  elements ( $|\Omega_{T_m}| = m$ ) and let  $T_i$  represent the compressed space with  $|\Omega_{T_i}| = i < m$  elements, where  $i = m - 1$  and the number of merged elements is  $n = 2$ . We then merge elements  $\{t'_1, \dots, t'_n\} \subseteq \Omega_{T_m}$  to a single element  $t \in \Omega_{T_i}$  to obtain  $T_i$ . The merger cost is given by  $\Delta\mathcal{L}_Y : 2^{\Omega_{T_m}} \times \mathfrak{N}_+ \rightarrow \mathfrak{N}$ , which can be written as [60]

$$\Delta\mathcal{L}_Y(\{t'_1, \dots, t'_n\}; \beta) = p(t) \left[ JS_\Pi(p(y|t'_1), \dots, p(y|t'_n)) - \frac{1}{\beta} H(\Pi) \right], \quad (17.26)$$

where  $H(\Pi)$  is the Shannon entropy, where  $\Pi$  is given by

$$\Pi = [\Pi_1, \dots, \Pi_n]^T \triangleq \left[ \frac{p(t'_1)}{p(t)}, \dots, \frac{p(t'_n)}{p(t)} \right]^T, \quad (17.27)$$

and  $\text{JS}_{\Pi}(p_1, \dots, p_n)$  is the Jensen–Shannon (JS) divergence between the distributions  $p_1, \dots, p_n$ , with weights  $\Pi$  defined as [47]

$$\text{JS}_{\Pi}(p_1, \dots, p_n) \triangleq \sum_{s=1}^n \Pi_s D_{\text{KL}}(p_s, \bar{p}), \quad (17.28)$$

where for each  $y \in \Omega_Y$ ,  $\bar{p}(y) = \sum_{s=1}^n \Pi_s p_s(y)$ . Furthermore, we have that

$$p(t) = \sum_{s=1}^n p(t'_s), \quad p(y|t) = \sum_{s=1}^n \Pi_s p(y|t'_s), \quad (17.29)$$

which can be found by realizing that  $p(t|x) \in \{0, 1\}$  for all  $x \in \Omega_X$  and  $t \in \Omega_T$  along with the Markov chain condition  $T \leftrightarrow X \leftrightarrow Y$  [60, 61].

### 17.4.2 Optimal Compression of Quadtrees

In this section, we show how to use the AIB ideas to provide a hierarchy of compressed representations of  $\mathcal{T}_W$  that depends on the parameter  $\beta$ . Similarly to (17.25), we can define the Lagrangian in the space of quadtrees as the mapping  $L_Y : \mathcal{T}^Q \times \mathfrak{R}_+ \rightarrow \mathfrak{R}$ , given by

$$L_Y(\mathcal{T}_q; \beta) \triangleq \mathcal{L}_Y(p^q(t|x); \beta), \quad (17.30)$$

where  $\mathcal{L}_Y(p(t|x); \beta)$  is defined in (17.25). Then, for a given  $\beta > 0$ , we can search the space of quadtrees for the one that maximizes (17.30). This optimization problem is formally given by

$$\mathcal{T}_{q^*} = \underset{\mathcal{T}_q \in \mathcal{T}^Q}{\operatorname{argmax}} L_Y(\mathcal{T}_q; \beta). \quad (17.31)$$

By posing the optimization problem as in (17.31), we have implicitly incorporated the constraints on the mapping  $p(t|x)$  in order for the resulting representation to be a quadtree depiction of the world  $\mathcal{W}$ . The resulting world representation is encoded by the mapping  $p^{q^*}(t|x)$ . That is, the leafs of  $\mathcal{T}_{q^*}$  determine the optimal multi-resolution representation of  $\mathcal{W}$  for the given  $\beta$ .

While the optimization problem given by (17.31) allows one to form an analogous problem to that in (17.24) over the space of trees, the drawback of this method is the need to exhaustively enumerate all feasible quadtrees which can represent the world space. In other words, (17.31) requires that  $p^q(t|x)$  be provided for each  $\mathcal{T}_q \in \mathcal{T}^Q$ .

Because of this, the problem becomes intractable for large grid sizes and thus requires reformulation to handle larger world maps. We propose a sequential characterization of (17.30), and we formulate an optimization problem requiring the generation of candidate solutions, which are all obtained by successive expansions starting from the root tree  $\mathcal{T}_{q^0}$  to the current tree  $\mathcal{T}_{q^m}$ . The optimization problem that maximizes retained information while compressing  $\mathcal{T}_{q^m}$  can then be reformulated as [43]

$$\max_m \max_{\mathcal{T}_{q^1}, \dots, \mathcal{T}_{q^m}} L_Y(\mathcal{T}_{q^0}; \beta) + \sum_{i=0}^{m-1} \Delta L_Y(\mathcal{T}_{q^i}, \mathcal{T}_{q^{i+1}}; \beta), \quad (17.32)$$

where

$$\Delta L_Y(\mathcal{T}_{q^i}, \mathcal{T}_{q^{i+1}}; \beta) \triangleq L_Y(\mathcal{T}_{q^{i+1}}; \beta) - L_Y(\mathcal{T}_{q^i}; \beta). \quad (17.33)$$

In this formulation, the constraint encoding that the resulting representation is a quadtree is handled implicitly by  $\mathcal{T}_{q^i} \in \mathcal{T}^Q$ . The additional maximization over  $m$  in (17.32) appears since the horizon of the problem is not known a-priori and is, instead, a free parameter in the optimization problem.

Next, we propose an efficient algorithm to solve the optimization problem in (17.32).

### 17.4.3 The Q-Tree Search Algorithm

One way to solve (17.32) is to maximize  $\Delta L_Y(\mathcal{T}_{q^i}, \mathcal{T}_{q^{i+1}}; \beta)$  myopically at each step. That is, provided that  $\mathcal{T}_{q^{i+1}} \in \mathcal{T}^Q$  is a neighbor<sup>1</sup> of  $\mathcal{T}_{q^i} \in \mathcal{T}^Q$ , we consider the next tree  $\mathcal{T}_{q^{i+1}}$  that maximizes the value of  $\Delta L_Y(\mathcal{T}_{q^i}, \mathcal{T}_{q^{i+1}}; \beta)$ , and we sequentially keep selecting trees ( $\mathcal{T}_{q^{i+1}} \rightarrow \mathcal{T}_{q^{i+2}} \rightarrow \dots$ ) until no further improvement is possible. This greedy algorithm is simple to implement and requires little pre-processing but it does not, in general, find globally optimal solutions. This is owing to the short-sightedness of the algorithm and its inability to realize that poor expansions at the current step may lead to much higher valued options in the future.

In order to improve performance, we introduce the Q-tree search algorithm that takes into account future tree expansions. This is analogous to problems in reinforcement learning and dynamic programming, where an action-value function ( $Q$ -function) is introduced to incorporate the notion of cost-to-go for selecting among feasible actions in a given state [11, 63]. The idea behind introducing such a function is to incorporate future costs, thus allowing agents to take actions that are not the most optimal with respect to the current one-step cost, but have lower total cost due to expansions that are possible in the future.

---

<sup>1</sup>A quadtree  $\mathcal{T}'_q$  is a neighbor of  $\mathcal{T}_q$  if  $\mathcal{T}'_q$  results from a  $\mathcal{T}_q$  by a single node expansion of a leaf node of  $\mathcal{T}_q$ , where by ‘‘expansion’’ of a node we mean adding to the tree all its children. For instance, the two quadtrees in Fig. 17.6 are not neighbors according to our definition, as more than one node expansion is needed to generate the quadtree shown on the right from the quadtree shown on the left in that figure.

To this end, we define the function

$$Q_Y(\mathcal{T}_{q^i}, \mathcal{T}_{q^{i+1}}; \beta) \triangleq \max \left\{ \Delta L_Y(\mathcal{T}_{q^i}, \mathcal{T}_{q^{i+1}}; \beta) + \sum_{\tau=1}^n Q_Y(\mathcal{T}_{q^{i+1}}, \mathcal{T}_{q_\tau^{i+2}}; \beta), 0 \right\}, \quad (17.34)$$

where  $\mathcal{T}_{q^{i+1}}$  is a neighbor of  $\mathcal{T}_{q^i}$  with higher leaf node cardinality and

$$Q_Y(\mathcal{T}_{q'}, \mathcal{T}_W; \beta) \triangleq \max \left\{ \Delta L_Y(\mathcal{T}_{q'}, \mathcal{T}_W; \beta), 0 \right\}, \quad (17.35)$$

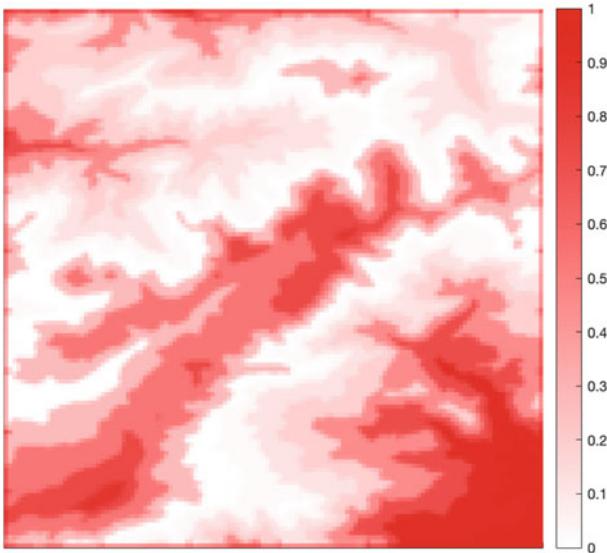
for all  $\mathcal{T}_{q'} \in \mathcal{T}^Q$  for which  $\mathcal{T}_W \in \mathcal{T}^Q$  is a neighbor. The quadtrees  $\mathcal{T}_{q_\tau^{i+2}}$ ,  $\tau \in \{1, \dots, n\}$  are neighbors of  $\mathcal{T}_{q^{i+1}}$  which are obtained by expanding each of the children of one of the leaf of  $\mathcal{T}_{q^i}$ . The next theorem formally establishes the optimality of solutions found by the Q-tree search algorithm.

**Theorem 17.1** *Let  $\mathcal{T}_{\tilde{q}} \in \mathcal{T}^Q$  to be an optimal with respect to the cost  $L_Y(\cdot; \beta)$ . Assume, without loss of generality, that the Q-tree search algorithm is initialized at the tree  $\mathcal{T}_{q^0} \in \mathcal{T}^Q$ , where  $\mathcal{T}_{q^0} \subseteq \mathcal{T}_{\tilde{q}}$  and let  $\mathcal{T}_{q^*} \in \mathcal{T}^Q$  be the solution returned by the Q-tree search algorithm. Then  $\mathcal{T}_{q^*} = \mathcal{T}_{\tilde{q}}$ .*

Below, we present a numerical example to demonstrate the emergence of abstractions in a grid-world setting using the Q-tree search algorithm with different values of  $\beta$ . To this end, consider the environment shown in Figure 17.7 having dimension  $128 \times 128$ . We view this map as representing an environment where the intensity of the color indicates the probability that a given cell is occupied. In this view, the map in Figure 17.7 can be thought of as an occupancy grid where the original space,  $X$ , is considered to be the elementary cells shown in the figure. We wish to compress  $X$  to an abstract representation  $T$  (a quadtree), while preserving as much information regarding cell occupancy as possible. Thus, the relevant random variable  $Y$  is the probability of occupancy. Consequently,  $\Omega_Y = \{0, 1\}$  where  $y = 0$  corresponds to free space and  $y = 1$  to occupied space. It is assumed that  $p(x)$  is provided and  $p(y|x)$  is given by the occupancy grid, where  $p(x, y) = p(y|x)p(x)$ . For simplicity, we assume that  $p(x)$  is uniform, but the results can be easily extended to the case when  $p(x)$  is not uniform.

Several environment depictions for various values of  $\beta$  obtained from the Q-tree search algorithm are shown in Fig. 17.8. As seen in these figures, the solution returned by the Q-tree search algorithm approaches that of the original space as  $\beta \rightarrow \infty$ , with a spectrum of solutions obtained as  $\beta$  is varied. These figures also show that areas containing high information content, as specified by  $Y$ , are refined first while leaving the regions with less information content to be refined at higher values of  $\beta$ .

We see that  $\beta$  resembles a sort of a “rationality parameter,” analogous to [27, 44, 67], where agents with low  $\beta$  are considered to be more resource limited, thus utilizing simpler, lower cardinality representations of the environment. Thus, once the map is given, changing *only* the value of  $\beta$  gives rise to a variety of world representations of varying resolution.



**Fig. 17.7** 128×128 original map of environment. Shading of red indicates the probability that a cell is occupied

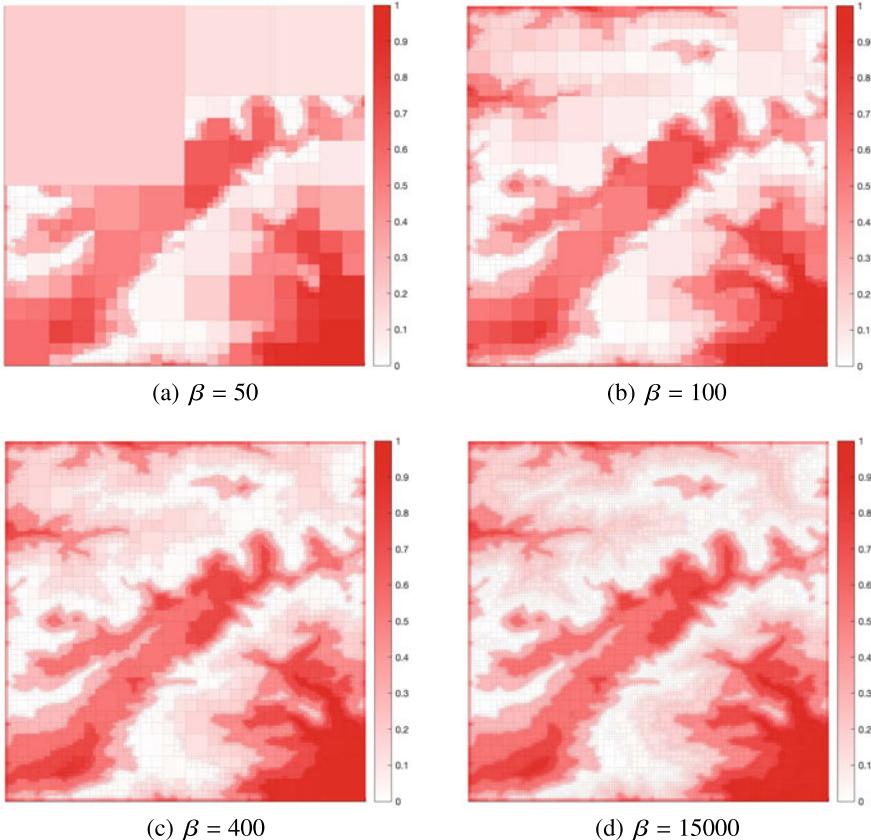
## 17.5 Stochastic Games and Bounded Rationality

Decision-making is often a process that involves multiple entities, whose actions are beyond the subjective control of the decision-maker. Either the environment or other agents affect the outcome of decision-making and hence the agent's deliberation process needs to account for those unpredictable factors as well. A natural way to capture these externalities that are beyond the decision-maker's own control is via game theory. Stochastic games, in particular, is the standard framework to model situations involving multiple agents [25, 51]. However, the solution of stochastic games is often considered to be an intractable undertaking mainly because of the “rationality” and “common knowledge” assumptions imposed on all agents (also referred to as “players” in the context of games). The manifestation of this axiomatic imposition of rationality is the celebrated Nash equilibrium (NE).<sup>2</sup> The Nash equilibrium involves a solution concept such that each player computes the best response strategy to the other players' best response strategies, so that no player has the incentive to unilaterally deviate from its own strategy.

Given the strategies of its opponents, each player can compute the value function (and hence the best response) by solving a single-sided MDP using, say, value iteration (see (17.5)). However, in games the value function of each agent (or the value of the game for the zero-sum case) is coupled to the strategies of the other

---

<sup>2</sup>Since different agents may have different objectives, the concept of optimality is not suitable for games. Instead, all agents seek to find an *equilibrium*.



**Fig. 17.8** Representations of the environment for different values of  $\beta$  [43]

players. In order to compute the NE, each player then needs to know the behavior (i.e., strategies) of the other player(s) and hence, implicitly, their value function. Solving for perfectly rational Nash equilibria thus leads to an infinite regress, as one player computes its own value function by inferring the value function of the other player, who, in turn, computes its own value function by inferring the first player's value function, which requires inferring what the other player infers about the value function of the first player, etc., at infinitum. This infinite regress is what makes the computation of Nash equilibria difficult in practice.

Although the Nash equilibrium definition seems natural, it has received a lot of criticism in practice since it is not at all clear as to why and under what conditions the players in an N-person game may be expected to play such an equilibrium [33]. Nash equilibria make sense only if one assumes that, for some reason, each player knows the strategies of the other players. But this assumption appears to be rather restrictive for many practical situations, especially if the players are not omnipotent (i.e., not

perfectly rational). In addition, the calculation an NE (not surprisingly perhaps in the light of the previous discussion) has been shown to be NP-hard [20].

There are two aspects of the problem that challenge this pessimistic point of view to solving stochastic games. The first is the realization that, in practice, equilibria that imply some form of interdependence between agents are not only plausible but also more likely [2]. The basic tenet of simultaneous action by all players without any collaboration whatsoever, deeply engrained in the Nash equilibrium concept, is too stringent and may be relaxed, leading to more efficient computations [3, 28]. The second aspect (which is emphasized here) is a departure from imposing perfect rationality among all agents and, instead, take into consideration the different sophistication levels of the agents, their perceived knowledge of the environment and of the other agents, or their limited computational resources. Indeed, the assumption of perfect rationality is considered to be too strong and perhaps unrealistic for many applications [29, 59].

One way to resolve this dilemma is by adopting a bounded rationality framework. Several manifestations of bounded rationality exist in the literature, but here we adopt the one that makes use of the model of *cognitive hierarchy* (CH) [16] and *level-k thinking* [38], motivated by the so-called “theory of mind” [26, 72]. Under this notion, the agent no longer seeks the (Nash) equilibrium, but instead seeks a best response to some (believed or inferred) strategy of its opponent(s) according to an assumed level- $k$  responding structure, thus restricting each player’s depth of recursive inference. This imposes a level of sophistication for each agent, with a totally rational agent employing infinite regress. Less sophisticated (i.e., bounded rational) players use regress up to a finite depth. The level of regress of each player induces a “type” for each player (its level of sophistication) [72].

In order to demonstrate the main ideas behind bounded rational stochastic games, in this section, we apply the theory to a pursuit-evasion game in a stochastic environment where both players are assumed to be bounded rational. The generalization to general non-zero-sum multi-player games is conceptually straightforward, but the formulas can quickly become somewhat unwieldy.

### 17.5.1 Stochastic Pursuit–Evasion

In this section, we consider a two-player pursuit-evasion game (PEG) in a stochastic environment to demonstrate the theory. Pursuit–evasion games are a special class of dynamic games [7, 39, 41, 71], which capture a wide range of diverse behaviors among adversarial agents. An extensive amount of literature exists on the topic. Some notable examples include [5–7, 13, 37, 39, 48, 58, 69]. A *stochastic* pursuit–evasion game is a special case of a two-player ( $N = 2$ ) zero-sum stochastic dynamic game for which the running reward is zero ( $R(x, x', u) = 0$ ) and the objective is to maximize/minimize a *terminal cost/reward*  $g(x_T)$  (typically the miss-distance) at some future time  $T$ , where  $x_T \in \mathcal{X}_a$  with  $\mathcal{X}_a \subset \mathcal{X} \triangleq \mathcal{X}_1 \times \mathcal{X}_2$  the set of the absorbing states, that enforce game termination. Here  $\mathcal{X}_i$  is the state space of agent  $i \in \{1, 2\}$ .

The time  $T$  when the game ends is not known a priori, so the PEG is an infinite horizon game with no discount ( $\gamma = 1$ ) and absorbing states.

In a PEG, the maximizer player is called the Evader and the minimizer player is called the Pursuer. Let the policy (or strategy) of player  $i = 1$  (the Evader) be denoted by  $\pi^1$  and let the set of the Evader strategies be  $\Pi^1$ . Similarly, let the policy (or strategy) of player  $i = 2$  (the Pursuer) be denoted by  $\pi^2$  and let the set of the Pursuers strategies be  $\Pi^2$ . The expected reward-to-go function for agent  $i$  under the joint policy  $\pi \triangleq (\pi^1, \pi^2) \in \Pi \triangleq \Pi^1 \times \Pi^2$  and initial state  $x$  is

$$V_\pi^i(x) \triangleq V_{\pi^1, \pi^2}^i(x) = \mathbb{E}_{x, \pi}[g^i(x(T))], \quad i = 1, 2, \quad (17.36)$$

where the terminal cost for a two-player, zero-sum game satisfies  $g^1 = -g^2 = g$ . Note that  $T$  depends, in general, on the joint strategy  $\pi$ . Both agents try to maximize their own expected reward functions by choosing their respective policies  $\pi^i$ . Specifically, for a given initial state  $x$ , the optimization problem can be written as

$$\max_{\pi^1 \in \Pi^1} V_{\pi^1, \pi^2}^1(x), \quad \max_{\pi^2 \in \Pi^2} V_{\pi^1, \pi^2}^2(x). \quad (17.37)$$

Since the game is zero-sum, we have that  $V_{\pi^1, \pi^2}^1(x) = -V_{\pi^1, \pi^2}^2(x) = V_{\pi^1, \pi^2}(x)$  for all joint policies  $(\pi^1, \pi^2) \in \Pi$  and all initial states  $x \in \mathcal{X}$ . As a result, the previous optimization problem reduces to

$$\max_{\pi^1 \in \Pi^1} \min_{\pi^2 \in \Pi^2} V_{\pi^1, \pi^2}(x) = \min_{\pi^2 \in \Pi^2} \max_{\pi^1 \in \Pi^1} V_{\pi^1, \pi^2}(x). \quad (17.38)$$

Given the policy of the minimizing player  $\pi^2$ , let us consider the MDP for the maximizer player. To calculate the value function at a state  $x \in \mathcal{X}$ , we solve the following Bellman equation:

$$V_{\pi^2}(x) \triangleq \max_{\pi^1 \in \Pi^1} \mathbb{E}_P[V_{\pi^1, \pi^2}^1(x')|x, \pi^2] = \max_{\pi^1 \in \Pi^1} \sum_{x' \in \mathcal{X}} P(x'|x, \pi^1(x), \pi^2(x)) V_{\pi^2}(x'), \quad (17.39)$$

where the subscript in  $V_{\pi^2}$  denotes the dependence on the opponent's strategy  $\pi^2$  (compare with notation in (17.3)). For the minimizer player, the Bellman equation is similar but with a min operator instead of a max operator, that is,

$$V_{\pi^1}(x) \triangleq \min_{\pi^2 \in \Pi^2} \mathbb{E}_P[-V_{\pi^1, \pi^2}^1(x')|x, \pi^1] = \min_{\pi^2 \in \Pi^2} \sum_{x' \in \mathcal{X}} P(x'|x, \pi^1(x), \pi^2(x)) V_{\pi^1}(x'). \quad (17.40)$$

Let us assume that (17.39) can be solved given the opponent's policy, and similarly for (17.40). The result is the unique value function  $V_{\pi^{1,*}, \pi^{2,*}}$  along with the optimal (Nash) policies  $\pi^{1,*}$  and  $\pi^{2,*}$ .

### 17.5.2 Level- $k$ Thinking

In the level- $k$  framework, we construct iterative decision rules for the agents. The iterative process begins with “level-0” agents, who have no prior assumptions about their opponents or the structure of the game, and merely act randomly; i.e., they implement a random policy where each action is taken equally likely. The “level- $k$ ” agents presume that their opponent(s) are of “level- $(k-1)$ ”, and they respond accordingly based on this assumption. The notion of rationality is captured through the parameter  $k$ ; the higher the value of  $k$ , the higher the level of rationality. Consequently, in order to capture the notion of bounded rationality [38, 46], each agent has a maximum level  $k_{\max}^i$  beyond which it cannot operate (different agents may have different maximum levels). Under this framework, the computational complexity of finding the optimal policy for an agent is significantly reduced [46], since the game can be cast as a single-sided MDP problem [38] for each agent, which can be solved efficiently [25].

Let  $\pi^i$  denote the strategy of player  $i \in \mathcal{I} = \{1, 2, \dots, N\}$ , and let  $\Pi^i$  be the set of all such strategies. Let also  $\pi^{-i} = (\pi^1, \pi^2, \dots, \pi^{i-1}, \pi^{i+1}, \dots, \pi^N)$  denote the collection of all the strategies of the opponents of player  $i$ . In what follows, the superscript  $k \in \{0, 1, 2, \dots\}$  within a parenthesis will denote the level of rationality of the agent. The premise of level- $k$  thinking is that, if  $\pi^{-i,(k)}$  is a level- $k$  strategy for the opponent(s) of agent  $i$ , the agent will choose its best response to  $\pi^{-i,(k)}$ , which is a level- $(k+1)$  strategy computed, for every state  $x$ , by

$$\pi^{i,(k+1)}(x) \in \operatorname{argmax}_{\pi^{i,(k+1)} \in \Pi^i} \sum_{x' \in \mathcal{X}} P(x'|x, \pi^{i,(k+1)}(x), \pi^{-i,(k)}(x)) V^{i,(k+1)}, \quad (17.41)$$

which is precisely the best response of agent  $i$  to its opponent’s strategy  $\pi^{-i,(k)}$  (and hence it depends on  $\pi^{-i,(k)}$ ), and where for notational simplicity we have set  $V^{i,(k+1)} = V_{\pi^{-i,(k)}}^i$ . We write, accordingly,

$$\pi^{i,(k+1)} \in \operatorname{BestResponse}(\pi^{-i,(k)}), \quad (17.42)$$

to represent the strategy of agent  $i$  obtained from (17.41). Note that the best response to an opponent’s level- $k$  strategy  $\pi^{-i,(k)}$  is by definition the level- $(k+1)$  strategy  $\pi^{i,(k+1)}$ , which is not necessarily the NE strategy  $\pi^{i,*}$ . Recall that the set of strategies  $(\pi^{1,*}, \pi^{2,*}, \dots, \pi^{N,*})$  is a Nash equilibrium if the following conditions hold

$$\pi^{i,*} \in \operatorname{BestResponse}(\pi^{-i,*}), \quad i \in \mathcal{I}, \quad (17.43)$$

with the corresponding value functions  $V^{i,*}$  obtained from

$$\begin{aligned} V_{m+1}^{i,*}(x) &= \max_{\pi^i \in \Pi^i} \sum_{x' \in \mathcal{X}} P(x'|x, \pi^i(x), \pi^{-i,*}(x)) V_m^{i,*}(x'), \quad x \in \mathcal{X}, \\ V_m^{i,*}(x) &= g(x), \quad x \in \mathcal{X}_a, \end{aligned} \quad (17.44)$$

for  $m = 1, 2, \dots$ . If all players are rational, by induction of (17.42) they will all let  $k \rightarrow \infty$  to ensure that  $V^{i,(k)} \approx V^{i,(k+1)} = V^{i,(\infty)}$ , since they all know that they should be using a strategy that is one level higher than the strategy of their opponents, and they also all know that if their opponents, being rational, will do the same. If computational constraints are present, which do not allow the agents to perform this induction till convergence to  $V^{i,(\infty)}$ , they will have to terminate the process at some finite order, say  $k_{\max}$ . As a result, in practice an agent with limited computational resources (e.g., a human<sup>3</sup>) can continue this process only up to a certain level  $k_{\max}$ . This process of building strategies level after level as in (17.41), therefore, continues until the prescribed maximum rationality level  $k_{\max}^i \geq 1$  of agent  $i$  is achieved.

In general, the level- $k$  policy for agent  $i$  is calculated via (compare with (17.39) and (17.40))

$$V^{i,(k)}(x) = \max_{\pi^{i,(k)} \in \Pi^i} \sum_{x' \in \mathcal{X}} P(x'|x, \pi^{i,(k)}, \pi^{-i,(k-1)}) V^{i,(k)}(x'), \quad (17.45)$$

$$\pi^{i,(k)}(x) \in \operatorname{argmax}_{\pi^{i,(k)} \in \Pi^i} \sum_{x' \in \mathcal{X}} P(x'|x, \pi^{i,(k)}(x), \pi^{-i,(k-1)}(x)) V^{i,(k)}(x'), \quad (17.46)$$

where  $k^i \in \{1, 2, \dots, k_{\max}^i\}$  is the rationality level of agent  $i$ . Notice that, in order to calculate the strategy  $\pi^{i,(k)}$ , for  $k^i \in \{1, \dots, k_{\max}^i\}$ , agent  $i$  needs to calculate not only its own strategy  $\pi^{i,(k)}$  but also its opponents' strategies  $\pi^{-i,(k-1)}$ , for all  $k^i \in \{1, \dots, k_{\max}^i - 1\}$ .

In general, there is no guarantee that the previous iterative level- $k$  construction will converge without extra assumptions. For the case when  $\mathcal{I} = \{1, 2\}$ , however, we have the following result.

**Lemma 17.1** *If  $\pi^{i,(k+2)} = \pi^{i,(k)}$  for agent  $i$  at some level  $k$ , then the two agents reach a Nash equilibrium by applying the strategy pair  $(\pi^{i,(k)}, \pi^{-i,(k+1)})$ .*

**Proof** By the construction of level- $k$  strategies in (17.41), we know that

$$\begin{aligned} \pi^{i,(k+2)} &\in \operatorname{BestResponse}(\pi^{-i,(k+1)}), \\ \pi^{-i,(k+1)} &\in \operatorname{BestResponse}(\pi^{-i,(k)}). \end{aligned}$$

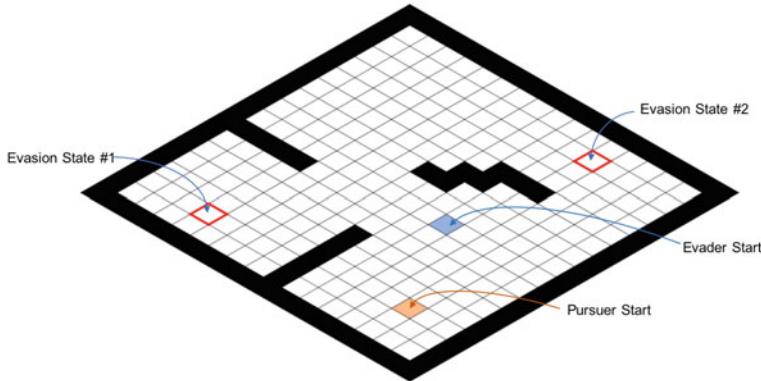
Since  $\pi^{i,(k+2)} = \pi^{i,(k)}$ , we have the following fixed point property:

$$\begin{aligned} \pi^{i,(k)} &\in \operatorname{BestResponse}(\pi^{-i,(k+1)}), \\ \pi^{-i,(k+1)} &\in \operatorname{BestResponse}(\pi^{-i,(k)}). \end{aligned}$$

The strategy pair  $(\pi^{i,(k)}, \pi^{-i,(k+1)})$  then corresponds to a Nash equilibrium, by definition.

---

<sup>3</sup>Behavioral studies in economics seem to indicate that humans apply  $k$ -level reasoning only up to levels 3–4 at a maximum [1].



**Fig. 17.9** Grid world for PEG

**Corollary 17.1** Assume no constraint of the maximum level is imposed. The level- $k$  iterative process terminates at some level  $\bar{k}$  if the condition  $\pi^{i,(\bar{k}+2)} = \pi^{i,(\bar{k})}$  is satisfied for some  $i \in \{1, 2\}$ .

### 17.5.3 A Pursuit–Evasion Game in a Stochastic Environment

In order to apply the theory, we assume a PEG between two bounded-rational players (e.g., UAVs) in a stochastic environment (e.g., wind field). The game is taking place in a discrete world of size  $18 \times 18$ , as shown in Fig. 17.9. The objective of the Evader is to reach one of the two evade states, while avoiding crashing into the obstacles and avoiding capture by the Pursuer. The Pursuer's objective is to capture the Evader before it reaches one of the evade states, while also not crashing into the obstacles. If the Pursuer and the Evader occupy the same cell capture ensues, and the Evader loses the game. As a result, the Evader wins the game if it reaches one of the two evade states alone (not co-occupied with the Pursuer). The terminal reward is assigned as follows

$$g(x) = \begin{cases} +1, & \text{if successful capture or if Evader crashes,} \\ -1, & \text{if successful evasion or if Pursuer crashes,} \\ 0, & \text{if both players crash,} \end{cases} \quad (17.47)$$

for all  $x \in X_a$ .

The motion of both players is affected by a stochastic wind. The mean velocity of the wind is generated randomly, while the covariance is set to  $\sigma_w = 0.4$  with no spatial correlation. Both agents have the same speed (assumed unity) and the same action set, namely,  $\mathcal{U}^1 = \mathcal{U}^2 = \{\text{UP}, \text{DN}, \text{L}, \text{R}\}$  corresponding to the four neighboring

**Table 17.1** Pursuer win percentages against level-2 evader. Results from 1500 simulated games

$k^2 = 2$	$k^1 = 1$	$k^1 = 2$	$k^1 = 3$	$k^1 = 4$	$k^1 = 5$	$k^1 = 6$
Pursuer wins	48.5	47.6	51.7	50.9	51.2	51.1
Due to capture	36.4	38.2	45.3	43.1	42.7	42.9

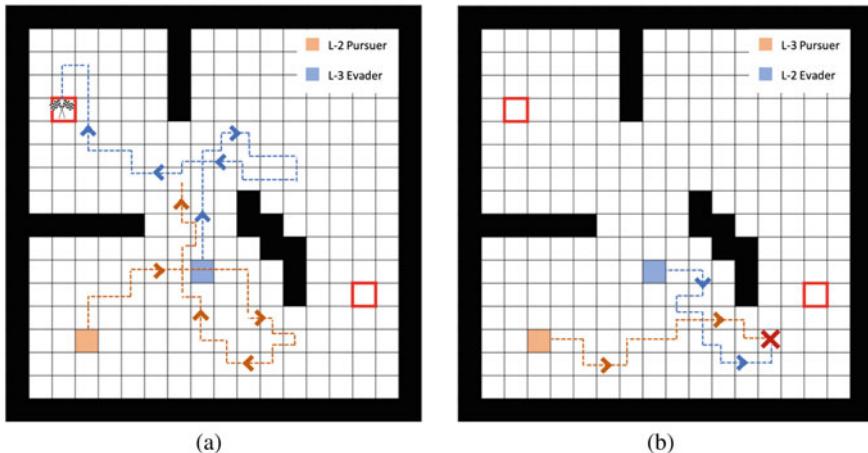
**Table 17.2** Evader win percentages against level-2 pursuer. Results from 1500 simulated games

$k^1 = 2$	$k^2 = 1$	$k^2 = 2$	$k^2 = 3$	$k^2 = 4$	$k^2 = 5$	$k^2 = 6$
Evader wins	47.5	52.4	53.8	52.9	53.2	53.1
Due to evasion	35.6	42.2	45.3	44.5	44.7	44.2

cells (4-connectivity). The starting positions of the agents, the evasion states, and the obstacles are shown in Fig. 17.9.

We assume that the level-0 agents pick their actions such that they will not cause an immediate crash, while choosing their action uniformly randomly. We construct the level- $k$  thinking hierarchy and calculate the strategies of both agents at different levels. We then simulate the resulted strategies at some selected level pairs over 1500 games. To illustrate the results, Tables 17.1 and 17.2 show the win percentages for each player of the selected scenarios. In Tables 17.1 and 17.2, we fix one agent to be level 2 while varying the level of the other agent. In both tables, the highest winning rates are attained at level 3. This result is expected, since level-3 strategies are by definition the best response to the level-2 strategies. It can also be observed that after level 5 the performance only varies slightly, since the strategies at higher levels do not differ much from each other (as they are converging to the NE).

In Fig. 17.10 two sample trajectories are illustrated. The first trajectory depicts a level-3 Evader against a level-2 Pursuer. One observes the “deceiving” behavior of the more sophisticated Evader in this case: it first moves toward evasion state #2 (on the right), which tricks the level-2 Pursuer to also go right and take the shorter route beneath the obstacles to defend evade state #2. The Evader then suddenly turns left and goes to the evade state #1 (on the left). When the Evader reveals that its true intention is to move toward evasion state #1, it is too late for the Pursuer to capture the Evader. The second trajectory depicts a successful capture. Different from the previous scenario, the level-3 Pursuer has perfect information about the strategy of the level-2 Evader, and as a result it can predict well what the Evader will do in the next time step. The Pursuer then simply acts accordingly to maximize the probability of successful capture.



**Fig. 17.10** Sample trajectories of agents playing fixed rationality levels. The first game is won by the evader; the second is won by the pursuer. **a** Level-2 pursuer versus level-3 evader; **b** Level-3 pursuer versus level-2 evader

## 17.6 Conclusions

Future autonomous systems will integrate data-driven and model-based methods for perception, learning, planning, and control. Current algorithms in these areas are computationally intensive, a fact that hinders their use in many applications that require fast (re)planning and decision-making, especially under uncertain, time-varying, and complex environments. Enabling “human-like” planning (e.g., flexible, robust, resilient, in sync with changing situations) requires a different look, beyond the current practice which is heavily based on the assumption that the decision-maker is a rational agent who is a perfect optimizer. That is, in the majority of current theory assumes that the agent has ample computational and/or time resources to process all information and compute the optimal solution unhindered. In many practical applications, this is a strong and often unrealistic assumption. For example, biological systems are adaptive in nature and excel at finding solutions to complex problems that may not be globally optimal but are “good enough” given the constraints and/or available sensory information.

The lingering issue of how to best deal with the computational complexity for “human-like autonomy,” especially under strict timing constraints suggests an alternative approach that abandons the perfect optimality/rationality paradigm for a bounded rationality one. Bounded rationality is supported by ample empirical evidence and captures better human decision-making than the more traditional maximum expected utility (MEU) paradigm.

The bounded rational paradigm is especially relevant when the agent operates in uncertain, complex, dynamic environments under limited time, computational resources, or energy. It is a good option, in particular, when strong priors (heuristics)

are available. A great source of heuristics is via abstractions that organize the collected information according to their relevance to the task at hand. In fact, a distinctive property of human intelligence is its ability to form abstractions by neglecting irrelevant information, allowing to separate structure from noise and allocate attention and processing capabilities tailored to the given resource constraints.

In this work, we first provide an overview of the existing framework for bounded rational decision-making based on the variational free-energy principle that allows the construction of state abstractions, which can then be utilized to solve sequential decision-making problems subject to computational and informational constraints. We discuss the connection of the proposed approach with information-theoretic signal compression (i.e., the Information Bottleneck principle), and formulate a novel optimization problem to obtain tree-based abstractions for path-planning problems as a function of the agent's computational resources. Finally, we investigate the effect of bounded rationality in pursuit–evasion stochastic games. Specifically, we show that limited-depth iterative best response strategies (e.g., the “theory of mind”) avoid the infinite regress implied for a Nash equilibrium and can provide implementable solutions to the problem at low computational complexity.

We believe that there are still many avenues open for investigation in this area. Task-aware and/or computation-aware algorithms for perception, planning, and learning with provable performance guarantees are indispensable for all autonomous systems. In particular, we argue that the bounded rational model is highly relevant for the development of the next generation of autonomous systems, having (expert) human-like perception and decision-making capabilities, especially when these autonomous agents need to interact with humans.

**Acknowledgements** The author would like to acknowledge Daniel Larsson, Yue Guan, Daniel Braun, Pedro Ortega, and Dipankar Maity for many insightful discussions. Support for this work has been provided by NSF award 1849130, by ONR awards N00014-13-0563, N00014-18-12375, N00014-18-1-2828, and by ARL under DCIST CRA W911NF-17-2-018.

## References

1. Arad, A., Rubinstein, A.: The 11–20 money request game: a level- $k$  reasoning study. *Am. Econ. Rev.* **102**(7), 3561–3573 (2012)
2. Aumann, R.J.: Subjectivity and correlation in randomized strategies. *J. Math. Econ.* **1**, 67–96 (1974)
3. Aumann, R.J.: Correlated equilibrium as an expression of Bayesian rationality. *Econometrica: J. Econ. Soc.* **55**(1), 1–18 (1987)
4. Aumann, R.J.: Rationality and bounded rationality. *Games Econ. Behav.* **21**(1–2), 2–14 (1997)
5. Başar, T., Olsder, G.: *Dynamic Noncooperative Game Theory: Second Edition*. Classics in Applied Mathematics. Society for Industrial and Applied Mathematics (1999)
6. Balakrishnan, A.V.: *Stochastic Differential Systems I*. Springer, Berlin (1973)
7. Behn, R., Ho, Y.-C.: On a class of linear stochastic differential games. *IEEE Trans. Autom. Control* **13**(3), 227–240 (1968). <https://doi.org/10.1109/TAC.1968.1098898>
8. Behnke, S.: Local multiresolution path planning. *Lect. Notes Artif. Intell.* **3020**(1), 332–343 (2004)

9. Bellman, R.: *Dynamic Programming*. Princeton University Press, Princeton (1957)
10. Bennett, C.H.: The thermodynamics of computation - a review. *Int. J. Theor. Phys.* **21**(12), 905–940 (1982)
11. Bertsekas, D.P.: *Dynamic Programming and Optimal Control*, vol. 2, 4th edn. Athena Scientific (2012)
12. Bodis-Wollner, I.: Pre-emptive perception. *Perception* **37**, 462–478 (2008). <https://doi.org/10.1088/p5880>
13. Bopardikar, S.D., Bullo, F., Hespanha, J.P.: On discrete-time pursuit-evasion games with sensing limitations. *IEEE Trans. Robot.* **24**(6), 1429–1439 (2008)
14. Borji, A., Sihite, D., Itti, L.: Modeling task-driven visual attention. In: *Proceedings of the British Machine Vision Conference*. Dundee, Scotland (2011)
15. Braun, D.A., Ortega, P.A., Theodorou, E., Schaal, S.: Path integral control and bounded rationality. France, Paris (2011)
16. Camerer, C.F., Ho, T.H., Chong, J.K.: A cognitive hierarchy model of games. *Q. J. Econ.* **119**(3), 861–898 (2004)
17. Cover, T.M., Thomas, J.A.: *Elements of Information Theory*. Wiley, New Jersey (2006)
18. Cowlagi, R., Tsotras, P.: Hierarchical motion planning with kinodynamic feasibility guarantees. *IEEE Trans. Robot.* **28**(2), 379–395 (2012)
19. Cowlagi, R., Tsotras, P.: Multiresolution path- and motion planning for autonomous agents via wavelet-based cell decompositions. *IEEE Trans. Syst. Man Cybern., Part B: Cybern.* **42**(5), 1455–1469 (2012)
20. Daskalakis, C., Goldberg, P.W., Papadimitriou, C.H.: The complexity of computing a Nash equilibrium. *SIAM J. Comput.* **39**(1), 195–259 (2009)
21. Deco, G., Heinke, D.: Attention and spatial resolution: a theoretical and experimental study of visual search in hierarchical patterns. *Perception* **36**, 335–354 (2007)
22. Einhorn, E., Schröter, C., Gross, H.M.: Finding the adequate resolution for grid mapping - cell sizes locally adapting on-the-fly. In: *IEEE Conference on Robotics and Automation*. Shanghai, China (2011)
23. Feldman, H., Friston, K.J.: Attention, uncertainty, and free-energy. *Frontier Human Neurosci.* **4**(215) (2010)
24. Feynman, R.P.: *Feynman Lectures on Computation*. Addison-Wesley Longman Publishing Co., Inc. (1998)
25. Filar, J., Vrieze, K.: *Competitive Markov Decision Processes*. Springer, Berlin (1996)
26. Frith, U., Frith, C.D.: Development and neurophysiology of mentalizing. *Philos. Trans. R. Soc. Lond. Ser. B: Biol. Sci.* **358**(1431), 459–473 (2003)
27. Genewein, T., Leibfried, F., Grau-Moya, J., Braun, D.A.: Bounded rationality, abstraction, and hierarchical decision-making: an information-theoretic optimality principle. *Front. Robot. AI* **2**, 27 (2015)
28. Gilboa, I., Zemel, E.: Nash and correlated equilibria: some complexity considerations. *Games Econ. Behav.* **1**(1), 80–93 (1989)
29. Gilovich, T., Griffin, D., Kahneman, D.: *Heuristics and Biases: The Psychology of Intuitive Judgment*. Cambridge University Press, Cambridge (2002)
30. Goodrich, M.A., Stirling, W.C., Frost, R.L.: A theory of satisficing decisions and control. *IEEE Trans. Syst., Man Cybern., Part A: Syst. Humans* **28**(6), 763–779 (1998)
31. Grafton, S.T., Hamilton, A.: Evidence for a distributed hierarchy of action representation in the brain. *Human Movement Sci.* **26**(4), 590–616 (2007)
32. Grau-Moya, J., Leibfried, F., Genewein, T., Braun, D.A.: Planning with information-processing constraints and model uncertainty in Markov decision processes. In: *Joint European Conference on Machine Learning and Knowledge Discovery in Databases*, pp. 475–491. Springer (2016)
33. Harsanyi, J.C.: *Rational Behavior and Bargaining Equilibrium in Games and Situations*, Cambridge (1977)
34. Hassanpour, S., Wübben, D., Dekorsy, A.: Overview and investigation of algorithms for the information bottleneck method. In: *International ITG Conference on Systems. Communications and Coding*, pp. 1–6. Hamburg, Germany (2017)

35. Hauer, F., Kundu, A., Rehg, J.M., Tsiotras, P.: Multi-scale perception and path planning on probabilistic obstacle maps. In: IEEE International Conference on Robotics and Automation, pp. 4210–4215. Seattle (2015)
36. Hauer, F., Tsiotras, P.: Reduced complexity multi-scale path-planning on probabilistic maps. In: IEEE Conference on Robotics and Automation, pp. 83–88. Stockholm, Sweden (2016)
37. Hespanha, J.P., Hyoun Jin Kim, Sastry, S.: Multiple-agent probabilistic pursuit-evasion games. In: Proceedings of the 38th IEEE Conference on Decision and Control, vol. 3, pp. 2432–2437 (1999)
38. Ho, T.H., Su, X.: A dynamic level-k model in sequential games. *Manag. Sci.* **59**(2), 452–469 (2013)
39. Ho, Y., Bryson, A., Baron, S.: Differential games and optimal pursuit-evasion strategies. *IEEE Trans. Autom. Control* **10**(4), 385–389 (1965)
40. Horvitz, E., Zilberstein, S.: Computational tradeoffs under bounded resources. *Artif. Intell.* **126**(1), 1–4 (2001)
41. Isaacs, R.: Differential Games. Wiley, New York (1965)
42. Itti, L., Baldi, P.: Bayesian surprise attracts human attention. *Vis. Res.* **49**(10), 1295–1306 (2009)
43. Larsson, D., Maity, D., Tsiotras, P.: Q-search trees: an information-theoretic approach towards hierarchical abstractions for agents with computational limitations (2019). <https://arxiv.org/abs/1910.00063>
44. Larsson, D.T., Braun, D., Tsiotras, P.: Hierarchical state abstractions for decision-making problems with computational constraints. In: 56th IEEE Conference on Decision and Control, pp. 1138–1143. Melbourne, Australia (2017)
45. Lee, T.S., Mumford, D.: Hierarchical Bayesian inference in the visual cortex. *J. Opt. Soc. Am. A* **20**(7), 1434–1448 (2003)
46. Li, J., Kendall, G., John, R.: Computing Nash equilibria and evolutionarily stable states of evolutionary games. *IEEE Trans. Evolut. Comput.* **20**(3), 460–469 (2016)
47. Lin, J.: Divergence measures based on the Shannon entropy. *IEEE Trans. Inf. Theory* **37**(1), 145–151 (1991)
48. Lin, W., Qu, Z., Simaan, M.A.: Nash strategies for pursuit-evasion differential games involving limited observations. *IEEE Trans. Aerosp. Electron. Syst.* **51**(2), 1347–1356 (2015)
49. Lipman, B.L.: Information Processing and Bounded Rationality: a Survey, vol. 28, pp. 42–67. Wiley on behalf of the Canadian Economics Association (1995)
50. Nelson, E., Corah, M., Michael, N.: Environment model adaptation for mobile robot exploration. *Auton Robot* **42**, 257–272 (2015)
51. von Neumann, J., Morgenstern, O.: Theory of Games and Economic Behavior. Princeton University Press, Princeton (1945)
52. Nguyen, A., Yosinski, J., Clune, J.: Deep neural networks are easily fooled: High confidence predictions for unrecognizable images. In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, pp. 427–436. Boston (2015)
53. Ortega, P.A., Braun, D.A.: Information, Utility and Bounded Rationality, pp. 269–274. Springer, Berlin (2011)
54. Ortega, P., Stocker, A.: Human decision-making under limited time (2016). [arXiv:1610.01698v1](https://arxiv.org/abs/1610.01698v1)
55. Ortega, P.A., Braun, D.A.: Thermodynamics as a theory of decision-making with information-processing costs. In: Proceedings of the Royal Society. Royal Society (2013)
56. Rubin, J., Shamir, O., Tishby, N.: Trading value and information in MDPs. *Decision Making with Imperfect Decision Makers*, Chap. 3, pp. 57–74. Springer, Heidelberg (2012)
57. Rubinstein, A.: Modeling Bounded Rationality. MIT Press, Cambridge (1998)
58. Shinar, J., Gutman, S.: Three-dimensional optimal pursuit and evasion with bounded controls. *IEEE Trans. Autom. Control* **25**(3), 492–496 (1980)
59. Simon, H.A.: Models of Bounded Rationality, vol. 3. MIT Press, Cambridge (1982)
60. Slonim, N.: The information bottleneck: Theory and applications. Ph.D. thesis, The Hebrew University (2002)

61. Slonim, N., Tishby, N.: Agglomerative information bottleneck. In: Advances in Neural Information Processing, pp. 617–623. Denver (2000)
62. Strouse, D.J., Schwab, D.J.: The deterministic information bottleneck. *Neural Comput.* **29**, 1611–1630 (2017)
63. Sutton, R.S., Barto, A.G.: Reinforcement Learning: An Introduction. The MIT Press, Cambridge (1998)
64. Szegedy, C., Zaremba, W., Sutskever, I., Bruna, J., Erhan, D., Goodfellow, I., Fergus, R.: Intriguing properties of neural networks (2013). <https://arxiv.org/abs/1312.6199>
65. Taylor, P., Hobbs, J.N., Burroni, J., Siegelmann, H.T.: The global landscape of cognition: hierarchical aggregation as an organizational principle of human cortical networks and functions. *Sci. Rep.* **5**, 18112 (2015)
66. Tishby, N., Pereira, F.C., Bialek, W.: The information bottleneck method. In: The 37th Annual Allerton Conference on Communication. Control and Computing, pp. 368–377. Monticello (1999)
67. Tishby, N., Polani, D.: Information theory of decisions and actions. In: Perception-Action Cycle: Model, Architectures and Hardware, Springer Series in Cognitive and Neural Systems 1, chap. 19, pp. 601–636. Springer Science & Business Media, Berlin (2011)
68. Tsiotras, P., Jung, D., Bakolas, E.: Multiresolution hierarchical path-planning for small UAVs using wavelet decompositions. *J. Intell. Robot. Syst.* **66**(4), 505–522 (2012)
69. Vidal, R., Shakernia, O., Kim, H.J., Shim, D.H., Sastry, S.: Probabilistic pursuit-evasion games: theory, implementation, and experimental evaluation. *IEEE Trans. Robot. Autom.* **18**(5), 662–669 (2002)
70. Wierzbicki, A.P.: A mathematical basis for satisficing decision making. *Math. Model.* **3**(5), 391–405 (1982)
71. Willman, W.: Formal solutions for a class of stochastic pursuit-evasion games. *IEEE Trans. Autom. Control* **14**(5), 504–509 (1969)
72. Yoshida, W., Dolan, R.J., Friston, K.J.: Game theory of mind. *PLOS Comput. Biol.* **4**(12), 1–14 (2008)
73. Zilberstein, S.: Satisficing and bounded optimality. In: AAAI Spring Symposium on Satisficing Models, pp. 91–94 (1998)

# Chapter 18

## Fairness in Learning-Based Sequential Decision Algorithms: A Survey



Xueru Zhang and Mingyan Liu

**Abstract** Algorithmic fairness in decision-making has been studied extensively in static settings where one-shot decisions are made on tasks such as classification. However, in practice most decision-making processes are of a sequential nature, where decisions made in the past may have an impact on future data. This is particularly the case when decisions affect the individuals or users generating the data used for future decisions. In this survey, we review existing literature on the fairness of data-driven sequential decision-making. We will focus on two types of sequential decisions: (1) past decisions have no impact on the underlying user population and thus no impact on future data; (2) past decisions have an impact on the underlying user population, and therefore the future data, which can then impact future decisions. In each case the impact of various fairness interventions on the underlying population is examined.

### 18.1 Introduction

Decision-making algorithms that are built from real-world datasets have been widely used in various applications. When these algorithms are used to inform decisions involving human beings (e.g., college admission, criminal justice, and resume screening), which are typically done by predicting certain variable of interest from observable features, they may inherit the potential, pre-existing bias in the dataset and exhibit similar discrimination against protected attributes such as race and gender. For example, the COMPAS algorithm used by courts for predicting recidivism in the United States has been shown to be biased against black defendants [11]; job searching platform XING ranks less qualified male applicants higher than female

---

X. Zhang (✉) · M. Liu  
University of Michigan, Ann Arbor, MI, USA  
e-mail: [xueru@umich.edu](mailto:xueru@umich.edu)

M. Liu  
e-mail: [mingyan@umich.edu](mailto:mingyan@umich.edu)

applicants who are more qualified [29]; a nationwide algorithm used for allocating medical resources in US is biased against black patients [35].

There are various potential causes for such bias. It may have been introduced when data is collected. For instance, if data sampled from a minority group is much smaller in size than that from a majority group, then the model could be more in favor of the majority group due to this representation disparity (e.g., more than a third of data in ImageNet and Open Images, two datasets widely used in machine learning research communities, is US-based [37]). Another example is when the data collection decision itself reflects bias, which then impacts the collected data (e.g., if more police officers are dispatched to places with higher crime rate to begin with, then crimes are more likely to be recorded in these places [12]). Even when the data collection process is unbiased, bias may already exist in the data. Historical prejudice and stereotypes can be preserved in data (e.g., the relationship between “man” and “computer programmers” were found to be similar to that between “woman” and “homemaker” [7]). An interested reader can find more detailed categorization of bias in the survey [34].

The problem does not merely stop here. On one hand, decisions made about humans can affect their behavior and reshape the statistics of the underlying population. On the other hand, decision-making algorithms are updated periodically to assure high performance on the targeted populations. This complex interplay between algorithmic decisions and the underlying population can lead to pernicious long-term effects by allowing biases to perpetuate and reinforcing pre-existing social injustice. For example, [2] shows that incarceration can significantly reduce people’s access to finance, which in turn leads to substantial increase in recidivism; this forms a credit-driven crime cycle. Another example is speech recognition: products such as Amazon’s Alexa and Google Home are shown to have accent bias with native speakers experiencing much higher quality than non-native speakers [18]. If this difference in user experience leads to more native speakers using such products while driving away non-native speakers, then over time the data used to train the algorithms may become even more skewed toward native speakers, with fewer and fewer non-native samples. Without intervention, the resulting model may become even more accurate for the former and less for the latter, which then reinforces their respective user experience [19]. Similar negative feedback loops have been observed in various settings such as recommendation system [9], credit market [13], and policing prediction [12]. Preventing discrimination and guaranteeing fairness in decision-making is thus both an ethical and a legal imperatives.

To address the fairness issues highlighted above, a first step is to define fairness. Anti-discrimination laws (e.g., Title VII of the Civil Rights Act of 1964) typically assess fairness based on disparate impact and disparate treatment. The former happens when outcomes disproportionately benefit one group while the latter occurs when the decisions rely on sensitive attributes such as gender and race. Similarly, various notions of fairness have been formulated mathematically for decision-making systems and they can be categorized roughly into two classes:

- **Individual fairness:** this requires that similar individuals are treated similarly.
- **Group fairness:** this requires (approximate) parity of certain statistical measures (e.g., positive classification rate, true positive rate, etc.) across different demographic groups.

In Sect. 18.2 we present the definitions of a number of commonly used fairness measures. Their suitability for use is often application dependent, and many of them are incompatible with each other [28].

To satisfy the requirement of a given definition of fairness, various approaches have been proposed and they generally fall under three categories:

1. **Pre-processing:** by changing the original dataset such as removing certain features, reweighing, and so on, e.g., [8, 15, 26, 42].
2. **In-processing:** by modifying the decision-making algorithms such as imposing fairness constraints or changing objective functions, e.g., [1, 5, 40, 41].
3. **Post-processing:** by adjusting the output of the algorithms based on sensitive attributes, e.g., [17].

While the effectiveness of these approaches have been shown in various domains, most of these studies are done using a static framework where only the immediate impact of the learning algorithm is assessed but not its long-term consequences. Consider an example where a lender decides whether or not to issue a loan based on the applicant’s credit score. Decisions satisfying an identical true positive rate (equal opportunity) across different racial groups can make the outcome seem fairer [17]. However, this can potentially result in more loans issued to less qualified applicants in the group whose score distribution skews toward higher default risk. The lower repayment among these individuals causes their future credit scores to drop, which moves the score distribution of that group further toward higher default risk [31]. This shows that intervention by imposing seemingly fair decisions in the short term can lead to undesirable results in the long run [43]. As such, it is critical to understand the long-term impacts of fairness interventions on the underlying population when developing and using such decision systems.

In this survey, we focus on fairness in sequential decision systems. We introduce the framework of sequential decision-making and commonly used fairness notions in Sect. 18.2. The literature review is done in two parts. We first consider sequential settings where decisions do not explicitly impact the underlying population in Sect. 18.3, and then consider sequential settings where decisions and the underlying population interact with each other in Sect. 18.4. The impact of fairness interventions is examined in each case. For consistency of the survey, we may use a set of notations different from the original works.

## 18.2 Preliminaries

### 18.2.1 Sequential Decision Algorithms

The type of decision algorithms surveyed in this paper are essentially classification/prediction algorithms used by a decision-maker to predict some variable of interest (label) based on a set of observable features. For example, judges predict whether or not a defendant will re-offend based on its criminal records; college admission committee decides whether or not to admit an applicant based on its SAT; lender decides whether or not to issue a loan based on an applicant's credit score.

To develop such an algorithm, data are collected consisting of both features and labels, from which the best mapping (decision rule) is obtained, which is then used to predict unseen, new data points. Every time a prediction is made, it can either be correct (referred to as a gain) or incorrect (referred to as a loss). The optimal decision rule without fairness consideration is typically the one that minimizes losses or maximizes gains.

In a sequential framework, data arrive and are observed sequentially and there is feedback on past predictions (loss or gain), and we are generally interested in optimizing the performance of the algorithm over a certain time horizon. Such a sequential formulation roughly falls into one of two categories.

- P1:** The goal of the algorithm is to learn a near-optimal decision rule quickly, noting that at each time step only partial information is available, while minimizing (or maximizing) the total loss (or gain) over the entire horizon. Furthermore, within the context of fairness, an additional goal is to understand how a fairness requirement impacts such a decision rule.
- P2:** This is a setting where not only do data arrives sequentially, but decisions made in the past can affect the feature space of the underlying population, thereby changing the nature of future observations. The goal in this case is to learn an optimal decision rule at each time step and understand the impact it has on the population and how fairness requirement further adds to the impact.

### 18.2.2 Notions of Fairness

As mentioned in Sect. 18.1, different notions of fairness can be generally classified into *individual fairness* and *group fairness*.

**Group fairness:** For simplicity of exposition and without loss of generality, we will limit ourselves to the case of two demographic groups  $G_a, G_b$ , distinguished based on some sensitive attribute  $Z \in \{a, b\}$  representing group membership (e.g., gender and race). Group fairness typically requires certain statistical measures to be equal across these groups. Mathematically, denote by random variable  $Y \in \{0, 1\}$  an

individual's true label and  $\hat{Y}$  its prediction generated from a certain decision rule. Then the following is a list of commonly used group fairness criteria.

1. Demographic Parity (DP): it requires that the positive prediction rate be equal across different demographic groups, i.e.,  $\mathbb{P}(\hat{Y} = 1|Z = a) = \mathbb{P}(\hat{Y} = 1|Z = b)$ .
2. Equal of Opportunity (EqOpt): it requires true positive rate (TPR)<sup>1</sup> be equal across different demographic groups, i.e.,  $\mathbb{P}(\hat{Y} = 1|Y = 1, Z = a) = \mathbb{P}(\hat{Y} = 1|Y = 1, Z = b)$ .
3. Equalized Odds (EO): it requires both the false positive rate and true positive rate be equal across different demographic groups, i.e.,  $\mathbb{P}(\hat{Y} = 1|Y = y, Z = a) = \mathbb{P}(\hat{Y} = 1|Y = y, Z = b), \forall y \in \{0, 1\}$ .
4. Equalized Loss (EqLos): it requires different demographic groups experience the same total prediction error, i.e.,  $\mathbb{P}(\hat{Y} \neq Y|Z = a) = \mathbb{P}(\hat{Y} \neq Y|Z = b)$ .

**Individual fairness:** Such a criterion targets the individual, rather than group level. Commonly used examples are as follows.

1. Fairness through awareness (FA): this requires that similar individuals be treated similarly.
2. Meritocratic fairness (MF): this requires that less qualified individuals not be favored over more qualified individuals.

The above definitions do not specify how similarity among individuals or qualification of individuals are measured, which can be context dependent.

### 18.3 (Fair) Sequential Decision When Decisions Do Not Affect Underlying Population

We first focus on a class of sequential decision problems (**P1**) where the decision at each time step does not explicitly affect the underlying population; a list of these studied are summarized in Table 18.1. Most of these works have developed algorithms that can learn a decision rule with sufficient accuracy/performance subject to certain fairness constraint, and the impact of fairness on these sequential decision-making problems is reflected through its (negative) effect on the achievable performance.

#### 18.3.1 Bandits, Regret, and Fair Regret

We begin with [4, 16, 20] on online learning problems, where a decision-maker at each time  $t$  receives data from one individual and makes decision according to some decision rule. It then observes the loss (resp. utility) incurred from that decision. The

---

<sup>1</sup>Based on the context, this criterion can also refer to equal false negative rate (FNR), false positive rate (FPR), or true negative rate (TNR).

**Table 18.1** Summary of related work when decisions do not affect the underlying population. \* represents the use of fairness definitions or interventions not included in Sect. 18.2.2

	Fairness definition		Data type	Problem type
	Group fairness	Individual fairness		
Reference [20]		FA	i.i.d.	<b>P1</b>
Reference [16]		FA	non-i.i.d.	<b>P1</b>
Reference [4]	EqOpt		i.i.d.	<b>P1</b>
Reference [25]		MF	i.i.d.	<b>P1</b>
Reference [24]		MF	i.i.d.	<b>P1</b>
Reference [33]		FA	i.i.d.	<b>P1</b>
Reference [38]	*		i.i.d.	<b>P1</b>
Reference [6]	EqOpt, EO, EqLos		non-i.i.d.	<b>P1</b>
Reference [10]		*	i.i.d.	<b>P1</b>
Reference [30]		*	i.i.d.	<b>P1</b>
Reference [36]		*	i.i.d.	<b>P1</b>
Reference [14]		FA	non-i.i.d.	<b>P1</b>

goal is to learn a decision rule from a set of data collected over  $T$  time steps under which (1) the accumulated expected loss (resp. utility) over  $T$  steps is upper (resp. lower) bounded and (2) certain fairness criterion is satisfied. Specifically, [16, 20] focus on individual fairness which ensures that similar individuals (who arrive at different time steps) be treated similarly, by comparing each individual with either all individuals within a time epoch [16, 20] or only those who have arrived in the past [16]. By contrast, [4] focuses on group fairness (EqOpt), where at each time the arriving individual belongs to one demographic group and the goal is to ensure different demographic groups in general receive similar performance over the entire time horizon. Moreover, [4] considered a partial feedback scenario where the loss (resp. utility) is revealed to the decision-maker only when certain decisions are made (e.g., whether an applicant is qualified for a certain job is only known when he/she is hired). In each of these settings, the impact of fairness constraint on accumulated expected loss/utility is examined and quantified and an algorithm that satisfies both (approximate) fairness and certain loss/utility is developed.

In some applications, the decision-maker at each time makes a selection from multiple choices. For example, hiring employees from multiple demographic groups, selecting candidates from a school for certain competitions, etc. Specifically, the decision-maker at each time receives features of multiple individuals (potentially from different demographic groups) and the corresponding sequential decision problems can be formulated as a multi-armed bandit problem, where each arm represents either one specific individual or one demographic group and choosing an arm represents selecting one individual (from one demographic group). In a classic stochastic

bandit problem, there is a set of arms  $\mathcal{Z} = \{1, \dots, K\}$ . The decision-maker selects an arm  $k_t$  at time  $t$  from  $\mathcal{Z}$  and receives a random reward  $r_t^{k_t}$ , drawn from a distribution  $r_t^{k_t} \sim \mathbb{P}^{k_t}(\cdot; g^{k_t})$  with unknown mean  $\mathbb{E}(r_t^{k_t}) = g^{k_t} \in [0, 1]$ .

Let  $h_t = \{(k_s, r_s^{k_s})\}_{s=1}^{t-1}$  represent all history information received by the decision-maker up to time  $t$ . Then the decision rule  $\tau_t$  at  $t$  is a probability distribution over all arms. Denote by  $\tau_t(k|h_t)$  the probability of selecting arm  $k$  at time  $t$  given history  $h_t$ . The *regret* of applying the decision rule  $\{\tau_t\}_{t=1}^T$  over  $T$  time steps is defined as

$$\text{Regret}^T(\{\tau_t\}_t) = \sum_{t=1}^T \max_k g^k - \sum_{t=1}^T \mathbb{E}_{k_t \sim \tau_t}[g^{k_t}].$$

The goal of a *fair* decision-maker in this context is to select  $\{\tau_t\}_{t=1}^T$  such that the regret over  $T$  time steps is minimized, while certain fairness constraint is satisfied.

Joseph et al. in [25] proposed the use of meritocratic fairness in the above bandit setting as follows. Consider a multi-armed bandit problem where each arm represents an individual and the decision-maker selects one individual at each time. Let the mean reward  $\mathbb{E}(r_i^k)$  represent the average qualification of individuals (e.g., hiring more qualified applicant can bring higher benefit to a company); then it is unfair if the decision-maker preferentially chooses an individual less qualified in expectation over another. Formally, the decision-maker is defined to be  $\delta$ -fair over  $T$  time steps if with probability  $1 - \delta$ , for all pairs of arms  $k, k' \in \mathcal{Z}$  and  $\forall t$ , the following holds.

$$\tau_t(k|h_t) > \tau_t(k'|h_t) \text{ only if } g^k > g^{k'}. \quad (18.1)$$

Reference [25] developed an algorithm to find optimal decision rules in classic stochastic setting that is  $\delta$ -fair. To ensure  $\delta$ -fairness, for any two arms  $k, k'$ , they should be selected with equal probability unless  $g^k > g^{k'}$ . Let  $u_t^k, l_t^k$  be the upper and lower confidence bounds of arm  $k$  at time  $t$ . Then arms  $k$  and  $k'$  are *linked* if  $[l_t^k, u_t^k] \cap [l_t^{k'}, u_t^{k'}] \neq \emptyset$ ; arms  $k$  and  $k'$  are *chained* if they are in the same component of the transitive closure of the linked relation. The algorithm in [25] first identifies the arm with the highest upper confidence bound and finds all arms chained to it ( $\mathcal{S}_t$ ). For arms not in  $\mathcal{S}_t$ , the decision-maker has sufficient confidence to claim that they are less qualified than others, while for arms in  $\mathcal{S}_t$ , the decision-maker randomly selects one at uniform to ensure fairness.

Reference [25] shows that if  $\delta < 1/\sqrt{T}$ , then the algorithm can achieve  $\text{Regret}^T(\{\tau_t\}_t) = O(\sqrt{K^3 T \ln \frac{TK}{\delta}})$ . In contrast, without fairness consideration, the original upper confidence bound (UCB) algorithm proposed by Auer et al. [3] achieves  $\text{Regret}^T(\{\tau_t\}_t) = O(K \log T / \Delta_a)$ , where  $\Delta_a$  is the difference between the expected rewards of the optimal arm and a sub-optimal arm. The cubic dependence on  $K$  (the number of arms) in the former is due to the fact that any fair decision rule must experience constant per-step regret for  $T \gg K^3$  steps on some instances, i.e., the average per-step regret is  $\gg 1$  for  $T = \Omega(K^3)$ .

The idea of this *chaining strategy* can also be adapted to develop fair algorithms for more general scenarios such as contextual bandit problems [25] and bandits with different (or even infinite) number of arms at each time among which multiple arms can be selected [24]. Similar to constraint (18.1), fairness metrics in these generalized settings are also defined in terms of individual's expected qualification, and stipulate that two similar individuals with the same expected reward be treated similarly, even though their reward distributions can be significantly different.

In contrast, Liu et al. [33] proposes smooth fairness based on individuals' reward distributions rather than expected reward, which requires that individuals with similar reward distributions be selected with similar probabilities. Formally,  $\forall \epsilon_1, \epsilon_2 \geq 0$  and  $\forall \delta \in [0, 1]$ , the decision rule  $\tau = \{\tau_t\}_{t=1}^T$  is  $(\epsilon_1, \epsilon_2, \delta)$ -smooth fair w.r.t. a divergence function  $D$ , if  $\forall t$  and for any pair of arms  $k, k'$ , the following holds with probability at least  $1 - \delta$ :

$$D\left(\text{Ber}(\tau_t(k|h_t)) \middle\| \text{Ber}(\tau_t(k'|h_t))\right) \leq \epsilon_1 D\left(\mathbb{P}^k(\cdot; g^k) \middle\| \mathbb{P}^{k'}(\cdot; g^{k'})\right) + \epsilon_2, \quad (18.2)$$

where  $\text{Ber}(\tau_t(k|h_t))$  denotes a Bernoulli distribution with parameter  $\tau_t(k|h_t)$ .

Compared with meritocratic fairness, smooth fairness is weaker in the sense that it allows a worse arm to be selected with higher probability. To quantify such violation, [33] further proposes a concept of fairness regret, where a violation occurs when the arm with the highest reward realization at a given time is not selected with the highest probability. Based on this idea, the fairness regret of decision rule  $\tau_t$  at time  $t$  is defined as

$$R_t^{fair} = \mathbb{E}\left[\sum_{k=1}^K \max\left(\mathbb{P}^*(k) - \tau_t(k|h_t), 0\right) \middle| \{g^k\}_{k=1}^K\right],$$

and the cumulative fairness regret is defined as  $R_{1:T}^{fair} = \sum_{t=1}^T R_t^{fair}$ , where  $\mathbb{P}^*(k) = \mathbb{P}(k = \underset{k' \in \mathcal{Z}}{\text{argmax}} r_t^{k'})$  is the probability that the reward realization of arm  $k$  is the highest among all arms.

Two algorithms were developed in [33] for special types of bandit problems: (1) Bernoulli bandit, where the reward distributions satisfy  $\mathbb{P}^k(\cdot; g^k) = \text{Ber}(g^k)$  and (2) Dueling bandit:  $\forall t$  the decision maker selects two arms  $k_t^1, k_t^2$  and only observes the outcome  $\mathbf{1}(r_t^{k_t^1} > r_t^{k_t^2})$ . These algorithms satisfy smooth fairness w.r.t. total variation distance with low fairness regret.

In satisfying FA that similar individuals be treated similarly, one challenge is to define the appropriate context-dependent metric to quantify "similarity". Most studies in this space assume such a metric is given. Reference [14] proposes to learn such a similarity metric from the decision process itself. Specifically, it considers a linear contextual bandit problem where each arm corresponds to an unknown parameter  $\theta \in \mathbb{R}^d$ . At each time  $t$ , the decision-maker observes  $K$  arbitrarily and possibly adversarially selected contexts  $x_t^1, \dots, x_t^K \in \mathbb{R}^d$  from  $K$  arms, each representing features of an individual. It selects one (say arm  $i$ ) among them according to some

decision rule  $\tau_t$  and receives reward  $r_t^i$  with mean  $\mathbb{E}(r_t^i) = \langle x_t^i, \theta \rangle$ . Reference [14] focuses on individual fairness that individuals with similar contexts (features) be selected with similar probabilities, i.e.,  $|\tau_t(k|h_t) - \tau_t(k'|h_t)| \leq D(x_t^k, x_t^{k'})$ ,  $\forall k, k'$ , for some unknown metric  $D(\cdot, \cdot)$ . Similar to reference [33] and reference [14] also defines a fairness regret to quantify fairness violation over  $T$  time steps. Specifically, let  $R_t^{fair}(\Delta) = \sum_{i=1}^{K-1} \sum_{j=i+1}^K \mathbf{1}(|\tau_t(i|h_t) - \tau_t(j|h_t)| > D(x_t^i, x_t^j) + \Delta)$  be the total number of arm pairs violating  $\Delta$ -fairness and the total fairness regret over  $T$  steps is  $R_{1:T}^{fair}(\Delta) = \sum_{t=1}^T R_t^{fair}(\Delta)$ , where  $\Delta$  represents the error tolerance. The goal is to find a decision rule with low fairness regret that is also near-optimal (w.r.t. the best fair decision rule).

However, since  $D(\cdot, \cdot)$  is unknown, to achieve the above objective,  $D(\cdot, \cdot)$  also needs to be learned. To do so, it assumes that in addition to reward  $r_t^i$ , the decision-maker at each time receives feedback  $\{(k, k') : |\tau_t(k|h_t) - \tau_t(k'|h_t)| > D(x_t^k, x_t^{k'})\}$ , i.e., the set of all pairs of individuals for which the decision rule violates the fairness constraint. With such (weak) feedback, a computationally efficient algorithm is developed in [14] that for any metric  $D(\cdot, \cdot)$  following the form of Mahalanobis distance, i.e.,  $D(x_1, x_2) = \|Ax_1 - Ax_2\|_2$  for some matrix  $A$ , any time horizon  $T$  and any  $\Delta$ , with high probability it (i) obtains regret  $\tilde{O}(K^2 d^2 \log(T) + d\sqrt{T})$  w.r.t. the best fair decision rule; and (ii) violates unknown fairness constraints by more than  $\Delta$  on at most  $O(K^2 d^2 \log(d/\Delta))$  steps.

Other studies, such as [10, 30, 36] also use a bandit formulation with fairness consideration, where the fairness constraint requires either each arm be pulled for at least a certain fraction of the total available steps, or the selection rate of each arm be above a threshold. Algorithms that satisfy both (approximate) fairness and low regret are developed in these studies.

### 18.3.2 Fair Experts and Expert Opinions

In some sequential decision problems, decision-maker at each time may follow advice from multiple experts  $\mathcal{V} = \{v_k\}_{k=1}^K$  and at each  $t$  it selects expert according to a decision rule  $\tau_t$ , where  $\tau_t(k)$  denotes the probability of selecting expert  $v_k$  at time  $t$ . Blum et al. [6] considers a sequential setting where at each time a set of experts  $\mathcal{V}$  all make predictions about an individual (possibly based on sensitive attribute  $Z_t \in \{a, b\}$ ). Let  $Y_t \in \{0, 1\}$  be the individual's true label and expert  $v_k$ 's prediction be  $\hat{Y}_t^k$ , then the corresponding loss of expert  $v_k$  is measured as  $l(Y_t, \hat{Y}_t^k) \in [0, 1]$ . By following decision rule  $\tau_t$ , the decision-maker takes  $v_k$ 's advice with probability  $\tau_t(k)$ , and the overall expected loss at time  $t$  is given by  $\sum_{v_k \in \mathcal{V}} \tau_t(k) l(Y_t, \hat{Y}_t^k)$ . The decision-maker is assumed to observe  $l(Y_t, \hat{Y}_t^k) \in [0, 1], \forall v_k \in \mathcal{V} \text{ and } \forall t$ .

In [6], each expert in isolation is assumed to satisfy certain fairness criterion  $C \in \{\text{EqOpt}, \text{EO}, \text{EqLos}\}$  over a horizon. Specifically, given a sequence of individuals  $\{(y_t, z_t)\}_{t=1}^T$ , let  $\mathcal{T}_z^y = \{t | z_t = z, y_t = y\}$  be the set of time steps at which corresponding individuals are from  $G_z$  and have label  $y \in \{0, 1\}$ , expert  $v_k$  satisfies

$\text{EqOpt}$  if  $\frac{1}{|\mathcal{T}_a^1|} \sum_{t \in \mathcal{T}_a^1} l(y_t, \hat{y}_t^k) = \frac{1}{|\mathcal{T}_b^1|} \sum_{t \in \mathcal{T}_b^1} l(y_t, \hat{y}_t^k)$  holds. The decision-maker following  $\tau = \{\tau_t\}$  is said to be  $\Delta$ -fair w.r.t.  $\text{EqOpt}$  if the following holds,

$$\left| \mathbb{E}\left[\frac{1}{|\mathcal{T}_a^1|} \sum_{t \in \mathcal{T}_a^1} \sum_{v_k \in \mathcal{V}} \tau_t(k) l(Y_t, \hat{Y}_t^k)\right] - \mathbb{E}\left[\frac{1}{|\mathcal{T}_b^1|} \sum_{t \in \mathcal{T}_b^1} \sum_{v_k \in \mathcal{V}} \tau_t(k) l(Y_t, \hat{Y}_t^k)\right] \right| \leq \Delta .$$

Similar formula can be derived for the  $\text{EO}$  and  $\text{EqLos}$  criteria. The goal of the decision-maker is to find  $\Delta$ -fair  $\tau$  w.r.t.  $C$  from a set of fair experts that all satisfy fairness  $C$  in isolation, and at the same time perform as (approximate) good as the best expert in hindsight. Formally, define  $\epsilon$ -approximate regret of  $\tau$  over  $T$  time steps with respect to decision-maker  $v_i \in \mathcal{V}$  as follows:

$$\text{Regret}^T(\tau, v_i, \epsilon) = \sum_{t=1}^T \sum_{v_k \in \mathcal{V}} \tau_t(k) l(y_t, \hat{y}_t^k) - (1 + \epsilon) \sum_{t=1}^T l(y_t, \hat{y}_t^i) . \quad (18.3)$$

Then the goal is to achieve vanishing regret  $\mathbb{E}[\text{Regret}^T(\tau, v_i, \epsilon)] = o(T)$ ,  $\forall \epsilon > 0$  and  $\forall v_i \in \mathcal{V}$ .

When the input is i.i.d., the above setting is trivial because the best expert can be learned in  $O(\log |\mathcal{V}|)$  rounds and the decision-maker can follow its advice afterwards. Because each expert is fair in isolation, this also guarantees vanishing discrimination.

However, when input is non-i.i.d., achieving such objective is challenging. Reference [6] considers an adversarial setting where both  $Z_t$  and  $Y_t$  can be adaptively chosen over time according to  $\{(Z_s, Y_s, \hat{Y}_s)\}_{s=1}^{t-1}$ . It first examines the property of  $\text{EqOpt}$  and shows that given a set of experts that satisfies  $\text{EqOpt}$ , it is impossible to find a decision rule  $\tau$  with vanishing regret that can also preserve  $\Delta$ -fairness w.r.t.  $\text{EqOpt}$ . This negative result holds for both the cases when group identity information  $Z_t$  is used in determining  $\tau$  (group-aware) and the cases when the group information is not used (group-unaware). Specifically, for both cases, [6] constructs scenarios (about how an adversarial selects  $(Z_t, Y_t)$  over time) under which for any  $\Delta$  that is smaller than a constant  $c < 0.5$ ,  $\exists \epsilon > 0$  such that for any  $\tau$  that satisfies  $\mathbb{E}[\text{Regret}^T(\tau, v_i, \epsilon)] = o(T)$ ,  $\forall v_i \in \mathcal{V}$ , violates the  $\Delta$ -fairness w.r.t.  $\text{EqOpt}$ .

Since  $\text{EqOpt}$  is strictly weaker than  $\text{EO}$ , the above impossibility result in  $\text{EqOpt}$  naturally generalizes to  $\text{EO}$ . In contrast, under  $\text{EqLos}$ , given a set of experts that satisfies  $\text{EqLos}$  fairness,  $\forall \Delta > 0$ , there exists group-aware  $\tau$  that can simultaneously attain  $\Delta$ -fairness and the vanishing regret. The idea is to run two separate multiplicative weights algorithms for two groups. Because one property of the multiplicative weights algorithm is that it performs no worse than the best expert in hindsight but also no better. Therefore, the average performance of each group is approximately equal to the average performance attained by the best expert for that group. Because each expert is  $\text{EqLos}$  fair, the average performance attained by best experts of two groups are the same. Consequently, both vanishing regret and  $\Delta$ -fairness are satisfied. This positive result is due to the consistency between performance and fairness mea-

sure for EqLos. However, such positive result does not generally hold for EqLos. If only one multiplicative algorithm is performed without separating two groups, i.e., run in group-unaware manner, then it can be shown that  $\forall \epsilon > 0$  and  $\forall \Delta > 0$ , any algorithm satisfying vanishing regret also violates  $\Delta$ -fairness w.r.t. EqLos.

Valera et al. [38] studied a matching problem in a sequential framework, where a set of experts  $\mathcal{V}$  need to make predictions about  $m$  individuals from two demographic groups over  $T$  time steps, where at time step  $t$  individual  $i$ 's decision is made by expert  $v_i(t) \in \mathcal{V}$ . Different from [6] where experts are all fair (w.r.t. a particular metric) over a horizon and at each time only one expert's advice is followed on one individual, experts in [38] can be biased and at each time predictions from  $m$  decision-makers are all used and each is assigned to one individual. The algorithms for finding the optimal assignments are developed for cases with and without fairness intervention, which can improve both the overall accuracy and fairness as compared to random assignment, and fairness is guaranteed even when a significant percentage (e.g., 50%) of experts are biased against certain groups.

### 18.3.3 Fair Policing

Ensign et al. [12] studied a predictive policing problem, where the decision-maker at each time decides how to allocate patrol officers to different areas to detect crime based on historical crime incident data. The goal is to send officers to each area in numbers proportional to the true underlying crime rate of that area, i.e., areas with higher crime rate are allocated more officers. Reference [12] first characterizes the long-term property of existing predictive policing strategies (e.g., PredPol software), in which more officers are sent to areas with the higher predicted crime rates and the resulting incident data is fed back into the system. By modeling this problem using Pólya urn model, [12] shows that under such a method, one area can eventually consume all officers, even though the true crime rates may be similar across areas. This is because by allocating more officers to an area, more crimes are likely to be detected in that area; allocating more officers based on more detected crimes is thus not a proper method. To address this issue, effective approaches are proposed in [12], e.g., by intentionally normalizing the detected crime rates according to the rates at which police are sent.

## 18.4 (Fair) Sequential Decision When Decisions Affect Underlying Population

We next examine a second class of sequential decision problems (**P2**) where the decisions affect the underlying population; a list of these studied are summarized in Table 18.2. We will start with a set of papers that use a two-stage model, followed by a set of papers focusing on finite-horizon and infinite-horizon models.

**Table 18.2** Summary of related work when decisions affect underlying population.  $\star$  represents some other fairness notions or interventions that are not introduced in Sect. 18.2.2

	Fairness notion		Problem type
	Group fairness	Individual fairness	
Reference [31]	EqOpt, DP		<b>P2</b>
Reference [21]	$\star$		<b>P2</b>
Reference [27]	EqOpt, ...		<b>P2</b>
Reference [34]	DP		<b>P2</b>
Reference [32]	$\star$		<b>P2</b>
Reference [22]	DP, ...		<b>P2</b>
Reference [19]	$\star$		<b>P2</b>
Reference [44]	EqOpt, DP		<b>P2</b>
Reference [23]		MF	<b>P1</b>
Reference [39]	DP		<b>P1</b>

### 18.4.1 Two-Stage Models

To examine the long-term impact of fairness intervention on the underlying population, some studies [21, 27, 31] construct two-stage models, whereby the first stage decisions (under certain fairness criterion) are imposed on individuals from two demographic groups  $G_a, G_b$ , which may cause individuals to take certain actions, and the overall impact of this one-step intervention on the entire group is then examined in the second stage.

Let  $\alpha_k$  be the size of  $G_k$  as the fraction of the entire population and  $\alpha_a + \alpha_b = 1$ . Reference [31] focuses on a one-dimensional setting where an individual from either group has feature  $X \in \mathcal{X}$  with  $\mathcal{X} = \{1, 2, \dots, M\}$  and sensitive attribute  $Z \in \{a, b\}$  representing his/her group membership. Let  $\pi(x|k) = \mathbb{P}(X = x|Z = k)$ ,  $x \in \mathcal{X}$  be  $G_k$ 's feature distribution and  $Y \in \{0, 1\}$  the individual's true label. The decision-maker makes predictions on individuals using the decision rule  $\tau(x, k) = \mathbb{P}(\hat{Y} = 1|X = x, Z = k)$  and receives expected utility  $u(x)$  for making a positive prediction  $\hat{Y} = 1$  of an individual with feature  $x$  (e.g., average profit of a lender by issuing a loan to applicants whose credit score is 760). The expected utility of the decision-maker under  $\tau$  is given by

$$U(\tau) = \sum_{k \in \{a, b\}} \alpha_k \sum_{x \in \mathcal{X}} u(x) \tau(x, k) \pi(x|k).$$

Define the *selection rate* of  $G_k$  under a decision rule as  $\gamma(k) = \mathbb{P}(\hat{Y} = 1|Z = k) = \sum_{x \in \mathcal{X}} \tau(x, k) \pi(x|k)$ . Then given feature distributions, the relationship between  $\gamma(k)$  and  $\tau(\cdot, k)$  can be described by an invertible mapping  $g(\cdot)$  so that  $\gamma(k) = g(\tau(\cdot, k); \pi(\cdot, k))$  and  $\tau(\cdot, k) = g^{-1}(\gamma(k); \pi(\cdot, k))$ .

In [31], decision rules for  $G_a, G_b$  are selected such that  $U(\tau)$  is maximized under fairness constraints defined as follows:

- Simple: it requires the same decision rule be used by  $G_a, G_b$ , i.e.,  $\tau(\cdot, a) = \tau(\cdot, b)$ .
- Demographic Parity (DP): it requires the selection rates of  $G_a, G_b$  are equalized, i.e.,  $\gamma(a) = \gamma(b)$ .
- Equal of Opportunity (EqOpt): it requires the true positive rate (TPR) of  $G_a, G_b$  are equalized, i.e.,  $\mathbb{P}(\hat{Y} = 1|Y = 1, Z = a) = \mathbb{P}(\hat{Y} = 1|Y = 1, Z = b)$ .

Once an individual with feature  $X = x$  is predicted as positive ( $\hat{Y} = 1$ ) in the first stage, its feature may be affected; denote the average of such change as  $\Delta(x)$ . For example, consider a lending scenario where a lender decides whether or not to issue loans to applicants based on their credit scores. Among applicants who are issued loans, those with the higher (resp. lower) credit score are more likely to repay (resp. default); as a result, the credit scores may increase for applicants who can repay the loans ( $\Delta(x) > 0$ ) but decrease for those who default ( $\Delta(x) < 0$ ). Consequently, the feature distribution of the entire group can be skewed. Let the impact of a decision rule  $\tau(x, k)$  on  $G_k$  be captured by the average change of  $X$  in  $G_k$ , defined as  $\Delta\mu(\tau, k) = \sum_{x \in \mathcal{X}} \tau(x, z)\pi(x|k)\Delta(x)$ . It can be shown that  $\Delta\mu(\tau, k)$  is a concave function in the selection rate  $\gamma(k)$ .

Let the optimal fair decision rule that maximizes  $U(\tau)$  under fairness criterion  $C \in \{\text{Simple}, \text{DP}, \text{EqOpt}\}$  be noted as  $\tau^C$ , and the corresponding selection rate be noted as  $\gamma^C$ . Let group labels  $a, b$  be assigned such that  $G_b$  is the disadvantaged group in the sense that  $\gamma^{\text{Simple}}(a) > \gamma^{\text{Simple}}(b)$ . Given  $\Delta\mu(\tau, b)$ , a decision rule  $\tau$  causes

- active harm to  $G_b$  if  $\Delta\mu(\tau, b) < 0$ ;
- relative harm if  $\Delta\mu(\tau, b) < \Delta\mu(\tau^{\text{Simple}}, b)$ ;
- relative improvement if  $\Delta\mu(\tau, b) > \Delta\mu(\tau^{\text{Simple}}, b)$ .

Due to the one-to-one mapping between the decision rule and the selection rate, the notation  $\Delta\mu(\tau, k) = \Delta\mu(g^{-1}(\gamma(k); \pi(\cdot, k)), k)$  in the following is simplified as  $\Delta\tilde{\mu}(\gamma(k), k)$ . Let  $\gamma_0(b)$  be the *harmful* threshold for  $G_b$  such that  $\Delta\tilde{\mu}(\gamma_0(b), b) = 0$ ; let  $\gamma^*(b)$  be the *max-improvement* threshold such that  $\gamma^*(b) = \operatorname{argmax}_\gamma \Delta\tilde{\mu}(\gamma, b)$ ; let  $\bar{\gamma}(b)$  be the *complementary* threshold such that  $\Delta\tilde{\mu}(\bar{\gamma}(b), b) = \Delta\tilde{\mu}(\gamma^{\text{Simple}}(b), b)$  and  $\gamma^{\text{Simple}}(b) < \bar{\gamma}(b)$ .

The goal of [31] is to understand the impact of imposing DP or EqOpt fairness constraint on  $\Delta\mu(\tau, k)$ , whether these fairness interventions can really benefit the disadvantaged group  $G_b$  as compared to the Simple decision rule.

Reference [31] first examined the impact of Simple decision rule, and showed that if  $u(x) > 0 \implies \Delta(x) > 0$ , then Simple threshold does not cause active harm, i.e.,  $\Delta\mu(\tau^{\text{Simple}}, b) \geq 0$ . In lending example, the condition  $u(x) > 0 \implies \Delta(x) > 0$  means that the lender takes a greater risk by issuing a loan to an applicant than the applicant does by applying.

For DP and EqOpt fairness, [31] showed that both could cause relative improvement, relative harm and active harm, under different conditions. We summarize these results below, for  $C \in \{\text{DP}, \text{EqOpt}\}$ ,

1. Under certain conditions, there exists  $\alpha_0 < \alpha_1 < 1$  such that  $\forall \alpha_b \in [\alpha_0, \alpha_1]$ ,  $\tau^C$  causes relatively improvement, i.e.,  $\gamma^{\text{Simple}}(b) < \gamma^C(b) < \bar{\gamma}(b)$ .
2. Under certain conditions, positive predictions can be over-assigned to  $G_b$  for satisfying  $C$ . There exists  $\alpha_0$  such that  $\forall \alpha_b \in [0, \alpha_0]$ ,  $\tau^C$  causes relatively harm or active harm, i.e.,  $\gamma^C > \bar{\gamma}(b)$  or  $\gamma^C > \gamma_0(b)$ .

These results show that although it seems fair to impose DP and EqOpt constraints on decisions (e.g., by issuing more loans to the disadvantaged group), it may have unintended consequences and harm the disadvantaged group (e.g., features of disadvantaged group may deteriorate after being selected).

Reference [31] makes further comparisons between DP and EqOpt fairness. Generally speaking, DP and EqOpt cannot be compared in terms of  $\Delta\mu(\tau, b)$ . Because there exist both settings when DP causes harm while EqOpt causes improvement, and settings when EqOpt causes harm while DP causes improvement. However, for some special cases when  $\pi(\cdot|a)$  and  $\pi(\cdot|b)$  satisfy a specific condition, there exists  $\alpha_0, \alpha_1$  such that  $\forall \alpha_b \in [\alpha_0, \alpha_1]$ , DP causes active harm while EqOpt causes improvement. Moreover, if under Simple decision rule,  $\gamma^{\text{Simple}}(a) > \gamma^{\text{Simple}}(b)$  and  $\mathbb{P}(\hat{Y} = 1|Y = 1, Z = b) > \mathbb{P}(\hat{Y} = 1|Y = 1, Z = a)$  hold, then  $\gamma^{\text{EqOpt}}(b) < \gamma^{\text{Simple}}(b) < \gamma^{\text{DP}}(b)$  can be satisfied, i.e., EqOpt can cause relative harm by selecting less than Simple rule.

An interested reader is referred to [31] for details of the specific conditions mentioned above. It shows that temporal modeling and a good understanding of how individuals react to decisions are necessary to accurately evaluate the impact of different fairness criteria on the population.

#### 18.4.1.1 Effort-Based Fairness

Essentially, the issues of unfairness described in the preceding section may come from the fact that different demographic groups have different feature distributions, leading to different treatments. However, this difference in feature distributions is not necessary because one group is inherently inferior to another; rather, it may be the result of the fact that advantaged group can achieve better features/outcomes with less effort. For example, if changing one's school type from public to private can improve one's SAT score, then such change would require much higher effort for the low-income population. From this point of view, Heidari et al. [21] proposes an effort-based notion of fairness, which measures unfairness as the disparity in the average effort individuals from each group have to exert to obtain a desirable outcome.

Consider a decision-maker who makes a prediction about an individual using decision rule  $h(\cdot)$  based on its  $d$ -dimensional feature vector  $X \in \mathcal{X}$ . Let  $Y \in \mathcal{Y}$  be the individual's true label,  $Z \in \{a, b\}$  its sensitive attribute, and  $\hat{Y} = h(X)$  the predicted

label. Define a benefit function  $w(h(X), Y) \in \mathbb{R}$  that quantifies the benefit received by an individual with feature  $X$  and label  $Y$  if he/she is predicted as  $h(X)$ .

For an individual from  $G_k$  who changes his/her data from  $(x, y)$  to  $(x', y')$ , the total effort it needs to take is measured as  $E_k((x, y), (x', y')) = \frac{1}{d} \sum_{i=1}^d e_k^i(x_i, x'_i)$ , where  $x = (x_1, \dots, x_d)$ ,  $x' = (x'_1, \dots, x'_d)$  and  $e_k^i(x_i, x'_i)$  denote the effort needed for an individual from  $G_k$  to change its  $i$ th feature from  $x_i$  to  $x'_i$ . Accordingly, the change in the individual's benefit by making such an effort is  $\Delta w((x, y), (x', y')) = w(h(x), y) - w(h(x'), y')$ , and the total utility received by an individual from  $G_k$  in changing his/her data is

$$U_k((x, y), (x', y')) = \Delta w((x, y), (x', y')) - E_k((x, y), (x', y')) .$$

Define  $\widehat{U}_k = \mathbb{E}[\max_{(x', y') \in \mathcal{X} \times \mathcal{Y}} U_k((x, y), (x', y')) | Z = k]$  as the expected highest utility  $G_k$  can possibly reach by exerting effort. Reference [21] suggests the use of the disparity between  $\widehat{U}_a$  and  $\widehat{U}_b$  as a measure of group unfairness.

The microscopic impact of decisions on each individual can be modeled using the above unfairness measure. Intuitively, if individuals can observe the behaviors of others similar to them, then they would have more incentive to imitate behaviors of those (social models) who receive higher benefit, as long as in doing so individuals receive positive utility.

Let  $\mathcal{D}_k$  be the training dataset representing samples of population in  $G_k$ . Then  $(x^*, y^*) = \operatorname{argmax}_{(x', y') \in \mathcal{D}_k} U_k((x, y), (x', y'))$  can be regarded as a social model's profile that an individual  $(x, y)$  from  $G_k$  aims to achieve, as long as  $U_k((x, y), (x^*, y^*)) > 0$ . Given the change of each individual in  $\mathcal{D}_k$ , a new dataset  $\mathcal{D}'_k$  in the next time step can be constructed accordingly.

Given  $\mathcal{D}_k, \mathcal{D}'_k$ , the datasets before and after imposing decisions according to  $h(\cdot)$ , the macroscopic impact of decisions on the overall underlying population can be quantified. Reference [21] adopts the concept of segregation from sociology to measure the degree to which multiple groups are separate from each other. Specifically, the segregation of  $\mathcal{D}_k$  and  $\mathcal{D}'_k$  are compared from three perspectives: *Evenness*, *Clustering* and *Centralization*. The details of each can be found in [21]; here we only introduce *Centralization* as an example: this is measured as the proportion of individuals from a minority group whose prediction  $h(X)$  is above the average. The impact of decisions on the entire group is examined empirically by comparing *Evenness*, *Clustering*, and *Centralization* of  $\mathcal{D}_k$  and  $\mathcal{D}'_k$ .

Reference [21] first trained various models  $h(\cdot)$  such as neural network, linear regressor, and decision tree over a real-world dataset without imposing a fairness constraint. It shows that individuals by imitating social model's data profile can either increase or decrease the segregation of the overall population, and different models may shift the segregation toward different directions. Next, [21] examined the impact of imposing fairness constraint on a linear regression model. Specifically, the fairness constraint requires each group's average utility be above the same threshold, a higher threshold indicating a stronger fairness requirement. Empirical results show that segregation under different levels of fairness can change in completely

different directions (decrease or increase), and impacts on *Evenness*, *Centralization*, and *Clustering* are also different.

Indeed, fairness intervention affects segregation in two competing ways. If more desirable outcomes are assigned to a disadvantaged group intentionally, then on one hand individuals from the disadvantaged group may have less motivation to change their features, on the other hand, the same individuals may serve as social models, which in turn can incentivize others from the same disadvantaged group to change their features. Both impacts are at play simultaneously and which one is dominant depends on the specific context. This paper highlights the fact that modifying decision algorithm is not the only way to address segregation and unfairness issues; imposing mechanisms before individuals enter the decision system may be another effective way, e.g., by decreasing the costs for individuals from the disadvantaged group to change their features.

#### 18.4.1.2 A Two-Stage Model in College Admissions

Kannan et al. [27] studied a two-stage model in the case of college admissions and hiring. In the first stage, students from two demographic groups are admitted to a college based on their entrance exam scores; in the second stage an employer chooses to hire students from those who were admitted to the college based on their college grades. Specifically, let  $Z \in \{a, b\}$  denote a student's group membership and  $Y \sim \mathcal{N}(\mu_k, \sigma_k^2)$ ,  $k \in \{a, b\}$  his/her qualification drawn from a group-specific Gaussian distribution. Let variable  $X = Y + \nu$  be the student's entrance exam score with independent noise  $\nu \sim \mathcal{N}(0, 1)$ ,  $\forall k \in \{a, b\}$ .

Denote by  $\hat{Y}^c \in \{0, 1\}$  the college's admissions decision about a student. Let  $\tau^c(x, k) = \mathbb{P}(\hat{Y}^c = 1 | X = x, Z = k) \in [0, 1]$  be the admissions rule representing the probability a student from  $G_k$  with score  $x$  gets admitted, which is monotone non-decreasing in  $x$  for  $k \in \{a, b\}$ . Consider a threshold decision rule of the following form:

$$\tau^c(x, k) = \begin{cases} 1, & \text{if } x \geq \theta_k \\ 0, & \text{if } x < \theta_k. \end{cases} \quad (18.4)$$

For a student who is admitted, he/she receives a grade  $G = Y + \mu$  with  $\mu \sim \mathcal{N}(0, \sigma_c^2)$ ,  $\forall k \in \{a, b\}$ , where the variance  $\sigma_c^2 > 0$  is determined by some grading rule. Specifically,  $\sigma_c^2 \rightarrow \infty$  can be regarded as a case where students' grades are not revealed to the employer, whereas  $\sigma_c^2 \rightarrow 0$  represents a case where the employer has perfect knowledge of the students' qualifications. The employer decides whether or not to hire a student based on his/her grade. Let  $c \in [c^-, c^+]$  be the cost for the employer for hiring a student, which can either be known or unknown to the college. Then a student from  $G_k$  with grade  $g$  gets hired if the employer can achieve a non-negative expected utility, i.e.,  $\mathbb{E}[Y|G = g, \hat{Y}^c = 1, Z = k] \geq c$ .

The goal of [27] is to understand what admission rules and grading rules should be adopted by the college in the first stage so that the following fairness goals may be attained in the second stage:

- Equal of Opportunity (EqOpt): it requires the probability of a student being hired by the employer conditional on the qualification  $Y$  is independent of group membership  $Z$ .
- Irrelevance of Group Membership (IGM): it requires the employer's hiring decision, conditional on  $\hat{Y}^c$  and  $G$ , should be independent of group membership, i.e.,  $\forall g \in \mathbb{R}, \mathbb{E}[Y|G = g, \hat{Y}^c = 1, Z = a] \geq c \iff \mathbb{E}[Y|G = g, \hat{Y}^c = 1, Z = b] \geq c$ .
- Strong Irrelevance of Group Membership (sIGM): it requires the employer's posterior about students' qualifications, conditional on  $\hat{Y}^c$  and  $G$ , should be independent of group membership, i.e.,  $\forall g \in \mathbb{R}$  and  $\forall y \in \mathbb{R}, \mathbb{P}[Y = y|G = g, \hat{Y}^c = 1, Z = a] = \mathbb{P}[Y = y|G = g, \hat{Y}^c = 1, Z = b]$ .

Below, we present two simple scenarios found in [27], in which both EqOpt and IGM can be satisfied in the second phase under some admission rules.

### 1. Noiseless entrance exam score, i.e., $X = Y$ .

In this scenario, the admission decision is determined by the student's qualification  $Y$  completely. Reference [27] shows that as long as the threshold in the admission decision rule is set as  $\theta_k = c^+, \forall k \in \{a, b\}$  in Eq. (18.4), then  $\forall [c^-, c^+] \subset \mathbb{R}$  and with any grading rule, both EqOpt and IGM can be satisfied.

### 2. No grade is revealed to the employer, i.e., $\sigma_c^2 \rightarrow \infty$ .

In this case, as long as the threshold in the admission decision rule is set as  $\theta_a = \theta_b = \theta$  for some sufficiently large  $\theta$  (e.g., highly selective MBA programs) in Eq. (18.4), then  $\forall [c^-, c^+] \subset \mathbb{R}$ , both EqOpt and IGM can be satisfied.

Reference [27] also studied more general scenarios when noises  $\mu$  and  $\nu$  are both of finite variance, i.e., noisy entrance exam scores and when colleges report informative grades to the employer. When employer's hiring cost  $c$  is known to the college,  $\forall c \in \mathbb{R}$ , there always exist two thresholds  $\theta_a^*, \theta_b^*$  and a grade  $g^*$  for college, under which  $\mathbb{E}[Y|G = g^*, X \geq \theta_a^*, Z = a] = \mathbb{E}[Y|G = g^*, X \geq \theta_b^*, Z = b] = c$  always holds, i.e., IGM can always be satisfied.

However, if we consider the employer's posterior distributions on students' qualification, as long as two groups have different prior distributions, for any two thresholds  $\theta_a, \theta_b$  in the admission rule, there always exists  $y$  such that  $\mathbb{P}[Y = y|G = g, X \geq \theta_a, Z = a] \neq \mathbb{P}[Y = y|G = g, X \geq \theta_b, Z = b]$ , i.e., satisfying sIGM is impossible.

Moreover, suppose prior distributions of two groups' qualifications are Gaussian distributed with different mean but the same variance, then  $\forall c$ , there exists no threshold decision rule  $\tau^c$  that can satisfy both EqOpt and IGM simultaneously. For EqOpt under some fixed hiring cost  $c$ , in cases when grading rule has variance  $\sigma_c^2 \neq 1$ , there is no threshold decision rule  $\tau^c$  such that EqOpt can be satisfied. For cases when  $\sigma_c^2 = 1$ , EqOpt can be satisfied only if the admission rule and grading rule can satisfy  $\mathbb{E}[Y|G = \theta_b, X \geq \theta_a, Z = a] = \mathbb{E}[Y|G = \theta_a, X \geq \theta_b, Z = b] = c$ . Such condition is generally impossible to hold. It concludes that EqOpt is generally impossible to achieve.

If employer's hiring cost  $c$  is uncertain that college only knows the interval  $[c^-, c^+]$ , when two groups have different priors, [27] shows that  $\forall c \in [c^-, c^+]$ , neither EqOpt nor IGM can be satisfied even in isolation under a threshold admission rule.

The above results show that even with a simple model studied in [27], many common and natural fairness goals are impossible to achieve in general. Such negative results are likely to hold true in more complex models that capture more realistic aspects of the problem.

### 18.4.2 Long-Term Impacts on the Underlying Population

Decisions made about humans affect their actions. Bias in decisions can induce certain behavior, which is then captured in the dataset used to develop decision algorithms in the future. The work [21, 27, 31] introduced in the previous section studied such one-step impact of decisions on the population. However, when newly developed algorithms are then used to make decisions about humans in the future, those humans will be affected and biases in the datasets generated by humans can perpetuate. This closed feedback loop becomes self-reinforcing and can lead to highly undesirable outcomes over time. In this section, we focus on the long-term impacts of decisions on population groups. The goal is to understand what happens to the underlying population when decisions and people interact with each other and what interventions are effective in sustaining equality in the long run.

#### 18.4.2.1 Effects of Decisions on the Evolution of Features

One reason why decisions are made in favor of one group is that the favored group is believed to bring more benefit to the decision-maker. For example, a lender issues more loans to a group believed to be more likely to repay, a company hires more from a group perceived to be more qualified, and so on. In other words, disparate treatment received by different groups is due to the disparity in their (perceived) abilities to produce good outcomes (qualifications). From this perspective, the ultimate social equality is attained when different demographic groups possess the same abilities/qualifications. In this section, we present studies reported in [22, 32, 34] to understand how qualifications of different groups evolve over time under various fairness interventions, and under what conditions social equality may be attained.

Let  $G_a, G_b$  be two demographic groups,  $\alpha_k$  the size of  $G_k$  as a fraction of the entire population and assumed constant, and  $\alpha_a + \alpha_b = 1$ . Each individual has feature  $X$ , sensitive attribute  $Z \in \{a, b\}$ , and label  $Y \in \{0, 1\}$  representing his/her qualification or the ability to produce certain good outcome. Define the *qualification profile* of  $G_k$  at time  $t$  as the probability distribution  $\pi_t(y|k) = \mathbb{P}_t(Y = y|Z = k)$ ,  $y \in \{0, 1\}$ . Changes in feature  $X$  induced by decisions are captured by change in the qualification profile.

Using the definition of qualification profiles of two groups, social equality can be defined formally as equalized qualification profiles, i.e.,

$$\lim_{t \rightarrow \infty} |\pi_t(1|a) - \pi_t(1|b)| = 0. \quad (18.5)$$

Reference [32, 34] assume that the qualification profiles at each time are known to the decision-maker, who makes prediction about each individual according to a decision rule  $\tau_t(y, k) = \mathbb{P}_t(\hat{Y} = 1|Y = y, Z = k)$  and receives utility  $u(y)$  for making positive prediction  $\hat{Y} = 1$ , where  $u(0) \leq 0$  and  $u(1) \geq 0$  correspond to the loss and benefit, respectively. Define the *selection rate* of  $G_k$  under a decision rule at time  $t$  as  $\gamma_t(k) = \mathbb{P}_t(\hat{Y} = 1|Z = k) = \sum_{y \in \{0,1\}} \gamma_t(y|k) = \sum_{y \in \{0,1\}} \mathbb{P}_t(\hat{Y} = 1|Y = y, Z = k) \mathbb{P}_t(Y = y|Z = k)$ . Then the expected utility of the decision-maker at  $t$  is

$$\begin{aligned} U_t(\tau_t) &= \sum_{k \in \{a,b\}} \alpha_k \sum_{y \in \{0,1\}} u(y) \mathbb{P}_t(\hat{Y} = 1|Y = y, Z = k) \mathbb{P}_t(Y = y|Z = k) \\ &= \sum_{k \in \{a,b\}} \alpha_k \sum_{y \in \{0,1\}} u(y) \tau_t(y, k) \pi_t(y|k). \end{aligned} \quad (18.6)$$

Upon receiving a decision, a qualified individual can either remain qualified or become unqualified, and an unqualified individual can either become qualified or remain unqualified for the next time step. In [34], the evolution of a group's qualification profile is modeled as a dynamical system as follows:

$$\pi_{t+1}(1|k) = \pi_t(1|k)v(\gamma_t(0|k), \gamma_t(1|k)) + \pi_t(0|k)\mu(\gamma_t(0|k), \gamma_t(1|k)), \quad (18.7)$$

where  $v(\cdot, \cdot) : [0, 1] \times [0, 1] \rightarrow [0, 1]$  represents the retention rate of subgroup who are qualified ( $Y = 1$ ) in time  $t$  that are still qualified in  $t + 1$ , while  $\mu(\cdot, \cdot) : [0, 1] \times [0, 1] \rightarrow [0, 1]$  represents the improvement rate of subgroup who are unqualified ( $Y = 0$ ) at time  $t$  but make progress to be qualified ( $Y = 1$ ) at time  $t + 1$ . Due to the mapping between the decision rule and the selection rate, the impact of decisions on individuals' future qualifications are captured by the impact of selection rates on the overall qualification profiles via some general functions  $v(\cdot, \cdot)$  and  $\mu(\cdot, \cdot)$  in model (18.7).

The goal of the decision-maker is to find a decision rule  $\tau_t$  with or without fairness consideration, so as to maximize  $U_t(\tau_t)$ . It examines what happens to the qualification profiles of two groups when these decisions are applied at each time, and under what conditions social equality is attained under these decisions.

Without fairness considerations, the corresponding optimal decision at time  $t$  for  $G_k, k \in \{a, b\}$ , is given by<sup>2</sup>:

---

<sup>2</sup>Note that such an ideal decision rule assumes the knowledge of  $y$ , which is not actually observable. In this sense this decision rule, which has 0 error, is not practically feasible. Our understanding is that the goal in [34] is to analyze what happens in such an ideal scenario when applying the perfect decision.

$$\tau_t^*(y, k) = \operatorname{argmax}_{\tau_t} U_t(\tau_t) = \begin{cases} 0, & \text{if } y = 0 \\ 1, & \text{if } y = 1 \end{cases}. \quad (18.8)$$

Using this decision rule, the selection rate  $\gamma_t(k) = \pi_t(1|k)$ . Since the decision rules for the two groups are not constrained by each other, the dynamics (18.7) can be simplified as follows:  $\forall k \in \{a, b\}$ ,

$$\pi_{t+1}(1|k) = \Phi(\pi_t(1|k)) \text{ with } \Phi(\pi) = \pi v(0, \pi) + (1 - \pi)\mu(0, \pi). \quad (18.9)$$

Social equality can be attained for any starting profiles  $\pi_0(1|a), \pi_0(1|b)$  in this unconstrained case if and only if the system  $\pi_{t+1} = \Phi(\pi_t)$  has a unique globally attracting equilibrium point  $\pi^*$  and a sufficient condition is given in [34].

Reference [34] also studied impact of fairness intervention on dynamics. It focuses on the notion of demographic parity (DP), which requires the selection rates of two groups to be equal, i.e.,  $\gamma_t(a) = \gamma_t(b), \forall t$ . Depending on group proportions  $\alpha_a, \alpha_b$  and utilities  $u(1), u(0)$ , there are two possibilities for the fair optimal decision rule  $\tau_t^*$ . If group labels  $a, b$  are assigned such that  $G_a$  is the advantaged group, i.e.,  $\pi_t(1|a) \geq \pi_t(1|b)$ , then we have

$$\text{if } \alpha_a u(1) + \alpha_b u(0) \leq 0 : \tau_t^*(0, a) = \tau_t^*(0, b) = 0 \quad (18.10)$$

$$\text{(under-selected)} \quad \tau_t^*(1, a) = \frac{\pi_t(1|b)}{\pi_t(1|a)}; \tau_t^*(1, b) = 1 \quad (18.11)$$

$$\text{if } \alpha_a u(1) + \alpha_b u(0) \geq 0 : \tau_t^*(0, a) = 0; \tau_t^*(0, b) = \frac{\pi_t(1|a) - \pi_t(1|b)}{1 - \pi_t(1|b)} \quad (18.12)$$

$$\text{(over-selected)} \quad \tau_t^*(1, a) = \tau_t^*(1, b) = 1. \quad (18.13)$$

To guarantee equalized selection rates, Eqs. (18.10), (18.11) show the case where  $G_a$  is under-selected, while Eqs. (18.12), (18.13) show the case where  $G_b$  is over-selected. Dynamics (18.7) can then be expressed as follows:

$$\begin{aligned} \pi_{t+1}(1|a) &= \Phi_a(\pi_t(1|a), \pi_t(1|b)) \\ \pi_{t+1}(1|b) &= \Phi_b(\pi_t(1|a), \pi_t(1|b)). \end{aligned}$$

Similar to the unconstrained case, sufficient conditions for reaching social equality in these case can also be derived and are given in [34].

By comparing these sufficient conditions, [34] shows that unconstrained optimal decision rules may reach social equality on its own in some cases. However, if DP fair decisions are used instead in these special cases, then the equality may be violated. Specifically, if disadvantaged group  $G_b$  is over selected, social equality may or may not be attained by using DP fair decisions. Moreover, for settings where equality can be attained under both types of decisions, [34] further shows that DP fair decisions may lead to the higher total utility as well as the more qualified population in the long run. In contrast, if advantaged group  $G_a$  is under-selected, social equality will

definitely be attained by using DP fair decisions. However, imposing this additional fairness constraint may decrease the decision-maker's utility and the population's overall qualification level.

Liu et al. [32] also studied a similar problem on the evolution of qualification profiles of different demographic groups. In their setting, decisions applied to each group can incentivize each individual to rationally invest in his/her qualifications, as long as the expected reward received from the decision-maker's prediction outweighs the investment cost.

Formally, the impact of decisions on the underlying population is captured by *individual's best response*. Let random variable  $c_k$  be the cost incurred by an individual from  $G_k$  in order to obtain  $Y = 1$  (be qualified). Let cumulative distribution function (CDF) of  $c_k$  be denoted as  $\mathbb{F}_k(\cdot)$ . For any individual, regardless of the group membership  $Z$  and actual qualification  $Y$ , he/she receives a reward  $w > 0$  only if he/she is predicted as positive (qualified)  $\hat{Y} = 1$ . Therefore, an individual from  $G_k$  at  $t$  acquires qualification  $Y = 1$  if and only if the resulting utility of investing outweighs the utility of not investing, i.e.,

$$\underbrace{w\mathbb{P}_t(\hat{Y} = 1|Y = 1, Z = k) - c_k}_{\text{utility if investing}} - \underbrace{w\mathbb{P}_t(\hat{Y} = 1|Y = 0, Z = k)}_{\text{utility if not investing}} \\ = w(\tau_t(1, k) - \tau_t(0, k)) - c_k > 0. \quad (18.14)$$

Note that the qualification status  $Y$  of each individual depends completely on whether he/she invests: given decision rule  $\tau_t$ , individuals become qualified as long as Eq. (18.14) is satisfied. The overall qualification profile of  $G_k$  is the probability of individuals being qualified, i.e.,  $\mathbb{P}(Y = 1|Z = k)$ , or equivalently, the probability of investment cost being sufficiently small (Eq. (18.14)). Therefore, the update of qualification profile of  $G_k$  at  $t + 1$  can be captured by the CDF of cost variable  $c_k$  according to the following:

$$\pi_{t+1}(1|k) = \mathbb{P}(c_k < w(\tau_t(1, k) - \tau_t(0, k))) = \mathbb{F}_k(w(\tau_t(1, k) - \tau_t(0, k))). \quad (18.15)$$

Consider the decision rule that maximizes the decision-maker's utility as given in Eq. (18.6) at each time, i.e.,  $\tau_t(y, k) = \operatorname{argmax}_{\tau} U_t(\tau)$ . Then the ideal (though infeasible) decision is the same as in Eq. (18.8) and is given by the following,<sup>3</sup>  $\forall k \in \{a, b\}$ ,

$$\tau_t(y, k) = \operatorname{argmax}_{\tau} U_t(\tau) = \begin{cases} 0, & \text{if } y = 0 \\ 1, & \text{if } y = 1. \end{cases} \quad (18.16)$$

---

<sup>3</sup>In [32] the assumption that such a perfect decision rule with 0 error is feasible is formally stated as “realizability”.

Given initial qualification profiles  $\pi_0(1|a)$  and  $\pi_0(1|b)$ ,  $\pi_t(1|k)$  can be captured by a dynamic system  $\pi_{t+1}(1|k) = \Phi(\pi_t(1|k))$  for some  $\Phi(\cdot)$ . We first present the results in [32] under the assumption that CDF of cost variables for two groups are the same, i.e.,  $\mathbb{F}_a(\cdot) = \mathbb{F}_b(\cdot) = \mathbb{F}(\cdot)$ .

If the perfect decision rule shown in Eq. (18.16) is feasible, then this dynamic system has a unique non-zero equilibrium  $\pi^*$  and the corresponding qualification profile  $\pi^*(1|k) = \lim_{t \rightarrow \infty} \Phi^t(\pi_0(1|k)) = \mathbb{F}(w)$  is also the optimal for  $G_k$ .<sup>4</sup> However, since this ideal decision is not generally feasible in practice, the evolution of equilibria for more realistic cases is further examined in [32]. Let prediction  $\hat{Y}$  be calculated from features  $X$  via a mapping  $h(\cdot) : X \rightarrow \{0, 1\}$ . Reference [32] focused on two special cases: (1) uniformly distributed  $X$ ; (2) spherical multivariate Gaussian distributed  $X$ . For both cases, every group  $G_k$  in isolation can be perfectly predicted by some mapping  $h_k(\cdot)$  but when both groups are combined, such perfect mapping does not exist. Reference [32] shows that for both cases, under certain conditions, decision-maker by applying  $h_k(\cdot)$  to both groups at each time can result in a stable equilibrium at which  $\pi^*(1|k) = \mathbb{F}(w) > \pi^*(1|\{a, b\} \setminus k)$ , i.e., the qualification profile of  $G_k$  is optimal, decision is always in favor of  $G_k$  and social equality is violated. Although there exists a unique decision rule  $\hat{h}(\cdot)$  for both cases, following which at each time can result in an equilibrium satisfying  $\pi^*(1|a) = \pi^*(1|b)$  (social equality), such equilibrium is unfortunately shown to be unstable.

Both cases show that as long as the initial decision rule is not  $\hat{h}(\cdot)$ , equilibria of the dynamic system can be in favor of one group and biased against the other, social equality cannot be attained. Above results hold under the case when CDF of cost variables for two groups are the same, i.e.,  $\mathbb{F}_a(\cdot) = \mathbb{F}_b(\cdot)$ . If we remove this assumption and let  $G_b$  be disadvantaged in the sense that its investment cost is sufficiently higher than  $G_a$ , then [32] shows that there is no stable equilibrium that is in favor of  $G_b$  and no equilibrium can result in social equality. This conclusion, although is negative, suggests an effective intervention that can potentially improve the qualification profile of disadvantaged group at the equilibrium: by subsidizing the cost of investment for disadvantaged group.

Another effective intervention proposed in [32] is by decoupling the decision rules by group, i.e., each group is predicted by its own group-specific decision rule instead of sharing the same decision rule for all groups. In this case, different from (18.16), at each time  $t$  decision-maker chooses two decision rules for two groups,

$$\forall k \in \{a, b\} : \tau_t(y, k) = \operatorname{argmax}_{\tau} \sum_{y \in \{0, 1\}} u(y) \tau(y, k) \pi_t(y|k)$$

and qualification profile of  $G_k$  at  $t + 1$  is updated in the same way as (18.15). Under this new dynamic system, [32] shows that  $\forall k \in \{a, b\}$ , if there exists a perfect decision rule for  $G_k$  such that  $\tau_t(1, k) = 1$  and  $\tau_t(0, k) = 0$ , then the resulting unique equilibrium  $\pi^*$  is stable and satisfies  $\pi^*(1|k) = \mathbb{F}(w)$ , i.e., both groups have the optimal qualification profiles. If there is no perfect decision rule for both groups, i.e.,

---

<sup>4</sup> $\Phi^t$  is a  $t$ -fold composition of  $\Phi$ .

$\max_{\tau} \sum_{k \in \{a, b\}} \alpha_k (\tau(1, k) - \tau(0, k)) < 1$ , then we can still guarantee that at least one group's qualification profile can be strictly improved at an equilibrium as compared to the case when both groups use the same decision rule.

### 18.4.2.2 Fairness Intervention on Labor Market

Hu and Chen [22] studied the impact of fairness intervention on labor market. In their setting, each individual from either group is a worker. All workers pass through a sequence of two markets: a temporary labor market (TLM) and a permanent labor market (PLM). Inside the labor market, workers who get hired by an employer will produce an outcome that is either “good” or “bad”.  $\forall k \in \{a, b\}$ , how well can group  $G_k$  perform in general at time  $t$  is measured by the group's *reputation*, defined as the proportion of all workers in  $G_k$  (including those who are not employed) who can produce “good” outcomes in the labor market over the time interval  $[t - t_0, t]$ , noted as  $\pi_t^k$ . Within this context, social equality introduced in Eq. (18.5) earlier is re-defined: it is attained if the group reputation is equalized, i.e.,

$$\lim_{t \rightarrow \infty} |\pi_t^a - \pi_t^b| = 0 .$$

Reference [22] shows that social equality can be attained by imposing short-term fairness intervention in the TLM. Below, we take a closer look at this dual labor market model.

In order to compete for a certain job in the future, worker  $i$  from  $G_k$  at time  $t$  may choose to make education investment  $\eta_i \geq 0$  based on expected wage  $w_t$  of the job and its personal cost in the investment  $c_{\pi_t^k}(\mu_i, \eta_i)$ . The cost depends on two factors:

- Worker's ability  $\mu_i$ : it is an intrinsic attribute of workers with CDF  $\mathbb{F}_\mu(\cdot)$ , which is identical to both groups.
- Reputation of the group ( $\pi_t^k$ ) that worker  $i$  belongs to; workers from a group with better reputation face better cost conditions.

Let variable  $\rho \in \{Q, U\}$  denote a worker's qualification status and the probability of a worker being qualified for the job is  $\gamma(\eta_i) \in [0, 1]$  where  $\gamma(\cdot)$  is a monotonic increasing function. Whether or not a worker can be hired in the TLM is determined by the worker's investment and his/her group membership via a mapping  $\tau_{TLM}(\eta_i, k) \in \{0, 1\}$ . If  $\tau_{TLM}(\eta_i, k) = 1$ , then worker  $i$  that is hired in the TLM is eligible to enter the PLM.<sup>5</sup> Specifically, worker  $i$  keeps the same job in the TLM until a Poisson process selects him/her to enter the PLM. Upon entering the PLM, at each time he/she cycles through jobs.

In order to be hired in the PLM, workers build their own personal reputation  $\Gamma^s$  by consistently exerting efforts  $E$  and producing outcomes  $O$  in labor markets. Specifically, workers can exert either high ( $H$ ) or low ( $L$ ) effort with cost  $e_\rho(\mu_i)$  or

---

<sup>5</sup>  $\tau_{TLM} = 1$  only ensures a worker's eligibility to be hired in the PLM (a necessary condition); whether the worker is indeed hired in the PLM is determined by the hiring strategy in the PLM.

0, and produce either good ( $G$ ) or bad ( $B$ ) outcome. Denote  $p_H$  as the probability a worker producing good outcome with high effort and  $p_\rho$  as the probability a worker producing good outcome with low effort and qualification status  $\rho$ . Worker  $i$ 's personal reputation  $\Gamma_i^s \in [0, 1]$  is the proportion that he/she produces good outcomes during the recent length- $s$  history in the labor market, which determines whether or not he/she can be hired at each time in the PLM via a mapping  $\tau_{PLM}(\Gamma_i^s) \in \{0, 1\}$ .

Group  $G_k$ 's reputation  $\pi_{t'}^k$  at time  $t'$ , which determines the worker's cost in education before entering the TLM, and will also be updated based on the outcomes produced by all workers from  $G_k$  during time lag  $[t' - t_0, t']$ . Moreover, the expected wage of the job  $w_t$  that determines workers' investments before entering the TLM is also updated in a Poisson manner based on  $g_t$ , the proportion of workers that are hired in the labor market producing good outcomes in their jobs at  $t' < t$ . The above form a feedback loop between the labor market and workers.

Reference [22] studied the long-term impact of imposing fairness constraints in determining  $\tau_{TLM}$ . Specifically, it compares hiring strategies in the TLM under three constraints:

- Demographic Parity (DP): among workers hired in the TLM, a  $\alpha_k$  fraction of them are from  $G_k$ .
- Simple: both groups are subject to the same hiring strategy, i.e.,  $\tau_{TLM}(\cdot, a) = \tau_{TLM}(\cdot, b)$ .
- Statistical Discrimination (SD): this is a hiring strategy based on the firm's belief of worker qualifications, e.g.,  $\mathbb{P}(\rho = Q|Z = k, \eta) = \frac{p_Q(\eta)\rho_k^p}{p_Q(\eta)\rho_k^p + p_U(\eta)(1-\rho_k^p)}$ , where  $\rho_k^p$  denotes the prior of  $G_k$ 's capabilities, and  $p_Q(\eta)$ ,  $p_U(\eta)$  denote the probabilities of a qualified and unqualified worker investing  $\eta$ , respectively.

Reference [22] analyzed the optimal hiring strategies of firms in the TLM and PLM, as well as the workers' optimal effort/investment strategy; it also examined the group reputation  $(\pi_t^a, \pi_t^b)$  over time when DP fairness intervention is imposed in the TLM. They show that there exists a unique stable equilibrium and  $T$  such that  $\pi_t^a = \pi_t^b, \forall t > T$ , i.e., short-term fairness intervention in the TLM can result in two groups gradually approaching the same reputation level and achieving social equality. Without fairness intervention, workers from the group with better reputation are more likely to invest in education (which is cheaper), enter the PLM and produce good outcomes, which further improves their group reputation. With the DP constraint, the hiring thresholds take into account the differences in costs of investment, and the fractions of workers from two groups that enter PLM are maintained at  $\alpha_a, \alpha_b$ . As a result, the proportions of workers producing good outcomes do not diverge and social equality can be reached.

In contrast, under either the Simple or SD hiring strategy in the TLM, the two groups will not be proportionally represented in the labor market according to  $\alpha_a, \alpha_b$  as they have different costs in investment. Their group reputations will diverge eventually and cannot reach social equality.

### 18.4.2.3 Effects of Decisions on Group Representation

Decision algorithms developed from multiple demographic groups can inherit representation disparity that may exist in the data: the algorithm may be less favorable to groups contributing less to the training process; this in turn can degrade population retention in these groups over time, and exacerbate representation disparity in the long run. Hashimoto et al. [19] are among the first to show that the (unconstrained) empirical risk minimization (ERM) formulation, which is widely used in training machine learning models, can amplify group representation disparity over time.

Consider two demographic groups  $G_a, G_b$ . An individual from either group has feature  $X \in \mathcal{X}$  and label  $Y \in \mathcal{Y}$ . Let  $f_a(x, y)$  and  $f_b(x, y)$  be the joint distributions of  $(X, Y)$  for individuals in  $G_a$  and  $G_b$ , respectively. At each time  $t$ , a decision-maker receives data  $\mathcal{D}_t$  from a set of individuals. Specifically,  $\forall k \in \{a, b\}$ , let  $N_k(t)$  be the expected number of individuals in  $\mathcal{D}_t$  that are from  $G_k$  and  $\alpha_k(t) = \frac{N_k(t)}{N_a(t) + N_b(t)}$  is how much  $G_k$  is represented in the data. Then the overall feature distribution of the entire population at  $t$  is given by  $f_t(x, y) = \alpha_a(t)f_a(x, y) + \alpha_b(t)f_b(x, y)$ . Denote  $\alpha(t) = [\alpha_a(t); \alpha_b(t)]$ .

Let  $h_\theta : \mathcal{X} \rightarrow \mathcal{Y}$  be a decision rule for predicting label from features, which is parameterized by some parameter  $\theta \in \mathbb{R}^d$ . Let  $l(h_\theta(X), Y)$  be the prediction loss incurred by predicting  $(X, Y)$  using  $h_\theta(\cdot)$  where  $l(\cdot, \cdot)$  is the loss function measuring the discrepancy between predictions and true labels. The goal of the decision-maker at time  $t$  is to find a  $\theta(t)$  for both groups such that the overall prediction loss is minimized:

$$\theta(t) = \theta(\alpha(t)) = \operatorname{argmin}_{\theta} L(\theta) = \mathbb{E}_{(X, Y) \sim f_t(x, y)}[l(h_\theta(X), Y)]. \quad (18.17)$$

Individuals after receiving their predictions may choose to either leave the decision system or stay. For those who experience low accuracy, they have a higher probability of leaving the system. As a result, the impact of decisions on the overall group representation can be captured by a discrete-time user retention model:

$$\begin{aligned} N_k(t+1) &= \Phi(N_k(t)) = N_k(t) \cdot v(L_k(\theta(\alpha(t)))) + \beta_k \\ \alpha_k(t+1) &= \frac{N_k(t+1)}{N_a(t+1) + N_b(t+1)}, \end{aligned} \quad (18.18)$$

where  $L_k(\theta(\alpha(t))) = \mathbb{E}_{(X, Y) \sim f_k(x, y)}[l(h_{\theta(t)}(X), Y)]$  is the expected loss experienced by  $G_k$  from decision  $\theta(t)$ , retention rate  $v(\cdot) \in [0, 1]$  represents the probability of an individual who was in system at  $t$  remaining in the system at  $t+1$ , and  $\beta_k$  is the number of new users from  $G_k$ .

Under the systems given in Eqs. (18.17), (18.18), [19] first finds the condition under which a fixed point of the system is unstable; the representation disparity under such unstable systems will be amplified over time.

To prevent one group from diminishing, or, to ensure  $\alpha_k(t) > \alpha_{\min}, \forall t$ , for some  $\alpha_{\min}$ , instead of minimizing the overall prediction loss, [19] suggests bounding the

worst-case group loss  $L_{\max}(\theta(\alpha(t))) = \max\{L_a(\theta(\alpha(t))), L_b(\theta(\alpha(t)))\}$ ,  $\forall t$ . This can be challenging as the true sensitive attribute  $Z$  of each data point is unknown to the decision-maker. To address this, a distributionally robust optimization (DRO) is formulated in [19]. Instead of controlling  $L_{\max}(\theta(\alpha(t)))$  directly, it controls an upper bound on it. Specifically, it considers the worst-case loss among all perturbed distributions  $\tilde{f}_r(x, y)$  that are within a chi-squared ball  $\mathcal{B}(f(x, y), r)$  around real distribution  $f(x, y)$ . Let  $\mathcal{B}(f(x, y), r) = \{\tilde{f}_r(x, y) | D_{\chi^2}(f || \tilde{f}_r) \leq r\}$ , where  $D_{\chi^2}(f || \tilde{f}_r)$  is  $\chi^2$ -divergence between distributions  $f(x, y)$  and  $\tilde{f}_r(x, y)$ , then  $\forall \theta$  and  $f_t(x, y)$ , loss experienced by  $G_k$  can be upper bounded by

$$L_{dro}(\theta; r_k) = \sup_{\tilde{f}_r(x, y) \in \mathcal{B}(f_t(x, y), r_k)} \mathbb{E}_{(X, Y) \sim \tilde{f}_r(x, y)} [l(h_\theta(X), Y)] \geq \mathbb{E}_{(X, Y) \sim f_k(x, y)} [l(h_\theta(X), Y)]$$

with robustness radius  $r_k = (1/\alpha_k(t) - 1)^2$ . Consequently,  $L_{\max}(\theta(\alpha(t)))$  can be controlled by choosing

$$\theta(\alpha(t)) = \operatorname{argmin}_\theta L_{dro}(\theta; r_{\max}) \quad (18.19)$$

with  $r_{\max} = (1/\min\{\alpha_a(t), \alpha_b(t)\} - 1)^2$ .

Suppose  $\forall k \in \{a, b\}$ , the initial states satisfy  $\alpha_k(1) > \alpha_{\min}$ . Using the above method, [19] shows that  $\alpha_k(t) > \alpha_{\min}$ ,  $\forall t$ , can be guaranteed for the entire horizon under the following condition:

$$L_{dro}(\theta(\alpha(t)); r_{\max}) \leq v^{-1} \left( 1 - \frac{(1 - v_{\max})\beta_k}{\alpha_{\min}(\beta_a + \beta_b)} \right),$$

where  $v_{\max} = \max\{v(L_a(\theta(t))), v(L_b(\theta(t)))\}$ . While the above condition is hard to verify in practice, experiments in [19] show that the decisions selected according to the DRO formulation (18.19) result in stronger stability of group representation than that selected by ERM formulation (18.17).

Reference [19] shows that the group representation disparity can worsen over time when no fairness is imposed when making a decision. In contrast, Zhang et al. [44] show that it can worsen even when fairness criteria are imposed. They consider a similar sequential framework where at each time  $t$  two (potentially different) decision rules  $h_{\theta_a(t)}(\cdot)$ ,  $h_{\theta_b(t)}(\cdot)$  are applied to  $G_a$ ,  $G_b$  and parameters  $\theta_a(t)$ ,  $\theta_b(t)$  are selected to optimize an objective, subject to certain fairness criterion  $C$ :

$$\operatorname{argmin}_{(\theta_a, \theta_b)} \mathbf{O}_t(\theta_a, \theta_b; \alpha_a(t), \alpha_b(t)) = \alpha_a(t) O_{a,t}(\theta_a) + \alpha_b(t) O_{b,t}(\theta_b) \quad (18.20)$$

$$\text{s.t. } \Gamma_{C,t}(\theta_a, \theta_b) = 0.$$

Note that the overall objective at time  $t$  consists of sub-objectives from two groups weighted by their group proportions at  $t$ , and empirical risk minimization (18.17) studied in [19] is a special case of (18.20), with  $\theta_a = \theta_b$  and  $O_{k,t}(\theta_k) = L_k(\theta)$  being

$G_k$ 's empirical loss  $\forall t$ . Similar to [19], group representation is affected by decisions according to a user retention model and are updated over time,

$$\begin{aligned} N_k(t+1) &= N_k(t) \cdot \pi_{k,t}(\theta_k(t)) + \beta_k \\ \alpha_k(t+1) &= \frac{N_k(t+1)}{N_a(t+1) + N_b(t+1)}. \end{aligned} \quad (18.21)$$

As compared to (18.18), the retention rate  $\pi_{k,t}(\theta_k(t))$  of  $G_k$  can be any function that depends on the decision, which means the analysis and conclusions obtained in [44] are not limited to applications where user retention is driven by model accuracy (e.g., speech recognition, medical diagnosis); instead they are more generally applicable (e.g., in lending/hiring, user retention is more likely to be driven by positive classification rate rather than the expected loss.)

The goal of [44] is to characterize long-term property of group representation disparity  $\frac{\alpha_a(t)}{\alpha_b(t)}$ , and understand what is the impact of imposing various fairness constraints in this process. It turns out that even with fairness intervention, group representation disparity can still change monotonically and one group may diminish over time from the system. Specifically, given a sequence of one-shot problems  $\{\mathcal{O}_t(\theta_a, \theta_b; \alpha_a(t), \alpha_b(t))\}_{t=1}^T$ , if  $\forall t$ ,  $\mathcal{O}_t$  is defined over the same sub-objectives  $O_a(\theta_a), O_b(\theta_b)$  with different group proportions  $(\alpha_a(t), \alpha_b(t))$ , and the dynamics satisfy  $\pi_{k,t}(\theta_k) = h_k(O_k(\theta_k))$  for some decreasing function  $h_k(\cdot)$ , i.e., user departure is driven by the value of sub-objective function, then the group representation disparity  $\frac{\alpha_a(t)}{\alpha_b(t)}$  changes monotonically over time and the discrepancy between  $\pi_{a,t}(\theta_a(t))$  and  $\pi_{b,t}(\theta_b(t))$  increases over time. Intuitively, whenever one group's proportion (e.g.,  $\alpha_a(t)$ ) starts to increase, the decision-maker in minimizing the overall objective would select a decision pair such that  $O_a(\theta_a(t))$  decreases. Consequently,  $G_a$ 's retention as determined by  $h_a(O_a(\theta_a(t)))$  increases, i.e.,  $G_a$ 's proportion increases further and representation disparity worsens.

This condition that leads to exacerbating representation disparity can be easily satisfied under commonly used objectives (e.g., minimizing overall expected loss), common fairness constraints (e.g., EqOpt, DP, etc.), and various dynamics (e.g., user participation driven by model accuracy or intra-group disparity); an interested reader is referred to [44] for more details. It highlights the fact that common fairness interventions fail to preserve representation parity. This is ultimately because what are being equalized by those fairness criteria often do not match what drives user retention; thus applying seemingly fair decisions may worsen the situation. A main takeaway is that fairness must be defined with a good understanding of the underlying user retention model, which can be challenging in practice as we typically have only incomplete/imperfect information. However, if user dynamics model is available, [44] presents the following method for finding a proper fairness criterion that mitigates representation disparity.

Consider a general dynamics model  $N_k(t+1) = \Phi(N_k(t), \{\pi_k^m(\theta_k(t))\}_{m=1}^M, \beta_k)$ ,  $\forall k \in \{a, b\}$ , where user departures and arrivals are driven by  $M$  different factors  $\{\pi_k^m(\theta_k(t))\}_{m=1}^M$  (e.g., accuracy, false positive rate, positive rate, etc.). Let  $\Theta$  be the

set of all possible decisions, if there exists a pair of decisions  $(\theta_a, \theta_b) \in \Theta \times \Theta$  under which dynamics have stable fixed points, then a set  $C$  of decisions  $(\theta_a, \theta_b)$  that can sustain group representation can be found via an optimization problem:

$$\begin{aligned} C = \operatorname{argmin}_{(\theta_a, \theta_b)} & \left| \frac{\tilde{N}_a}{\tilde{N}_b} - \frac{\beta_a}{\beta_b} \right| \\ \text{s.t. } & \tilde{N}_k = \Phi(\tilde{N}_k, \{\pi_k^m(\theta_k)\}_{m=1}^M, \beta_k) \in \mathbb{R}_+, \theta_k \in \Theta, \forall k \in \{a, b\}. \end{aligned}$$

The idea is to first select decision pairs whose corresponding dynamics can lead to stable fixed points  $(\tilde{N}_a, \tilde{N}_b)$ ; we can then select among them those that are best in sustaining group representation.

#### 18.4.2.4 Combined Effects on Group Representation and Features

In practice, decisions can simultaneously impact both group representation and the evolution of features, (potentially) making a bad situation worse. Consider the lending example where a lender decides whether or not to approve a loan application based on the applicant's credit score. It has been shown in [31] that decisions under either EqOpt or DP can potentially lead to over issuance of loans to the less qualified (disadvantaged) group. As a result, the disadvantaged group's score distribution will skew toward higher default risk. Over time, more people from this group may stop applying for loans. The increased disproportionality between the two groups will then lead the lender to actually issue more loans (relatively) to the less qualified group to satisfy EqOpt or DP fairness, leading its score distribution to skew more toward higher default risk over time.

Reference [44] studies the combination of these two effects on the underlying population, i.e., the effect on group representation and the effect on how features evolve. Specifically, they consider the case where feature distributions  $f_{k,t}(x, y)$  are allowed to change over time, and try to understand what happens to group representation disparity  $\frac{\alpha_a(t)}{\alpha_b(t)}$  when  $f_{k,t}(x, y)$  are also affected by decisions.

Let  $f_{k,t}(x, y) = g_{k,t}^0 f_{k,t}^0(x) + g_{k,t}^1 f_{k,t}^1(x)$  be  $G_k$ 's feature distribution at  $t$ , where  $g_{k,t}^j = \mathbb{P}(Y = j | Z = k)$  and  $f_{k,t}^j(x) = \mathbb{P}(X = x | Y = j, Z = k)$  at  $t$ . Let  $G_k^j$  be the subgroup of  $G_k$  with label  $Y = j$ . Based on the facts that individuals from the same demographic group with different labels may react differently to the same decision rule, [44] considered two scenarios of how feature distributions are reshaped by decisions: (1)  $\forall k \in \{a, b\}$ ,  $f_{k,t}^j(x) = f_k^j(x)$  remain fixed but  $g_{k,t}^j$  changes over time according to  $G_k^j$ 's own perceived loss and (2)  $\forall k \in \{a, b\}$ ,  $g_{k,t}^j = g_k^j$  remain fixed but for subgroup  $G_k^j$  that is less favored by decision over time (experience an increased loss), its members make extra effort such that  $f_{k,t}^j(x)$  skews toward the direction of lowering their losses. In both cases, [44] shows that representation disparity can worsen over time under common fairness intervention and such exacerbation accelerates as compared to the case when feature distributions are fixed.

### 18.4.2.5 Fairness in Reinforcement Learning Problems

Studies in [23, 39] capture the interaction between decisions and the underlying population via a reinforcement learning framework, where the environment is described by a Markov decision process (MDP), defined by a tuple  $(\mathcal{S}, \mathcal{A}, \mathbb{P}, R, \gamma)$ .  $\mathcal{S}$  is the set of states representing certain properties of individuals in the system and  $\mathcal{A}$  the set of actions representing available decisions. At time  $t$ , the decision-maker by taking action  $a_t \in \mathcal{A}$  in state  $s_t \in \mathcal{S}$  receives a reward  $r_t = R(s_t, a_t) \in [0, 1]$ . The probability of the decision-maker being in state  $s_{t+1}$  at time  $t + 1$  is given by the transition probability matrix  $\mathbb{P}(s_{t+1}|a_t, s_t)$ ; this is what captures the impact of decisions on the underlying population. Reference [23] generalizes the bandits problem studied in [24, 25] to the above reinforcement learning framework, by taking into account the effects of decisions on the individuals' future states and future rewards. It slightly modifies the meritocratic fairness defined in [25] based on long-term rewards: a decision is preferentially selected over another only if the long-term reward of the former is higher than the latter. Under such a fairness constraint, an algorithm is proposed that can achieve near-optimality within  $T_0$  time steps. The impact of fairness is reflected in  $T_0$ : it takes more time steps to learn a near-optimal decision rule when the fairness requirement is stricter.

Reference [39] studied a reinforcement learning problem under group fairness (DP) constraint, where the state  $s_t = (x_t, z_t)$  consists of both the feature  $x_t$  and the sensitive attribute  $z_t \in \{a, b\}$  of the individual who is subject to the decision-maker's decision at  $t$ . When action  $a_t$  is taken in state  $s_t$ , in addition to reward  $R(s_t, a_t)$  received by the decision-maker, the individual also receives a reward  $\rho(s_t, a_t)$ . The DP constraint in [39] requires that the expected (discounted) cumulative reward of individuals from the two groups to be approximately equal. Algorithms (model-free and model-based) are developed in [39] for learning a decision rule that is both DP-fair and near-optimal.

**Acknowledgements** This work is supported by the NSF under grants CNS-1616575, CNS-1646019, CNS-1739517, CNS-2040800, and by the ARO under contract W911NF1810208.

## References

1. Agarwal, A., Beygelzimer, A., Dudik, M., Langford, J., Wallach, H.: A reductions approach to fair classification. In: International Conference on Machine Learning, pp. 60–69 (2018)
2. Aneja, A.P., Avenancio-León, C.F.: No credit for time served? incarceration and credit-driven crime cycles (2019)
3. Auer, P., Cesa-Bianchi, N., Fischer, P.: Finite-time analysis of the multiarmed bandit problem. *Mach. Learn.* **47**(2–3), 235–256 (2002)
4. Bechavod, Y., Ligett, K., Roth, A., Waggoner, B., Steven, Z.: Equal opportunity in online classification with partial feedback. *Adv. Neural Inf. Process. Syst.* **32**, 8972–8982 (2019)
5. Berk, R., Heidari, H., Jabbari, S., Joseph, M., Kearns, M., Morgenstern, J., Neel, S., Roth, A.: A convex framework for fair regression (2017). [arXiv:1706.02409](https://arxiv.org/abs/1706.02409)

6. Blum, A., Gunasekar, S., Lykouris, T., Srebro, N.: On preserving non-discrimination when combining expert advice. In: Advances in Neural Information Processing Systems, pp. 8376–8387 (2018)
7. Bolukbasi, T., Chang, K.-W., Zou, J.Y., Saligrama, V., Kalai, A.T.: Man is to computer programmer as woman is to homemaker? debiasing word embeddings. Advances in Neural Information Processing Systems **29**, 4349–4357 (2016)
8. Calders, T., Žliobaitė, I.: Why unbiased computational processes can lead to discriminative decision procedures. In: Discrimination and Privacy in the Information Society, pp. 43–57. Springer (2013)
9. Chaney, A.J.B., Stewart, B.M., Engelhardt, B.E.: How algorithmic confounding in recommendation systems increases homogeneity and decreases utility. In: Proceedings of the 12th ACM Conference on Recommender Systems, pp. 224–232. ACM (2018)
10. Chen, Y., Cuellar, A., Luo, H., Modi, J., Nemlekar, H., Nikolaidis, S.: Fair contextual multi-armed bandits: Theory and experiments (2019). [arXiv:1912.08055](https://arxiv.org/abs/1912.08055), 2019
11. Dressel, J., Farid, H.: The accuracy, fairness, and limits of predicting recidivism. Sci. Adv. **4**(1), eaao5580 (2018)
12. Ensign, D., Friedler, S.A., Neville, S., Scheidegger, C., Venkatasubramanian, S.: Runaway feedback loops in predictive policing. In: Conference of Fairness, Accountability, and Transparency (2018)
13. Fuster, A., Goldsmith-Pinkham, P., Ramadorai, T., Walther, A.: Predictably unequal? the effects of machine learning on credit markets. The Effects of Machine Learning on Credit Markets (2018)
14. Gillen, S., Jung, C., Kearns, M., Roth, A.: Online learning with an unknown fairness metric. In: Advances in Neural Information Processing Systems, pp. 2600–2609 (2018)
15. Gordaliza, P., Del Barrio, E., Fabrice, G., Jean-Michel, L.: Obtaining fairness using optimal transport theory. In: International Conference on Machine Learning, pp. 2357–2365 (2019)
16. Gupta, S., Kamble, V.: Individual fairness in hindsight. In: Proceedings of the 2019 ACM Conference on Economics and Computation, pp. 805–806. ACM (2019)
17. Hardt, M., Price, E., Srebro, N. et al.: Equality of opportunity in supervised learning. In: Advances in neural information processing systems, pp. 3315–3323 (2016)
18. Harwell, D.: Amazon’s alexa and google home show accent bias, with Chinese and Spanish hardest to understand (2018). <http://bit.ly/2QFA1MR>
19. Hashimoto, T., Srivastava, M., Namkoong, H., Liang, P.: Fairness without demographics in repeated loss minimization. In: Dy, J., Krause, A. (eds.), Proceedings of the 35th International Conference on Machine Learning, Proceedings of Machine Learning Research, vol. 80, pp. 1929–1938. PMLR (2018)
20. Heidari, H., Krause, A.: Preventing disparate treatment in sequential decision making. In: Proceedings of the 27th International Joint Conference on Artificial Intelligence, pp. 2248–2254 (2018)
21. Heidari, H., Nanda, V., Gummadi, K.: On the long-term impact of algorithmic decision policies: effort unfairness and feature segregation through social learning. In: International Conference on Machine Learning, pp. 2692–2701 (2019)
22. Hu, L., Chen, Y.: A short-term intervention for long-term fairness in the labor market. In: Proceedings of the 2018 World Wide Web Conference on World Wide Web, pp. 1389–1398. International World Wide Web Conferences Steering Committee (2018)
23. Jabbari, S., Joseph, M., Kearns, M., Morgenstern, J., Roth, A.: Fairness in reinforcement learning. In: Proceedings of the 34th International Conference on Machine Learning-Volume 70, pp. 1617–1626. JMLR. org, (2017)
24. Joseph, M., Kearns, M., Morgenstern, J., Neel, S., Roth, A.: Meritocratic fairness for infinite and contextual bandits. In: Proceedings of the 2018 AAAI/ACM Conference on AI, Ethics, and Society, pp. 158–163. ACM, (2018)
25. Joseph, M., Kearns, M., Morgenstern, J.H., Roth, A.: Fairness in learning: classic and contextual bandits. In: Advances in Neural Information Processing Systems, pp. 325–333 (2016)

26. Kamiran, F., Calders, T.: Data preprocessing techniques for classification without discrimination. *Knowl. Inf. Syst.* **33**(1), 1–33 (2012)
27. Kannan, S., Roth, A., Ziani, J.: Downstream effects of affirmative action. In: Proceedings of the Conference on Fairness, Accountability, and Transparency, pp. 240–248. ACM (2019)
28. Kleinberg, J., Mullainathan, S., Raghavan, M.: Inherent trade-offs in the fair determination of risk scores. In 8th Innovations in Theoretical Computer Science Conference (ITCS 2017). Schloss Dagstuhl-Leibniz-Zentrum fuer Informatik (2017)
29. Lahoti, P., Gummadi, K.P., Weikum, G.: ifair: learning individually fair data representations for algorithmic decision making. In: 2019 IEEE 35th International Conference on Data Engineering (ICDE), pp. 1334–1345. IEEE, (2019)
30. Li, F., Liu, J., Ji, B.: Combinatorial sleeping bandits with fairness constraints. In: IEEE INFOCOM 2019-IEEE Conference on Computer Communications, pp. 1702–1710. IEEE (2019)
31. Liu, L.T., Dean, S., Rolf, E., Simchowitz, M., Hardt, M.: Delayed impact of fair machine learning. In: Dy, J., Krause, A. (eds.), Proceedings of the 35th International Conference on Machine Learning, Proceedings of Machine Learning Research, vol. 80, pp. 3150–3158. PMLR (2018)
32. Liu, L.T., Wilson, A., Haghtalab, N., Tauman Kalai, A., Borgs, C., Jennifer Chayes. The disparate equilibria of algorithmic decision making when individuals invest rationally. *arXiv preprint arXiv:1910.04123*, 2019
33. Liu, Y., Radanovic, G., Dimitrakakis, C., Mandal, D., Parkes, D.C.: Calibrated fairness in bandits (2017). *arXiv:1707.01875*
34. Mehrabi, N., Morstatter, F., Saxena, N., Lerman, K., Galstyan, A.: A survey on bias and fairness in machine learning (2019). *arXiv:1908.09635*
35. Obermeyer, Z., Mullainathan, S.: Dissecting racial bias in an algorithm that guides health decisions for 70 million people. In: *Proceedings of the Conference on Fairness, Accountability, and Transparency*, p. 89. ACM, 2019
36. Patil, V., Ghalme, G., Nair, V., Narahari, Y.: Achieving fairness in the stochastic multi-armed bandit problem (2019). *arXiv:1907.10516*
37. Shankar, S., Halpern, Y., Breck, E., Atwood, J., Wilson, J., Sculley, D.: No classification without representation: Assessing geodiversity issues in open data sets for the developing world. *stat* **1050**, 22 (2017)
38. Valera, I., Singla, A., Gomez Rodriguez, M.: Enhancing the accuracy and fairness of human decision making. In: Advances in Neural Information Processing Systems 31, pp. 1774–1783. Curran Associates, Inc. (2018)
39. Wen, M., Bastani, O., Topeu, U.: Fairness with dynamics (2019). *arXiv:1901.08568*
40. Zafar, M.B., Valera, I., Gomez Rodriguez, M., Gummadi, K.P.: Fairness beyond disparate treatment & disparate impact: lclassification without disparate mistreatment. In: Proceedings of the 26th International Conference on World Wide Web, pp. 1171–1180. International World Wide Web Conferences Steering Committee (2017)
41. Zafar, M.B., Valera, I., Gomez-Rodriguez, M., Gummadi, K.P.: Fairness constraints: a flexible approach for fair classification. *J. Mach. Learn. Res.* **20**(75), 1–42 (2019)
42. Zemel, R., Wu, Y., Swersky, K., Pitassi, T., Dwork, C.: Learning fair representations. In: International Conference on Machine Learning, pp. 325–333 (2013)
43. Zhang, X., Mahdi Khalili, M., Liu, M.: Long-term impacts of fair machine learning. *Ergonom, Des* (2019)
44. Zhang, X., Mahdi Khalili, M., Tekin, C., Liu, M.: Group retention when using machine learning in sequential decision making: the interplay between user dynamics and fairness. *Adv. Neural Inf. Process. Syst.* **32**, 15243–15252 (2019)

## Chapter 19

# Trading Utility and Uncertainty: Applying the Value of Information to Resolve the Exploration–Exploitation Dilemma in Reinforcement Learning



Isaac J. Sledge and José C. Príncipe

**Abstract** A fundamental problem in reinforcement learning is the exploration–exploitation dilemma: a search problem that entails sufficiently investigating the possible action choices and exploiting those that work well for certain contexts. Few exploration mechanisms, however, provide expected performance guarantees for a given search amount. Here, we show that this dilemma can be addressed and the expected agent performance quantified by optimizing Stratonovich’s value of information. The value of information is an information-theoretic criterion that specifies the greatest increase in rewards, from the worst case, subject to a certain uncertainty amount. In the context of reinforcement learning, uncertainty is quantified by a constrained mutual dependence between random variables. When the mutual dependence between the random variables go to zero, agents tend to exploit its acquired knowledge about the environment; little to no improvements in policy performance are obtained in this case. As the mutual dependence increases, a great amount of exploration is permitted and the policy can converge to the global-best action-selection strategy. Optimizing the value of information yields action-selection update strategies that, in the limit, is theoretically guaranteed to uncover the optimal policy for a given mutual dependence amount. We show that, in a finite number of episodes, the value of information yields policies that outperform conventional exploration mechanisms for both single-state and multi-state, multi-action environment abstractions based on Markov decision processes.

---

José C. Príncipe—is the director of the Computational NeuroEngineering Laboratory (CNEL) at the University of Florida.

---

I. J. Sledge (✉)

Department of Electrical and Computer Engineering, University of Florida, 32611 Gainesville, FL, USA

Computational NeuroEngineering Laboratory (CNEL), University of Florida, Gainesville, FL, USA

e-mail: [isledge@ufl.edu](mailto:isledge@ufl.edu)

J. C. Príncipe

Department of Electrical and Computer Engineering and Department of Biomedical Engineering, University of Florida, 32611 Gainesville, FL, USA

e-mail: [principe@cnel.ufl.edu](mailto:principe@cnel.ufl.edu)

## 19.1 Introduction

Information theory has been dominated by Claude Shannon's contributions. The early theories that he advanced involved the use of quantitative measures on the amount of information for a given problem. These measures, like entropy, are dependent only on the probabilities of events associated with the problem; they consider no additional contextual details that may exist to relate information with the reality of the embodiment where the observations are being collected.

There are certain application domains for which knowing only event probabilities is insufficient. One example is agent decision-making, which can take the form of determining what advertisements to show on a website or even specifying how an articulated robot should maneuver through a cluttered environment. For these applications, no theory that just involves the use of outcome probabilities, without considering the consequences of those outcomes, could possibly be adequate for describing the uncertainty importance to the decision-maker. It is hence necessary to be concerned not only with the probabilistic nature of the uncertainties, but also with the impact of those uncertainties on the agent's obtainable costs or rewards across a series of action choices.

While relatively unknown to much of the western world, Stratonovich independently made meaningful contributions to information theory that allows for its use in decision-making. In his seminal work, he coupled the definitions of entropy and mutual information with variational frameworks to emphasize the generality and optimality of the concepts. In doing so, he also elucidated the practicality of this combination, which extends well beyond the strict field of information processing.

The common theme of these variational problems is that they lead up to a notion of the value of information. The value of information is a new facet of information descriptors that relates values and the optimization under the Bayesian system of decisions and risks, which has been deemphasized in the western world since the days of Shannon. This behavior of the value of information is very timely for machine learning because it extends their set of tools with a principled framework and an emphasis on bounds.

More specifically, Stratonovich proposed three variational problems, the first two being restatements of classical information-theoretic results. The first is a presentation of channel-capacity theory as the optimization of entropy under a constant loss constraint; this loss constraint can be viewed as energy, under a thermodynamics interpretation. The second variational problem is related to Shannon's famous result that information transmission via noisy channels can be made asymptotically error free. This result can also be achieved by minimizing mutual information with a constraint on a loss bound. Kolmogorov's work on epsilon entropy exploits the variational framework and reaches the same solution. That is, they advocated minimizing either the uncertainty of one random variable or the uncertainty of one random variable conditioned on another random variable such that a certain level of costs could be obtained. In either case, these constrained problems can be viewed as measures that assesses the value of information. This is because entropy quantifies the amount

of surprise associated with an event, as described by either the random variable or a random variable conditioned on another, and thus the information one receives about the event. For distinct loss bound amounts, different amounts of surprise may be obtained. The constrained criterion given by Shannon and Kolmogorov therefore trades off between uncertainty and losses: it specifies how much uncertainty can be tolerated to achieve a given level of performance.

The third variational problem [1, 2] involves quantifying the maximum benefit that can be obtained for a given information amount to minimize expected losses. This problem is, again, posed as a constrained optimization: It involves determining extremal probability distributions that arise by minimizing losses subject to an information bound. This criterion is an optimization-based statement of Shannon's theories and generalizes the theory of rate-distortion. Mutual information, not entropy, is typically used in his criterion, which implements a constrained optimization problem over either a single- or dual-variable space whereby the aim is to minimize losses subject to a mutual information bound. This is equivalent to the dual problem of maximizing the dependence of one random variable on another subject to an improvement constraint. It is apparent that either version of this criterion assesses the value of information for any general information measure, e.g., Hartley, Renyi, Shannon, or Boltzmann information. Such behavior emerges because the cost function specifies the amount of information that can be obtained from one random variable about another for an expected level of improvement. Different amounts of average improvement lead to potentially distinct information amounts; the converse holds for the dual problem. Consequently, the criterion is a conditional extremum between expected utility and random variable uncertainty as described by the chosen information measure.

It can be shown that the value of information, as proposed by Stratonovich, satisfies all of the axioms for von Neumann–Morgenstern expected utility theory [3]. It also mirrors the work of prospect theory. It hence yields a principled strategy of agent decision-making under uncertainty that is independent of any specific definition of information [4]. Such behavior makes the Stratonovich's value-of-information framework particularly appealing compared to the work of either Shannon or Kolmogorov. More specifically, the work of Shannon and Kolmogorov relies on an optimization over a space of a single random variable, which complicates choosing an optimal action for given agent situations. The work of Stratonovich, in contrast, relies on solving an optimization problem in the joint space of the random variables, for certain information measures. It therefore is possible to specify the mutual dependence between environmental situations, or states, and actions. The information obtained by one random variable from another can be used to have agents make cost-/reward-optimal decisions when only partial information is available about the agent's environment through either one or both of the random variables.

Stratonovich never exploited this capability in a practical sense, though. He merely focused on establishing some of the criterion's theoretical properties. Moreover, he did not explicitly show how to iteratively find extremal distributions for the value of information, which is necessary for it to be used in practical settings.

In some of our recent work, we have begun to empirically investigate the value-of-information's potential for reinforcement learning [5]. Thus far, we have predominantly focused on its use for resolving the exploration-exploitation dilemma in Markov decision process-based reinforcement learning [6–8]. We have also shown how to optimize the criterion for this problem and other general decision-making problems [9].

Reinforcement learning is a branch of artificial intelligence that considers the problem of how an autonomous agent, which exists in a potentially unknown environment, can come to act intelligently for solving tasks in that environment. An agent refers to an entity that can perceive either the state of the environment or make observations about it and take actions that affect the environment's state. In return for taking an action, the agent receives a response, referred to as a reinforcement. The objective of the agent is to optimize the reinforcements that it receives for the amount of time that it acts in the environment. This entails determining the best actions, for given situations, that maximize the expected value of the cumulative reinforcement responses, assuming that the reinforcements are viewed as positive rewards. Such a problem is equivalent to maximizing the response of a so-called value function.

For many reinforcement-learning applications, complete details about the environment are not provided to the agent. Due to this lack of information, the agent needs to interact with the environment to acquire more knowledge about its action-state transition dynamics and their effects on the obtainable reinforcements. That is, the agent has to purposefully try, or explore, actions that are currently deemed sub-optimal for a given set of states. An agent that purely explores seemingly sub-optimal action choices is undesirable, though: it would regularly pick actions that deviate from its current policy, thereby potentially yielding small long-term reinforcements. An agent that eschews trying seemingly sub-optimal actions will also not address various tasks well either. It would consistently choose actions from its possibly limited understanding of the environment dynamics, which may not align well with the true dynamics; poor reinforcements would then be obtained.

A reasonable compromise between exploration and exploitation is hence crucial for obtaining good performance: the agent has to explore well enough that it understands the environment but not overly so. It is also needed to ensure that the search times are not unnecessarily burdensome for a given application. This hence leads to a dilemma of choosing the right amount of information, at any given moment during learning, for a certain problem.

The value of information is particularly well suited for optimally addressing the exploration–exploitation dilemma, which is inherently a compromise between utility and uncertainty. This is because it directly quantifies the benefit of exploring or exploiting at a given rate. The amount of exploration is implicitly dictated by the bound on the mutual dependence on states and actions, or, rather, the imposed uncertainty associated with selecting an action in a given state. As the information bound is increased to either the state-variable entropy or the action-variable entropy, the uncertainty associated with a particular action choice increases: the agent can eventually, given enough experience, consider all of the action choices and uncover the one that is specialized to a particular situation and yields the best reinforcement. This corresponds to an exploration-driven search. On the other hand, if the information bound is decreased to zero, then there is less uncertainty about what action should be taken in a given state. This corresponds to the case of an exploitation-driven search. Information bounds between these two extremes lead to a mixture of exploration and exploitation.

In the next section, we demonstrate the empirical effectiveness and theoretical optimality of the exploration behavior implemented by the value of information for single-state, multiple-action Markov decision processes [6]. Here, we consider only a single random variable, the action. The value of information, for this abstraction, has two equivalent interpretations. The first is that it describes the expected utility gain against the effort needed to transform an initial action-selection policy into the optimal policy. The second interpretation is that it describes the largest increase in expected reinforcements associated with actions that carry a certain amount of information about the environment. In either case, the value of information is defined by two terms that specify a modified negative-free-energy expression. The first term quantifies the expected reinforcements, while the second measures the divergence between the initial probabilistic policy and the optimal policy. Since we deal only with a single random variable, we work with the marginal of Shannon mutual information, which is the Kullback–Leibler divergence. The interplay between these two terms facilitates an optimal search for the best-performing action under the given bounds.

We face certain practical challenges when applying the value of information to single-state, multiple-action Markov decision processes. The first entails showing how the criterion can be optimized to yield an action-selection strategy. To deal with this issue, we demonstrate that the criterion can be reformulated into a form that can be seen to be convex. We then provide an iterative, coordinate-descent-based approach for uncovering global solutions of this variational expression. The coordinate-descent update leads to a formula for probabilistically weighting action choices according to the expected reinforcements. This formula contains a single hyperparameter, which accounts for the information bound amount. The effect of this hyperparameter is to modulate the uncertainty in the weighting process. Low hyperparameter values lead to more uncertainty in what action should be chosen, causing each action to be chosen at random, regardless of its expected performance. High values lead to less uncertainty, which implies that the current best action will be taken with increasing frequency. The second challenge entails determining how to best set this hyperparameter. We deal with this according to simulated-annealing

update schedules that start with a high-exploration search and eventually give way to an exploitation of the best-performing action. A sufficiently quick lowering of the hyperparameter value ensures that the optimal policy can be found at an optimal rate.

Following this treatment, we consider the more difficult environment abstraction of multiple-state, multiple-action Markov decision processes [7, 8]. The value of information, in this context, has an analogous interpretation as in the single-state, multiple-action case. The criterion is now given by three terms, though, that yield a modified, negative-free-energy optimization problem. The first two terms specify the average anticipated improvement in the agent’s performance over the baseline case where no information is available about the actions that should be taken. The remaining term, an optimization constraint, bounds the mutual dependence between states and actions according to a chosen information measure. That is, it describes the expected value of the divergence between the initial action-selection policy and the unknown, optimal policy.

As before, optimizing the value of information is necessary to explore this environment abstraction. We again show how this can be performed using a coordinate-descent-type approach, which yields a weighted-random action-selection update equation for constructing a stochastic policy. A single hyperparameter emerges from this optimization process that dictates how random the action choices will be when using this update. Setting this hyperparameter is crucial, as it is application dependent, just like in the single-state, multiple-action case. Unfortunately, the complicated nature of this more general environment abstraction precludes an easy, explicit adjustment schedule for the hyperparameter that yields optimal agent behaviors. We thus have resorted to the use of an information-theoretic quantity, the policy cross entropy, for deciding how to heuristically adjust the hyperparameter. The policy cross entropy is a bounded measure of change between pairs of stochastic policies. It hence roughly describe the rate of learning. Cross-entropy values near zero imply that few changes are being made to a policy. Values of cross entropy near one correspond to many policy changes. To facilitate good action specialization for given situations, we consider a heuristic that adjusts the hyperparameter value so that the cross entropy remains above a certain threshold for most of the learning process.

We have found that an interesting action-state quantization phenomenon emerges when applying using the value of information for this more complicated environment abstraction. For low information bound amounts, many states achieve highly similar action-state value function magnitudes and hence are grouped together. The same action is assigned to each state in the cluster. This behavior coincides with the exploitation-driven search: the action choices are not uniquely specialized to individual states due to the high action-state uncertainty. For high information bound amounts, each state has the potential to obtain a unique value function response. Many state groups emerge with either few states or even just a single state per cluster; a potentially unique action is assigned to each group. This corresponds to an exploration-dominant investigation: the actions become specialized to the dynamics of the environment to achieve the best reinforcements. The value of information hence performs constrained clustering according to the value function, where the information term acts as a regularization factor that implicitly determines the num-

ber of clusters [9]. This clustering behavior aids in the exploration process: it allows the criterion to outperform conventional exploration heuristics, typically by a large amount.

The remainder of this chapter is organized as follows. Section 19.2 deals with the value of information for exploring single-state, multiple-action Markov decision processes. We begin in Sect. 19.2.1 with an overview of existing search strategies for this abstraction. We predominantly focus on methodologies that rely on stochastic search, as the value of information also yields a stochastic exploration mechanism. We outline our methodology in Sect. 19.2.2. We start, in Sect. 19.2.2.1, by introducing the value of information for this simple abstraction. Interpretations for this criterion are provided. In Sect. 19.2.2.2, we provide the solution to the optimization problem, which takes the form of a parameterized Gibbs distribution. We convert this distribution into a mixture model to ease the theoretical analysis. We also provide two potential parameter annealing schedules. Our theoretical treatment of this Gibbs-based update is given [6]. We proved that these two algorithms can yield optimal and near-optimal-exploration strategies.

In Sect. 19.2.3, we assess the empirical capabilities of the value of information for the single-state, multiple-action environment abstraction. We begin this section by covering our experimental protocols in Sect. 19.2.3.1. In Sect. 19.2.3.2, we present our simulation results. We assess the performance of the value of information for the fixed-exploration case and the tuned case. Performance comparisons against other bandit algorithms are presented in Sect. 19.2.3.3. Since the value of information implements a weighted-random search, we consider a variety of stochastic-based sampling techniques, such as  $\epsilon$ -greedy exploration, soft max-based selection, and reinforcement comparison. We also compare against the deterministic upper confidence-bound method. Our results indicate that the value of information empirically outperforms these alternate approaches in a variety of circumstances. This is rather surprising, since these algorithms also implement an optimal-exploration strategy that is independent of the number of available actions and how the stochastic rewards generated for each action. Our findings are summarized in Sect. 19.2.4.

Section 19.3 describes the use of the value of information for exploring multiple-state, multiple-action Markov decision processes, which is a general abstraction for many important decision-making problems. In Sect. 19.3.1, we describe some of the existing methodologies for trial-and-error action selection during learning. We then, in Sect. 19.3.2.1, introduce a modified version of the value-of-information optimization problem given in Sect. 19.2.2.1. We focus on the case where the mutual dependence between states and actions is dictated by Shannon information, which we previously considered in [7, 8]. In Sect. 19.3.2.2, we outline how to iteratively solve this problem, which results in an expectation-maximization-like set of update equations for modifying the policy. These equations contain a single free hyperparameter. We combine the updates with a conventional reinforcement-learning algorithm, tabular  $Q$ -learning. At the end of Sect. 19.3.2.2, we draw upon our work from [8] and outline how to adjust the hyperparameter according to a cross-entropy heuristic.

We apply the value of information to the task of constructing good-performing agents for the game *Crossy Road* in Sect. 19.3.3. *Crossy Road* is a maze-like game

where the agent’s aim is to progress as far as possible into a procedurally generated environment while avoiding dynamic and stationary obstacles. The game requires the use of multiple strategies to maximize the agent’s score. We describe the game, its reward structure, and the chosen state–action feature space in Sect. 19.3.3.1. After covering our experimental protocols, we present our simulation results. When begin in Sect. 19.3.3.1, which covers the qualitative improvement in the agent’s behaviors. We then empirically demonstrate, in 19.3.3.2, the effects of changing the hyperparameter values on the quantitative performance; connections back to the qualitative agent behaviors are provided. Performance comparisons of our policies against epsilon-greedy and soft max-based search are provided in Sect. 19.3.3.3. We also consider the relative performance of uncertainty-based techniques that systematically explore the state–action space. Our findings are summarized in Sect. 19.3.4.

## 19.2 Exploring Single-State, Multiple-Action Markov Decision Processes

### 19.2.1 Literature Survey

Many exploration schemes have been advanced over the years for investigating general dynamical systems. Most of these schemes have been developed for rather models like multi-state, multi-action Markov decision processes. We refer to the survey of Kaelbling et al. [10] for overviews of such schemes. For now, we consider popular stochastic exploration tactics for the simpler single-state, multiple-action Markov decision process abstraction with a discrete number of action possibilities [11]. Such an abstraction is also referred to as a multi-armed bandit. This is due to its similarity to slot machines, which are colloquially referred to as one-armed bandits.

The stochastic, discrete multi-armed-bandit problem consists of a finite set of probability distributions with associated expected values and variances. Each probability distribution is an abstraction of a slot machine, which is colloquially referred to as a one-armed bandit. Initially, each of the distributions is unknown. The agent, a gambler, can choose at discrete-time instants to pull one of the slot-machine arms to receive a stochastic reward; in this context, pulling an arm is equivalent to taking an action. The objective of the gambler is to maximize his or her return, which is the sum of the rewards received over a sequence of arm pulls. Since the reward distributions may differ from machine to machine, the objective of the gambler is equivalent to discovering and continuously utilizing the machine with the best possible pay-off, as early as possible. That is, the gambler should achieve minimal regret, which is defined to be the expected total difference in reinforcements that is incurred due to not pulling the best-paying arm from the beginning. For the stochastic, discrete multi-armed bandit problem, a regret that is a logarithmic function of the number of arm pulls  $k$  is optimal.

We note that several other variants of this abstraction exist. These include adversarial bandits [12, 13], non-stationary bandits [12, 13], associative bandits [12, 14], and budgeted bandits [15], each of which makes different assumptions about the reinforcements and has pertinent exploration–exploitation strategies. Extensions of the bandit problem to the continuous action case have also been made [16–18].

One of the most widely employed exploration strategies for discrete, stochastic multi-armed bandits is the  $\epsilon$ -greedy method [19]. Epsilon-greedy exploration entails choosing either a random action, with a probability given by the value of  $\epsilon$ , or the action with the highest mean reward. For practical problems, the mean reward is estimated from the observed rewards up to the current pull.

The simplest form of  $\epsilon$ -greedy exploration is the  $\epsilon$ -first strategy. This approach consists of initially performing an exploration phase during learning, then switching to a pure exploitation phase. In the exploration phase, the arms are randomly selected and pulled during the first  $\epsilon k$  rounds, where  $k$  is a pre-specified number of rounds. Sub-optimal arms may or may not be dropped from the search process during this stage. During the remaining  $(1 - \epsilon)k$  rounds, the lever with the highest estimated mean from the exploration phase is pulled. This is referred to as the exploitation phase. As Even-Dar et al. have shown [20], a log-linear number of pulls  $O(|\mathcal{A}|/\alpha^2)\log(|\mathcal{A}|/\kappa))$  during the exploration phase is an upper bound to find an  $\alpha$ -optimal action with probability at least  $1 - \kappa$ . Here,  $|\mathcal{A}|$  represents the number of action choices. This result can be viewed as an analysis of the asymptotic behavior of the  $\epsilon$ -first mechanism from a probabilistically-absolutely-correct framework. Mannor and Tsitsiklis [21] provided complementary results for a lower bound and proved that it is also log-linear.

In certain circumstances, context information may be available, at each round, to help determine which arm should be pulled.  $\epsilon$ -first exploration can also be used for such contextual bandits [22]. More advanced schemes have been proposed that rely on Gaussian processes [23, 24]. Others, such as the exponential-weighting for exploration and exploitation using expert advice (Exp4) [13, 25] rely on expert advice to systematically remove irrelevant actions.

In the  $\epsilon$ -first method, the value of  $\epsilon$  remains fixed during the learning process. This is a rather restrictive assumption, since it prevents the decision-making process from getting arbitrarily close to the optimal lever. Researchers have proposed so-called  $\epsilon$ -changing strategies to deal with this issue. Many such methods entail modifying  $\epsilon$  across consecutive arm pulls so that the optimal action choice can be made asymptotically. For certain classes of reward distributions, Cesa-Bianchi and Fischer [26] showed that a squared-log regret of  $O(\vartheta \log(|\mathcal{A}|/\beta^2)/\beta^2 + \log(k) + \vartheta \log^2(k)/\beta^2)$  can be obtained for  $\epsilon$ -changing strategies after  $k$  rounds. Their approach was referred to as GreedyMix. Here,  $\beta, \vartheta$  are parameters, independent of the number of arm pulls  $k$ , that describe reward statistics for each of the  $|\mathcal{A}|$  actions. In GreedyMix, they assumed that  $\epsilon$  would be decreased logarithmically with respect to the number of rounds. Auer, Cesa-Bianchi, and Fischer [27] later showed that a log regret of  $O(\log(k/\vartheta) + (1 + \beta^2)\vartheta)$  could be obtained for GreedyMix by choosing good initial values for  $\epsilon$ .

Another widely utilized exploration mechanism is soft-max selection. Soft-max selection entails taking actions in a stochastic fashion according to a Gibbs distribution. This distribution specifies probabilities for choosing a given lever based upon its expected reward relative to all other possible rewards. Action with high expected rewards is assigned greater probability mass than those with lower rewards. A single, user-adjustable parameter  $\tau$  is present in the distribution that controls the degree of randomness in the lever-selection process. Values of  $\tau$  close to zero lead to a greedy selection strategy: the action with the highest expected reward is consistently favored over all other actions. Values of  $\tau$  close to one yield a uniform selection strategy: no action is favored over others, regardless of the expected rewards.  $\tau$  is sometimes referred to as temperature, which is due to the connections between the Gibbs distribution and the theory of thermodynamics.

Soft-max selection can be modified into  $\tau$ -first and  $\tau$ -changing strategies. In the former case,  $\tau$  is set so that exploration is performed only at the beginning of the learning process. The latter case involves monotonically adjusting the temperature  $\tau$  from some sufficiently high value to a low value according to either a linear or logarithmic schedule. A logarithmic cooling case was investigated by Cesa-Bianchi and Fischer [26]. They showed that a log-squared regret of  $O(\vartheta \log(|\mathcal{A}|/\beta^2)/\beta^2 + \log(k) + \vartheta \log^2(k)/\beta^2)$  can be obtained after  $k$  rounds by their SoftMix algorithm. As before,  $\beta$ ,  $\vartheta$  are parameters that depend on the expected values of the reward distributions for the  $|\mathcal{A}|$  actions.

A more complicated variant of the soft-max selection, the exponential-weight algorithm for exploration and exploitation (Exp3), was proposed by Auer, Cesa-Bianchi, and Fischer in [27]. Exp3 entails probabilistically choosing a lever based upon an amended Gibbs distribution. The role of the amended distribution is to divide the expected reward by the probability that the action was chosen. Auer et al. showed that a cumulative regret bound of  $O((|\mathcal{A}|k \log(|\mathcal{A}|k/\kappa))^{1/2})$ , with probability of at least  $1 - \kappa$ , was possible for finding the  $\alpha$ -optimal arm when using Exp3. This bound is optimal, as the authors demonstrated that a matching lower bound could be obtained under general circumstances. MacMahan and Streeter [28] later revised Exp3 and Exp4 to take advantage of expert advice and systematically remove irrelevant arms. They furnished a tighter bound of  $O((Sk \log(M))^{1/2})$ , where  $S \ll |\mathcal{A}|$  is a variable that measures the extent to which the expert recommendations agree and  $M$  is the number of experts.

In what follows, we provide a soft-max style of action selection for discrete, multi-armed bandits. Our methods are based on the value-of-information criterion proposed by Stratonovich. When applied to reinforcement learning, this criterion quantifies the expected trade-off between uncertainty, as quantified by policy information, and expected rewards. An appealing property of this criterion is that it directly specifies the obtainable performance for a given amount of exploration. To our knowledge, no other multi-armed bandit search algorithms share this property.

The value of information yields a search process governed by a Gibbs distribution parameterized according to the amount of policy information  $\tau_k$ . We show that a modified  $\tau_k$ -changing learning style yields a regret of  $O(\log(k))$ ; this is the best regret bound that can be achieved for a stochastic multi-armed bandit problem [29]. For this approach, we assume that knowledge of the mean slot-machine rewards are available to inform the choice of a hyperparameter. This regret bound is an interesting result, as Gibbs-based approaches with a monotone learning-rate sequence have previously only obtained polylogarithmic regret [26, 30]. We additionally propose a method with a straightforward means of selecting the hyperparameter. The second approach leads to a regret of  $O(\log(k)^{1+2\theta})$ , which can be made arbitrarily close to a logarithmic regret by choosing a small  $\theta$ . No prior knowledge of the reward statistics are needed, unlike in the first method, to achieve this regret bound. This makes it competitive against other Gibbs-based, distribution-free schemes, such as Exp3 and Exp4 [13], which only obtain linear-logarithmic regret.

### 19.2.2 Methodology

The value of information is a constrained optimization problem that can be used for decision-making when the agent-environment interactions are described by a general dynamical system. Here, we take this system to be a Markov decision process.

The Markov decision process framework is used heavily in the theory of stochastic decision-making for discrete-space and discrete-time settings. In a Markov decision process, the system being considered is assumed to be driven by underlying Markov chains. That is, the system jumps randomly from one state  $s_k^j \in \mathcal{S}$  to the next  $s_{k+1}^p \in \mathcal{S}$  for discrete-time steps  $k$ . Moreover, in a Markov decision process, the agent is required to choose an action  $a_k^i \in \mathcal{A}$  from a set of available actions. An immediate real-valued reinforcement  $r_k(a_k^i) : \mathcal{A} \rightarrow \mathbb{R}$  is earned during the transition between states  $s_k^j$  and  $s_{k+1}^p$  whenever an action  $a_k^i$  is taken; since these reinforcements will usually be positive, we also refer to them as rewards. For now, we assume that the Markov chain only has a single state, to which it always returns with probability one after an action is performed.

A corresponding policy dictates the action that is selected at each state. A policy  $p(a_i^k) = \pi_k^i, \pi_k^i : \mathcal{A} \rightarrow \mathbb{R}_+$ , characterizes the probability distribution over actions  $a_k^i \in \mathcal{A}$ . We assume that an initial probabilistic action-selection policy is provided  $p_0(a_k^i)$ . Such a policy could assign equal probabilities of choosing each action, for example. The value of information seeks a transformed version of this initial policy that yields the highest rewards. The initial policy is not necessarily allowed to change arbitrarily, though, after each taken action. Rather, the transformation is governed by a bounded information constraint term. The lower the bound, the less that the initial policy entries can be modified for a single action choice. This behavior can

impact the obtainable rewards if the initial policy is poor. Conversely, the higher the bound, the more that the initial policy can be adjusted in a given round and hence the greater the potential reward improvement. The degree of policy transformation must be delicately balanced to obtain the best pay-out.

As we will show, the information bounds are dictated by a single parameter that emerges from converting this single-term constrained optimization problem into a two-term unconstrained problem. This parameter simultaneously dictates how much weighted-random-based exploration is performed. We introduce another parameter that adds a uniform-random exploration component. Adjusting both parameters is crucial for exploring the action possibilities at a rate that minimizes the deviation in reinforcements compared to following the unknown optimal policy from the beginning. This deviation in expected reinforcements is referred to as regret. Lai and Robbins showed that regret is bounded above and below by the logarithm of the number of taken actions [31]. There is no exploration strategy with better asymptotic performance.

It is well known that a constant amount of exploration typically leads to linear regret [26]. A systematic adjustment of exploration is needed to achieve sub-linear regret. As we noted above, we consider a simulated-annealing-type approach for tuning the two parameters across each action choice. We provide two different annealing cooling schedules that switch between near-perfect exploration and near-perfect exploration for a finite number of taken actions. We proved in [6] that these schedules can achieve logarithmic regret. They hence explore the action space at the best possible rate for this problem.

### 19.2.2.1 Value of Information

For single-state, multiple-action Markov decision processes, we assume that an agent's initial behavior is described by a prior distribution  $p_0(a_k^i)$ . Here,  $a_k^i \in \mathcal{A}$  represents the  $i$ th action chosen in the  $k$ th round; a single round, which we also refer to as an iteration, is equivalent to taking a single action. The variable  $\mathcal{A}$  is the set of all actions.

The agent transforms its behavior to a posterior-like distribution  $p(a_k^i)$ , which is referred to as the policy. We will also use  $\pi_k^i$  to denote the policy distribution. This transformation occurs in a way that optimally trades off the expected utility against the transformation costs for going from  $p_0(a_k^i)$  to  $p(a_k^i)$ . This trade-off is given by the following extremization problem, which is the value of information,

$$\max_{p(a_k^i)} \underbrace{\left( \sum_{k=1,2,\dots} \sum_{a_k^i \in \mathcal{A}} p(a_k^i) r_k(a_k^i) \right)}_{\sum_{k=1,2,\dots} \mathbb{E}_{a_k^i} [r_k(a_k^i)]} \text{ such that } \underbrace{\sum_{a_k^i \in \mathcal{A}} p(a_k^i) \log \left( \frac{p(a_k^i)}{p_0(a_k^i)} \right)}_{D_{\text{KL}}(p(a_k^i) \| p_0(a_k^i))} \leq \varphi_{\text{inf}}. \quad (19.1)$$

This expression can be viewed as a problem of finding a version  $p(a_k^i)$  of the initial policy that achieves maximal expected rewards in the case where the initial policy  $p_0(a_k^i)$  can only change by a prescribed amount. Here,  $\varphi_{\text{inf}} \in \mathcal{R}_+$  is a non-negative value that represents the transformation amount bound. The variable  $r_k(a_k^i) \in \mathcal{R}$  represents the reinforcement for the  $i$ th action  $a_k^i \in \mathcal{A}$  chosen after  $k$  action choices; it can also be interpreted as a random variable.

The optimization term in (2.1) can be described in more detail as follows:

**Optimization Term: Expected Returns.** The term to be optimized represents the expected returns  $\sum_{k=1,2,\dots} \mathbb{E}_{a_k^i} [r_k(a_k^i)] = \sum_{k=1,2,\dots} p(a_k^i) r_k(a_k^i)$  associated with an action-selection strategy  $p(a_k^i)$  that provides the best pay-out  $r_k(a_k^i)$  over all  $k$  taken actions. The selection strategy  $p(a_k^i)$  is initially unknown. However, it is assumed to be related to the specified prior  $p_0(a_k^i)$ , in some manner, as described by the information constraint bound.

As the number of taken actions becomes infinite, the agent will have complete knowledge of the environment, assuming that the information constraint bound  $D_{\text{KL}}(p(a_k^i) \| p_0(a_k^i)) \leq \varphi_{\text{inf}}$  is equal to the action random variable entropy  $\varphi_{\text{inf}} = -\sum_{a^j \in \mathcal{A}} p(a^j) \log(p(a^j))$ . This term will eventually produce globally optimal pay-outs. Optimal pay-outs can also be achieved if the information constraint bound is annealed at a sufficiently rapid pace. The agent's action-selection behavior becomes entirely deterministic in such situations, as only the best action will be selected. The policy  $p(a_k^i)$  becomes a delta function in the limit. Otherwise, the agent's behavior will be stochastic. That is, the policy  $p(a_k^i)$  will be described by a discrete unimodal distribution that is not quite uniform and not quite a delta function.

The constraint term in (2.1) can be interpreted as follows:

**Constraint Term: Transformation Cost.** The constraint term,  $D_{\text{KL}}(p(a_k^i) \| p_0(a_k^i))$ , a Kullback–Leibler divergence, quantifies the divergence between the posterior  $p(a_k^i)$  and the prior  $p_0(a_k^i)$ . We use this term, as we work only with a single random variable; the Kullback–Leibler divergence can be seen as the marginal of Shannon mutual information. This term is bounded above by some non-negative value  $\varphi_{\text{inf}}$ , which implies that the amount of overlap will be artificially limited by the chosen  $\varphi_{\text{inf}}$ .

The value of  $\varphi_{\text{inf}}$  dictates by how much the prior  $p_0(a_k^i)$  can change to become the posterior  $p(a_k^i)$ . If  $\varphi_{\text{inf}}$  is zero, then the transformation costs are infinite. The posterior will therefore be equivalent to the prior and no exploration will be performed. The expected pay-out will not be maximized for non-optimal priors. If  $\varphi_{\text{inf}}$  is larger than the random variable entropy, then the transformation costs are ignored. The prior is free to change to the optimal-reward policy, since the exploration of actions dominates. For values of  $\varphi_{\text{inf}}$  between these two extremes, the agent weighs the expected improvement in rewards against the transformation costs. A mixture of exploration and exploitation occurs.

The constraint can also be viewed as describing the amount of information available to the agent about the environment. That is, for small values of  $\varphi_{\text{inf}}$ , the agent has little knowledge about the problem domain. It is unable to determine how to best adapt its action-selection strategy. As  $\varphi_{\text{inf}}$  increases, more information is made available to the agent, which allows it to refine its behavior. In the limit, the agent has the potential for complete knowledge about the environment, which means the action choice will be reward-optimal.

The value of information, given by (2.1), therefore describes the best pay-out that can be achieved for a specified policy transformation cost. It has the dual interpretation of quantifying how sensitive the action-selection process will be to environment knowledge, as communicated through the Shannon information term.

This constrained criterion given in (2.1) can be converted into the following unconstrained variational expression that defines a free-energy difference, in a thermodynamics sense. The unconstrained problem follows from the theory of Lagrange multipliers,

$$\max_{p(a_k^i)} \underbrace{\left( \sum_{k=1,2,\dots} \sum_{a_k^i \in \mathcal{A}} p(a_k^i) r_k(a_k^i) \right) - \frac{1}{\tau_k} \sum_{a_k^i \in \mathcal{A}} p(a_k^i) \log \left( \frac{p(a_k^i)}{p_0(a_k^i)} \right)}_{\tau_k^{-1} \log(\sum_{a_k^i \in \mathcal{A}} p_0(a_k^i) e^{\tau_k r_k(a_k^i)})}, \quad (19.2)$$

for some non-negative  $\tau_k$ . The parameter  $\tau_k$  sets the relative importance between the transformation cost and the pay-out maximization.  $\tau_k^{-1}$  can be viewed as an inverse temperature that can change as the number of taken actions increases. Adjusting  $\tau_k$  across each action choice modifies the amount of exploration that is performed. We discuss ways this can be done, so as to achieve optimal regret, in the next subsection.

The preceding value of information formulation has some practical difficulties. In particular, it is not immediately apparent if the criterion in (2.2) is convex. It may therefore have a global extremum that is not easily obtained. We, therefore, consider the dual problem of maximizing the informational overlap subject to a cost constraint. This extremization problem can be seen to be convex, as it relies on the minimization of a Kullback–Leibler divergence,

$$-\min_{p(a_k^i)} \underbrace{\sum_{a_k^i \in \mathcal{A}} p(a_k^i) \log \left( \frac{p(a_k^i)}{p_0(a_k^i)} \right)}_{D_{\text{KL}}(p(a_k^i) \| p_0(a_k^i))} \text{ such that } \underbrace{\left( \sum_{k=1,2,\dots} \sum_{a_k^i \in \mathcal{A}} p(a_k^i) r_k(a_k^i) \right)}_{\sum_{k=1,2,\dots} \mathbb{E}_{a_k^i} [r_k(a_k^i)]} \leq \varphi_{\text{cost}} \quad (19.3)$$

for  $\varphi_{\text{cost}} \in \mathcal{R}_+$ . The equivalency of (2.1) and (2.3) follows from an application of Stjernvall's dominance theory and some additional arguments. We can state this constrained criterion in an unconstrained variational form as a negative free-energy difference,

$$-\arg \min_{p(a_k^i)} \underbrace{\sum_{a_k^i \in \mathcal{A}} p(a_k^i) \log \left( \frac{p(a_k^i)}{p_0(a_k^i)} \right) - \frac{1}{\tau_k} \left( \sum_{k=1,2,\dots} \sum_{a_k^i \in \mathcal{A}} p(a_k^i) r_k(a_k^i) \right)}_{-\tau_k^{-1} \log (\sum_{a_k^i \in \mathcal{A}} p_0(a_k^i) e^{\tau_k r_k(a_k^i)})}, \quad (19.4)$$

subject to some non-negative  $\tau_k$ . The criterion in (2.3) can be seen as one that attempts to maximize the divergence between the prior and the policy subject to a reward constraint.

### 19.2.2.2 Value of Information Optimization

The process of decision-making using the value of information proceeds as follows. At the beginning, the agent finds itself in an environment where it is performing optimally, given the information constraints, according to the prior. The agent then experiences a change in the environment, which leads to a change in the rewards. As a consequence of this environmental change, the previous policy is no longer optimal and a new one needs to be found. This requires the maximization of the negative free-energy difference.

It is known that the exponential-based Gibbs distribution maximizes the negative free-energy difference in (2.4),

$$p(a_{k+1}^i) = \underbrace{\left( \frac{e^{q_{k+1}^i / \tau_k}}{\sum_{a_k^i \in \mathcal{A}} e^{q_k^i / \tau_k}} \right)}_{\pi_{k+1}^i = \text{exponential component}}, \quad q_{k+1}^i = \frac{r_k(a_k^i)}{p(a_k^i)} I_k^i, \quad (19.5)$$

where  $q_{k+1}^i$  is probabilistically scaled version of the instantaneous reward for the  $i$ th action  $a^i \in \mathcal{A}$ .

The update given in (2.5) can be viewed as a type of coordinate descent where the step-size is automatically chosen at each iteration. We consider the following modification of the Gibbs distribution in (2.5), which is a kind of mixture model. This was done to ensure that the optimal-exploration rate could be easily obtained

$$p(a_{k+1}^i) = (1 - \gamma_k) \underbrace{\left( \frac{e^{\sum_{s=1}^k q_s^i / \tau_s}}{\sum_{a_k^j \in \mathcal{A}} e^{\sum_{s=1}^k q_s^j / \tau_s}} \right)}_{\pi_{k+1}^i = \text{exponential component + uniform component}} + \frac{\gamma_k}{|\mathcal{A}|}, \quad q_{k+1}^i = \frac{r_k(a_k^i)}{p(a_k^i)} I_k^i. \quad (19.6)$$

Here, we have introduced a uniform-random exploration component from [27]. This mixture model is parameterized by the inverse temperature  $\tau_k^{-1}$  and a mixing coefficient  $\gamma_k$ . For this update,  $r_s^i$  represents the current estimate of the expected reward for action  $a_s^i$ , and  $q_s^i$  is a version of it that is transformed by the allocation rule  $p(a_{k+1}^i) = \pi_k^i$  and an indicator variable  $I_k^i$ . Observe that this transformed reward is

modified by the parameters  $\tau_1, \dots, \tau_s$ , up to round  $s = k$ , and then aggregated across all previous taken actions  $a_1^i, \dots, a_s^i$ , up to round  $s = k$ . This differs from previous variants of exponential-weighting algorithms, such as soft-max selection, where only the current value of  $\tau_k$  modifies the sum of aggregated rewards for actions  $a_1^i, \dots, a_s^i$ , up to  $s = k$ .

---

**Algorithm 19.1**  $\tau_k$ -changing Exploration: VoIMix

---

**Input:**  $d$ : real number in the unit interval such that  $0 < d < \min_j \mu^* - \mu^j$ , for all  $a^j \in \mathcal{A} \setminus a^*$ .

- 1 Define the sequences  $\gamma_k$  and  $\tau_k^{-1}$  for  $k = 1, 2, \dots$  by  $\gamma_k = \min\left(1, \frac{5|\mathcal{A}|}{kd^2}\right)$  and either
 
$$\tau_k^{-1} = \frac{1}{|\mathcal{A}|/\gamma_k + 1} \log\left(1 + \frac{d(|\mathcal{A}|/\gamma_k + 1)}{2|\mathcal{A}|/\gamma_k - d^2}\right) \text{ or}$$

$$\tau_k^{-1} = \frac{1}{1 + 2|\mathcal{A}|/\gamma_k} \log\left(1 + \frac{d + 2d|\mathcal{A}|/\gamma_k}{2|\mathcal{A}|/\gamma_k}\right).$$
  - 2 Let  $q^j = 0$  for  $j = 1, \dots, |\mathcal{A}|$ .
  - 3 **for each**  $k = 1, 2, \dots$  **do**
    - 4 Take an action drawn from the distributions  $\pi_k^1, \dots$ , where
 
$$\pi_{k+1}^i = (1 - \gamma_k) \frac{e^{\sum_{s=1}^k q_s^i / \tau_s}}{\sum_{a^j \in \mathcal{A}} e^{\sum_{s=1}^k q_s^j / \tau_s}} + \frac{\gamma_k}{|\mathcal{A}|}.$$
    - 5 Let  $i$  be the index of the taken action and  $r_k(a_k^i)$  the obtained reward. Set
 
$$q_k^i = r_k(a_k^i) I_k^i / \pi_{k+1}^i, \text{ where } I_k^i = 1 \text{ and } I_k^j = 0 \text{ for all } j \neq i.$$
- 

---

**Algorithm 19.2**  $\tau_k$ -changing Exploration: AutoVoIMix

---

**Input:**  $\theta$ : real number in the range  $(0, 0.5)$ .

- 1 Define the sequences  $\gamma_k$  and  $\tau_k^{-1}$  for  $k = 1, 2, \dots$  by  $\gamma_k = \min\left(1, \frac{5|\mathcal{A}|}{k} \log^{2\theta}(k)\right)$  and
 
$$\tau_k^{-1} = \frac{1}{|\mathcal{A}|/\gamma_k + 1} \log\left(1 + \frac{\log^{-\theta}(k)(|\mathcal{A}|/\gamma_k + 1)}{2|\mathcal{A}|/\gamma_k}\right).$$
  - 2 Let  $q^j = 0$  for  $j = 1, \dots, |\mathcal{A}|$ .
  - 3 **for each**  $k = 1, 2, \dots$  **do**
    - 4 Taken an action drawn from the distributions  $\pi_k^1, \dots$ , where
 
$$\pi_{k+1}^i = (1 - \gamma_k) \frac{e^{\sum_{s=1}^k q_s^i / \tau_s}}{\sum_{a^j \in \mathcal{A}} e^{\sum_{s=1}^k q_s^j / \tau_s}} + \frac{\gamma_k}{|\mathcal{A}|}.$$
    - 5 Let  $i$  be the index of the taken action and  $r_k(a_k^i)$  the obtained reward. Set  $q_k^i = r_k(a_k^i) I_k^i / \pi_{k+1}^i$ , where  $I_k^i = 1$  and  $I_k^j = 0$  for all  $j \neq i$ .
- 

Both user-selectable parameters in (2.6) have the following functionality:

**$\tau_k$ : Inverse Temperature Term.** This parameter can be interpreted as an inverse-temperature-like parameter. As  $\tau_k^{-1}$  approaches zero, we arrive at a low-temperature thermodynamic system of particles. Such a collection particles will have low

kinetic energy and hence their position will not vary greatly. In the context of learning, this situation corresponds to policies  $\pi_k^i, \pi_{k+1}^i, \dots$  that will not change much due to a low-exploration search. Indeed, the allocation rule will increasingly begin to favor the action with the highest current reward estimate, with ties broken randomly. All other randomly selectable actions will be progressively ignored. On the other hand, as  $\tau_k^{-1}$  goes off to infinity, we have a high-temperature system of particles, which has a high kinetic energy. The action-selection policy  $\pi_{k+1}^i$  can change drastically as a result. In the limit, all actions are taken independently and uniformly at random. Values of the inverse-temperature parameter between these two extremes implement searches for actions that are a blend of exploration and exploitation.

**$\gamma_k$ : Mixing Coefficient Term.** This parameter can be viewed as a mixing coefficient for the model: it switches the preference between the exponential distribution component and the uniform-distribution component. For values of the mixing coefficient converging to one, the exponential component is increasingly ignored. The resulting action-choice probability distributions  $\pi_{k+1}^i$  become uniform. The chance of any action being selected is the same, regardless of the expected pay-out. This corresponds to a pure exploration strategy. As the mixing coefficient becomes zero, the influence of the exponential term consequently rises. The values of  $\pi_{k+1}^i$  will hence begin to dictate how frequently a given action will be chosen based upon its energy, as described by the reward pay-out, and the inverse temperature. Depending on the value of the inverse-temperature parameter,  $\tau_k^{-1}$ , the resulting search will either be exploration-intensive, exploitation-intensive, or somewhere in between.

To take advantage of this model, a principled means of choosing both the inverse temperature and the mixing coefficient parameters is required. One way to do this is by employing a type of simulated annealing. Under this scheme, empirical convergence is realized by letting both parameters monotonically decrease to zero according to some cooling schedule.

We have introduced two possible cooling schedules, in Algorithms 19.1 and 19.2, which yield optimal and near-optimal regret for the bandit problem, respectively. These schedules are inspired from GreedyMix, but avoid factors that would lead to log-squared regret. For Algorithm 19.1, named VoIMix, we set  $\gamma_k = \Theta(1/k)$  and  $\tau_k^{-1} = \Theta(\log(k))$ .

Two cooling schedules are given for  $\tau_k^{-1}$  in Algorithm 19.1, both of which yield equivalent regret. It is important to note that Algorithm 19.1 requires the user to supply a real-valued input parameter  $d$ , which is used to update both  $\gamma_k$  and  $\tau_k^{-1}$ . Selecting initial values of this parameter can be difficult, in practice. It relies on knowledge about the slot-machine average rewards and the global-best expected rewards, both of which are unknown to the gambler *a priori*. In Algorithm 19.2, called AutoVoIMix, the parameter  $d$  from Algorithm 19.1 has been removed and is implicitly calculated by way of another real-valued term  $\theta$ , which is more straightforward to specify. For values of  $\theta$  near zero, Algorithm 19.2 achieves nearly logarithmic regret. A great many action choices are needed for this to occur, though. Setting  $\theta$  near 0.5 achieves log-squared regret. The slot-machine arms need to be pulled only a few times before this happens.

The parameter update schedules given in these algorithms can be explained as follows. Initially, there is a long period of pure exploration. The length of this period is dictated by  $d$ , in VoIMix, and  $\theta$ , for AutoVoIMix; higher values of either parameter increase the period. It also depends on the number of actions, with larger amounts of them leading to a lengthier period. The mixing coefficient,  $\gamma_k$ , is set to one during this time. The inverse-temperature term,  $\tau_k^{-1}$ , is set to some value greater than zero. The exponential mixture component is therefore largely ignored in favor of the uniform component. The action-selection policy  $\pi_{k+1}^i$  will sample all of the arms in a uniform manner during this time. This period of exploration corresponds to the case where the inverse temperature is infinite, if we had used a purely exponential formulation. As the number of taken actions increases, the weight of the exponential term on the policy  $\pi_{k+1}^i$  becomes greater.  $\gamma_k$  begins to decay toward zero at a rate determined by both the number of actions and the user-supplied parameter. This, in turn, causes  $\tau_k^{-1}$  to decrease toward zero. Higher paying actions are chosen more frequently, which is due to the increased importance of the exponential component. Eventually, only the action with the best pay-out will be chosen.

In [6], we showed that these hybrid allocation rules did not come about arbitrarily. They were designed to yield logarithmic regret for the discrete, stochastic multi-armed bandit problem.

In particular, we applied a variety of inequalities to systematically bound the expected value of the policy probabilities for sub-optimal arms. We eventually obtained that  $\mathbb{E}[\pi_k^i] = O(1/k)$  for non-optimal arms  $a_k^i$  at round  $k$ . It follows from this bound that the regret is  $O(\log(k))$  after  $k$  arm pulls. This is an improvement from the regret bound  $O(\log^2(k))$  obtained for soft-max-based selection when employing monotone inverse-temperature sequences [26]. Note that it was necessary to perform this regret analysis, as the results obtained by Auer et al. [27] do not automatically extend to our hybrid uniform- and exponential-component update. In [27], the authors only considered a uniform-random exploration term that was driven by a monotone exploration parameter decrease schedule.

We were also able to quantify the upper and lower probability bounds for choosing a sub-optimal slot-machine arm. Such results are important, as they can be used to assess the quality of the exploration process at any arm pull past a given point during learning. Through these bounds, we were able to demonstrate that the value of information has a much higher chance of pulling the optimal arm than other schemes, including that of Auer et al. [27].

For the distribution-free AutoVoIMix, we also systematically bounded the expected value of the policy probabilities for sub-optimal arms to get that  $\mathbb{E}[\pi_k^i] = O(\log^{2\theta}(k)/k)$ . We obtained a regret of  $O(\log(k)^{1+2\theta})$  after  $k$  arm pulls. This regret bound becomes increasingly equivalent to that of SoftMix selection when  $\theta \rightarrow 0.5$  and approaches logarithmic regret as  $\theta \rightarrow 0$ . As before, it was necessary to perform this regret analysis. The weak bounds obtained by Auer et al. [13] for their distribution-free approaches were for the case where the exploration remained fixed across each arm pull. In our case, the hyperparameters are adjusted across a given simulation. As well, the exponential terms in their method and our own are differ-

ent. In either VoIMix or AutoVoIMix, the exponential terms are a function of the previously accrued rewards, not just the immediately received reward.

### 19.2.3 *Simulations and Analyses*

In the previous section, we provided an information-theoretic criterion for quantifying the expected increase in rewards associated with transformation-constrained policies. We also developed a means for optimizing this criterion, which provides a probabilistic mechanism for choosing a given arm based on its past history of expected rewards.

In this section, we quantify the empirical performance of the criterion. The aims of our simulations are multi-fold. First, we want to assess the relative performance of our value-of-information-based exploration approach with existing ones. We would also like to gauge how well our results align with theoretical expectations. Secondly, we want to determine how the value-of-information search mechanism investigates the problem domain.

Toward this end, we consider a series of difficult bandit problems. We demonstrate that the value-of-information-based VoIMix and AutoVoIMix can effectively address these problems. We first consider the case where the two exploration parameters of VoIMix and AutoVoIMix are fixed to constant values across the rounds. We then compare these results to the case where the parameters are allowed to adapt according to the cooling schedules given in the previous section. This is done to highlight the regret improvement and empirically justify the two algorithms.

We consider five other bandit algorithms for comparative purposes. Two of these algorithms,  $\epsilon$ -greedy and soft-max, were chosen due not only to their popularity, but also their commonalities with the value of information. Stochastic, soft-max-based derivatives, such as pursuit methods and reinforcement comparison, are also employed. Both approaches rely on different search mechanisms, which offers experimental variety. Lastly, we consider the deterministic, tuned and un-tuned upper confidence-bound approaches. Our simulation results reveal that the value of information outperforms these methods either in general or in specific instances. Explanations for this behavior are provided so that practitioners can determine when to potentially employ a given methodology.

#### 19.2.3.1 *Simulation Preliminaries*

The difficulty of the multi-armed bandit problem is fully characterized by two attributes: the distribution properties used to model the slot machines and the number of slot-machine arms.

For our simulations, the rewards for each of the slot-machine arms were sampled from normal distributions. We originally considered a range of additional distribu-

tions, such as uniform, inverse normal, and triangular, but found that the results were highly similar. We therefore only report the findings for the normal distribution.

The means of the slot-machine reward distributions were chosen uniformly from the unit interval and fixed for the duration of the simulation. We evaluate the algorithms for a reward distribution variance of  $\sigma^2 = 1$ , which is fixed. The value corresponds to standard deviations that are one-hundred percent of the interval containing the expected values. This yields a complex problem no matter the number of arms, as the reward distributions have a wide spread and overlap significantly. The agent must therefore judiciously sample each of the arms so that the one with the best average pay-out can be identified and consistently pulled.

Another aspect that changes the difficulty of the problem is the number of slot machines. We evaluate the algorithms for three, ten, and thirty slot-machine arms. Three arms lead to fairly easy tasks, while ten arms furnish marginally difficult ones. Over thirty arms begin to provide tasks that are challenging. Originally, we considered greater numbers of slot-machine arms. However, we found that their relative behaviors are mostly consistent with the thirty-armed case.

For the value of information, we assume that an initial policy is supplied. The results in the appendix indicate that regret is independent of the initial policy. Nevertheless, we specify that the initial policies have uniform probabilities, so as to not introduce bias that could lead to simpler problems.

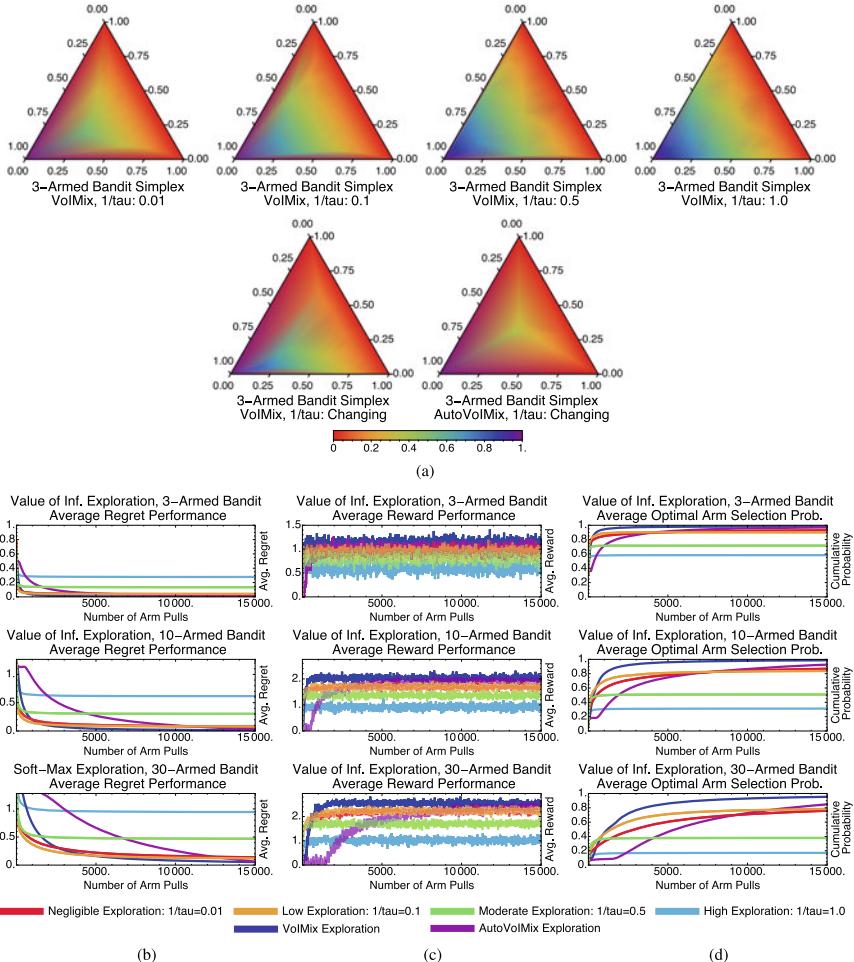
### 19.2.3.2 Value of Information Results and Analysis

We assess the performance of the value of information in two situations: when the parameters  $\tau_k$  and  $\epsilon_k$  remain fixed across arm pulls during learning and when they are updated automatically. Simulation results for both cases are presented in Fig. 19.1. Note that for all of the regret plots in this section, we divide the regret, at each round, by the number of arm pulls. Therefore, this modified regret should decrease toward zero as the number of rounds increases.

**Fixed-Exploration Amount and Performance.** Figure 19.1a contains probability simplex plots for VoIMix. Such plots highlight the arm-selection probability distributions of the iterates during the learning process for the three-armed bandit problem. They indicate how VoIMix is investigating the action space across all of the Monte Carlo trials and if it is focusing on sub-optimal arms.

The optimal action for these three-armed bandit problems is associated with the bottom, left corner of the simplex. We hence expect to see a purple-colored region near this corner of the plot, which would indicate that the optimal action is selected frequently during learning. All of the other corners should have either red- or orange-colored regions, as these actions should not be consistently chosen after a series of initial arm pulls.

The value of information operates under the premise that more information can lead to an increase in accrued rewards due to a more thorough exploration of the action space. The plots given in Fig. 19.1a show that the empirical results are aligned with



**Fig. 19.1** Results for the value-of-information-based VoIMix for the fixed- and adaptive-parameter case. The plots in **a** highlight the regions in the action probability simplex that are visited during the simulations for the three-armed bandit problem. Cooler colors (purple, blue, and green) correspond to probability triples that are encountered often during learning. Cooler Warmer (yellow, orange, and red) correspond to probability triples that are not seen often. The correct action is the one associated with the bottom, left corner of the simplex. The simplex plots were produced for fixed-parameter values and averaged across independent simulations. The plots in **b**, **c**, and **d** give the regret, reward, and optimal-arm selection cumulative probability for the three-, ten-, and thirty-armed bandit problem when the distribution variance is  $\sigma^2 = 1$ . The red, orange, green, and blue curves correspond to fixed inverse temperatures of  $\tau^{-1} = 0.01, 0.1, 0.5$ , and  $1.0$ , respectively. The purple and dark blue curves correspond to VoIMix and AutoVoIMix, respectively, where the inverse temperature and mixing parameters are allowed to vary throughout the simulation, according to the cooling schedules given in the previous section

these expectations, at least for the three-armed bandit case. For a fixed low inverse temperature  $\tau_k^{-1}$ , VoIMix is reserved in its search due to the effects of the information constraint on the prior. The iterates briefly explore the two sub-optimal arms on the top and right sides of the simplex before congregating on the left half of the simplex. The distribution for the left-hand plot of Fig. 19.1a indicates that the iterates quickly converge to the optimal arm. For high  $\tau_k^{-1}$  values, larger portions of the simplex are investigated before converging to the optimal arm. Between  $\tau_k^{-1} \approx 0.15$  and  $\tau_k^{-1} \approx 1.0$ , the simplex distribution does not change greatly.

Figure 19.1 also contains plots of the average regret as a function of time, the average reward, and the cumulative chance that the optimal arm is chosen. The first two criteria summarize how the methods handled the problem. The third criterion is relevant to assess applications in which minimizing the number of sub-optimal arm plays is crucial. The averages reported in these plots were derived from a thousand Monte Carlo simulations so as to eliminate the bias of any overly poor or good runs. They are also a function of different parameter choices that correspond to varying amounts of exploration.

Figure 19.1b, c highlight the effects of the inverse-temperature parameter  $\tau_k^{-1}$  on the performance. As  $\tau_k^{-1}$  decreases, the episode-averaged regret tends to drop rapidly during the first few arm pulls. Likewise, the average reward increases more rapidly. The effect becomes much more pronounced as the number of bandit arms is small. Both occurrences are a product of quickly finding either the best-paying arm or well-paying arms. As the number of arms increases, there is a reversal in this trend: larger values of  $\tau_k^{-1}$  begin to yield better long-term results due to the increased exploration that is needed to sample all of the arms.

Figure 19.1d captures the cumulative probability of choosing the optimal arm. These plots show that the fixed-parameter case of VoIMix was unable to reliably find the best-paying arm with a high probability. As the number of arms increased, the short-term cumulative probability dropped, regardless of the amount of exploration  $\tau_k^{-1}$  that was used. It can be seen, however, that higher amounts of exploration more frequently uncover the optimal arm. This occurrence aligns with the preceding regret and reward results.

The inverse-temperature parameter  $\tau_k^{-1}$  is not the only way to control the search amount in VoIMix. The mixing coefficient  $\gamma_k$  also dictates how finely the action space will be investigated. For the simulations in Fig. 19.1, we assumed that the mixing coefficient was 0.1. One in ten arm pulls would therefore be random. Setting it higher, for these problems, often led to increasingly more exploration than was necessary. The rewards dropped and the regret was inflated, the results of which are presented in table B.1(c). Setting the mixing coefficient lower also impacted the results, as the pay-out became worse. Such results are given in Table B.1(d).

**Automatic Exploration Amount and Performance.** The results given above are for the case where the parameter values were constant throughout the simulation. This is useful for understanding the algorithmic behaviors in different scenarios. Leaving the parameters fixed can, however, lead to linear regret in the worst case. It

is therefore not indicative of the true capabilities of VoIMix, which is known to have logarithmic regret.

We now consider VoIMix and AutoVoIMix when the temperature changes across each arm pull. In Fig. 19.1, we provide results for both VoIMix and AutoVoIMix. These results are for the cases where  $\tau_k^{-1}$  and  $\gamma_k$  are either fixed or automatically tuned according to the cooling schedules from the previous section.

Plots of the probability simplexes for the tuned versions of VoIMix and AutoVoIMix are given in Fig. 19.1a. As before, these plots highlight the distribution of iterates during the learning process. The iterates should converge to the bottom, left corner of the simplex, which is the optimal action for the three-armed bandit case. This occurs for VoIMix: after about 250–750 arm pulls, there is high chance of selecting the optimal arm. The blue- and green-colored region in the bottom, left corner of the simplex plot corroborates this claim. The dark-purple-colored region in the left, bottom corner of the simplex indicates that the probability of selecting the optimal arm becomes near-unity and stays that way throughout a majority of the learning process. The yellow- and orange-colored portions of the plot indicate that VoIMix heavily considered the arm associated with the bottom-right corner of the simplex early during learning. This was the second best-paying arm in all of our Monte Carlo trials.

Different results were observed for AutoVoIMix. The corresponding probability simplex plot in Fig. 19.1a captures that a large portion of the iterates cluster near the middle of the simplex. Each arm has a near-equiprobable chance of being chosen. This region of yellow and orange colors arises from the iterates for the first 5000 arm pulls, which is when a majority of the exploration is performed. Eventually, the optimal arm was identified. The iterates traced an almost direct path from the middle of the simplex to the bottom, left corner. This path has a slightly more green color than the paths to the other two corners, suggesting that arm probabilities from this part of the simplex were consistently selected later in the learning process. The probability for choosing the optimal arm was high near the end of learning, which led to the purple-colored region in this corner.

As shown in Fig. 19.1b, the regret for the adaptive-parameter case typically improved beyond the results of the fixed-parameter case after several hundred to thousand iterations. The rate at which the regret dropped depended on the number of slot-machine arms. Such events signaled that a high-paying arm was reliably uncovered and consistently chosen with a high probability, which can be gleaned from Fig. 19.1d. The total rewards also jumped, after a few thousand arm pulls, compared to the best results for the fixed-parameter case. This can be seen in Fig. 19.1c. This event coincided with the end of the perfect exploration period and a decay of the inverse-temperature parameter and mixing coefficient.

A drawback of VoIMix is that setting the single hyperparameter can prove difficult. This is because it relies on knowledge of the reward distributions to which gamblers are not privy. We introduced AutoVoIMix as a way to circumvent this issue. However, AutoVoIMix tends to choose subpar arms compared to the non-fixed-parameter version of VoIMix, which can be seen in Fig. 19.1d. On average, the regret and total

rewards are worse, compared to VoIMix, when relying on parameter annealing, as indicated in Fig. 19.1b, c.

**Performance Discussions.** We considered two possible instances of VoIMix: one where its parameters were fixed another where they varied. Our simulations showed that the latter outperformed the former. This was because a constant amount of exploration might explore either too greatly or not thoroughly enough to achieve good performance. Without the optimal annealing schedule that we provided, substantial trial-and-error testing would be needed to find reasonable exploration amounts for practical problems.

Our simulation results highlight that the automatically tuned VoIMix outperforms the automatically tuned AutoVoIMix for these bandit problems. This was also witnessed when adjusting the methods' hyperparameters according to the cross-entropy heuristic. Such results align with our theoretical analyses. AutoVoIMix can only approach logarithmic regret when its hyperparameter value  $\theta_k$  tends toward zero. VoIMix can achieve logarithmic regret in the limit, no matter the value of its hyperparameter  $d_k$  in the valid range.

We have additionally provided distribution plots of the arm-selection probabilities overlaid on the simplex. These plots highlighted that the best-paying arm can be uncovered when the temperature  $\tau_k^{-1}$  is either fixed or automatically adjusted. We only considered simplex distribution plots for the three-armed bandit problem. It is not possible to produce exact plots of the probability simplex for bandit problems with more arms, since those probability vectors often cannot be losslessly embedded in a three-dimensional domain.

When manually lowering the temperature, the iterates became more focused and targeted the optimal arm more frequently. This is because the temperature value ensures that only the best-paying arm is chosen with a high probability. When raising the temperature, the search process near-uniformly considered a larger portion of the simplex. The chance of choosing an arm approaches that of any other arm. Even when considering  $\tau_k^{-1} = 1.0$ , which represents a high amount of policy change, the optimal arm could still be uncovered about half of the time. The corresponding simplex plot shows that the iterates do routinely pull the optimal arm after a few thousand plays.

For our simulations, we sorted the arm rewards so that the best-playing arm is in the bottom-left corner of the simplex and the worst playing arm is in the top-middle corner. We thus expected that the iterates would tend to cluster on the bottom half of the simplex. However, the worst-paying arm appears to be played just as frequently as the second best-paying arm. Additional analysis revealed that this was due to the high variance of the reward distributions: it was often possible for the worst arm to have a better rewards over many arm pulls than the second-best arm. Both arms would therefore have a high probability of being chosen across multiple iterations. The choice of one of these arms over the other changed frequently, which is why the search process near-uniformly investigated these regions of the simplex values of  $\tau_k^{-1}$  away from one.

When automatically tuning the inverse temperature according to VoIMix, the resulting simplex plot indicates that it quickly starts to play the optimal arm. Only

the two best-paying arms are routinely sampled, which did not happen in the fixed-parameter case. This materialized due to a steep drop in the exploration rate during training, which would emphasize the selection of arms with high expected rewards. We witnessed this behavior for problems with many more arms. The tuned version of VoIMix will thus likely perform well for real-world bandit problems. It automatically modifies the amount of search exploration in proportion to the intrinsic complexity of the problem. This is a major advantage compared to manually choosing the exploration amount.

For the tuned AutoVoIMix, the center regions of the simplex are more thoroughly explored before the optimal arm is consistently pulled. Such investigations may yield little useful insights about the slot-machine distributions after a few hundred to thousand arm pulls. Many arm pulls will hence be required before the regret can be decreased. This can prove problematic for application domains where the physical time and effort needed to supply rewards to the agent is high; the utility of drug-based treatments in clinical trials is one such example.

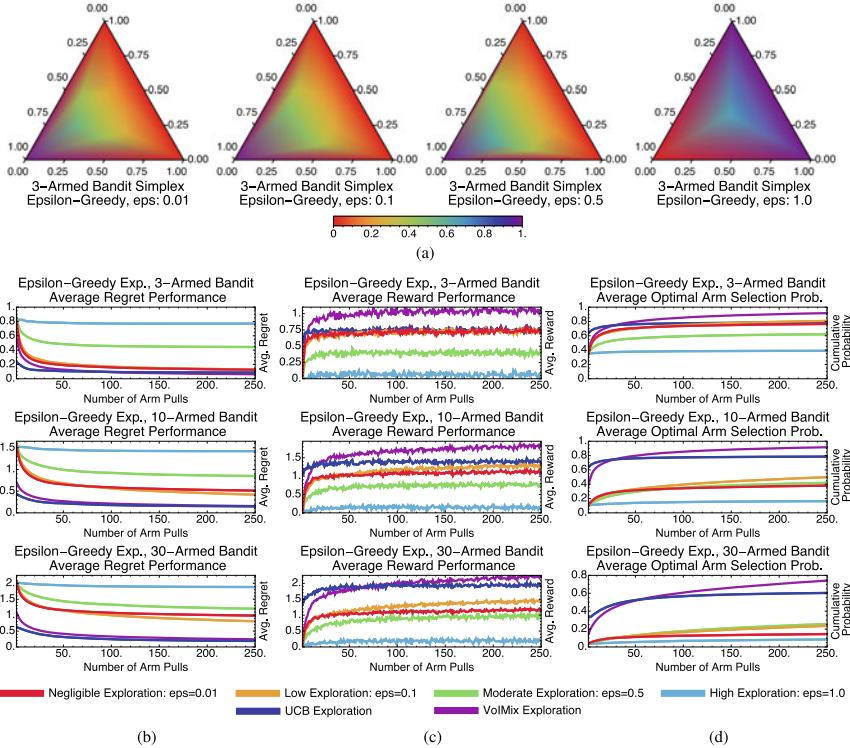
Both VoIMix and AutoVoIMix can achieve the same long-term rewards and appear to reliably identify the optimal arm. However, VoIMix does so at a much faster rate, leading to a better short-term regret. If practitioners can rely on auxiliary knowledge about their bandit problems to choose good hyperparameter values, then VoIMix should be favored compared to AutoVoIMix. Otherwise, AutoVoIMix should be utilized, with the understanding that the ease of choosing its hyperparameter is traded off against a, potentially, poor initial regret.

### 19.2.3.3 Methodological Comparisons

To provide context for these results, we have compared the value of information to other commonly used approaches for addressing the bandit problem. These include epsilon-greedy and soft-max exploration, both of which are stochastic. We also consider the deterministic upper confidence-bound method. As before, the results given are averaged over a thousand Monte Carlo trials.

**Stochastic Methods:  $\epsilon$ -Greedy and Soft-Max.** The  $\epsilon$ -greedy-based GreedyMix and soft-max-based SoftMix [26] are two of the more popular heuristic approaches for solving multi-armed bandits. GreedyMix is based upon the principle of taking random actions and random times and greedily choosing the best-paying arm in all other instances. SoftMix relies on a weighted-random approach to arm selection, similar to VoIMix and AutoVoIMix. That is, it weights each of the arms according to their expected pay-outs and stochastically chooses an arm using those weights.

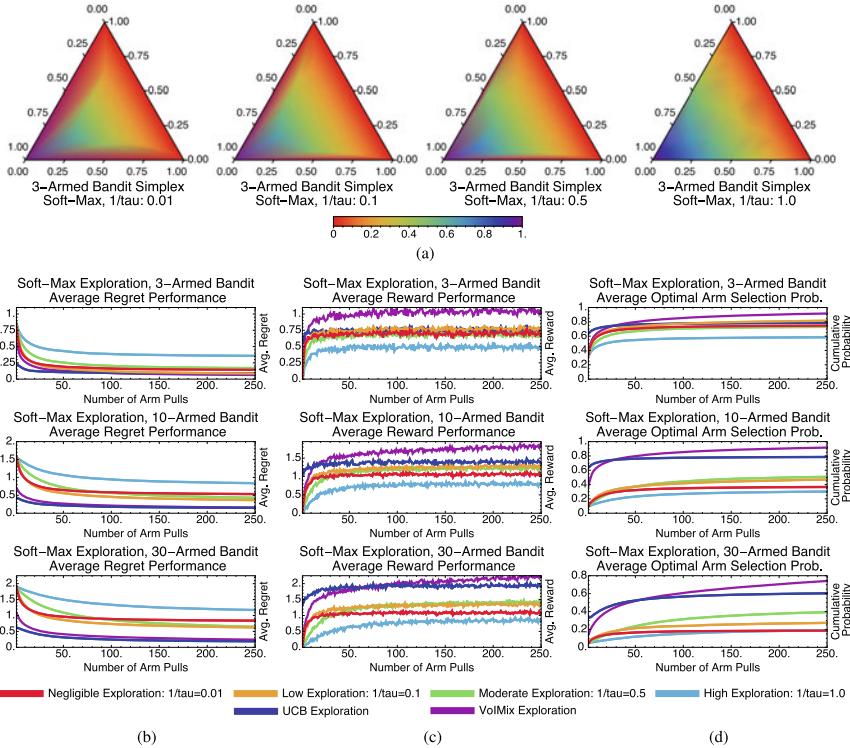
Results for GreedyMix and SoftMix are given in Figs. 19.2 and 19.3, respectively. Despite only achieving a log-squared regret, SoftMix does well in the short-term due to its weighted-random style of search. It accounts for the expected reward when deciding whether or not to explore new arms. Accurate means for well-paying arms are also created, which aids in choices that will initially reduce regret and improve rewards by a substantial amount; this can be seen in Fig. 19.2b, c.



**Fig. 19.2** Results for the  $\epsilon$ -greedy-based GreedyMix for the fixed-parameter case. The plots in **a** highlight the regions in the action probability simplex that are visited during the simulations for the three-armed bandit problem. The correct action is the one associated with the bottom, left corner of the simplex. The simplex plots were produced for fixed-parameter values and averaged across independent simulations. The plots in **b**, **c**, and **d** give the regret, reward, and optimal-arm selection cumulative probability for the three-, ten-, and thirty-armed bandit problem when the distribution variance is  $\sigma^2 = 1$ . The red, orange, green, and blue curves correspond to fixed-exploration values of  $\epsilon = 0.01, 0.1, 0.5$ , and  $1.0$ , respectively

GreedyMix, in contrast, merely selects arbitrary arms, regardless of their quality. This haphazard selection can impede exploration early in the learning process, as sub-optimal arms may be frequently chosen. Poorer results than SoftMix are usually returned. This can be seen when comparing the plots from Fig. 19.2b–d to those from Fig. 19.3b–d.

Figures 19.2a and 19.3a illustrate that GreedyMix and SoftMix yield probability distributions that resemble those for the fixed-parameter VoIMix. Low amounts of exploration perform best for the three-armed bandit problem. The iterates tend to congregate in bottom-left corner of the simplex, which corresponds to the best-paying arm. The only exception is the right-most simplex plot in Fig. 19.3a. The iterates continuously explore the entire simplex, since a new action is randomly chosen at each step.



**Fig. 19.3** Results for the soft-max-based SoftMix for the fixed-parameter case. The plots in **a** highlight the regions in the action probability simplex that are visited during the simulations for the three-armed bandit problem. The correct action is the one associated with the bottom, left corner of the simplex. The simplex plots were produced for fixed-parameter values and averaged across independent simulations. The plots in **b**, **c**, and **d** give the regret, reward, and optimal-arm selection cumulative probability for the three-, ten-, and thirty-armed bandit problem when the distribution variance is  $\sigma^2 = 1$ . The red, orange, green, and blue curves correspond to fixed inverse temperatures of  $\tau^{-1} = 0.01, 0.1, 0.5$ , and  $1.0$ , respectively

Compared to VoIMix, both SoftMix and GreedyMix perform worse. GreedyMix will sometimes uniformly sample sub-optimal arms during the exploration phase. Its regret for the best-performing policy is thus higher, while the rewards are also reduced compared to the best results from VoIMix and AutoVoIMix. This is highlighted in Fig. 19.2b–d for VoIMix in the case where both the parameter and hyperparameter are adjusted across arm pulls. SoftMix also has issues. It has a habit of oversampling sub-optimal arms, which was caused by improperly estimated average rewards. Both the regret and total pay-out are worse, respectively, than VoIMix and AutoVoIMix. This is captured in Fig. 19.3b–d.

**Deterministic Method: Upper Confidence Bound.** One of the most widely studied strategies for addressing multi-armed bandits are the deterministic upper confidence-

bound schemes [12, 32]. They operate under the principle of optimism in the face of uncertainty. This entails choosing actions under the assumption that the unknown mean pay-offs of each arm are as large as plausibly possible, based on the previously observed samples.

There are many versions of the upper confidence-bound idea. The simplest of these starts by pulling each arm once, from which it begins to construct an estimate of the arms' empirical means. It then greedily picks the arm with the highest pay-out, subject to an arm-frequency constraint [27]. This constraint ensures that arms that have not been played often, perhaps due to poorly estimated means, will eventually be selected. A permutation of this idea, called tuned upper confidence bound [33], takes into account both the means and the variance of the arm pay-outs, which purportedly aids in dealing with high-variance slot machines. Most, if not all, upper confidence-bound approaches are known to achieve logarithmic regret without the need for parameter tuning.

The un-tuned upper confidence-bound method results are presented in Figs. 19.2 and 19.3. The tuned variant produced virtually identical results for these problems, so those results are not shown in these plots. Both methods generated reasonably good time-averaged regret, per round, at the beginning of the simulations. In many cases, their initial performance exceeded that of the tuned VoIMix. The upper confidence-bound methods also outperformed the remaining stochastic exploration approaches. They did converge much more slowly to the optimal arm than VoIMix, though. This led to lower pay-outs over a much larger set of plays.

**Comparison Discussions.** These results indicate that the tuned version of VoIMix outperforms many conventional exploration schemes in the short term. Long term it does too. We did not compare the results to the tuned version of AutoVoIMix, as its initial exploration period was longer than the number of pulls that we considered in these simulations; its results would be quite poor during this time. Over a greater number of arm pulls, the performance of AutoVoIMix approaches that of VoIMix. It will, consequently, yield better results than some of these alternate schemes.

The performance that we witnessed for many of these algorithms is predicated on choosing reasonable parameter or hyperparameter values. The average increase in total regret for improperly tuned algorithms was approximately thirty percent for VoIMix and forty percent for SoftMix and GeedyMix. Reinforcement comparison likewise suffered an increase in regret of about forty percent, as did the pursuit method, which saw a seventy percent raise. Moreover, seemingly optimal parameter values for one multi-armed bandit scenario could suddenly become one of the worst in another. This finding indicates a need for online parameter adjustment, akin to what is done by the information-theoretic scheme that we introduced for adapting the value-of-information hyperparameter.

Our simulations indicated that the performance of SoftMix exceeded that GreedyMix. In the long term, GreedyMix should improve beyond SoftMix, especially if the currently known regret bounds are tight. This is because an unlucky sampling of arms in SoftMix can bias the expected returns and favor sub-optimal arms. The optimal arm may have a low probability of being sampled, especially after

many pulls. In contrast, all arms have the equal chance of being selected during the exploration phase of GreedyMix. Any poorly estimated reward means, especially that of the optimal arm, can hence be made more precise. The optimal arm will therefore be chosen more frequently once this occurs.

Despite both methods achieving a logarithmic regret, VoIMix outperformed the fixed-parameter GreedyMix over the first few hundred arm pulls. This was also observed in the case where the parameters for GreedyMix were annealed across arm pulls. This discrepancy is likely due to the constant regret complexity factors, which are typically excluded. Those factors for VoIMix have a lower magnitude than those for GreedyMix.

In short, as noted by Cesa-Bianchi, et al. [26, 30], and echoed by several other authors in the literature, finding good annealing schedules for stochastic methods can be difficult. Some cooling schedules may consistently draw sub-optimal arms during the learning process, even after having estimated all of the means correctly. In some cases, they may focus too early to a sub-optimal arm and never recover afterward. The value of information appears to avoid such issues, due to its mixture-model formulation. That is, there is always a chance that a non-optimal arm can be selected uniformly at random, even if its current reward estimate is incorrect.

Our simulations also revealed that while upper confidence-bound methods achieve superior performance during the first hundred arm pulls, it lags behind VoIMix over a longer number of plays. This behavior likely stems from the greedy arm-selection process of the upper confidence-bound methods. Initially, only the highest performing slot machine will be chosen by these methods, as the arm-frequency constraint will do little to sway the selection. This arm may be sub-optimal, yet have an unusually high reward due to the variance. The optimal arm, in contrast, may have had an uncharacteristically low reward the round that it was chosen, causing it to be ignored until later in the learning phase when the arm-frequency constraint will ensure that it is picked. Stochastic approaches have more opportunities to choose optimal arms that, initially, appear to be sub-optimal.

Both the tuned and un-tuned upper confidence-bound methods yielded near-identical regrets and rewards in our simulations. Increasing the arm variance did little to change the results. The pay-outs accrued by each of the algorithms that we consider thus appear to only be affected by two characteristics. These are the number of slot-machine arms and the mean separation of the corresponding machine distributions.

This finding has ramifications for the types of regret bounds that investigators can expect to obtain. It also informs about the methodological traits that should be emphasized when developing new bandit algorithms. Such a suggestion runs counter to recent theoretical endeavors that have focused on obtaining improved regret bounds by considering the second-order reward moments of specific distributions. Nevertheless, our results indicate that accounting for high-order moments of the reward distribution will not prove fruitful, as the type of reward distribution had only a marginal effect on the algorithm behaviors and performance. This is a trait exhibited not only by the value of information, but also for the other stochastic and deterministic approaches that we considered here.

### 19.2.4 Conclusions

In this section, we considered the exploration of single-state, multiple-action Markov decision processes. We showed that an information-theoretic means of choosing optimal actions was possible for this environment abstraction. Our approach is based on Stratonovich's value-of-information criterion for a single random variable, which defines a type of free-energy extremization problem.

As we noted at the beginning of this chapter, Stratonovich never considered practical applications of his criterion. This was, in part, because he did not explicitly derive expressions for updating the decision-making probabilities; he only considered the algebraic solutions for plotting the normal and abnormal branches of the criterion for simple Bayesian systems. In this section, we showed that the optimization of the value of information gives rise to a soft-max-like policy search. Exploration is therefore driven by weighted randomness. The weights are determined by a parameterized Gibbs distribution, which operates on the rewards associated with pulling various arms. The parameter, a type of inverse temperature, dictates the amount of exploration that is conducted in the policy space.

A purely soft-max-based action selection suffers from a well-known flaw. That is, an infinite number adjustment of the inverse-temperature parameter is needed to convert from a purely random exploration of the policy space to pure exploitation. To resolve this issue, we created a mixture model from the Gibbs distribution that can more effectively transition from one extreme to the other. This model defines the action-selection probabilities to be a mixture of the exponential-based Gibbs-distribution component and a uniform-distribution component. The influence of either of these components is determined by a single mixing parameter.

Both mixture-model parameters have a profound impact on the accrued rewards. We considered two possible means of automatically tuning the parameters during the search process. The first approach, VoIMix, is optimal, as it achieves logarithmic regret for the multi-armed bandit problem. It does, however, rely on a user-specified term that is difficult to set since it relies on details about the problem that are not normally available *a priori*. This limits its practical use. The second approach, AutoVoIMix, is nearly optimal, as it can be arbitrarily close to logarithmic regret. The user-selectable term is much easier to set for this case, though, as it does not depend on any *a priori* unknown quantities.

We assessed the search capability of the value of information through a battery of simulations. This is another contribution of our paper. When manually choosing the amount of exploration, VoIMix either performs similarly or outperforms conventional heuristics, such as  $\epsilon$ -greedy and soft-max selection, in terms of regret and overall reward. When annealing the exploration amount, both VoIMix and AutoVoIMix outperformed these criteria. This was expected.  $\epsilon$ -greedy does not account for the reward associated with action choices. It is therefore possible for poor rounds of initial exploration to occur, which limits the short-term obtainable rewards. This occurs whether the amount of exploration is either manually fixed or automatically chosen so as to achieve logarithmic regret. Soft-max selection, in contrast, is not

known to exhibit logarithmic regret. Existing implementations of soft-max selection also have the flaw that the consistent sampling some sub-optimal arm may prevent the optimal arm from being sampled enough.

We also analyzed the performance of the tuned and un-tuned versions of VoIMix against AutoVoIMix. The former leads to steep drops in regret due to an effective action exploration, provided that the parameter values are in the valid range. The regret for the tuned VoIMix approach is often better than the best fixed-exploration-amount cases after only a few hundred arm pulls. The optimal arms are routinely identified with a probability near one. AutoVoIMix, in contrast, has the tendency to unnecessarily search the action space. A great many arm pulls are needed for the regret to become small, even when its hyperparameters are tuned according to our cross-entropy heuristic. While both VoIMix and AutoVoIMix can achieve the same regret and instantaneous pay-out in the limit, the former would be poorly suited for real-world, time-sensitive bandit problems where few arm pulls are made before learning stops.

Our theoretical analyses indicate that an adaptation of the exploration rate is needed for the value of information to achieve logarithmic regret. This was corroborated by our simulation results: fixed amounts of exploration led to poorer results. In our future work, we will ascertain if any of this theory can be applied to the multi-state case so that the optimal average cost improvement can be obtained across a series of episodes.

## 19.3 Exploring Multiple-state, Multiple-Action Markov Decision Processes

### 19.3.1 Literature Survey

Many of the techniques developed for exploring single-state, multiple-action Markov decision processes have excellent theoretical guarantees. Their utility for practical problems is, however, lacking. This is largely due to the simplifying assumptions of the multi-armed bandit problem, the most prudent of which is that action choices in multi-armed bandits do not affect future states and hence future obtainable reinforcements. The actions agents take in certain environments also often affect the future states that are visited. Many of these formally justified approaches hence cannot be applied to multiple-state, multiple-action Markov decision processes.

Several competing strategies have been proposed to deal with these issues. One example involves a Bayesian treatment of the exploration process during reinforcement learning [34–36]. Starting with a prior distribution of parameters in the underlying Markov decision process, these methods repeatedly update the posterior distribution of the parameters conditional on the observation history. The action that maximizes the expected future rewards, with respect to the posterior distribution, is taken at each step [37].

An advantage of Bayesian reinforcement learning is that it provides a principled means of addressing the exploration–exploitation problem. The posterior distribution arising from Bayes’ rule naturally captures the complete knowledge of the states, subject to the chosen parameterization. It is therefore possible for an agent to choose actions that maximize the expected gain with respect to this knowledge.

Some of the early foundations for Bayesian exploration were provided by Silver [38]. Silver considered three different scenarios. The first involves the concept of multi-matrix Markov processes. Such processes utilize one of several known transition matrices. A Bayesian approach is then used to sequentially modify a probability vector over candidate matrices. Silver studied properties of multi-matrix Markov processes, such as mean recurrence times, and used these properties to explain aspects of decision-making in this context. The second scenario assumes that the transition probabilities are known exactly. The rewards, however, are random variables.

Silver [38] also considers the more general case where the transition probabilities are random variables. He shows that the multi-dimensional beta distribution is the most convenient distribution, in a Bayesian sense, to place over probabilities of a single row of the transition matrix. He includes a strategy for assigning prior distributions and examines the effect of randomly distributed transition probabilities upon steady-state probabilities, state occupancy times, first-passage times, and transient behavior. Such findings are utilized to assess decision problems, with a particular emphasis on steady-state behavior and observation costs. Much of this work has been generalized and made more rigorous by Martin [39].

Cozzolino et al. [40] consider a practical treatment of Silver’s Bayesian treatment of Markov decision processes with unknown transition probabilities. They propose a dynamic programming formulation for determining optimal strategies. However, they concluded that applying their approach to non-trivial problems is impractical. They then investigated heuristics for making sequential decision and conduct Monte Carlo simulation studies.

Satia et al. [41, 42] developed a Bayesian formulation of decision-making for which the transition probability uncertainty is expressed as conjugate probability distributions. This permits the enumeration of the decision tree corresponding to a sequence of alternate decisions. The authors propose branch-and-bound techniques for using this tree to determine the optimal action sequence. Such a technique works by maintaining upper and lower bounds on the optimal discounted future return for each hyperstate node in the existing tree. At any stage during the optimization process, if the upper bound for some decision is less than the lower bound for some other decision, then the branches emanating from that state need not be enumerated and further explored. This ability to avoid unnecessary action sequence branches significantly reduces the amount of computation that needs to be performed, making it possible to apply Bayesian reinforcement learning to non-trivial problems.

Bayesian action selection is optimal in the sense that it takes the posterior into account and finds a Bayesian-optimal balance between exploration and exploitation. However, the use of Bayesian techniques in practice is limited for two main reasons. First, such approaches all require a prior distribution over candidate Markov decision processes. Such priors are difficult to choose for a variety of real-world problems.

Even with a seemingly appropriate prior selection, there may be instances where the agent fails to uncover the optimal policy in finite time. Secondly, computing the optimal value function over posterior distributions can be computationally expensive [43]. Some progress has been made, recently, to deal with this issue [44–46]. However, the computational performance still is not adequate for many complicated problems.

Due to the computational scaling issues of Bayesian methods, many practitioners defer to so-called ad hoc or heuristic approaches [47]. These heuristics may be interpreted as trying to efficiently investigate portions of the policy space by trying new actions and retrying previously taken actions for different scenarios. The amount of exploration is typically application- and parameter-dependent. If the parameters are properly tuned, then the value function estimates are guaranteed to converge to their optimal values. Additionally, if the exploratory action-selection tendencies are slowly phased out, then the optimal policy will asymptotically align itself with the greedy policy.

One of the simplest exploration strategies simply selects actions randomly according to a uniform distribution. This leads to an undirected form of exploration, which, one may suspect, would lead to rather inefficient learning. Whitehead [48] demonstrated otherwise, however. In his research, he examined the consequences of random-walk exploration in the context of deterministic grid-like environments with explicit goal states. He showed that, in such settings, learning techniques using uniform-random exploration are expected to scale exponentially with the size of the state space. If investigators adopt a simple counter-based action-selection rule that drives the agent to randomly choose the least-taken action, then the goal state is always found in time that is polynomial in the size of the state space. This assumes either access to the transition probabilities or that such transitions can be estimated, which is not always possible for real-world environments.

Perhaps the most widely used heuristic for virtually every incarnation of reinforcement learning is epsilon-greedy action selection [49–51]. Epsilon-greedy involves selecting an action, uniform randomly, with probability epsilon. If a random action is not taken, then the agent instead chooses the action with the highest value function magnitude. Albeit simple, it has been shown that epsilon-greedy is greedy in the limit, which implies that it will eventually yield actions that are sampled from the greedy policy [52]. This assumes a reasonable decay schedule for epsilon across episodes. For many practical problems, though, it can be difficult to a priori define such a schedule, which means that this approach may produce policies with sub-optimal actions.

The epsilon-greedy approach is simple, but it is not ideal in practical problems where only a limited amount of agent-environment interaction is possible. It also ignores the influence of the actions on the expected costs. Another widely used strategy, which can overcome these issues, is the soft-max approach [53]. This method entails weighting the action choices according to a Boltzmann distribution that is parameterized by the action's influence on the expected cost. An action for each new state is chosen in a weighted-random fashion, where actions that have a lower cost are more likely to be selected. An advantage of the soft-max-based selection

is that it can implement a continuum of action choices from completely random to completely deterministic and cost driven. The underlying action-selection strategy is determined by a user-selectable parameter. Barto et al. [54] provided some of the earliest attempts of deterministically tuning this parameter across episodes. Optimal parameter tuning is still, however, an open problem.

Many more ad hoc exploration schemes exist [55, 56] and have been successfully applied to a gamut of practical problems. This is, in part, because they make few operational assumptions about the environments. The same schemes can therefore be applied to many problems without any changes, save for parameter choices. This contrasts with Bayesian exploration approaches, where much of the prior knowledge may change from problem to problem. These ad hoc schemes also tend to introduce little computational overhead in the reinforcement-learning process.

There are, however, shortcomings of these heuristic schemes. The most glaring example is that they do not directly quantify the trade-off between the exploration rate and the obtainable reinforcements. They hence may be searching rather poorly, despite their convergence guarantees. For instance, it has been demonstrated they often perform either highly local or highly global surveys of the policy space. That is, they tend to produce sequential series of policy iterates that are either clustered together in small neighborhoods or greatly spread apart. Only small sections of the policy space may therefore be explored, which can impact the agent's performance in finite-episode situations. The amount of computation needed to more globally investigate the policy space can be immense. Another example of their shortcomings is that there are, currently, no principled ways to set and optimally tune the parameters during the learning process. This is a massive oversight, since the exploration strategy is application-dependent and highly sensitive to these values. We correct both of these deficiencies with the value of information.

### 19.3.2 Methodology

In the previous section, we considered the case where the Markov decision process contained only a single state; each action would lead to a transition back to the same state. Now, each action has a chance of leading to a transition between different states. We hence have two random variables on which we can operate. We would like to utilize the value of information to explore the action choices for various states in this more general abstraction. In doing so, the effect of exploring a certain amount on the obtainable reinforcements can be quantified, which is a trait that no other search mechanisms appear to share.

In this setting, probabilistic policies therefore are no longer marginal distributions over the actions. Rather, they are conditional distributions over states  $s_k^j \in \mathcal{S}$  and actions  $a_k^i \in \mathcal{A}$ :  $\pi(s_k^j, a_k^i) : \mathcal{S} \times \mathcal{A} \rightarrow \mathbb{R}_+$ ,  $\pi(s_k^j, a_k^i) = p(a_k^i | s_k^j)$  at discrete-time  $k$ . If the transition probabilities between states are known, then the policy can be found using dynamic programming. On the other hand, if they are not known, then

reinforcement-learning techniques are applied to learn the policy from experience. This typically takes the form of learning an associated action-state value function  $Q(s_k^j, a_k^i) : \mathcal{S} \times \mathcal{A} \rightarrow \mathbb{R}$ , which quantifies the reinforcements obtained by taking a given action in a given state at a specified time, then following the current policy thereafter. The value function is defined in terms of the discounted future rewards associated with a particular sequence of actions

$$Q(s, a) = \mathbb{E}_{\pi(s, a)} \left( \sum_{k'=0}^{\infty} \gamma^t r_{k+k'+1}(s_{k+k'}^j, a_{k+k'}^i) \middle| s_k^j = s, a_k^i = a \right),$$

for some discount factor  $\gamma$  in the unit interval, which ensures that the summation is finite. Here,  $r_k(s_k^j, a_k^i) : \mathcal{S} \times \mathcal{A} \rightarrow \mathbb{R}$  represents the reward obtained for experiencing the transition associated with taking action  $a^i$  in state  $s^j$  at time  $k$ .

The problem of finding the best policy can be written in terms of the action-state value function as

$$\max_{\pi_{a_k^i}(s_k^j) : \mathcal{S} \rightarrow \mathcal{A}} \left( \sum_{s_k^j \in \mathcal{S}} \sum_{a_k^i \in \mathcal{A}} p(s_k^j) \pi(s_k^j, a_k^i) Q(s_k^j, \pi_{a_k^i}(s_k^j)) \right) = \sum_{s_k^j \in \mathcal{S}} p(s_k^j) \max_{a_k^i \in \mathcal{A}} \left( \pi(s_k^j, a_k^i) Q(s_k^j, a_k^i) \right)$$

assuming that the reinforcements are treated as positive rewards. Markov decision processes are considered to be solved whenever the policy maximizes the future expected rewards from any starting state. As we noted above, we will use the value of information to explore the action choices so that this environment abstraction can be solved.

Toward this end, we assume that a default probabilistic action-selection policy  $p(a_k^i) = \pi(a_k^i)$ ,  $a_k^i \in \mathcal{A}$ , is provided. The value of information seeks to change this default agent behavior into a policy  $\pi(s_k^j, a_k^i) = p(a_k^i | s_k^j)$  that yields the highest improvement in reinforcements. The way in which the default behavior changes is dictated by the expected divergence bound between it and the policy. This is equivalent to saying that the mutual dependence between states and actions cannot exceed a specified bound. When there is less dependence between states and actions, the choice of the best action becomes increasingly uncertain. Actions are hence chosen at random, which leads to an exploration-intensive search. The agent will therefore favor risky actions which improves the chances of it obtaining more reinforcements. The higher the bound, the greater the dependence between states and actions. The best-performing action will be chosen with increasing frequency, which causes the agent to implement risk-adverse action-selection strategies. This leads to an exploitation-intensive search, which can impact the obtainable rewards if the current policy does not encode the environment dynamics well.

As we will show, the information bounds are dictated by a single parameter that emerges from converting this constrained optimization problem into an unconstrained optimization problem. This parameter simultaneously dictates how frequently actions that differ from the current policy are taken and hence how risky the agent's actions will be.

### 19.3.2.1 Value of Information

There are two extreme cases to consider when finding an action-selection policy that solves the total expected penalty criterion. The first case is when no information about the value of the random variable  $s_k^j \in \mathcal{S}$  is available. That is, states carry no information about the actions that should be selected. There is only one way to choose the optimal estimator  $a_k^i \in \mathcal{A}$  when this occurs: maximize the average action-value function  $\mathbb{E}[\max_{a_k^i \in \mathcal{A}} \mathbb{E}[Q(s_k^j, a_k^i) | s_k^j]] = \max_{a_k^i \in \mathcal{A}} \mathbb{E}[Q(s_k^j, a_k^i)]$ . The action-selection policy is hence a uniform distribution, there is complete uncertainty over what action should be chosen. On the other hand, if the states carry total information about the actions, then  $\mathbb{E}[\max_{a_k^i \in \mathcal{A}} \mathbb{E}[Q(s_k^j, a_k^i) | s_k^j]] = \mathbb{E}[\max_{a_k^i \in \mathcal{A}} Q(s_k^j, a_k^i)]$ . In this instance, the action-selection policy is a delta function, as there is no uncertainty about the optimal estimator.

For many problems, only partial information is available. There is a challenge of how exactly to best use it. One way of doing this is through the value of information, which connects both the complete-information and no-information cases with a smooth, nonlinear transition that relates obtainable costs and information. That is, it provides an optimal conversion between varying degrees of partial information and expected obtainable costs.

The value of information combines Shannon's information theory with the concept of average losses or risk, which characterizes the quality of decisions being made. This criterion can be defined as the maximum benefit that can be gained by a given quantity of information in order to minimize average losses. As such it is an independent branch of information theory that can be used in many practical applications.

For Markov decision process-based reinforcement learning, the criterion can be utilized to choose optimal-cost actions for each state. This is done by minimizing the difference in average agent costs for the no-information case with the total expected agent costs for the partial-information case,

$$\underbrace{\max_{a_k^i \in \mathcal{A}} \left( \sum_{s_k^j \in \mathcal{S}} p(s_k^j) Q(s_k^j, a_k^i) \right)}_{\max_{a_k^i \in \mathcal{A}} \mathbb{E}\left[Q(s_k^j, a_k^i)\right]} + \underbrace{\max_{\pi(s_k^j, a_k')} \left( \sum_{s_k^j \in \mathcal{S}} \sum_{a_k' \in \mathcal{A}} p(s_k^j) \pi(s_k^j, a_k') Q(s_k^j, \pi_{a_k'}^*(s_k^j)) \right)}_{\max_{\pi(s_k^j, a_k')} \mathbb{E}\left[\max_{a_k' \in \mathcal{A}} \mathbb{E}\left[Q(s_k^j, a_k') \mid s_k^j\right]\right]} \quad (19.1)$$

Here, the term  $\pi_{a_k'}^*(s_k)$  represents the best action  $a_k' \in \mathcal{A}$  chosen when using the optimal policy.

For the second term in (19.1), we have that the conditional probabilities representing the policy are subject to an information constraint, which we take to be Shannon mutual information

$$\pi(s_k^j, a_k^i) \text{ such that : } \underbrace{\sum_{s_k^j \in \mathcal{S}} p(s_k^j) \sum_{a_k^i \in \mathcal{A}} \pi(s_k^j, a_k^i) \log \left( \frac{\pi(s_k^j, a_k^i)}{p(a_k^i)} \right)}_{\mathbb{E} \left[ D_{\text{KL}}(\pi(s_k^j, a_k^i) \| p(a_k^i)) \right]} \leq \varphi_{\text{inf}}, \quad \varphi_{\text{inf}} > 0. \quad (19.2)$$

This constraint is parameterized by a positive, user-selectable value  $\varphi_{\text{inf}}$ . This value artificially limits how much information the states should carry about what actions should be taken.

The criterion defined by (19.1) and (19.2) define a difference in expected costs. They can be described as follows:

- **First Optimization Term: No-Information Returns.** The first term  $\max_{a_k^i \in \mathcal{A}} \mathbb{E}[Q(s_k^j, a_k^i)]$  captures the possible returns for a policy in which no information about the actions can be inferred from the states. This is used to establish the baseline agent performance, as it is the worse possible cost that can be obtained by the agent. If the states are not informative, then the optimal action is based solely on the state random variable distribution. If, however, the states are informative, then the returns for the simplest policy will be offset by a second term.
- **Second Optimization Term: Informative Returns.** The second term is based on the expected return using a modified action-value function,  $\max_{\pi(s_k^j, a_k^i)} \mathbb{E}[\max_{a_k^i \in \mathcal{A}} \mathbb{E}[Q(s_k^j, a_k^i)]]$ . It is attempting to find a policy  $\pi(s_k^j, a_k^i)$  that produces the best costs and changes only by a certain amount from an action-selection prior  $p(a_k^i)$ . For this term, the divergence bound between states and actions is assumed to be nonzero. States therefore carry some information about what action should be chosen, which allows for the formation of a non-uniform policy. In this instance, the magnitude of this second term  $\max_{\pi(s_k^j, a_k^i)} \mathbb{E}[\max_{a_k^i \in \mathcal{A}} \mathbb{E}[Q(s_k^j, a_k^i)]]$  will be larger than the first  $\max_{a_k^i \in \mathcal{A}} \mathbb{E}[Q(s_k^j, a_k^i)]$ . If the bound is zero, then the value of information is zero.

As the number of training episodes becomes unbounded, the agent will have complete knowledge of the environment, assuming that the complexity-control parameter  $\varphi_{\text{inf}}$  is equal to the state random variable entropy  $\varphi_{\text{inf}} = -\sum_{s_k^j \in \mathcal{S}} p(s_k^j) \log(p(s_k^j))$ . This second term will eventually produce returns that converge to those from the full-information case. The agent's behavior becomes entirely deterministic in such a situation, as the policy is a delta function for each state. Otherwise, the agent's behavior will be semi-random.

The constraint term in (19.2) can be described as follows:

- **Constraint Term: Information Bound.** The policy complexity is determined by a Shannon information constraint  $\mathbb{E}[D_{\text{KL}}(\pi(s_k^j, a_k^i) \| p(a_k^i))]$ . This constraint determines how much information the states carry about the actions. This term is bounded above by  $\varphi_{\text{inf}}$ , which implies that the mutual dependence between the states and actions will be artificially limited.

Naturally, the benefit yielded by the received information is related to the reduction in costs. As  $\varphi_{\text{inf}}$  is increased, the states carry more information about what actions should be taken. The agent costs, as defined by the difference in the first and second terms, are expected to monotonically decrease. The converse is true for decreasing values of  $\varphi_{\text{inf}}$ . The costs are expected to increase, as there is less information about what action should be taken in a given state.

The constraint parameter  $\varphi_{\text{inf}}$  has an impact on the amount of exploration. As  $\varphi_{\text{inf}}$  decreases toward zero, the available search space is explored coarsely, and the agent begins to exploit more often than it explores. The size of the jumps that the policy iterates take in neighborhoods of the search space can increase compared to higher values of  $\varphi_{\text{inf}}$ . The frequency of the jumps decreases compared to higher values of  $\varphi_{\text{inf}}$ . When  $\varphi_{\text{inf}}$  is zero, exploration becomes a random walk. The agent may therefore take an inordinate amount of time to complete an objective. When the parameter  $\varphi_{\text{inf}}$  is set near to or above the state entropy, the action-state space is finely quantized. The agent therefore attempts to maximize the possible returns. The policy iterates move around frequently in small regions of the policy space.

The value of information, as defined by (3.1) and (3.2), therefore describes the best improvement in reinforcements that can be achieved for a specified information bound.

The objective function given by (3.1) and (3.2), unfortunately, has some practical difficulties. Investigators must know the minimum expected cost associated with the globally optimal policy. To avoid needing such details, we reformulate (3.1) in a way that depends only on the estimate of the optimal policy. An advantage of doing this is that the resulting criterion is easily seen to be convex. This criterion is given by

$$-\min_{\pi(s_k^j, a_k^i)} \underbrace{\sum_{s_k^j \in \mathcal{S}} p(s_k^j) \sum_{a_k^i \in \mathcal{A}} \pi(s_k^j, a_k^i) \log \left( \frac{\pi(s_k^j, a_k^i)}{p(a_k^i)} \right)}_{\mathbb{E} \left[ D_{\text{KL}}(\pi(s_k^j, a_k^i) \| p(a_k^i)) \right]}, \quad (19.3)$$

where the conditional probabilities have the constraint

$$\pi(s_k^j, a_k^i) \text{ such that : } \underbrace{\max_{a_k^i \in \mathcal{A}} \left( \sum_{s_k^j \in \mathcal{S}} p(s_k^j) Q(s_k^j, a_k^i) \right)}_{\max_{a_k^i \in \mathcal{A}} \mathbb{E} [Q(s_k^j, a_k^i)]} + \underbrace{\left( \sum_{s_k^j \in \mathcal{S}} \sum_{a_k^i \in \mathcal{A}} p(s_k^j) \pi(s_k^j, a_k^i) Q(s_k^j, \pi_{a_k'}(s_k^j)) \right)}_{\mathbb{E} \left[ \max_{a_k' \in \mathcal{A}} \mathbb{E} [Q(s_k^j, a_k') | s_k^j] \right]} \leq \varphi_{\text{cost}} \quad (19.4)$$

for some positive reinforcement bound  $\varphi_{\text{cost}}$ . We can state this constrained optimization problem in an unconstrained variational form, which is a modified negative free-energy difference,

$$-\arg \min_{\pi(s_k^j, a_k^i)} \underbrace{\sum_{s_k^j \in \mathcal{S}} p(s_k^j) \sum_{a_k^i \in \mathcal{A}} \pi(s_k^j, a_k^i) \log \left( \frac{\pi(s_k^j, a_k^i)}{p(a_k^i)} \right) - \frac{1}{\tau_k} \left( \sum_{s_k^j} \sum_{a_k^i} p(s_k^j) \pi(s_k^j, a_k^i) Q(s_k^j, \pi_{a_k^i}(s_k^j)) \right)}_{\sum_{s_k^j \in \mathcal{S}} \sum_{a_k^i \in \mathcal{A}} p(s_k^j) p(a_k^i) e^{Q(s_k^j, a_k^i) / \tau_k} (Q(s_k^j, a_k^i) / \tau_k - \log(Z)) / Z} - \tau_k^{-1} \sum_{s_k^j} \sum_{a_k^i} p(s_k^j) p(a_k^i) e^{Q(s_k^j, a_k^i) / \tau_k + \log(Q(s_k^j, a_k^i)) / Z}, \quad (19.5)$$

where  $Z = \sum_{a_k^m \in \mathcal{A}} p(a_k^m) e^{Q(s_k^j, a_k^m)/\tau_k}$  is a normalization factor. This equivalent form of the value of information hence seeks a policy with the greatest dependence between states and actions such that the improvement in reinforcements is constrained by a certain amount.

### 19.3.2.2 Value of Information Optimization

Just as in the single-state case, the Gibbs distribution maximizes the negative free-energy difference in (19.5),

$$\underbrace{\pi(s_k^j, a_{k+1}^i) = \left( \frac{p(a_k^i) e^{Q(s_k^j, a_k^i)/\tau_k}}{\sum_{a_k^m \in \mathcal{A}} p(a_k^m) e^{Q(s_k^j, a_k^m)/\tau_k}} \right)}_{\pi(s_k^j, a_{k+1}^i) = \text{exponential component}}, \quad p(a_k^i) = \sum_{s^j \in \mathcal{S}} p(s_k^j) \pi(s_k^j, a_k^i), \quad (19.6)$$

where  $p(s_k^j)$  is the marginal state-visitation probability. The update given in (3.6) provides a means of altering the policy based upon any new details about the environment dynamics encoded in the new action-state value function magnitudes  $Q(s_k^j, a_k^i)$ . This expression can be viewed as a type of grouped coordinate descent where the step-size is chosen automatically; it can be equivalently interpreted as an expectation–maximization algorithm.

The single user-selectable parameter in (19.6) has the following functionality:

**$\tau_k$ : Inverse Temperature Term.** The parameter  $\tau_k$  that arises upon minimization of the unconstrained value of information dictates the rate of change in the reward difference with respect to the Shannon information constraint. As  $\tau_k^{-1}$  goes to zero, the policy complexity constraints are ignored. As  $\tau_k^{-1}$  goes to infinity, the relative entropy between the actions and states is minimized. This implies that all states elicit the same response, since the transformation costs are so high that the agent cannot change its behavior. Despite this limitation, though, the agent will still select actions that attempt to optimize the expected policy returns. For values of the rationality parameter between these two extremes, there is a trade-off that occurs. As the parameter value increases, the agent may favor state-specific actions that yield a better expected return for particular states. As the parameter value decreases, the agent will favor actions that yield high rewards for many observations, which produces a lower relative entropy.

The above policy update can be coupled with various reinforcement-learning approaches to promote an exploration of the state-action space, e.g., SARSA or  $Q$ -learning. Here, we have used tabular  $Q$ -learning, which relies on the estimation of action-value functions for each state-action combination. The resulting discrete, probabilistic policy can be found from these value functions, in an iterative manner, as shown in Algorithm 19.3. We assume that all of the actions for a given state in the policy are initialized to have discrete uniform probabilities.

---

**Algorithm 19.3** Value-of-Information-Based  $Q$ -Learning

---

- 1 Choose values for the discount and step-size parameters  $\gamma_t \in \mathbb{R}_+$ ,  $\alpha_t \in \mathbb{R}_+;$
  - 2 Choose values for the agent risk-taking parameters  $\tau_k \in \mathbb{R}_+;$
  - 3 Initialize the action-value function  $Q(a, s)$  and hence the policy  $\pi(s, a), s \in \mathcal{S}, a \in \mathcal{A};$
  - 4 **for**  $k = 0, 1, 2, \dots$  **do**
  - 5   For all relevant  $s_k^j \in \mathcal{S}$  and  $a_k^i \in \mathcal{A}$ , update the state probabilities  

$$p(s_k^j) \leftarrow p(s_0^j) \sum_{s_0^j, a_0^i, \dots, s_{k-1}^j, a_{k-1}^i} \prod_{t=1}^k p(s_t^j | a_t^i, s_{t-1}^j) \pi(s_{t-1}^j, a_{t-1}^i);$$
  - 6   **for**  $t = 0, 1, 2, \dots$  **do**
  - 7     Update  $p^{(t)}(a_k^i) \leftarrow \sum_{s_k^j \in \mathcal{S}} \pi^{(t)}(s_k^j, a_k^i) p(s_k^j), a_k^i \in \mathcal{A};$
  - 8     For  $s_k^j \in \mathcal{S}, a_k^i \in \mathcal{A}$ , update the policy  

$$\pi^{(t+1)}(s_k^j, a_k^i) \leftarrow p^{(t)}(a_k^i) e^{Q(s_k^j, a_k^i)/\tau_k} / \sum_{a_k^i \in \mathcal{A}} p^{(t)}(a_k^i) e^{Q(s_k^j, a_k^i)/\tau_k};$$
  - 9   Choose an action, in a weighted random fashion, according to the policy  
 $\pi^{(t+1)}(s_k^j, a_k^i);$  perform a state transition  $s_k^j \rightarrow s_{k+1}^p \in \mathcal{S}.$  Obtain a reward  
 $r_{k+1}(s_k^j, a_k^i) \in \mathcal{R};$
  - 10   Update the action-value function estimates  

$$Q(s_k^j, a_k^i) \leftarrow Q(s_k^j, a_k^i) + \alpha_k \left( r_{k+1}(s_k^j, a_k^i) + \gamma_k \max_{a \in \mathcal{A}} Q(s_{k+1}^p, a) - Q(s_k^j, a_k^i) \right);$$
- 

The corresponding optimization steps for the value of information are given in Algorithm 19.3, Steps 7 and 8. The equation in Step 7 is the expected value of the conditional action-selection probabilities over all states. It defines the average action-selection probability. The update in Step 8 leads to a process for weighting action choices in each state according to a Boltzmann distribution parameterized by the expected reward. This resembles soft-max-based action selection. The difference is that an extra term, whose form is given in Step 7, has been included to account for the promotion or suppression of risky actions. This term, along with an associated parameter, weights the expected rewards according to the level of risk that an investigator wishes an agent to possess. It is straightforward to show that this action weighting is greedy in the limit, under some relatively mild conditions. Therefore, when relying on SARSA, it will produce optimal-reward policies [52]. For  $Q$ -learning, the

value-of-information probabilistic policy update needs to be modified to produce a non-expansive mapping; from this property, convergence to the optimal policy easily follows.

The inner loop of Algorithm 19.3, given from Steps 7 to 8, covers the expectation-maximization-like update of the policy according to the value of information. We have shown that the policy approximation error for this loop is proportional to one over the number of updates; for most problems, ten to fifty iterations of this loop are sufficient to either achieve or almost achieve convergence. The outer loop of Algorithm 19.3, given from steps 5 to 10, defines the main reinforcement-learning process. The number of iterations needed to uncover a good policy is highly application dependent. For simple problems, only tens to hundreds of iterations may be required. For more complicated applications, thousands to hundreds of thousands of steps may be necessary. Note that alternate stopping criteria for both of these loops can be defined based upon reaching some steady-state solution.

We have shown that, when using Algorithm 19.3, the value of information is implicitly partitioning the state space according to the action-value function [9]. Partitioning occurs whenever the constraint parameter is below the state-variable entropy. This quantization can make the search process more computationally efficient than other common exploration strategies. This is because similar states are grouped together and treated as a single state for the purposes of exploration. Fewer states therefore need to be investigated. It becomes possible to uncover high-performing policies early during learning, as a consequence. In contrast, no state grouping is performed for the conventional search heuristics in reinforcement learning.

Step 8 relies on the inverse-temperature parameter, which simultaneously dictates how risky the agent's action-selection process will be. Choosing good parameter values can be challenging, as their influence over agent risk is application dependent. There are a few strategies that can be used for this purpose, however. One example would involve using expert knowledge. Investigators often have expectations about the scores they wish for the policies to obtain. A reasonable upper and lower bound for these scores could then be used to derive a corresponding parameter value. The parameter values would then be fixed during the learning process, which leads to an unchanging amount of risk. Albeit a simple strategy, the amount of risk may not be appropriate either early or late into the learning process; we saw this was true for the single-state, multiple-action abstraction. The policy returns may be adversely impacted as a consequence.

A more principled strategy would be to employ deterministic annealing to adjust the amount of agent risk. Initially, a high value could be selected for the parameter, which promotes deviating from the policy when there is potential for reward improvement. Steps 5–10 in Algorithm 19.3 would then be executed until convergence is achieved. Once this happens, this solution would be taken as the initial iterate of a new search process. The parameter value would simultaneously be reduced according to an annealing rate parameter. This forces the agent to take less risks as the learning progresses. This process repeats until the parameter reaches some minimal value. By this time, assuming that the annealing rate is not too fast, the agent should choose its actions in a deterministic manner. Although this procedure is

more computationally demanding, it ensures that the agent's actions will be chosen in a way that maximizes the optimization criterion. If an investigator is willing to sacrifice guaranteed optimality in favor of reduced computation, a simpler heuristic of lowering the parameter value across episodes could be utilized.

A downside to all of these schemes is that they do not adapt to the agent's experiences during a single learning episode. An alternate procedure that we have found to work well involves the use of the policy cross entropy. When applied to reinforcement learning, cross entropy

$$\mathbb{E}_{\pi^{(t)}(s_k^j, a_k^i)} \left[ -\log \left( \pi^{(t+1)}(s_k^j, a_k^i) \right) \right] = - \sum_{s_k^j \in \mathcal{S}} \pi^{(t)}(s_k^j, a_k^i) \log \left( \pi^{(t+1)}(s_k^j, a_k^i) \right)$$

quantifies the amount of overlap between two policies. It is a bounded measure of the amount of change between policies when acting on policies  $\pi^{(t)}(s_k^j, a_k^i)$  and  $\pi^{(t+1)}(s_k^j, a_k^i)$  across consecutive episodes. This reflects the rate of learning about the environment dynamics. If the action-selection probabilities change by a great amount, then the cross entropy will be near one. The acquisition of new behaviors can therefore be prevalent, which typically precedes a steep decrease in cost. If action-selection probabilities change only slightly, then the cross entropy will be near zero. Existing behaviors are likely to be retained, leading to moderate or negligible improvements in costs.

Policy cross entropy can be employed to yield an efficient hyperparameter adjustment scheme when using value-of-information exploration. A low action-state complexity should be used, in the beginning, to sparsely survey the entire solution space. High-performing regions will be quickly uncovered when doing so, as the state-action space will be coarsely quantized. The policy tables will begin to be populated with meaningful entries, and basic reinforcement-accruing agent behaviors will emerge. Eventually, the cross entropy will dwindle and reach a near-steady-state value. Before this occurs, the policy complexity should be gradually increased to promote a more thorough investigation of the policy space. The action-state space quantization will be relaxed, allowing existing action strategies to be refined. Further cost improvements can often be realized.

### 19.3.3 Simulations and Analyses

We now assess our exploration strategy on the game *Crossy Road*. Learning good policies for *Crossy Road* is a highly challenging problem. Foremost, the game takes place in a stochastic, dynamic environment. As well, the game requires a shift in play styles throughout certain sections. Without an ability to adequately explore the state space, the development of good agent behaviors that can deal with various game-play issues may not occur early in the learning process.

The aims of our simulations with this game are twofold. First, we want to discern what agent behaviors emerge and understand the impact of the search processes on those behaviors. Secondly, we want to quantify the performance of policies obtained with our stochastic-based exploration strategy with those that are also stochastic. For our purposes, performance will be measured according to cost and cross entropy.

### 19.3.3.1 Simulation Preliminaries

**Agent Gameplay and Rewards.** *Crossy Road* is an endless dynamic maze game. The objective is for the agent to move forward through a semi-confined environment while avoiding collisions with any of the dynamic obstacles. Dynamic obstacles, such as automobiles and trains, appear on roadway and track tiles. Colliding with a dynamic obstacle results in a loss of a life. A loss of life can also occur if the agent either waits too long to move forward, moves backward more than three rows, or jumps into a water tile. When the agent runs out of lives, the position of the agent is reset, and a new procedurally generated layout is created.

The scoring system for our implementation of *Crossy Road* is as follows. If a move leads the agent to a new row in the environment, a negative cost is incurred ( $-100$ ). Taking any movement results in a positive cost ( $+10$ ), as does waiting ( $+10$ ), which helps ensure that the agent does not remain unnecessarily stationary. A negative cost is added when the agent picks up any randomly spawning gold coins ( $-500$ ). Red coins are worth even more ( $-750$ ). These coins have no effect on game-play; they simply unlock various agent customization options. Here, coins serve as secondary objectives to make the reinforcement-learning problem more interesting. Colliding with a dynamic obstacle results in a high penalty ( $+500$ ). Losing via other means is also penalized ( $+300$ ). A good policy should attempt to minimize the total cost.

**Extracted Features.** We consider two separate state space feature sets for *Crossy Road*. This is done to ensure that our exploration mechanism works well in different contexts.

The first feature set is a hybrid mixture of local and holistic features. The local features provide spatial information about the grid occupancy three grid locations away from the agent in the forward direction. Grid occupancy is only provided two grid locations away for to the left, right, and rear of the agent. Information about diagonal grid occupancy is also provided. Such features outline which grid locations are safely reachable, contain a coins, or contain either static or dynamic obstacles. The holistic features describe more global information. One example is the direction of a traveling tree log, provided that the agent is currently on a log. Other examples are the existence and travel direction of any dynamic obstacles up to two rows away and the distances to the two nearest dynamic obstacles. In total, twenty-one features are used.

Using holistic with local features ensures that the agent does not aimlessly wander through the event. When using purely local features, such undesirable behaviors could materialize, as the agent can only make observations a few grid locations away.

Increasing the observation radius for the local features would avoid such behaviors. It would also greatly increase the state space size, though, and hence complicate training.

The second state space feature set relies predominantly on holistic features. Such features include the quantized distance to the three nearest dynamic obstacles, the direction of those obstacles relative to the agent, and the direction that the obstacles are moving. We include attributes for assessing if the current and next two rows contain dynamic obstacles and the direction that those obstacles are traveling. The quantized distance and direction to the nearest coin is also used. We additionally provide information about the agent's direction of travel if it is on a moving log. Lastly, we have local features that provide information about the type and occupancy of the cells up to one grid location away from the agent. In total, twenty-one features are extracted. Like the hybrid features outlined above, the holistic features are insensitive to the spatial location of the agent in the environment and therefore should generalize across different levels.

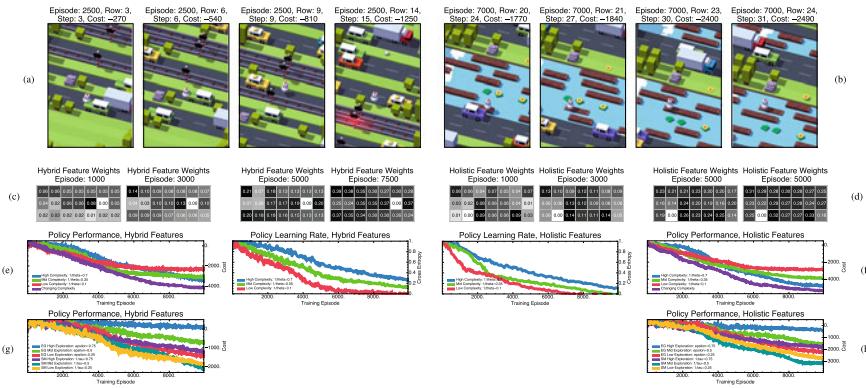
### 19.3.3.2 Value of Information Results and Analyses

For our simulations, we consider a two-dimensional, grid-based level configuration composed of  $15 \times \infty$  grid cells. These cells can be populated by static and dynamic obstacles. At any time, only a  $9 \times 19$  subsection of this grid is visible. We impose the constraint that the obstacles cannot be positioned on a row such that the agent does not have a path forward.

Three free parameters must again be set. We consider different values for the exploration parameter  $\tau_t$  throughout our simulations. The remaining two parameters are the learning rate and the discount factor, which dictate the action-value update behavior when using  $Q$ -learning. We set the learning rate  $\alpha$  to 0.45 so that more weight is given to previously acquired action-value magnitudes than those that were more recently acquired. We temporarily increase the learning rate to 0.7 for a single episode if that episode has the lowest cost compared to all previous episodes. This ensures that the policy quickly implements new agent behaviors to better play the game. A discount factor  $\gamma$  of 0.9 was used so that the agent would seek action sequences with low long-term costs. Each of these values was chosen based upon our previous applications of  $Q$ -learning to stochastic game environments.

**Fixed-Exploration Amount and Performance.** In this subsection, we analyze the performance for this game environment when using fixed values of  $\tau_t$ . This is done to establish a cost baseline for the cross-entropy-tuned results presented in the next section.

For  $\theta_t$ , we consider three different values: 0.1, 0.35, and 0.7. These values yield low-complexity, moderate-complexity, and high-complexity policies, respectively. Our simulation results are provided by the plots in Fig. 19.4. Due to the high dimensionality of the feature space, we are unable to plot the state-visitation trajectories.



**Fig. 19.4** Results of the agent’s performance during training for the game *Crossy Road*. In rows **a** and **b** are snapshots of the agent’s game-playing behaviors. Early during training, as shown in **a**, the agent performs many unnecessary actions in an attempt to understand the environment dynamics. It will often take random actions and get hit by cars frequently. As the training progresses, the agent better understands the relationship between states and actions. As shown in **b**, the agent is able to make it far into a given level. It can dodge cars and navigate across waterways. This latter behavior requires the implementation of a multi-modal game-play strategy, which is not trivial to implement and requires a thorough investigation of the state–action space. Quantitative results of the learning process are provided in **c–h**. In rows **c** and **d** are plots of the feature weights for the hybrid and holistic cases, respectively. Darker colors, and hence higher weights, indicate that a given feature is important for the decision-making process. The feature ordering is from left-to-right and top-to-bottom in the same order that they were presented in the discussions. In rows **e** and **f** we plot the policy costs and learning rate for different values of  $\tau_t$  for the hybrid and holistic feature cases, respectively. Each colored curve corresponds to a unique test case. We also consider the adaptation of  $\tau_t$  according to our cross-entropy heuristic. Lower costs indicate a better-performing policy. Lower cross-entropy values indicate fewer changes being made to the policy. The best costs are obtained when  $\tau_t$  is allowed to change across each episode versus remaining fixed. Lastly, in rows **g** and **h** are the policy costs produced when using epsilon-greedy and soft-max exploration for the hybrid and holistic feature cases, respectively. Neither of these methods approaches the performance of the uncertainty-based value of information, regardless of the chosen parameters

We therefore consider only average costs and the policy learning rates as described by cross entropy.

Figure 19.4e, f capture both the holistic and hybrid policy performance and learning rate across 10000 episodes for 100 Monte Carlo simulations. The results indicate an initial sharp decrease in cost for three policy complexities. This is due to an effective exploration of the search space and the exploitation of high-performing actions. Later in learning process, there is a tempering of the cost drop-off, depending on the employed policy complexity.

The rate at which the cost decreases corresponds to two factors. The first is the feature representation. Contrasting the left-hand plot in Fig. 19.4e with the right-hand plot in Fig. 19.4f, it can be seen that the performance of policies for the hybrid feature case lags behind that for the holistic feature case. The second factor is the policy complexity. When going from mid-complexity policies to those with high

complexity, the rate of decrease is lower and the early agent performance is worse. When going from moderately complex policies to those with lower complexity, the situation is reversed. The rate of cost decrease is higher early in the learning process. This suggests a connection between policy complexity and the implementation of agent behaviors.

This connection is partly captured by the policy cross-entropy plots in Fig. 19.4e, f. For both feature sets, the cross entropy is initially close to one, which suggests a great many changes are being made to the policies. The acquisition of new behaviors is prevalent. As the number of episodes grows, the cross entropy tends toward zero. The rate at which this occurs depends on the policy complexity. Low-complexity policies exhibit the quickest decay of cross entropy, followed by those that encode longer action sequences. Such results imply that fewer changes are being made to the policies across episodes. Existing behaviors are hence retained.

**Automatic Exploration Amount and Performance.** In this subsection, we assess the performance changes when relying on our cross-entropy heuristic to tune the exploration parameter  $\tau_t$  throughout the learning process.

In the previous subsection, we found a relationship between cross entropy and cost decreases. This relationship suggests an efficient scheme to perform reinforcement learning when using uncertainty-based value-of-information exploration. A low action-state complexity should be used, in the beginning, to sparsely survey the entire solution space. High-performing regions will be quickly uncovered when doing so. The policy tables will begin to be populated with meaningful entries, and basic game-play behaviors will emerge. Eventually, the cross entropy should dwindle and reach a near-steady-state value. Before this occurs, the policy complexity should be gradually increased to promote a more thorough investigation of the policy space. This determination can be done using threshold tests. Existing game-play strategies can be refined if this change is made, potentially leading to further cost improvements.

Such a scheme is a departure from the current conventions that involve performing high-exploration searches and progressively scaling back to the exploitation of the current policy. However, when following it, we found that high-performing policies could be found in roughly a third the number of episodes required for the fixed-parameter case. We refer to Fig. 19.4e, f for associated cost statistics. For the simulations in Fig. 19.4e, f, we initially considered a value of 0.125 for  $\tau_t$ . When the cross entropy reached a chosen threshold, the policy complexity  $\tau_t$  was increased by 0.035. Learning was performed for another 20 episodes using these new values. After the conclusion of 20 episodes, another cross-entropy test was performed to determine if the policy complexity should be further increased. This iterative update process repeated until either a maximum value of 1.0 was reached for  $\tau_t$ . The adjustment also stopped if there was no meaningful policy improvements across 100 episodes.

A cross-entropy threshold value of 0.475 was chosen after experimentation. It represents a point at which the policy is not changing greatly and therefore a slowing of the learning process. Using this threshold value permitted the agent to travel much further into the environment once training concluded. These results can be inferred

from Fig. A.2b. Choosing a high-valued cross-entropy threshold means that the policy complexity will change quickly. The agent may not be capable of sufficiently exploring the state-action space before the policy complexity increases again. Costs can be adversely affected. Not adjusting the policy complexity quickly enough can cause learning stagnation.

**Performance Discussions.** The cost-decrease change for the different values of  $\tau_t$  was expected based upon our discussions on the effects of the exploration parameter. For the highest value of  $\tau_t$ , the policy space is being finely quantized and searched. The agent is therefore frequently attempting actions that diverge from the current policy. This helps it to uncover a low-cost policy. For lower values of  $\tau_t$ , the space is coarsely quantized. The exploitation of learned behaviors dominates during learning.

It can be seen that deviating from the policy impacts some of the fundamental game-play behaviors. Initially, the agent behaviors were poor for each of the chosen policy complexities, as shown in Fig. 19.4a. Later during training, the distance between the agent and dynamic obstacles on a given row of tiles increased for higher complexity policies. The amount of time spent waiting on grassy tiles for dynamical obstacles to pass decreased, while the time spent waiting on roadway or railway tiles rose. Each of these changes heightened the chance that the agent would not progress far into the environment. Given enough episodes, policies associated with higher values of  $\tau_t$  could perform just as well as, if not better than, those that came about from lower values of  $\tau_t$ . This claim can be seen by relating the costs at different episodes in Fig. 19.4e, f.

For lower values of  $\tau_t$ , the agent was, initially, better able to cope with the challenge of dodging dynamic obstacles. The agent spent more iterations on grassy tiles planning its future actions. It additionally stayed further away from dynamic obstacles. However, when relying on low-complexity policies, the agent may only possess the most basic behaviors for reducing the costs. Rarely were these abilities honed so that the agent could consistently reach tens of rows. The agent would also never seek out coins, even when they were only a tile away. This is because the policy space is being coarsely searched. The exploitation of previously learned behaviors will hence take precedence, after a certain point, over the acquisition of new behaviors. For instance, if the process of seeking coins has not been learned early on, then it likely will not in the later stages of learning. Policies with a higher complexity will thus overtake those of lower complexity after enough episodes have elapsed, as evidenced when contrasting the costs at different points in either Fig. 19.4e or 19.4f.

From the above simulations, the choice of a feature representation also played a major role in the agent behaviors and hence the policy performance. We found that holistic features, whose results are given in figure 19.4(f), largely outperformed hybrid features, whose results were captured in figure 19.4(e). This was largely due to the informativeness of the former. The holistic features were providing global information about the environment. The agent could therefore better plan actions that would allow it to progress further into the level. It could also better dodge incoming dynamic obstacles and navigate across water tiles, as indicated in figure 19.4(b). Moreover, the holistic features enabled the agent to more effectively seek out

coins. Each coin is approximately equal to progressing five to seven rows through the environment. The hybrid features permitted only a modest local view of the environment, in comparison. The agent had less forewarning about dynamic obstacles and an ability to react to them. The agent was also unable to seek coins outside of its limited viewing area, which further limited its cost decrease.

As we noted in the previous subsection, the rate of cost reduction depends on the policy complexity. *A priori* determining good policy complexities is a challenge. We therefore utilized a heuristic for adjusting the policy complexity using cross entropy. This adjustment is deforming the free energy of the system from the mutual information term in the Lagrangian (at  $\tau_t = 0$ ) to the distortion term (at  $\tau_t = \infty$ ). It is therefore obtaining the global minimizer of the value of information at low values of  $\tau_t$  and tracking the evolution of this minimum as  $\tau_t$  is increased.

This heuristic is conceptually supported by statistical thermodynamics. It is known that the entropy of an isolated system can only increase. This leads to the conclusion that entropy is maximized at thermal equilibrium. When the system is not isolated, the maximum entropy principle is replaced by the minimum free-energy principle. In the context of the value of information, the Lagrangian is the free energy. This principle states that the Lagrangian cannot increase and therefore reaches its minimum value at thermal equilibrium. By annealing the exploration hyperparameter, the system is always kept either at or near equilibrium. At zero temperature ( $1/\tau_t \rightarrow 0$ , where  $\tau_t \rightarrow \infty$ ), the system reaches the minimum energy configuration. This corresponds to the situation where we have the best crisp partitioning of the state space and hence the best policy.

Our simulations demonstrated that iteratively adjusting  $\tau_t$  had a significant impact on agent performance compared to the fixed-parameter case. This performance improvement was made possible due to successive quantizations of the policy space. Without the ability to sparsely sample such spaces, it becomes incredibly difficult to encounter high-performing regions early in the learning process. This partly explains why other search mechanisms must traditionally favor large levels of exploration at first. Likewise, the ability to deterministically explore helped in effectively investigating the quantized space.

Through the use of our heuristic, certain agent behaviors emerged more quickly. The agent learned to avoid static and dynamic obstacles earlier. It also effectively navigated across moving logs. Coins were acquired more frequently too. These implemented agent behaviors were inherently connected with a reduction in costs.

In our simulations, the cross entropy decayed across many episodes. For certain environments, the cross entropy may steadily increase after a certain number of episodes. This corresponds to the case where we have increased uncertainty about the action choices due to an influx of so-called neg-information. Such neg-information could be due to widespread changes in the environment dynamics; for instance, the Markov chain underlying the Markov decision process may have vastly different transition probabilities compared to earlier in the learning process. To adequately learn these new dynamics, the exploration hyperparameter  $\tau_t$  should be reduced until such a time that the cross entropy begins to decrease again. Increases in  $\tau_t$  according to our above heuristic can then resume.

### 19.3.3.3 Methodological Comparisons

To provide context for the preceding results, we compare against epsilon-greedy exploration and soft-max selection. These are two of the most well-known stochastic search strategies for reinforcement learning [47].

Epsilon-greedy is based upon the principle of taking random actions and random times and greedily choosing the best action in all other instances. Soft-max selection relies on a weighted-random approach to action selection. It weights each of the actions according to their expected costs and stochastically chooses an arm using those weights.

It is well known that epsilon-greedy exploration can converge to optimal values, in certain situations, as the number of episodes grows. For our application, convergence did not occur within ten thousand episodes, as indicated by Fig. 19.4g, h. Moreover, the results are worse than for the uncertainty-based value of information by at least half. The curves in these plots correspond to the following values of  $\epsilon$ : 0.25, 0.5, and 0.75. Such values correspond to taking a random action every fourth iteration, every other iteration, and three times over four iterations, respectively.

As highlighted in Fig. 19.4g, h, soft-max selection yielded modest benefits over epsilon-greedy exploration. It did not, however, achieve similar costs as the uncertainty-based value of information. The curves in these plots correspond to the case of adjusting the following values of  $\tau^{-1}$ : 0.25, 0.5, and 0.75. These parameter magnitudes, respectively, favor taking the best-cost action frequently, balancing exploration and exploitation, and promoting the near-equi-probable selection of actions.

We attempted to improve these results by making a variety of adjustments. We increased the chance of taking a random action, which only hurt performance over the number of episodes that we considered. We also lowered the random-action-selection rate, but that only yielded a marginal improvement. Training over several thousand more episodes did little to enhance the agent's performance. Adjusting the learning rate and discount factor had only a minimal impact too. The only protocol that we found to consistently improve the policy cost was to slowly anneal  $\epsilon$  and  $\tau^{-1}$  from a high to a low values. Even then, tens of thousands of episodes were needed to reach the same cost levels as uncertainty-based value-of-information exploration.

**Methodological Comparisons Discussions.** These results highlight the utility of our uncertainty-based value of information search mechanism. Regardless of how we tuned various parameters for either epsilon-greedy or soft-max selection, neither approach was able to reach similar costs as our method in the same number of episodes.

There are two reasons for this. First, such heuristics are unable to determine if a particular region of the policy space has been sufficiently visited. They may therefore randomly select actions that will neither visit new regions of the policy space nor lead to improved costs. Secondly, these heuristics must search over the entire policy space, not a quantized variant of it. It may therefore spend an inordinate number of

episodes dealing with related groups of states and not make meaningful revisions to the policy.

In terms of game-play, both exploration heuristics failed to implement crucial behaviors during the specified episodes. The agents were typically stymied by moving logs. Such sections of the environment can appear early in the level, which limits the potential score if the agents are unable to successfully navigate through them. The agents also typically made unnecessary moves. They would sometimes move laterally, with no apparent game-play benefit, thereby increasing costs. They would also either prematurely jump into roadways or railways or not move quickly enough away from a dynamic obstacle. Lastly, the agents often failed to collect easily reached coins. The only coins that were picked up were those in the direct path of the agent.

### 19.3.4 Conclusions

In this section, we considered the use of the value of information for exploring multiple-state, multiple-action Markov decision processes. That is, we considered the version of the criterion that Stratonovich proposed for two random variables. We showed that optimizing this revised criterion again led to a Gibbs-distribution update the conditional probabilities of a stochastic policy. A second update, which accounts for the change in the default action-selection behavior, also emerged.

Both update equations for globally optimizing the value of information rely a free hyperparameter whose value must be selected. This hyperparameter dictates the dependence of actions on the states. It also dictates the policy quantization. Small hyperparameter values correspond with low-complexity policies and induce risk-avoiding agent behaviors. A coarse quantization of the state–action space, according to the value function, is obtained; this implies that the same action is used across many states with similar feature values. Higher hyperparameter values lead to lower policy compression levels. Agents are more likely to frequently take risks in these cases, which can promote the formation of multi-modal strategies. A finely grained quantization is obtained in this case; a potentially unique action can be assigned to each state. What constitutes large and small values depends on the problem domain, so a means of automatically adapting the hyperparameter during learning is crucial for obtaining good-performing agents without significant trial-and-error experimentation.

To gauge the performance of the value of information, we considered the creation of game-playing agents for *Crossy Road*. We showed that the risk hyperparameter had a profound influence on the policy return quality for this game. High levels of risk yielded policies whose expected costs were low. They also facilitated the switching of game-play strategies, which, for this environment, involved pursuing coins that were not on the most direct path. Crossing over waterways also required a change in behavior. A great many learning episodes were needed to construct such multi-modal policies, though. Lower levels of risk allowed for the quick generation of policies that performed reasonably well. As the number of episodes increased,

the policies' qualities lagged behind those found with higher levels of risk. They were also unable to reliably produce multi-modal game-play strategies. Adapting the hyperparameter value during learning, according to a cross-entropy heuristic, led to the best performance. This is in line with the results in the previous section, which showed that a constant exploration rate is not effective, even for simple problems.

Such findings indicate that the value of information offers a good framework to understand the variables at play. We have only a single hyperparameter whose value must be chosen. This hyperparameter dictates a great deal of functionality, such as the amount of action-selection risk and hence the amount of exploration. Many other approaches that could be used for producing *Crossy Road* agents, such as deep, temporal neural networks, have multiple parameters and hyperparameters whose values must be chosen. Such a large number of parameters makes the connections between scores and variable values more difficult to understand. It can also impede the construction of automated, principled parameter-tuning methods.

In our simulations, we additionally compared the results of value-of-information-based exploration with two other reinforcement-learning search mechanisms: soft-max-based and epsilon-greedy action selection. Both schemes produced worse policies. They also did not implement multi-modal behaviors. This is because such selection procedures assume that the policies can be unboundedly complex. A great many episodes would therefore be needed to explore the joint state-action space. For many problems, value-of-information-based policies with finite complexity can approach the same levels of performance in fewer episodes. This is because not every state in a given problem may need to be paired with a unique action. It may hence be possible to group several states and assign them the same action without impacting performance.

These comparative results illustrate that the value-of-information criterion provides an excellent means of addressing the exploration dilemma, as it is inherently a problem of decision-making under uncertainty. Neither epsilon-greedy exploration nor soft-max selection, let al. one many other search mechanisms, adequately account for uncertainty when choosing actions. They can therefore be ill equipped to understand the dynamics of highly complex, stochastic environments like the *Crossy Road* game world in only a few episodes.

## References

1. Stratonovich, R.L.: On value of information. *Izvestiya of USSR Acad. Sci. Tech. Cybern.* **5**(1), 3–12 (1965)
2. Stratonovich, R.L., Grishanin, B.A.: Value of information when an estimated random variable is hidden. *Izvestiya of USSR Acad. Sci. Tech. Cybern.* **6**(1), 3–15 (1966)
3. von Neumann, J., Morgenstern, O.: *Theory of Games and Economic Behavior*. Princeton University Press, Princeton (1953)
4. Belavkin, R.V.: Asymmetry of risk and value of information. In: Vogiatzis, C., Walteros, J., Pardalos, P. (eds.) *Dynamics of Information Systems*, pp. 1–20. Springer, New York (2014)
5. Sledge, I.J., Principe, J.C.: Balancing exploration and exploitation in reinforcement learning using a value of information criterion. In: *Proceedings of the IEEE International Conference on*

- Acoustics, Speech, and Signal Processing (ICASSP), New Orleans, LA, USA, pp. 1–5 (2017). Accessed by 5–9 March 2017, <https://doi.org/10.1109/ICASSP.2017.7952670>
6. Sledge, I.J., Príncipe, J.C.: Quantifying the trade-offs between policy exploration and exploitation in Markov decision processes. *Entropy* **20**(3), 155(1–34) (2018). <https://dx.doi.org/10.3390/e20030155>
7. Sledge, I.J., Príncipe, J.C.: Analysis of agent expertise in Ms. Pac-Man using value-of-information-based policies. *IEEE Trans. Comput. Intell. Artif. Intell. Games* (2018). (accepted, in press). <https://dx.doi.org/10.1109/TG.2018.2808201>
8. Sledge, I.J., Emigh, Príncipe, J.C.: Guided policy exploration for Markov decision processes using an uncertainty-based value-of-information criterion. *IEEE Trans. Neural Netw. Learn. Syst.* **26**(9), 1–19 (2018). <https://dx.doi.org/10.1109/TNNLS.2018.2812709>
9. Sledge, I.J., Príncipe, J.C.: Partitioning relational matrices of similarities or dissimilarities using the value of information. In: Proceedings of the IEEE International Conference on Acoustics, Speech, and Signal Processing (ICASSP), Calgary, Canada, pp. 1–5 (2018). 15–20 Apr 2018. <https://arxiv.org/abs/1710.10381>
10. Kaelbling, L.P., Littman, M.L., Moore, A.W.: Reinforcement learning: a survey. *J. Artif. Intell. Res.* **4**(1), 237–285 (1996). <http://dx.doi.org/10.1613/jair.301>
11. Salganicoff, M., Ungar, L.H.: Active exploration and learning in real-valued spaces using multi-armed bandit allocation indices. In: Proceedings of the International Conference on Machine Learning (ICML), Tahoe City, CA, USA, pp. 480–487 (1995). Accessed 9–12 July 1995. <http://dx.doi.org/10.1101/B978-1-55860-377-6.50066-9>
12. Auer, P.: Using confidence bounds for exploration-exploitation trade-offs. *J. Mach. Learn. Res.* **3**(1), 397–422 (2002)
13. Auer, P., Cesa-Bianchi, N., Freund, Y., Schapire, R.E.: The nonstochastic multi-armed bandit problem. *SIAM J. Comput.* **32**(1), 48–77, 2002. <http://dx.doi.org/10.1137/S0097539701398375>
14. Strehl, A.L., Mesterharm, C., Littman, M.L., Hirsh, H.: Experience-efficient learning in associative bandit problems. In: Proceedings of the International Conference on Machine Learning (ICML), Pittsburgh, PA, USA, pp. 889–896 (2006). Accessed 25–29 June 2006. <http://dx.doi.org/10.1145/1143844.1143956>
15. Madani, O., Lizotte, S.J., Greiner, R.: The budgeted multi-armed bandit problem. In: Proceedings of the Conference on Learning Theory (COLT), New Brunswick, NJ, USA, pp. 643–645 (2004). Accessed 12–15 July 2004. Available: <http://dx.doi.org/10.1007/978-3-540-27819-1>
16. Kleinberg, R.D.: Nearly tight bounds for the continuum-armed bandit problem. In: Saul, L.K., Weiss, Y., Bottou, L. (eds.) *Advances in Neural Information Processing Systems (NIPS)*, pp. 697–704 (2008). MIT Press, Cambridge, MA, USA (2008)
17. Wang, Y., Audibert, J., Munos, R.: Algorithms for infinitely many-armed bandits. In: Koller, D., Schuurmans, D., Bengio, Y., Bottou, L. (eds.) *Advances in Neural Information Processing Systems (NIPS)*, pp. 1729–1736. MIT Press, Cambridge, MA, USA (2008)
18. Bubeck, S., Munos, R., Stoltz, G.: Pure exploration in finitely-armed and continuous-armed bandits. *Theor. Comput. Sci.* **412**(19), 1876–1902 (2011). <http://dx.doi.org/10.1016/j.tcs.2010.12.059>
19. Vermorel, J., Mohri, M.: Multi-armed bandit algorithms and empirical evaluation. In: Gama, J., Camacho, R., Brazdil, P.B., Jorge, A.M., Torgo, L. (eds.) *Machine Learning: ECML*, pp. 437–448. Springer-Verlag, New York City, NY USA (2005)
20. Even-Dar, E., Mannor, S., Mansour, Y.: PAC bounds for multi-armed bandit and markov decision processes. In: Proceedings of the Conference on Learning Theory (COLT), Sydney, Australia, pp. 255–270 (2002). Accessed 8–10 July 2002. <http://dx.doi.org/10.1007/3-540-45435-7>
21. Mannor, S., Tsitsiklis, J.N.: The sample complexity of exploration in the multi-armed bandit problem. *Journal of Machine Learning Research* **5**(12), 623–648 (2004)
22. Langford, J., Zhang, T.: The epoch-greedy algorithm for multi-armed bandits. In: Platt, J.C., Koller, D., Singer, Y., Roweis, S.T. (eds.) *Advances in Neural Information Processing Systems (NIPS)*, pp. 817–824. MIT Press, Cambridge, MA, USA (2008)

23. Srinivas, S., Krause, A., Seeger, M., Kakade, S.M.: Gaussian process optimization in the bandit setting: No regret and experimental design. In: Proceedings of the International Conference on Machine Learning (ICML), Haifa, Israel, pp. 1015–1022 (2010). Accessed 21–24 June 2010
24. Krause, A., Ong, C.S.: Contextual Gaussian process bandit optimization. In: Shawe-Taylor, J., Zemel, R.S., Bartlett, P.L., Pereira, F., Weinberger, K.Q. (eds.) Advances in Neural Information Processing Systems (NIPS), pp. 2447–2455. MIT Press, Cambridge, MA, USA (2011)
25. Beygelzimer, A., Langford, J., Li, L., Reyzin, L., Schapire, R.E.: Contextual bandit algorithms with supervised learning guarantees. In: Proceedings of the International Conference on Artificial Intelligence and Statistics (AISTATS), Ft. Lauderdale, FL, USA, pp. 19–26. Accessed 20–22 April 2011
26. Cesa-Bianchi, N., Fischer, P.: Finite-time regret bounds for the multi-armed bandit problem. In: Proceedings of the International Conference on Machine Learning (ICML), Helsinki, Finland, pp. 100–108 (1998). Accessed 5–9 1998
27. Auer, P., Cesa-Bianchi, N., Fischer, P.: Finite-time analysis of the multi-armed bandit problem. *Mach. Learn.* **47**(2), 235–256 (2002). <http://dx.doi.org/10.1023/A:1013689704352>
28. McMahan, H.B., Streeter, M.: Tight bounds for multi-armed bandits with expert advice,” in *Proceedings of the Conference on Learning Theory (COLT)*, Montreal, Canada, June 18–21 2009, pp. 1–10
29. Bubeck, S., Cesa-Bianchi, N.: Regret analysis of stochastic and non-stochastic multi-armed bandit problems. *Found. Trends Mach. Learn.* **5**(1), 1–122 (2012). <http://dx.doi.org/10.1561/2200000024>
30. Cesa-Bianchi, N., Gentile, C., Neu, G., Lugosi, G.: Boltzmann exploration done right. In: Guyon, I., Luxburg, U.V., Bengio, S., Wallach, H., Fergus, R., Vishwanathan, S., Garnett, R. (eds.) Advances in Neural Information Processing Systems (NIPS), pp. 6287–6296. MIT Press, Cambridge, MA, USA (2017)
31. Lai, T.L., Robbins, H.: Asymptotically efficient adaptive allocation rules. *Adv. Appl. Math.* **6**(1), 4–22 (1985). [http://dx.doi.org/10.1016/0196-8858\(85\)90002-8](http://dx.doi.org/10.1016/0196-8858(85)90002-8)
32. Auer, P., Ortner, R.: UCB revisited: improved regret bounds for the stochastic multi-armed bandit problem. *Periodica Math. Hung.* **61**(1–2), 55–65 (2010). <http://dx.doi.org/10.1007/s10998-010-3055-6>
33. Audibert, J.Y., Munos, R., Szepesvári, C.: Exploration-exploitation tradeoff using variance estimates in multi-armed bandits. *Theor. Comput. Sci.* **410**(19), 1876–1902 (2009). <http://dx.doi.org/10.1016/j.tcs.2009.01.016>
34. Kuss, M., Rasmussen, C.E.: Gaussian processes in reinforcement learning. In: Thrun, S., Saul, L.K., Schölkopf, P.B. (eds.) Advances in Neural Information Processing Systems (NIPS), pp. 751–758. MIT Press, Cambridge, MA, USA (2003)
35. Engel, Y., Mannor, S., Meir, R.: Bayes meets Bellman: the Gaussian process approach to temporal difference learning. In: Proceedings of the International Conference on Machine Learning (ICML), Washington D.C., USA, pp. 154–161 (2003). Accessed 21–24 August 2003
36. Engel, Y., Mannor, S., Meir, R.: Reinforcement learning with Gaussian processes. In: Proceedings of the International Conference on Machine Learning (ICML), Bonn, Germany, pp. 201–208 (2005). Accessed 7–11 August 2005. <http://dx.doi.org/10.1145/1102351.1102377>
37. Ghavamzadeh, M., Mannor, S., Pineau, J., Tamar, A.: Bayesian reinforcement learning: a survey. *Found. Trends Mach. Learn.* **8**(5–6), 359–483 (2015). <http://dx.doi.org/10.1561/2200000049>
38. Silver, E.A.: Markovian decision processes with uncertain transition probabilities and rewards. Ph.D. dissertation, Massachusetts Institute of Technology, Cambridge, MA, USA (1963)
39. Martin, J.J.: Bayesian Decision Problems and Markov Chains. Wiley, New York City, NY, USA (1967)
40. Cozzolino, J.M., Gonzalez-Zubieta, R., Miller, R.L.: Markovian decision processes with uncertain transition probabilities. Massachusetts Institute of Technology, Cambridge, MA, USA, Operations Research Center Technical Report 11 (1965)
41. Satia, J.K.: Markovian decision processes with uncertain transition matrices and/or probabilistic observation of states. Ph.D. dissertation, Stanford University, Stanford, CA, USA (1963)

42. Satia, J.K., Lave Jr., R.E.: Markovian decision processes with uncertain transition probabilities. *Oper. Res.* **21**(3), 728–740 (1973). <http://dx.doi.org/10.1287/opre.21.3.728>
43. Guez, A., Silver, D., Dayan, P.: Efficient Bayes-adaptive reinforcement learning using sample-based search. In: Pereira, F., Burges, C.J.C., Bottou, L., Weinberger, K.Q. (eds.) *Advances in Neural Information Processing Systems (NIPS)*, pp. 1025–1033. MIT Press, Cambridge, MA, USA (2012)
44. Dearden, R., Friedman, N., Andre, D.: Model based Bayesian exploration. In: *Proceedings of the Conference on Uncertainty in Artificial Intelligence (UAI)*, Stockholm, Sweden, pp. 150–159 (1999). Accessed July 30–August 1
45. Poupart, P., Vlassis, N., Hoey, J., Regan, K.: An analytic solution to discrete Bayesian reinforcement learning. In: *Proceedings of the International Conference on Machine Learning (ICML)*, Pittsburgh, PA, USA, pp. 697–704 (2006). Accessed 25–29 June 2006. <http://dx.doi.org/10.1145/1143844.1143932>
46. Kolter, J.Z., Ng, A.Y.: Near-Bayesian exploration in polynomial time. In: *Proceedings of the International Conference on Machine Learning (ICML)*, Montreal, Canada, pp. 513–520 (2009). Accessed 14–18 June 2009. <http://dx.doi.org/10.1145/1553374.1553441>
47. Sutton, R.S., Barto, A.G.: *Reinforcement Learning: An Introduction*. MIT Press, Cambridge, MA, USA (1998)
48. Whitehead, S.D.: Complexity and cooperation in  $Q$ -learning. In: *Proceedings of the International Conference on Machine Learning (ICML)*, Evanston, IL, USA, pp. 363–367 (1991). Accessed 20–25 June 1991. <http://dx.doi.org/10.1016/B978-1-55860-200-7.50075-1>
49. Hernandez-Gardiol, N., Mahadevan, S.: Hierarchical memory-based reinforcement learning. In: Leen, T.K., Dietterich, T.G., Tresp, V. (eds.) *Advances in Neural Information Processing Systems (NIPS)*, pp. 1047–1053. MIT Press, Cambridge, MA, USA (2000)
50. Faulkner, R., Precup, D.: Dyna planning using a feature based generative model. In: Lee, H., Ranzato, M., Bengio, Y., Hinton, G.E., LeCun, Y., Ng, A.Y. (eds.) *Advances in Neural Information Processing Systems (NIPS) Workshop*, pp. 10–19. MIT Press, Cambridge, MA, USA (2010)
51. Kulkarni, T.D., Narasimhan, K., Saeedi, A., Tenenbaum, J.B.: Hierarchical deep reinforcement learning: Integrating temporal abstraction and intrinsic motivation. In: Lee, D.D., Sugiyama, M., Luxburg, U.V., Guyon, I., Garnett, R. (eds.) *Advances in Neural Information Processing Systems (NIPS)*, pp. 1047–1053. MIT Press, Cambridge, MA, USA (2016)
52. Singh, S.P., Jaakkola, T., Littman, M.L., Szepesvári, C.: Convergence results for single-step on-policy reinforcement-learning algorithms. *Mach. Learn.* **38**(3), 287–308 (2000). <http://dx.doi.org/10.1023/A:1007678930559>
53. Watkins, C.J. C.H.: Learning from delayed rewards. Ph.D. dissertation, King's College, Cambridge University, Cambridge, UK (1989)
54. Barto, A.G., Bradtko, S.J., Singh, S.P.: Learning to act using real-time dynamic programming. *Artif. Intell.* **72**(1–2), 81–138, 1995. [http://dx.doi.org/10.1016/0004-3702\(94\)00011-O](http://dx.doi.org/10.1016/0004-3702(94)00011-O)
55. Sutton, R.S.: Integrated architecture for learning, planning, and reacting based on approximating dynamic programming. In: *Proceedings of the International Conference on Machine Learning (ICML)*, Austin, TX, USA, pp. 216–224 (1990). Accessed 21–23 June 1990. <http://dx.doi.org/10.1016/B978-1-55860-141-3.50030-4>
56. Cybenko, G., Gray, R., Moizumi, K.:  $Q$ -learning: A tutorial and extensions. In: Ellacott, S.W., Mason, J.C., Anderson, I.J. (eds.) *Mathematics of Neural Networks*, pp. 24–33. Springer-Verlag, New York City (1997)

## **Part V**

# **Applications of RL**

# Chapter 20

## Map-Based Planning for Small Unmanned Aircraft Rooftop Landing



J. Castagno and E. Atkins

**Abstract** New risk posed by unmanned aircraft systems (UASs) is a pressing concern for industry, regulators, and the public. Failures 480711 and anomalies may force a near-term landing, requiring fast real-time identification of low-risk landing sites and feasible low-risk paths to those sites. Although sensors can scan an immediate area, no safe landing site might be within the current field of view in which case a map of more distant sites is essential. In urban areas, small UAS touchdown on a flat rooftop may pose less risk to people and property than landing on crowded streets, sidewalks, or parks. This paper describes the offline construction of a landing site database using publicly available data sources with focus on rooftop sites. Flat roofs are identified from LiDAR and image data and then processed to extract obstacle-free surface(s) for landing. Flight path and landing site risk maps are defined. Pareto frontiers over these two risk types are examined. A multi-goal planner considers landing site risk and path risk as a weighted sum to minimize. Statistics on case studies in three diverse communities are presented.

### 20.1 Introduction

Unmanned aircraft systems (UASs) or drones are expected to proliferate in low-altitude airspace over the coming decade. Drones have been proposed to offer fast package delivery, perform inspections, and entertain. The FAA estimates over 1.25 million small UAS were flown in 2018 [10], far exceeding the number of registered manned aircraft in the USA. Low-altitude operation of UAS in urban areas will require flight near buildings and over people. Safety must be a top priority for system designers and regulators as UAS will pose new risks to the overflowed population.

A primary safety concern is ensuring a robust emergency landing capability [6, 44]. Emergency landing requires landing site selection, trajectory planning, and stable flight control to actually reach the selected site [2]. It is possible a UAS may identify

---

J. Castagno (✉) · E. Atkins  
University of Michigan, Ann Arbor, MI, USA  
e-mail: [jdcasta@umich.edu](mailto:jdcasta@umich.edu)

a safe site within sensor range allowing for an immediate landing. However, when no safe site is within range the UAS must devote time and energy to exploring sites beyond sensor range or else utilize pre-processed data to identify a safe site [27, 38]. An onboard database of maps including landing sites can be incorporated into an efficient autonomous decision-making framework. For example, Refs. [2, 21] and [35] utilize airborne flight risk models to build emergency landing plans for fixed-wing and urban flight operations, respectively. We call such a decision-making framework a map-based planner.

Urban areas typically do not offer classic emergency landing sites such as unpopulated open fields. This requires a planner to consider unconventional yet safe alternatives. We propose flat building rooftops as viable urgent landing sites for small UAS. These UAS will likely operate at low altitudes and at times even in urban canyons. During landing site selection, a map-based planner must be able to assess *landing site risk* posed to the aircraft and bystanders at touchdown. The UAS may pose risk to people and property it overflies enroute, so the planner must also assess *path risk* once the landing flight plan is known. Landing site and path risk together offer an estimate of *total risk*.

Map data used for emergency landing planning must have high integrity and low-latency access to support timely decision-making. This paper proposes an offline data processing pipeline and online multi-objective, multi-goal landing planner that enable a UAS to minimize total risk when a nearby emergency landing is required. Landing site and local area map information is pre-processed and stored onboard. The multi-goal onboard emergency landing planner explores available ground and rooftop landing sites likely to minimize overall total risk while greedily pruning options known to have higher risk. Pareto fronts over landing site and path risk are generated for three urban regions: Witten, Germany; Ann Arbor, Michigan; and mid-town Manhattan in New York City. We statistically analyze results with respect to availability and quality of landing sites as well as execution time of the map-based planner.

The first contribution of this paper is a novel method to identify flat rooftop surfaces from airborne LIDAR data to identify the largest clear small UAS landing location on each flat roof. Risk metrics are quantified from offline construction of a database using public data sources. A second contribution is our proposed approach to model and optimize plans over a combination of landing site and path risk metrics. Our third contribution is a multi-goal onboard planner that guarantees a risk-optimal solution is found rapidly by avoiding exploration of high-risk options. The proposed emergency planning framework enables a UAS to select an emergency landing site and corresponding flight plan with minimum total risk.

The paper is structured as follows. Section 20.2 provides background in emergency landing planning and multi-goal path planning. Section 20.3 reviews preliminaries in data sources and planning. Section 20.4 discusses offline construction of a risk-aware landing site database. Section 20.5 outlines methods for generating occupancy and risk maps used for 3D path planning, while Section 20.6 summarizes our map-based planning approach. Section 20.7 presents case studies with focus on analysis of trade-offs between landing site and path risk and statistics on required planning time. Section 20.8 presents conclusions and future work.

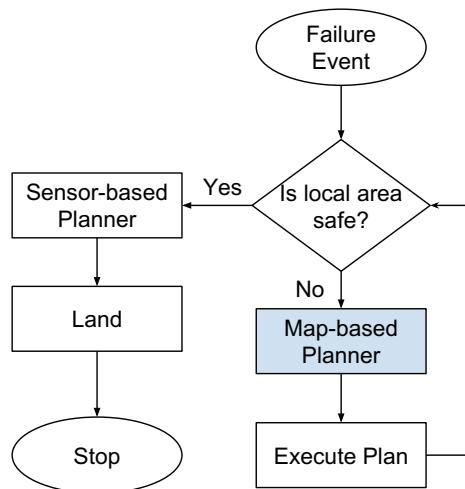
## 20.2 Background

Emergency landing planning is typically accomplished with either onboard sensor-based planning or map-based planning [38, 42]. Sensor-based planners rely strictly on real-time data streams while map-based planner use information previously gathered and stored onboard. Planners using a combination of maps and onboard sensors can capitalize on both data sources [38]. Section 20.2.1 outlines sensor-based planning while an overview of map-based planning is presented in Sect. 20.2.2. Section 20.2.3 provides background on multi-goal planning which our map-based planner utilizes. A meta-level framework to unify sensor and map-based planning has been proposed for general (fixed-wind) aviation [2] as well as multicopter UAS [38] as shown in Fig. 20.1. This planner relies on local sensor data to land if the immediate area is safe but calls a map-based planner otherwise. Sensor and map-based methods can be merged to capitalize on each other's strengths. Previous work all focuses on open landscapes and/or runways and considers only a single landing site in each planning epoch. We propose map-based planning with consideration of rooftop landing sites and dual risks from landing path and site for the first time in this paper. Previous research in sensor-based planning, map-based planning, and multi-goal planning is summarized below.

### 20.2.1 Sensor-Based Planning

Onboard exteroceptive sensors including camera, radar, and LiDAR can provide a wealth of information about the surrounding environment for use in emergency landing planning. Reference [42] uses downward facing camera data to identify and characterize possible landing sites according to size, shape, slope, and nearby obsta-

**Fig. 20.1** Emergency planning logic. Reprinted from Ref. [38], originally published open access under a CC-BY 4.0 license. <https://link.springer.com/article/10.1007/s10846-016-0370-z>



cles. Reference [39] provides methods and experimental results of autonomous local landing using video and LiDAR data. Reference [7] specifically identifies candidate landing sites on rooftops using a single camera, while Ref. [13] identifies terrain-based landing sites in an image plane from 2D probabilistic elevation maps generated over terrain. In all cases landing site identification is only possible within the sensor field of view.

### 20.2.2 Map-Based Planning

There are two main approaches in generating landing sites for use in map-based UAS emergency landing: one producing a specific landing site database and the other generating a risk grid from which landing sites can be selected. Both approaches, sometimes used together, rely on similar data sources such as census records, digital surface models (DSMs) and map vector data but differ in output representations. A georeferenced database contains vector geometries of landing sites with associated metadata used for risk evaluation. A risk grid is a two or three-dimensional data structure where each cell refers to the risk of a specific location on Earth.

Reference [8] uses data such as census records, open street maps (OSMs), and mobile phone records to quantify risk for unmanned aircraft requiring an emergency landing. Risk factors include risk to the vehicle, human population, property, and an area risk to assess the quality of a landing site. Landing sites such as open fields, grasslands, and highways are identified, risk evaluated, and stored in a landing site database. The proposed planning architecture uses this onboard database to select a risk-optimal landing site within a feasible flight footprint. After a landing site is chosen, path planning is performed at a constant altitude, assessing risk to people through a fusion of census data and mobile phone activity.

Reference [38] proposes the use of three-dimensional risk grids to identify landing sites and perform path planning for an energy-constrained multicopter in urban environments. A 3D occupancy grid to represent risk is generated as a combination of terrain, population, and obstacle costs. Terrain risk is evaluated using slope as well as terrain type, census data is used to determine population risk, and property risk is assigned equally to all buildings. A linear combination of these risks is used to generate a final 3D grid for flight planning. In this grid a landing site is a *terrain* cell that has a lower cost than a configurable threshold. All landing sites are treated equally, meaning the first landing site found by the planner is the “optimal” choice returned.

Reference [32] proposes generation of a 2D risk map that quantifies risk to population on the ground. The map is created taking into account an aircraft’s model parameters and the local environment conditions while considering their uncertainties. The risk map is defined for a specific altitude and is the combination of several risk layers including population density, obstacles, sheltering, and no fly zones.

We propose an emergency landing framework that uses both a landing site database and a 3D risk grid to evaluate landing site risk and path risk as independent metrics.

Our work is focused on vertical take-off and landing (VTOL) UAS that might require an urgent landing but still have sufficient flight control to stably follow a prescribed path to touchdown. We provide a multi-goal planner to trade-off risk of landing sites with risk-optimal paths to each site while efficiently finding the minimum total risk solution. We uniquely identify usable area on flat rooftops using machine learning and computational geometry techniques. Previous efforts to quantify an *area* risk of a landing site approximated risk as length or area of a site's buffered geometry [8]. However not all area in a landing site is suitable for landing. For example, many rooftops have obstacles such as air conditioning units, vents, and rooftop entrances. Additionally, the choice of a singular touchdown point at a chosen landing site is often simplified to be the centroid or pseudo-centroid of the site or cell [8, 19]. However both of these choices do not represent the optimal touchdown location with respect to ensuring flatness and distance from obstacles. We formulate the problem of choosing a touchdown position as finding the largest inscribed circle in its 2D polygon representation. Circle placement guarantees landing target coordinates are maximally separated from any obstacle or edge.

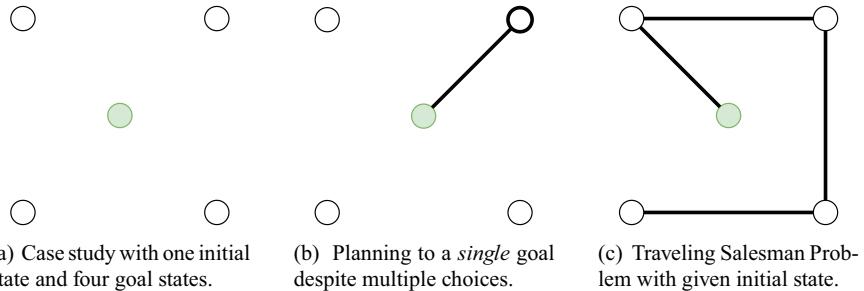
### 20.2.3 Multi-goal Planning

Multi-goal planning has two different definitions:

1. One start state and multiple goal states. The algorithm seeks to find the *singular* goal/path pair that minimizes/maximizes an objective function.
2. One start state, intermediary goal states, and a final end goal state. The algorithm seeks to find a connecting route, which includes *multiple* connected state pairs, that minimizes/maximizes some objective function. This formulation is commonly called the traveling salesman problem (TSP) [33].

A visual representation of these definitions is shown in Fig. 20.2. This paper focuses on the first definition per Fig. 20.2b. Work by Ref. [17] investigated a form of multi-goal search using uninformed planners in a 2D grid. In this work all goals are valued equally with the objective to return paths for all start/goal combinations. Each start/goal pair can be treated as an independent path planning problem. Lim et al. propose reusing previous information, e.g., node expansions and costs, to reduce search overhead for the next goal. Results indicate that retaining information from previous searches reduces overhead in 2D grids when using uninformed search.

An objective or cost function may consider both the cost of a start/goal path and also the worth of the goal itself. Reference [36] investigates efficient methods to conduct search for multiple agents seeking different products in a free market. The authors propose a multi-goal planner that maximizes expected overall utility from the set of opportunities found (e.g., products/goals) minus the costs associated with finding that set. In this work the worth or value of a goal is probabilistic and can only be ascertained through search. The objective function aims to balance goal achievement reward against the cost of obtaining that goal.



**Fig. 20.2** Comparison of multi-goal planning definitions. The green state depicts the initial state, e.g., where the UAS begins its emergency landing trajectory. Unfilled circles are goal states

In our work multiple landing sites (goals) will typically be identified. Risk is a function of path to each site as well as the site itself. Our objective is to find the singular goal/path pair which minimizes the combined risk of the landing site and a flight path from the initial UAS location to that site.

#### 20.2.4 Urban Landscape and Rooftop Landings

An emergency landing requires first identifying safe nearby landing sites. Cities lack conventional emergency landing sites such as open fields or grasslands. Empty lots are often sparsely distributed, and parks may be unexpectedly occupied. The satellite image in Fig. 20.3 illustrates a typical urban landscape that affords rooftop landing. Obstacles on a flat rooftop, e.g., air conditioning units, can be removed from potential landing site surfaces. We identify rooftop landing sites given sufficient flatness and distance from obstacles and edges.

Our proposed flight planner requires the vehicle to execute a stable approach to an emergency landing site. Failure scenarios can be detected and handled in time to execute a controlled landing. Scenarios in which an urgent landing can be achieved without loss-of-control include: low battery energy, lost communication link, adverse weather, non-essential sensor or actuator failure, operator emergency landing directive, and non-cooperative aircraft nearby. The methods and optimization techniques discussed in this paper can be used for any VTOL aircraft. The case studies and simulations presented are specific to a multicopter in an urban environment.

Steps required to construct a database of buildings and suitable ground-based landing zones (e.g., parks, fields, etc.) from OpenStreetMap (OSM) are described in our previous work [5]. This database is geospatial, in the sense that each row refers to a geographic entity (e.g., building or field) with polygonal shape. Metadata is also captured about each entity, e.g., height and land use.



**Fig. 20.3** Satellite image of an urban environment with multiple flat roof landing sites. Select roof shapes and obstacles are labeled

## 20.3 Preliminaries

### 20.3.1 Coordinates and Landing Sites

While map data will be globally georeferenced, low-altitude urban flights can be planned in a local Cartesian reference frame. Let orthogonal bases for this Cartesian coordinate frame be denoted  $\hat{\mathbf{e}}_x$ ,  $\hat{\mathbf{e}}_y$ , and  $\hat{\mathbf{e}}_z$ . The position of the UAS body frame with respect to the local Cartesian reference frame can then be defined as

$$\mathbf{O}_{UAS} = x \hat{\mathbf{e}}_x + y \hat{\mathbf{e}}_y + z \hat{\mathbf{e}}_z = [x, y, z]. \quad (20.1)$$

A set of candidate landing sites are generated within a radial footprint  $R$  defined as

$$\mathcal{S}_{ls} = \{l_i, \dots, l_n\} \quad (20.2)$$

where each  $l_i$  refers to a landing site with properties

$$l_i = \{\mathbf{c}, r_l, r_p\} \quad (20.3)$$

$$\mathbf{c} \in \mathbb{R}^3 \quad (20.4)$$

$$r_l, r_p \in \mathbb{R}, \quad (20.5)$$

where  $\mathbf{c}$  is landing site location in the Cartesian reference frame,  $r_l$  is landing site risk, and  $r_p$  is path risk. Both risk values are in domain [0, 1]. Landing site risk is calculated offline and represents the risk intrinsic to touching down at that landing site. Path risk must be calculated online and accounts for the path distance and proximity to obstacles. The calculation of  $r_l$  is shown in Sect. 20.4.4, while  $r_p$  is described in Sect. 20.5.

### 20.3.2 3D Path Planning with Mapped Obstacles

Path planning requires a cost function to guide search space exploration in pursuit of a feasible and optimal solution. For discrete search planners, the space must first be discretized into a graph  $G(V, E)$ , where  $V$  denotes graph vertices with edge set  $E$  and associated transition costs. This graph is defined implicitly for a 3D grid. The vertices are the cells/voxels accessed with indices  $(i, j, k)$ . The edge of each cell are dynamically computed from the 26 neighbors. In a mapped environment, A\* search applies a heuristic to reduce search overhead. A\* sums actual path cost  $g(n)$  and heuristic  $h(n)$  estimating cost-to-go to form node  $n$  total cost  $f(n)$ :

$$c(n', n) = \text{dist}(n', n) \cdot (1 + \text{risk}(n)) \quad (20.6)$$

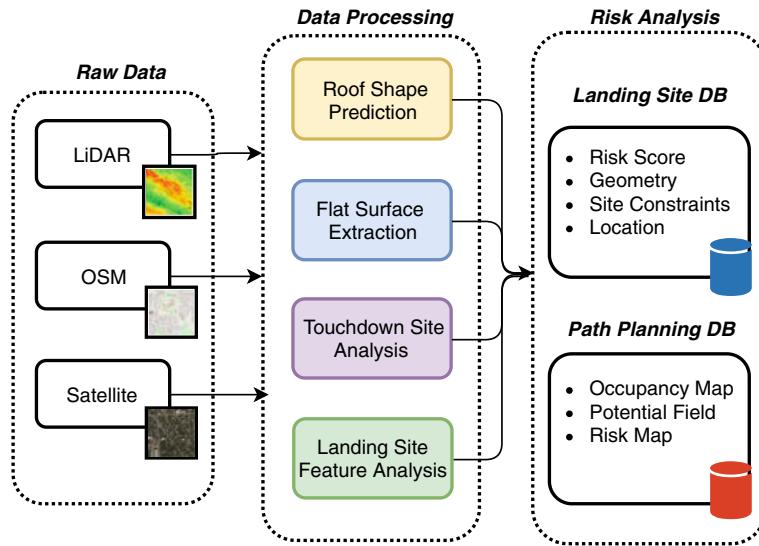
$$g(n) = g(n') + c(n', n) \quad (20.7)$$

$$f(n) = g(n) + h(n), \quad (20.8)$$

where  $c$  is transition cost from previous node  $n'$  to current node  $n$ ,  $\text{dist}(\cdot)$  is Euclidean distance between adjacent nodes  $n'$  and  $n$ , and  $\text{risk}(\cdot)$  represents normalized risk encoded in the 3D map. The search space is limited to cells within the radial footprint  $R$  to bound worst-case scenarios. We use a 3D octile distance heuristic  $h(n)$  which has been shown much more effective than Euclidean distance [24].

## 20.4 Landing Site Database

Data is fused from OSM, LiDAR, and satellite image sources to construct a feature-rich landing site database. Figure 20.4 provides an overview of the data processing pipeline that provides the features needed to construct a landing site database with risk metrics and maps. Below, Sect. 20.4.1 summarizes the machine learning (ML) process detailed in [5] to identify flat-like rooftops from which landing sites are identified. Section 20.4.2 describes our procedure to refine each landing site by determining usable area. Section 20.4.3 describes a method to compute the touchdown location on large flat surfaces. Landing site risk models are described in Sect. 20.4.4.



**Fig. 20.4** Data processing pipeline to construct landing site database and occupancy map databases with risk values

#### 20.4.1 Flat-Like Roof Identification

Information on building roof shape is sparse in existing databases. The authors have previously published data and ML models for predicting roof shape by fusing building outlines, satellite images, and airborne LiDAR data [3]. Our previous paper shows that a total accuracy of 86% is achievable when evaluating a trained ML pipeline on independent test sets. By adjusting the confidence threshold to 50%, precision and recall of 95%, and 75% can be achieved in classifying flat-like roofs in cities. This chapter employs our roof shape prediction model to label building roof shapes for subsequent landing site identification. High precision is necessary to assure that few false positives will be identified. Note that offline map building allows human inspection of city-wide rooftop landing sites to assure each identified unsafe site is pruned prior to landing site database use by a UAS.

Figure 20.5 shows results of our roof prediction model for the cities of Ann Arbor, Michigan, Witten, Germany, and mid-town Manhattan in New York City. In previous work only parks and grasslands were processed to identify emergency landing zones. This work adds new landing site options identified from the illustrated flat-like rooftops as described below.



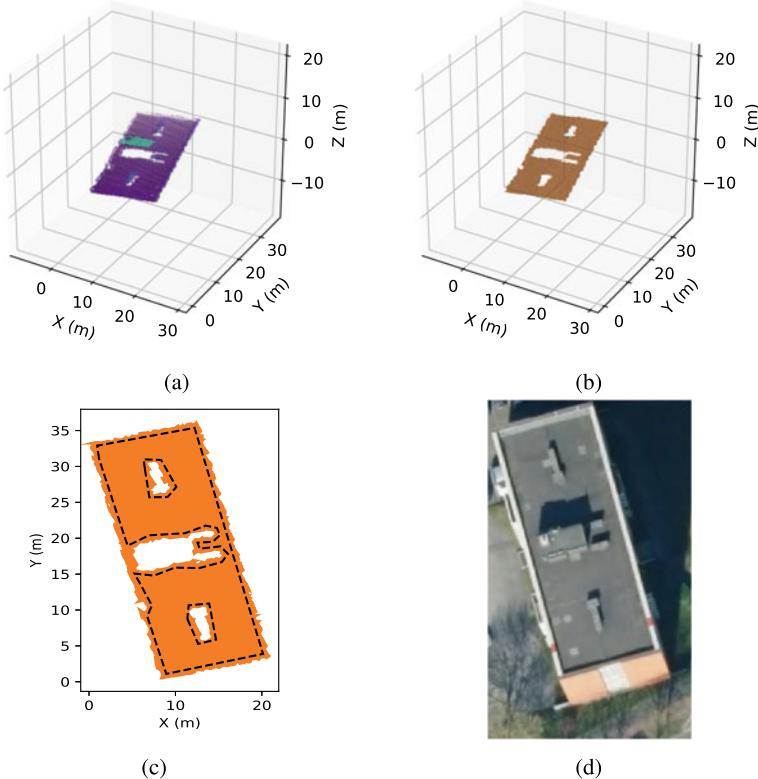
**Fig. 20.5** Three city maps. Buildings with a predicted flat-like roof shape are outlined in dark yellow with blue dashed lines through the center. Parks and grasslands are shown in green. Maps from ©OpenStreetMap contributors and ©CARTO. License: Open Database License: <https://www.openstreetmap.org/copyright>

#### 20.4.2 Flat Surface Extraction for Usable Landing Area

Airborne LiDAR point cloud data is used to determine planarity, shape, and extent of potential UAS landing site surfaces. Proposed polygonal landing sites are buildings with predicted flat-like roof shapes and terrain locations with land use keywords listed in Table 20.1. A point-in-polygon ray casting algorithm is used to generate the point set  $P_{ls}$  [34] where point clouds only reside in the outline of the landing site.

Next, flat surface extraction from  $P_{ls}$  is performed using the Polylidar algorithm developed by the authors in [4, 15]. The algorithm works by generating triangular meshes of an input point cloud, filtering triangles by planarity and edge length, extracting subsets of the mesh which are spatially connected, and finally converting the mesh to a polygon format. Polylidar is configurable by user provided planarity constraints and maximum triangle edge length. This guarantees the polygon can represent flat surfaces with interior holes denoting obstacles. Highlights of this procedure are shown in Fig. 20.6.

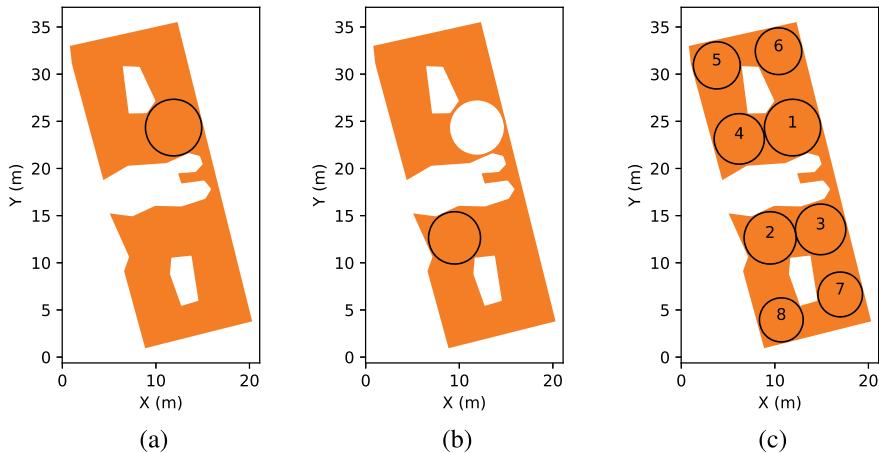
After the flat region is extracted, e.g., the orange polygon in Fig. 20.6c, this polygon is buffered inward and simplified as denoted by dashed lines. The buffering process is defined as the Minkowski difference of the polygon with a circle with radius equal to a buffer distance [12]. We set the buffer distance to 0.5 m which contracts the exterior hull and expands the interior holes. Afterward we use the Douglas–Peucker’s simplification algorithm to remove redundant vertices and “smooth” the polygon [43]. Narrow flat surfaces can be removed in this process as shown near the (6, 17) point in subfigure (c). The final output of this procedure is a set of polygons denoting flat and obstacle-free surfaces.



**Fig. 20.6** Flat rooftop surface extraction example. Point cloud data for a building is displayed in **a**; **b** shows the generated planar triangular mesh. Conversion of the 3D mesh to a 2D polygon is shown in **c** with subsequent polygon inward buffering and simplification shown in dashed lines. A reference satellite image is shown in **d**

#### 20.4.3 Touchdown Points

Once flat surface(s) have been extracted from potential rooftop and ground-based landing sites and represented as polygons, an ideal landing *touchdown point* must be defined. We define this ideal point to be farthest from any non-flat region or obstacle. In other words, the touchdown point is the furthest distance away from the exterior hull and any interior holes. This requirement may be framed as the Poles of Inaccessibility problem [14] and the Polylabel algorithm as proposed in [18] provides a solution. This algorithm aims at efficiently determining the largest inscribed circle in a polygon within a prescribed tolerance. The largest inscribed circle for the same rooftop in Fig. 20.6 is shown in Fig. 20.7a. There are additional suitable touchdown sites on this rooftop that can be used. We propose Algorithm 20.1 to capture the remaining touchdown points as a ranked list of circles. The algorithm begins by calling Polylabel to find the point and radius representing the largest circle inside an



**Fig. 20.7** Touchdown points on rooftop surface shown in orange. The best (largest circle) landing zone is shown in **a**. **b** Polygon subtraction of a previously found touchdown zone. The complete ranked touchdown site set is shown in **b**

input polygon  $P$ . A 16-sided polygon representation of this circle is created denoted  $P_c$ . If the radius is below a user provided minimum radius  $r_{min}$  then the empty set is returned.  $P_c$  is subtracted from the input polygon to create a smaller polygon  $P_{diff}$ . The procedure ends by returning the union of  $P_c$  and the result of a recursive call to TouchdownExtraction with  $P_{diff}$  as the input polygon. This recursive call continues the process of finding next largest circle. The end result is an ordered set of circular polygons with a radius greater than  $r_{min}$ . This minimal radius is user defined and should be determined by UAS characteristics. Visualization after the first and second procedure call is in Fig. 20.7a, b, respectively, with the final rankings shown in Fig. 20.7c.

---

#### Algorithm 20.1 TouchdownExtraction

---

```

Input : Polygon:  $P$ , Minimum Radius:  $r_{min}$ 
Output : Set of Polygon Circles
1 point,  $r = \text{Polylabel}(P)$ 
2  $P_c = \text{PolgonCircle}(\text{point}, r)$ 
3 if  $r < r_{min}$ :
4   | return  $\emptyset$ 
5  $P_{diff} = P - P_c$ 
6 return  $P_c \cup \text{TouchdownExtraction}(P_{diff}, r_{min})$ 

```

---

The final landing site chosen for each surface is the top ranked touchdown site. The remaining circles are kept in the database for further use in risk assessment.

### 20.4.4 Landing Site Risk Model

Each landing site corresponds to the largest clear touchdown area on either flat terrain or building rooftop surfaces. This paper adopts the risk model first presented in Ref. [5]. Risk is quantified as vehicle cost ( $C_v$ ), property cost ( $C_p$ ), and human occupancy cost ( $C_o$ ). Each of these risks are numeric values created from a functional composition of the attributes of each feature (land use, available area, etc.) as outlined in Sects. 20.4.4.1, 20.4.4.5, and 20.4.4.6, respectively. Landing site risk is the weighted sum

$$r_l = w_v \cdot C_v + w_s \cdot C_p + w_o \cdot C_o \quad (20.9)$$

$$w_v + w_s + w_o = 1,$$

where  $w_v$ ,  $w_s$ , and  $w_o$  are weights for vehicle, property, and human occupancy cost, respectively.

#### 20.4.4.1 Vehicle Cost

We denote risk to the UAS “vehicle” as

$$C_v = w_t C_t + w_a C_a + w_{ca} C_{ca} \quad (20.10)$$

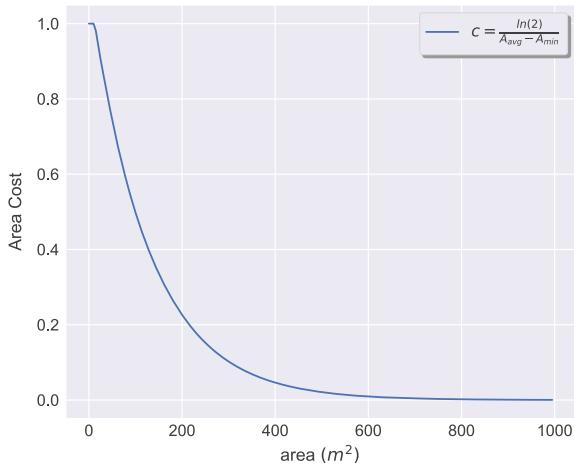
$$C_v, C_t, C_a, C_{ca} \in [0, 1], \quad (20.11)$$

where  $C_t$ ,  $C_a$ , and  $C_{ca}$  are risk-based costs associated with terrain type, usable area, and cumulative usable area, respectively. The variables  $w_t$ ,  $w_a$ , and  $w_{ca}$  are the user-defined weights aggregating these metrics into a total  $C_v$  value.

#### 20.4.4.2 Terrain Cost

Terrain cost  $C_t$  approximates the risk posed to the vehicle due to landing on a specific type of terrain. We use keywords gathered from OSM that describe type of terrain. Following a similar taxonomy from [8], these keywords are aggregated into groups and assigned costs as shown in Table 20.1. The trend of these costs is that groups with generally unoccupied open areas, such as Group 2, have lower risk than groups with possible cluttered areas, such as Group 5. Building rooftops, Group 1, have a slightly higher terrain cost than Group 2 because of increased risk at landing at higher altitudes. Group 7, industrial and commercial areas, have more diverse and uncertain terrain characteristics and assigned a higher cost. These costs are subjective in nature thus would be refined later by stakeholders.

**Fig. 20.8** Example function mapping area size to a cost value between [0, 1]. In this example  $A_{avg}$  and  $A_{min}$  are 100 and  $12.5\text{ m}^2$ , respectively



#### 20.4.4.3 Area Cost

Large flat surfaces pose less risk to UAS than small clearings. Our proposed risk model quantifies this as an area cost  $C_a$ . We propose Eq. 20.12 to map small areas to high risk (1), average areas to medium risk (0.5), and large areas to low risk (0). This piecewise exponentially decaying function is governed by user-defined minimum area  $A_{min}$  and maximum area  $A_{avg}$ . Note that  $A_{min}$  is the area of the circle with radius  $r_{min}$  used in Algorithm 20.1. These values would take UAS-specific landing area requirements into account.

$$\text{Area Cost}(a) = \begin{cases} 1 & a \leq A_{min} \\ e^{-c \cdot a} & a > A_{min} \end{cases} \quad (20.12)$$

$$c = \frac{\ln 2}{A_{avg} - A_{min}}. \quad (20.13)$$

An example of this mapping can be found in Fig. 20.8

#### 20.4.4.4 Cumulative Area Cost

A landing site that is nearby additional sites has advantage to landing sites with no other nearby options. We capture this metric as a cumulative area cost  $C_{ca}$ . The cumulative area of all touchdown sites available on an individual building or terrain surface is computed as shown in Fig. 20.7c. This area is then mapped to  $C_{ca}$  using Eq. 20.12.

**Table 20.1** Terrain type and property cost

Group	Keywords	Terrain ( $C_t$ )	Property ( $C_p$ )
Group 1	Building rooftops	0.25	0.5
Group 2	Brownfield, grass, grassland village green, greenfield	0.0	0.0
Group 3	Meadow, cemetery, scrub	0.25	0.0
Group 4	Water, riverbank	0.75	0.0
Group 5	Recreation ground, garden, golf course, track, pitch, playground, common, park	0.5	0.25
Group 6	Parking	0.75	0.75
Group 7	Industrial, commercial	1.0	1.0

#### 20.4.4.5 Property Cost

UAS may inadvertently damage a landing site in the event of an unplanned landing. Landing site characteristics can impact the likelihood, severity, and cost of damage. Table 20.1 provides example normalized estimates for the costs of similar landing sites. The proposed metric assigns increase cost to areas that may be damaged in the event of a high impact crash. Natural terrain areas have low cost while buildings and other maintained properties have higher cost. Parking lots and industrial areas are marked with the highest cost to people and property, e.g., cars, pedestrians.

#### 20.4.4.6 Human Occupancy Risk Mapping

Overflight risk to people is typically estimated with an occupancy grid based on Census records [1, 37, 38]. Census records in the United States are released every 10 years in an aggregated form in which the smallest areal unit is the census block. The size of a census block may vary from a city block to a much larger area in a rural community. For example, in New York City the average size of a census block is a  $121 \times 121$  meter square but with substantial variance over the city landscape. This average resolution is suitable for city-wide risk assessment but not ideal for higher resolution occupancy mapping. Techniques such as dasymetric modeling [23] are used to improve the spatial resolution of Census datasets at the cost of greater uncertainty [9].

Census records only provide information of where people reside which is most often representative of a region's nighttime population [8]. UAS will fly at all times of the day so time-varying population estimates are critical for accurate risk assessment. Work by Ref. [8] fused census data with publicly available mobile phone call detail records (CDRs) in Italy to generate a temporal population model for UAS risk assessment. Results indicated significant population migration throughout the day reinforcing the inadequacy of using a static population model. However, CDRs are

not generally open for public use, and access to other real-time data streams such as aggregated mobile phone GPS is limited. Most of the landing sites proposed in this paper are on flat rooftops likely to be unoccupied despite high building occupancy. Many population risk models introduce a shelter factor which estimates zero casualties whenever the building is not penetrated, e.g., during a UAS rooftop landing [20].

This paper requires risk assessment for landings sites and paths over small radial footprints ( $<250\text{m}$ ). The authors have previously used census data to assess population risk in this situation. However, a static low-resolution population model may provide misleading risk assessments. Therefore this paper does not use population risk metrics by setting  $w_o$  to 0 in Eq. 20.9. Our combined risk model will include path length, which when minimized, is a proxy to minimizing the risk to people during urgent landings so long as population is uniformly distributed rather than clustered, e.g., for special events not modeled in census data.

## 20.5 Three-Dimensional Maps for Path Planning

Let the city occupancy and risk map generated for path planning be denoted  $R_{map}$ . This map is a dense 3D voxel structure of size  $M \times N \times K$ , where the rows, columns, and slices are  $M$ ,  $N$ , and  $K$ , respectively. Each cell is indexed by triplet  $(i, j, k)$  returning occupancy and risk information for a specific position. Publicly available airborne point cloud data or digital surface maps (DSM) may be used as the primary data source to construct  $R_{map}$ . A DSM is a raster where each pixel holds a height value above Earth's mean sea level (MSL) including buildings and foliage. Such data sources are often georeferenced in a projected coordinate system which minimizes distortion of shape, area, or distance. This Cartesian coordinate system is ideal for path planning thus is carried into the voxel map. The rest of this procedure assumes the use of a DSM with equal pixel resolution and associated affine transformation matrix to convert from pixel space to the local Cartesian frame.

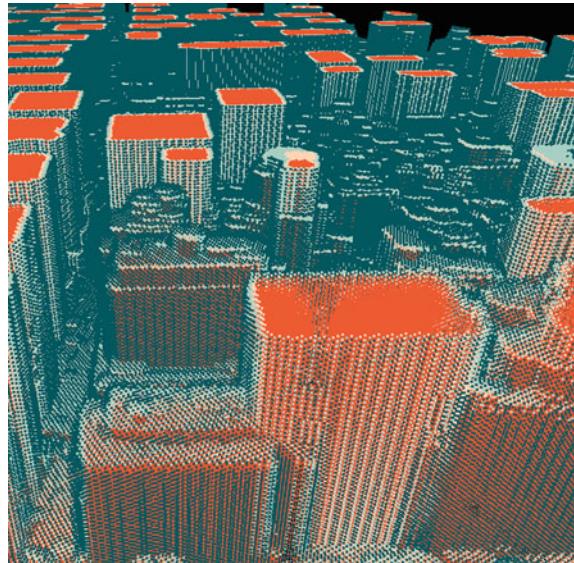
The procedure begins with a city DSM of size  $M \times N$ . The minimum and maximum height,  $z_{min}$  and  $z_{max}$ , are computed from the DSM. The value of  $z_{max}$  is bounded at 400 ft above local terrain level in this work. Note that FAA Part 107 restricts flying small UAS more than 400 ft above the tallest nearby obstacle [11]. An affine transformation matrix  $A$  is generated for  $R_{map}$ :

$$A = \begin{bmatrix} x_{res} & 0 & 0 & x_{min} \\ 0 & -y_{res} & 0 & y_{max} \\ 0 & 0 & z_{res} & z_{min} \end{bmatrix} \quad (20.14)$$

$$[x \ y \ z]^T = A \cdot [i \ j \ k]^T, \quad (20.15)$$

where  $x_{min}$ ,  $x_{res}$ ,  $y_{max}$ , and  $y_{res}$  are provided by the DSM affine matrix. Equation 20.15 performs the conversion from 3D voxel space to the local Caretesian coordinate system. The number of slices,  $K$ , is equal to  $\lfloor * \rfloor \frac{z_{max}-z_{min}}{z_{res}}$ . An  $M \times N \times K$  data structure storing unsigned 8 bit integers is zero initialized to represent  $R_{map}$ .

**Fig. 20.9** Example occupancy and risk map for New York City. Obstacles are colored orange with a surrounding potential field denoted by pink, light blue, and dark blue colors. Buildings which do not fit in the map (higher than 400 ft AGL), are shown with orange roofs



The occupancy map is generated similar to [38] where each  $(i, j)$  cell in the DSM is matched to an  $R_{map}$  cell  $(i, j, k)$ . The index  $k$  is calculated by  $\lfloor * \rfloor \frac{DSM(i, j) - z_{min}}{z_{res}}$ . All cells in  $R_{map}$  at the  $(i, j)$  position and below the  $k$  slice are then set to the value 255 to indicate an obstacle exists. This is done for each pixel in the DSM until a full 3D occupancy map is generated for  $R_{map}$ . Afterward a potential field cost is applied to  $R_{map}$  which fills empty cells near each obstacle cell with nonzero risk values. Three possible levels of potential field risk are assigned to an empty cell based upon its shortest Manhattan distance to an obstacle cell. Distances of one, two, and three provide integer risk values of 254, 170, and 85, respectively. An example 3D grid of New York City is shown in Fig. 20.9.

An A\* path planner is used to generate optimal collision-free trajectories inside  $R_{map}$ . Obstacle nodes, cells with a value of 255, are ignored for state transitions. The risk function described in Eq. 20.6 is defined as

$$\text{risk}(i, j, k) = \frac{R_{map}(i, j, k)}{255}. \quad (20.16)$$

The path risk to a goal cell,  $n_g$ , is the path cost to the goal node normalized by  $R$

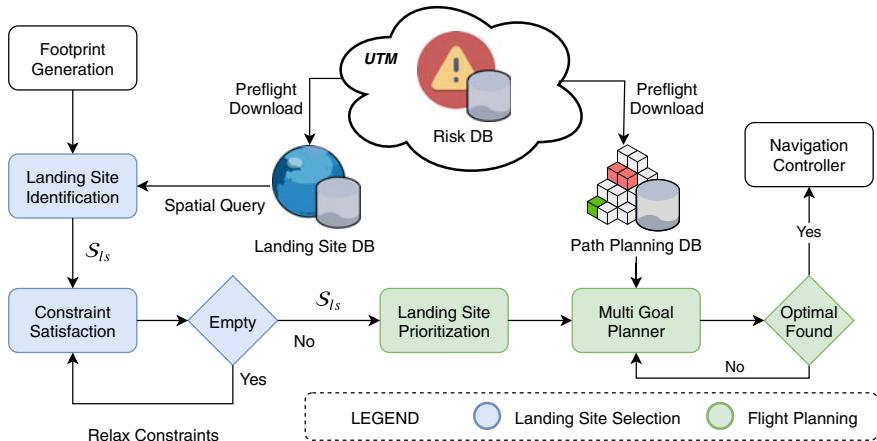
$$r_p = \frac{g(n_g)}{R}. \quad (20.17)$$

## 20.6 Planning Risk Metric Analysis and Integration

Section 20.6.1 describes the architecture of our proposed map-based planner. Section 20.6.2 outlines the inherent trade-off between landing site and path risk and our planning method. Section 20.6.3 provides underlying theoretical and algorithmic formulations of our multi-goal planner.

### 20.6.1 Real-Time Map-Based Planner Architecture

The proposed architecture for our map-based planner is shown in Fig. 20.10 which is modified from our previous work [2, 5]. The landing site and path planning database might be provided as part of NASA’s UAS Traffic Management (UTM) service [31]. Before mission operations begin, a preflight download commences from UTM servers to retrieve relevant data for the flight operational area. These data are lightweight thus can be stored onboard the UAS. In the event an urgent landing situation arises our map-based planner logic will be executed. First a footprint specifying the bounds of the reachable landing area is generated. Construction of such a footprint is not the focus of this paper; however, work by [2, 30] and [38] have investigated its generation for fixed-wing and multicopter aircraft, respectively. For simplification our footprint is a circle of radius  $R$  whose center is the UAS position. Next we efficiently query the spatially indexed landing site database to provide the set of risk-evaluated landing sites  $S_{ls}$ . Landing sites are filtered by user-defined constraints such as landing site height or area. If no valid landing sites are found, constraints are relaxed as needed.



**Fig. 20.10** Proposed map-based planner

The planner must identify a low-risk landing site and flight plan from the set of candidate landing sites  $\mathcal{S}_{ls}$ . In order to assess each landing site's path risk the physical path to each landing site is needed. Optimal collision-free path planning in three-dimensional space can take a significant amount of time thus is impractical to perform in real time for the numerous potential landing sites that may be available for a small UAS in a city environment. Therefore we use a heuristic to prioritize landing sites in  $\mathcal{S}_{ls}$  by minimum total risk. This sorted list is then sent to our multi-goal planner which efficiently searches over landing sites until the risk-optimal landing site/path pair is found. Finally the landing site and path is sent to a navigation controller.

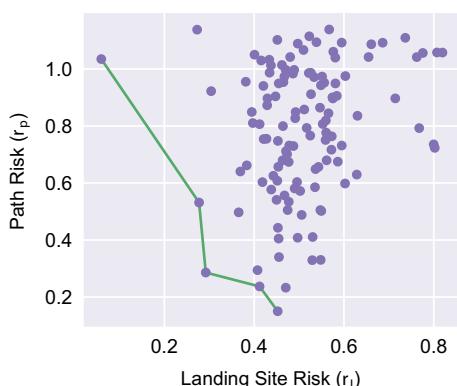
### 20.6.2 Trade-Off Between Landing Site and Path Risk

Minimizing the landing site risk and path risk to a site requires solving a multi-objective (MO) optimization problem from which there may not be a single solution simultaneously optimal over both objectives. An analysis of trade-offs is required for MO problems by computing and analyzing a Pareto frontier. Frontier visualization aids system designers in choosing the relative weighting of objective trade-offs to select a single “best” solution [26].

Figure 20.11 shows an example Pareto frontier that minimizes two objectives: landing site risk and path risk. Each purple dot represents a landing site. The  $x$ -axis represents landing site risk and the  $y$ -axis represents path risk to that site. The green line connects three points on the Pareto frontier, the set of non-dominated landing sites for which any improvement in one objective results in a negative trade-off in the other. Each of these three landing sites is “optimal”, and a quantifiable relationship between each objective must be constructed to select a final choice. A linear weighting scheme between the objectives is proposed below for each landing site  $l_i \in \mathcal{S}_{ls}$ :

$$r_t = w_l \cdot r_l + w_p \cdot r_p, \quad (20.18)$$

**Fig. 20.11** Example trade-off between landing site risk and path risk. Points in the Pareto frontier are connected by a green line



where  $r_t$  refers to the total risk and  $w_l$  and  $w_p$  are weights for landing site risk and path risk, respectively. The optimal landing site can then be found by solving the optimization problem shown in Eq. 20.19.

$$l_i^* = \arg \min_{l_i \in \mathcal{S}_{ls}} r_t. \quad (20.19)$$

### 20.6.3 Multi-goal Planner

Our multi-goal planner selects a landing site that minimizes total risk given a user-defined weighted trade-off between landing site risk and path risk. Each landing site's position and risk is assumed known a priori. Path risk cannot be known until the physical path is computed. Because this work relies on pre-processed map data rather than real-time perception, path planning has the highest real-time computational overhead. Multi-goal search allows exploration of many low-risk landing sites but will be computationally expensive. Our planner efficiently prunes high-risk goals/paths from the search space to reduce computational overhead. Ultimately the planner returns one goal/path pair minimizing combined total risk. The algorithm begins by creating an array of landing site data structures with the following form:

$$l_i = \{\mathbf{c}, r_l, r_p, r_{t,min}, \text{found}\} \quad (20.20)$$

$$\mathbf{c} \in \mathbb{R}^3 \quad (20.21)$$

$$r_l, r_p, r_{t,min} \in \mathbb{R} \quad (20.22)$$

$$\text{found} \in \{0, 1\}, \quad (20.23)$$

where  $\mathbf{c}$  is landing site position,  $r_l$  is landing site risk,  $r_p$  is path risk,  $r_{t,min}$  is minimum total risk, and  $\text{found}$  is a Boolean indicating whether a path has been found to landing site  $i$ . The minimum total risk is computed from

$$r_{t,min} = w_l \cdot r_l + w_p \cdot h(\mathbf{O}_{UAS}, \mathbf{c})/R, \quad (20.24)$$

where  $h()$  is an admissible heuristic, 3D octile distance in this work. We access elements in  $l_i$  through dot (.) notation, e.g.,  $l_i.r_{t,min}$  refers to the minimum total risk of the  $i$ th landing site in  $\mathcal{S}_{ls}$ . Section 20.6.3.1 provides definitions and a theorem for our multi-goal planner. Section 20.6.3.2 describes the planning algorithm and its implementation.

#### 20.6.3.1 Theory

**Definition 20.1** (*Total Ordered Set*) Binary relation,  $\leq$ , is a total order on set  $\mathcal{X}$  if  $\forall a, b \in \mathcal{X}$

1.  $a \leq b$  and  $b \leq a \implies a = b$  Anti-symmetry
2.  $a \leq b$  and  $b \leq c \implies a \leq c$  Transitivity
3.  $a \leq b$  or  $b \leq a$  Connexity.

We define binary operator  $\leq$  on the set  $\mathcal{S}_{ls}$   $\forall l_i, l_j \in \mathcal{S}_{ls}$ :

$$l_i \leq l_j : l_i.r_{t,min} \leq l_j.r_{t,min}. \quad (20.25)$$

This operator is used to sort  $\mathcal{S}_{ls}$  such that the natural numbers  $i, j \in [1, N]$  index with the following property:

$$\forall l_i, l_j \in \mathcal{S}_{ls} \ l_i \leq l_j \iff i \leq j, \quad (20.26)$$

where  $N = |\mathcal{S}_{ls}|$ . We use bracket operator  $[.]$  to index  $\mathcal{S}_{ls}$ , e.g.,  $l_i = S_{ls}[i]$ .

**Theorem 20.1** *Let  $i^* \in [1, N]$  and  $k \in [1, N]$  be natural numbers where  $i^* \leq k$ . If  $\forall j \in [1, k]$ ,  $l_{i^*}.r_t \leq l_j.r_t$  and  $l_{i^*}.r_t \leq l_{k+1}.r_{t,min}$  then  $l_{i^*}$  has the minimum total risk:*

$$l_{i^*} = \arg \min_{l_i \in \mathcal{S}_{ls}} l_i.r_t. \quad (20.27)$$

**Proof** The stated inequalities partition  $\mathcal{S}_{ls}$  into two ordered sets for some index  $k \in [1, N]$ . We denote  $\mathcal{S}_{ls}^{low} = \{l_1, \dots, l_k\}$  where  $l_{i^*} \in \mathcal{S}_{ls}^{low}$  and  $\mathcal{S}_{ls}^{hi} = \{l_{k+1}, \dots, l_N\} \cup \{l_{i^*}\}$ . We must show that  $l_{i^*}$  represents the minimum total risk in both sets. When  $\forall j \in [1, k] l_{i^*}.r_t \leq l_j.r_t$  holds true, by the definition of argmin we know that

$$(1) \quad l_{i^*} = \arg \min_{l_i \in \mathcal{S}_{ls}^{low}} l_i.r_t.$$

To show  $l_{i^*}$  has the minimum actual total risk of  $\mathcal{S}_{ls}^{hi}$  we begin by noting that for each landing site  $l_j$

$$(2) \quad \forall j \in [1, N], \ l_j.r_{t,min} \leq l_j.r_t.$$

If a landing site  $l_i$  has total risk  $l_i.r_t$  which is less than the minimum total risk of landing site  $l_j$  denoted  $l_j.r_{t,min}$  then

$$(3) \quad \forall i, j \in [1, N] \ l_i.r_t \leq l_j.r_{t,min} \implies l_i.r_t \leq l_j.r_t.$$

From the transitivity property of  $\mathcal{S}_{ls}$  we obtain

$$(4) \quad \exists i^*, k \in [1, N] \ s.t. \ l_{i^*}.r_t \leq l_{k+1}.r_{t,min} \implies \forall j \in [k+1, N] \ l_{i^*}.r_t \leq l_j.r_{t,min}.$$

Finally by combining (3) and (4) we obtain

$$(5) \quad \exists i^*, k \in [1, N] \ s.t. \ l_{i^*}.r_t \leq l_{k+1}.r_{t,min} \implies \forall j \in [k+1, N] \ l_{i^*}.r_t \leq l_j.r_t.$$

Using the definition of argmin we restate (5) as

$$(6) \quad \exists i^*, k \in [1, N] \text{ s.t. } l_{i^*} \cdot r_t \leq l_{k+1} \cdot r_{t,\min} \implies l_{i^*} = \arg \min_{l_i \in \mathcal{S}_{ls}^{hi}} l_i \cdot r_t.$$

Statements (1) and (6) show that  $l_{i^*}$  has the minimum total risk in  $\mathcal{S}_{ls}^{low}$  and  $\mathcal{S}_{ls}^{hi}$  if both qualifying predicates hold true. Therefore the union of these sets,  $\mathcal{S}_{ls}$ , has the same minimum  $l_{i^*} \cdot r_t$ .

**Remark 20.1** There may *not* exist a  $k$  for which the second clause  $l_{i^*} \cdot r_t \leq l_{k+1} \cdot r_{t,\min}$  in Theorem 20.1 holds true. In this case  $\mathcal{S}_{ls}^{low} = \mathcal{S}_{ls}$  and path planning must be performed for every landing site to guarantee risk-optimality.

### 20.6.3.2 Multi-goal Path Planning Algorithm

Our multi-goal path planner is shown in Algorithm 20.2. First,  $\mathcal{S}_{ls}$  is sorted by minimum total risk as described in Definition 20.1. The algorithm next initializes variables  $\hat{l}_{\min}$ ,  $\hat{p}_{\min}$ , and  $\hat{r}_t$  to track the minimum risk landing site, associated path, and total risk, respectively. These variables represent the current best landing site/path pair and are updated each time a lower risk landing site is found.

Our algorithm repeatedly investigates the next most promising landing site  $l_i$  from  $\mathcal{S}_{ls}$ . Line 11 starts a path planning sequence with 3D octile distance heuristic guiding the planner to  $l_i$ . The planner is opportunistic so that other landing sites (goals) may be found during the search. For this reason the identified landing site is returned from function `PathPlanning`,  $l_j$ , and will not always be equal to  $l_i$ . The true total risk of  $l_j$  is then calculated for the full flight plan and its *found* flag set. This landing site's total risk is then compared to the current best and updated if appropriate, ensuring the first clause of Theorem 20.1 is satisfied.

The first element in  $\mathcal{S}_{ls}$  which has not been found is returned in Line 18 as the next unplanned landing site that has minimum total risk. The tracked total risk is compared with this element's minimum total risk, and if less or equal will satisfy the second clause in Theorem 20.1. Once both predicates are satisfied iteration terminates and  $\hat{l}_{\min}$ ,  $\hat{p}_{\min}$  are returned as the risk-optimal landing site and plan, respectively. If the optimal site is not found then the procedure continues. Line 21 ensures that if  $l_i$  was not found ( $l_j$  is *not*  $l_i$ ) then the planner will retry  $l_i$  in the next iteration. This is accomplished by decrementing the loop variable  $i$ . The total number of iterations is equal to  $k$  in Theorem 20.1 which represents the number of landing sites searched.

Note that the algorithm can return the best found landing site and path at any iteration step if computational time becomes a concern. In addition a worst-case bound of unnecessary risk can be computed from the difference between the returned landing site's total risk and the minimum total risk of the next landing site to be searched.

**Algorithm 20.2** Multi-Goal Search

---

**Input :** Landing Site Set ( $\mathcal{S}_{ls}$ ),  
     Map ( $M$ ),  
     UAS Location ( $\mathbf{O}_{UAS}$ )  
     Footprint Radius ( $R$ ),  
     Weighting Trade-off ( $w_l, w_p$ ),  
     Heuristic ( $h(c_i, c_j)$ )

**Output :** Best landing site and path,  $\hat{l}_{min}$  and  $\hat{p}_{min}$

---

```

1  $\mathcal{S}_{ls} = \text{LandingSitePrioritization}(\mathcal{S}_{ls}, \mathbf{O}_{UAS}, R, w_l, w_p, h)$ 
2  $N = |\mathcal{S}_{ls}|$ 
3  $\hat{l}_{min} = \text{None}$ 
4  $\hat{p}_{min} = \text{None}$ 
5  $\hat{r}_t = \infty$ 
6 for  $i = 0$  to  $N$  do
7     $l_i = \mathcal{S}_{ls}[i]$ 
8    if  $l_i.\text{found}$  then
9      | continue
10      $goals = \{\forall l_i \in \mathcal{S}_{ls} \mid \text{not } l_i.\text{found}\}$ 
11      $l_j, p_j = \text{PathPlanning}(M, \mathbf{O}_{UAS}, R, h, l_i, goals)$ 
12      $l_j.\text{found} = \text{true}$ 
13      $l_j.r_t = w_l \cdot l_j.r_l + w_p \cdot \frac{\text{Cost}(p_j)}{R}$ 
14     if  $l_j.r_t < \hat{r}_t$  then
15       |  $\hat{r}_t = l_j.r_t$ 
16       |  $\hat{l}_{min} = l_j$ 
17       |  $\hat{p}_{min} = p_j$ 
18      $l_n = \text{FirstElement}(\{\forall l_i \in \mathcal{S}_{ls} \mid \text{not } l_i.\text{found}\})$ 
19     if  $\hat{r}_t <= l_n.r_{t,min}$  then
20       | break
21     if  $\text{not } l_i.\text{found}$  then
22       |  $i = i - 1$ 
23 end
24 return  $\hat{l}_{min}, \hat{p}_{min}$ 

```

---

**20.7 Maps and Simulation Results**

Landing site databases and path planning obstacle maps were generated for the cities of Ann Arbor, Michigan; Witten, Germany; and mid-town Manhattan, New York City, New York using the methods outlined in Sects. 20.4 and 20.5. Public data sources used for all three cities are shown in Table 20.2. Visualization and analysis of each city's databases and maps are found in Sect. 20.7.1. Section 20.7.2 presents two urgent landing scenarios for each city with results from our proposed framework. Section 20.7.3 provides statistical analysis of our planners speed and efficacy in all three cities. Table 20.3 displays parameters used throughout all case studies and simulations. The average touchdown site areas  $A_{avg}$  in each city were set to 150,

**Table 20.2** Satellite, LiDAR, and building data sources

City	Satellite	LiDAR	Buildings
Ann Arbor	Bing [22]	USGS [41]	OSM [29]
Witten	Land NRW [16]	Open NRW [28]	OSM [29]
New York	New York State [25]	USGS [40]	OSM [29]

**Table 20.3** All case study parameters

Group	Parameter	Value	Description
Landing site risk	$w_v$	0.8	Weight for vehicle cost
	$w_s$	0.2	Weight for property cost
	$w_o$	0.0	Weight for human occupancy cost
Vehicle cost	$w_t$	0.4	Weight for terrain type cost
	$w_a$	0.4	Weight for area cost
	$w_{ca}$	0.2	Weight for cumulative area cost
Multi-goal planner	$w_l$	0.6	Weight for landing site risk
	$w_p$	0.4	Weight for path risk
	$R$	250 m	Search radius footprint

100, and 100 square meters for Ann Arbor, Witten, and New York, respectively. The minimum touchdown radius and corresponding area were set to 2 m and 12.5 square meters, respectively, for all cases.

### 20.7.1 Landing Sites and Risk Maps

Figure 20.12 shows extracted landings sites and their associated risks for the cities of Ann Arbor (a), Witten (b), and New York (c). Landing site risk is color-coded from low (light yellow) to high (dark orange). Touchdown sites are displayed as blue circles. The operating regions are approximately  $1500 \times 1500$ ,  $1500 \times 1300$ , and  $1500 \times 3000$  meters for Ann Arbor (AA), Witten (WT), and New York (NY), respectively. Three-dimensional risk grids as described in Sect. 20.5 were generated for all three cities. Each voxel in  $R_{map}$  is a cube with two meter edge length. This resolution provides a balance between file size and providing sufficient resolution for use in path planning. The resulting file sizes are approximately 37, 26, and 67 MB for AA, WT, and NY, respectively.



**Fig. 20.12** Landings site risk for the cities of Ann Arbor (a), Witten (b), and New York (c). Landing sites are color-coded from low risk (light yellow) to high risk (dark orange). Touchdown sites are denoted by blue circles. Maps from ©OpenStreetMap contributors and ©CARTO. License: Open Database License: <https://www.openstreetmap.org/copyright>

### 20.7.2 Case Studies

We present two case studies for each city where an urgent landing is required for a small UAS. Figure 20.13 presents a map of each case study with locations shown in Table 20.4. The first row (a, b) is for Ann Arbor, the second row (c, d) is for Witten, and the final row (e, f) is for Manhattan. Position of the UAS during the urgent landing event is indicated by the green marker. Landing site risk is colorized from low (yellow) to high (dark orange) risk with associated touchdown sites marked as blue circles. The lowest risk landing sites, not considering path risk, are ranked and marked with blue numbered icons. Our planner's chosen landing site is marked

**Table 20.4** Case study locations

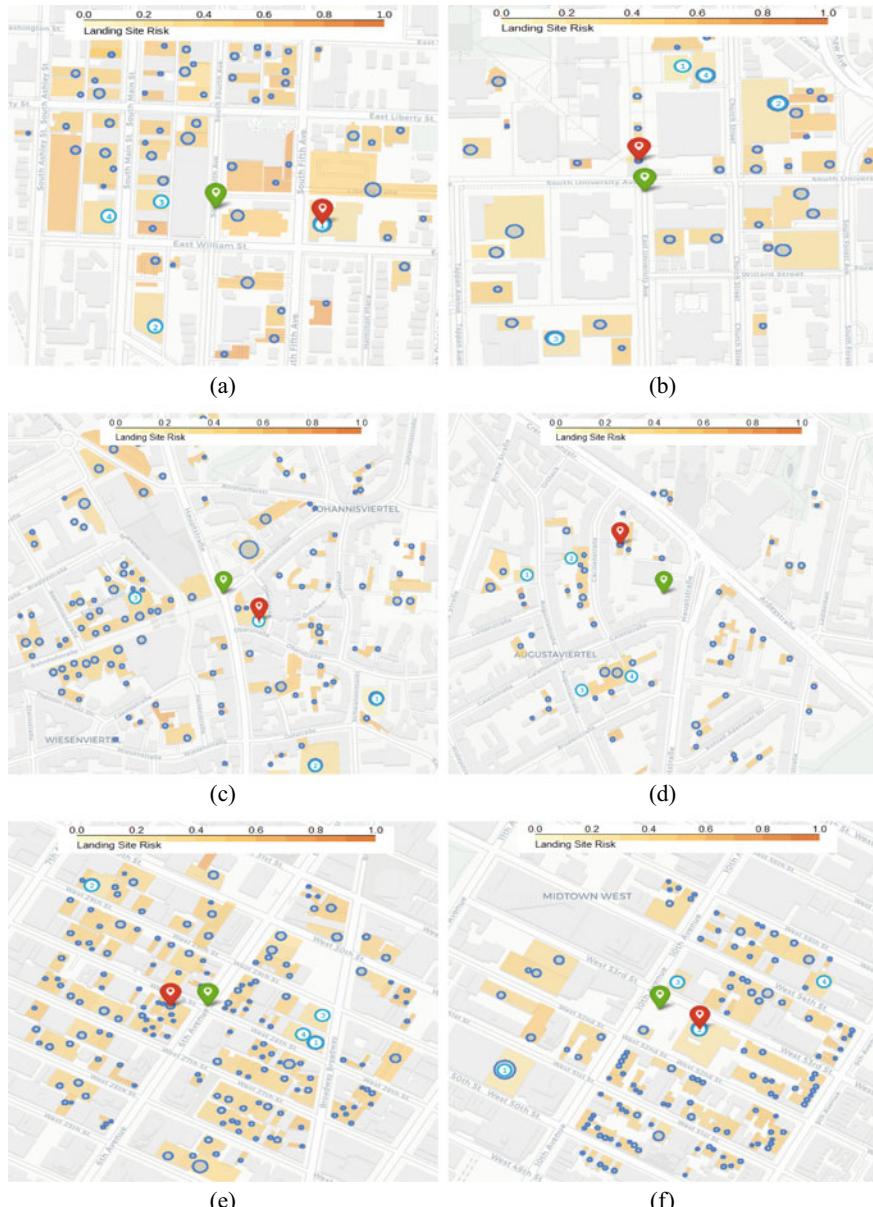
Scenario	Lng/Lat (degrees)	Height, MSL (m)
AA CS#1	42.2783, -83.7473	260
AA CS#2	42.2748, -83.7357	270
WT CS#1	51.4391, 7.3369	109
WT CS#2	51.4443, 7.3359	130
NY CS#1	40.7460, -73.9905	19
NY CS#2	40.7662, -73.9903	17

in red which trades off landing site and path risk. Pareto plots are also provided for each of these case studies in Fig. 20.14. To generate these plots, collision-free paths to *all* landing sites for each scenario were generated, providing their actual path risk. The first-, second-, and third-row correspond to Ann Arbor, Witten, and New York, respectively. Each scenario graph has the same axis limits, allowing the reader to compare each scenario visually. Each purple dot represents a landing site, while the red dot represents the landing site chosen by the map-based planner. The Pareto set for each scenario is depicted by the green line.

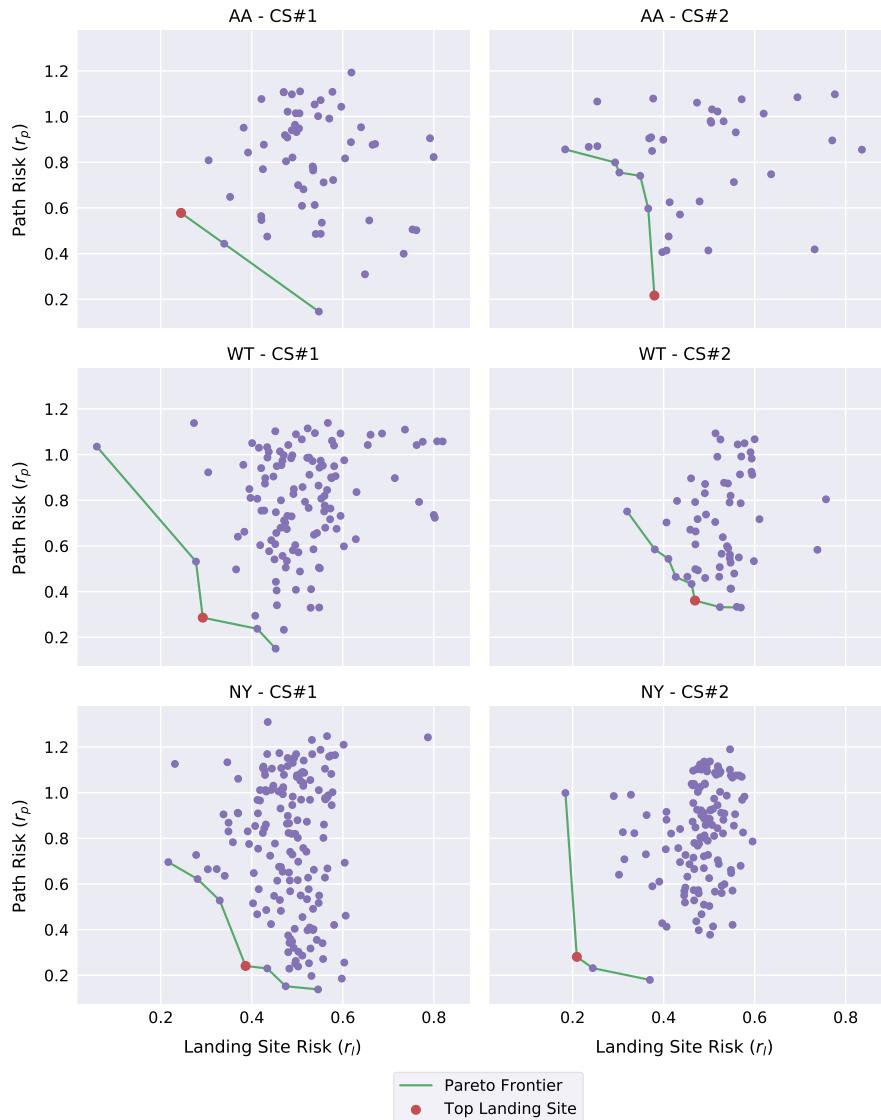
Figure 20.13a, b shows results of our map-based planner in Ann Arbor case studies 1 and 2. In case study 1 landing sites with low landing site risk are nearby, generating the Pareto frontier seen in the first row and column in Fig. 20.14. The resulting front is approximately linear and the multi-goal planner, which favors landing site risk per Table 20.3 chooses the landing site with lowest  $r_l$ . Case study 2 has the unfortunate situation where the best landing sites are far away, resulting in a Pareto front that has a sharp vertical drop. This drop allows the planner to make a trade-off between landing site risk and path risk which favors the point near the bottom of the front which represents a landing site near the UAS.

Witten case studies are shown in Fig. 20.13c, d. The Pareto front for case study 1 is nearly linear with the exception of a dip caused by one landing site (red point). The significant drop in path risk causes it to have the minimum total risk and be selected. Case study 2 shows a Pareto front shifted far to the right, indicating that few landing sites with low  $r_l$  are available. In addition few landing sites are immediately nearby for landing, forcing the planner to select a landing site which has a higher total risk than seen in case study 1.

New York City case studies are shown in Fig. 20.13e, f. A clear difference from the prior city case studies is the increased number of landing sites that are available. In New York hundreds of flat rooftops offer viable landing site options. However, it should be noted that quantity does not necessarily imply quality as many of these landing sites have high landing site risk. Case study 2 shows the fortunate situation where the UAS failure is next to a low-risk landing site causing the elbow shape Pareto front in the last row/col in Fig. 20.14. This point (marked in red) is selected by our planner because it provides the minimum total risk.



**Fig. 20.13** Maps of Ann Arbor (a, b), Witten (c, d), and New York (e, f). Failure position of the UAS is indicated by the green marker. Landing site risk is colorized from low (yellow) to high (dark orange) risk, with associated touchdown sites marked as blue circles. The lowest risk landing sites are ranked and marked with blue numbered icons. Our planner's chosen landing site is marked in red which trades off landing site and path risk. Maps from ©OpenStreetMap contributors and ©CARTO. License: Open Database License: <https://www.openstreetmap.org/copyright>



**Fig. 20.14** Simulation results for six case studies performed in Ann Arbor (first row), Witten (second row), and New York City (third row). The  $x$  and  $y$  axes are landing site risk and path risk, respectively. Each purple dots represents a landing site and its associated path, while the red dot signifies the planner's choice which minimizes total weighted risk

### 20.7.3 Urgent Landing Statistical Analysis

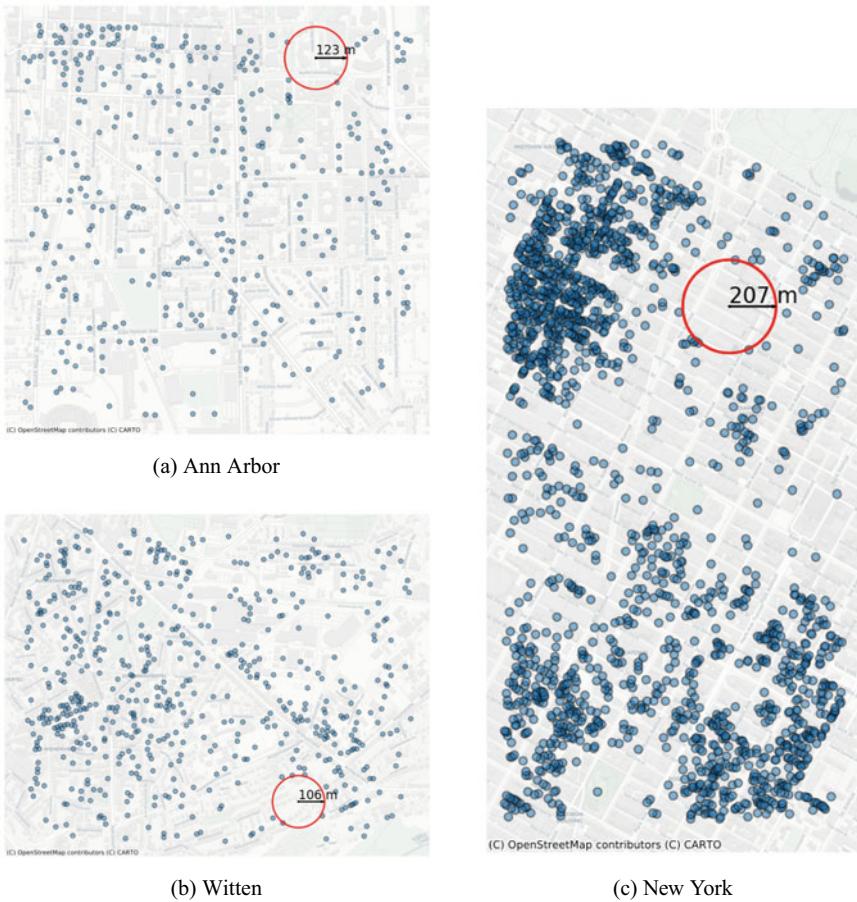
Two practical questions emerge from this study. First, how practical will it be for a small UAS to land in each analyzed city? Second, how quickly can the UAS identify a solution using the Algorithm 20.2 planner? Section 20.7.3.1 analyzes the minimum search radius needed to guarantee a landing site, offering a practical constraint on required UAS range for an urgent landing in a given region. Section 20.7.3.2 analyzes key performance metrics of our proposed planner.

#### 20.7.3.1 Minimum Radius Footprint

A search radius footprint,  $R$ , determines the maximum distance the planner will search for available landing sites. If this number is too small it is possible that no landing sites will be returned, potentially requiring an unsafe ditching/flight termination. It is therefore desirable to quantify the minimum radius footprint necessary to guarantee at least one landing site will be found anywhere in a mapped region. Figure 20.15 shows a planning area map of Ann Arbor, Witten, and New York in meters. All landing sites with a minimum radius of two meters are denoted by blue circles while the center of the red circle represents the point on the map farthest from any landing site. This point is found by finding the largest inscribed circle contained in the map that does not touch or contain any landing site. These maximum distances,  $d_{max}$ , are computed to be 123, 106, and 207 m for Ann Arbor, Witten, and New York, respectively. Note that this technique does not account for points near the edge of the map which may be farther from landing sites (such as northeast Manhattan). Therefore to guarantee a landing site is found with  $R \geq d_{max}$ , the operating region of this city map must be shrunk by each city's respective  $d_{max}$ . Alternatively, one can set  $R \geq 2 \cdot d_{max}$ .

#### 20.7.3.2 Performance Benchmarks

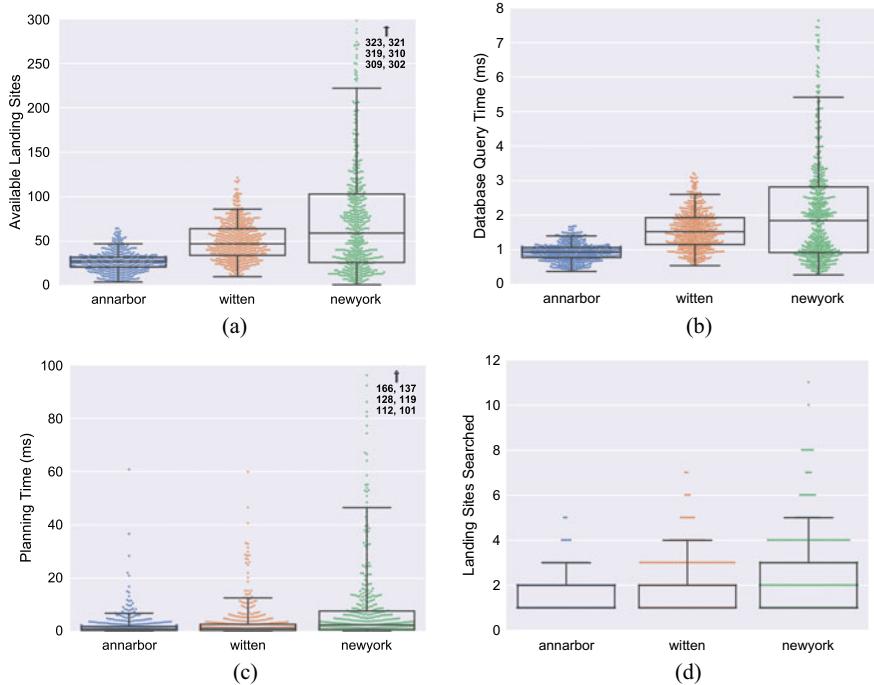
Monte Carlo simulations were performed to gather four key performance metrics of our proposed urgent landing planner: number of available landing sites in the landing footprint, database query time, multi-goal planning time, and number of landing site searched ( $k$  in Theorem 20.1). All results were computed using a desktop computer running an AMD 3900X 4.1 GHz processor. Each city had 500 uniformly sampled failure positions for which our framework provided a landing site and path with minimum total risk. Parameters were set to those from Table 20.3. Figure 20.16 displays a swarm plot with a box and whisker plot overlay showing results. Each data point is shown with the box capturing the inter-quartile range, the line in the box representing the median, and whiskers denoting 0–95 percentile. Outliers, if existing, are labeled in the top right of the graph.



**Fig. 20.15** The maximum distance to any landing site for each city is shown. Each landing site is displayed as a blue circle with the red circle center labeling the point farthest from any landing site. Maps from ©OpenStreetMap contributors and ©CARTO. License: Open Database License: <https://www.openstreetmap.org/copyright>

Figure 20.16a shows the number of available landing sites in the reachable footprint for each of the cities. Ann Arbor has the least number of landing sites, followed by Witten and then New York. Manhattan in particular has the highest number of possible landing sites. In some sections of the borough there are more than 300 landing sites within reach; this is most often near small clustered flat buildings found in the Northeast map region per Fig. 20.12c. However, in one particular case there is only one landing site available in Southern Central Park. In all simulations at least one landing site is available.

Figure 20.16b displays the number of milliseconds needed to query the database to provide available landing sites  $S_{ls}$ . The median time to execute this geospatial query



**Fig. 20.16** Comparison of key metrics between the cities of Ann Arbor, Witten, and New York in 500 random UAS initial positions with a search radius of 250 m. Each data point is shown with the box capturing inter-quartile range and whiskers denoting 0–95 percentile. Statistics are provided for number of available landing sites (a), time to query the database (b), time to find the risk-optimal landing site/path pair (c), and number of landing sites searched (d)

is under 2 ms for all cities. New York once again has a longer tail distribution since hundreds of possible landing sites are returned. This query is made efficient through the use of R\* trees for spatial indexing allowing for fast lookup in the database.

Figure 20.16c shows multi-goal planner execution time in milliseconds. Mean planning times are 2.0, 3.1, and 9.4 ms for Ann Arbor, Witten, and New York, respectively. Although the mean is low, all cities have a long tail distribution, with New York requiring up to 166 ms in one scenario. Some scenarios take longer than average because the 3D octile distance heuristic underestimates true path length particularly in New York due to its many high rise buildings presenting obstacles to be avoided. The degraded heuristic affects the planner in two ways: A\* path planning takes longer due to substantial search node expansion, and the multi-goal planner must search for more landing sites to prove the risk-optimal site is found. Figure 20.16d shows the number of landing sites searched by the multi-goal planner, i.e., the number of loop iterations in Algorithm 20.2. Each of these iterations requires an independent A\* path planning procedure. The worst case is found in New York with 11 planning iterations requiring an overall multi-goal planning time of 166 ms.

## 20.8 Conclusion

UAS operating in cities need to identify safe landing sites and associated paths in real-time whenever an urgent landing is required. This paper proposed the use of nearby flat rooftops to augment traditional emergency landing sites such as parks and fields. We showed that fusion of deep learning for roof shape identification and computational geometry for flat surface extraction results in suitable landing site identification. Landing site locations and associated risks were stored onboard UAS along with obstacle and risk maps of the local flight area. While previous risk-based planners have been proposed, our map-based planner is the first to explicitly trade-off landing site risk and path risk to minimize combined total risk. Our multi-goal planning algorithm efficiently selected the landing site/path pair guaranteed to minimize a weighted total risk function. Landing site databases and 3D risk maps were generated for three diverse cities with results presented from six case studies. Additional Monte Carlo simulations were run on all three cities to assess key performance metrics, showing that our planner finds risk-optimal landing sites and paths in less than 60 ms for 95% cases. Worst-case execution time across our tests was 174 ms in New York City. Future work can address this with a distributed or cloud-based path planner that offers speed up through parallelization. Additional risk factors such as wind patterns, rooftop material and strength, and dynamic population data will improve results. Ultimately, multiple UAS sensor data streams can be incorporated into mapping and real-time planning systems to refine risk databases.

**Acknowledgements** This work was supported in part under NSF grant CNS-1738714.

## References

1. Ancel, E., Capristan, F.M., Foster, J.V., Condotta, R.C.: Real-time Risk Assessment Framework for Unmanned Aircraft System (UAS) Traffic Management (UTM). In: 17th AIAA Aviation Technology, Integration, and Operations Conference. American Institute of Aeronautics and Astronautics (2017). <https://doi.org/10.2514/6.2017-3273>
2. Atkins, E.M., Portillo, I.A., Strube, M.J.: Emergency flight planning applied to total loss of thrust. *J. Aircr.* **43**(4), 1205–1216 (2006)
3. Castagno, J., Atkins, E.: Roof shape classification from lidar and satellite image data fusion using supervised learning. *Sensors* **18**(11), 3960 (2018)
4. Castagno, J., Atkins, E.: Polylidar - polygons from triangular meshes. *IEEE Robot. Autom. Lett.* **5**(3), 4634–4641 (2020). <https://doi.org/10.1109/LRA.2020.3002212>
5. Castagno, J., Ochoa, C., Atkins, E.: Comprehensive Risk-based Planning for Small Unmanned Aircraft System Rooftop Landing. In: 2018 International Conference on Unmanned Aircraft Systems (ICUAS), pp. 1031–1040 (2018). <https://doi.org/10.1109/ICUAS.2018.8453483>
6. DeGarmo, M.T.: Issues Concerning Integration of Unmanned Aerial Vehicles in Civil Airspace. Center for Advanced Aviation System Development (2004)
7. Desaraju, V.R., Michael, N., Humenberger, M., Brockers, R., Weiss, S., Nash, J., Matthies, L.: Vision-based landing site evaluation and informed optimal trajectory generation toward autonomous rooftop landing. *Auton. Robot.* **39**(3), 445–463 (2015). <https://doi.org/10.1007/s10514-015-9456-x>

8. Di Donato, P.F.A., Atkins, E.M.: Evaluating risk to people and property for aircraft emergency landing planning. *J. Aerosp. Inf. Syst.* (2017)
9. Dmowska, A., Stepinski, T.F.: A high resolution population grid for the conterminous United States: the 2010 edition. *Comput. Environm. Urban Syst.* **61**, 13–23 (2017). <https://doi.org/10.1016/j.compenvurbsys.2016.08.006>, <http://www.sciencedirect.com/science/article/pii/S0198971516301983>
10. Unmanned Aircraft Systems FY2019. [https://www.faa.gov/data\\_research/aviation/aerospace\\_forecasts/media/unmanned\\_aircraft\\_systems.pdf](https://www.faa.gov/data_research/aviation/aerospace_forecasts/media/unmanned_aircraft_systems.pdf). Accessed 10 Sept 2019
11. Federal Aviation Administration: Code of Federal Regulations (14 CFR) Part 107 (2016)
12. Flato, E., Halperin, D.: Robust and Efficient Construction of Planar Minkowski Sums. University of Tel-Aviv (2000)
13. Forster, C., Faessler, M., Fontana, F., Werlberger, M., Scaramuzza, D.: Continuous on-board monocular-vision-based elevation mapping applied to autonomous landing of micro aerial vehicles. In: 2015 IEEE International Conference on Robotics and Automation (ICRA). IEEE (2015). <https://doi.org/10.1109/icra.2015.7138988>
14. Garcia-Castellanos, D., Lombardo, U.: Poles of inaccessibility: a calculation algorithm for the remotest places on earth. *Scott. Geograph. J.* **123**(3), 227–233 (2007)
15. Polylidar - GitHub (2019). <https://github.com/JeremyBYU/polylidarv2>. Accessed 05 Sept 2019
16. Land NRW: Witten County Portal (2018). <https://www.land.nrw/de/tags/open-data>. Data was retrieved through ArcGIS World Imagery
17. Lim, K.L., Yeong, L.S., Ch'ng, S.I., Seng, K.P., Ang, L.M.: Uninformed multigoal pathfinding on grid maps. In: 2014 International Conference on Information Science, Electronics and Electrical Engineering. IEEE (2014). <https://doi.org/10.1109/infoseee.2014.6946181>
18. Github - polylabel (2018). <https://github.com/mapbox/polylabel>. Accessed 05 Jan 2018
19. Mejias, L., Fitzgerald, D.L., Eng, P.C., Xi, L.: Forced landing technologies for unmanned aerial vehicles: towards safer operations. *Aer. Veh.* **1**, 415–442 (2009)
20. Melnyk, R., Schrage, D., Volovoi, V., Jimenez, H.: A third-party casualty risk model for unmanned aircraft system operations. *Reliab. Eng. Syst. Saf.* **124**, 105–116 (2014). <https://doi.org/10.1016/j.ress.2013.11.016>, <http://www.sciencedirect.com/science/article/pii/S095183201300313X>
21. Meuleau, N., Plaunt, C., Smith, D.E., Smith, T.: An emergency landing planner for damaged aircraft. In: Proceedings of the 21st Innovative Applications of Artificial Intelligence Conference, pp. 71–80 (2009)
22. Microsoft: Bing Maps (2018). <https://www.bing.com/maps>. Accessed: 2018-09-05
23. Nagle, N.N., Buttenfield, B.P., Leyk, S., Speilman, S.: Dasymetric Modeling and uncertainty. annals of the association of american geographers. *Assoc. Am. Geograph.* **104**(1), 80–95 (2014). <https://doi.org/10.1080/00045608.2013.843439>, <https://www.ncbi.nlm.nih.gov/pmc/articles/PMC4109831/>
24. Nash, A., Koenig, S., Tovey, C.: Lazy Theta\*: Any-angle path planning and path length analysis in 3D. In: Twenty-Fourth AAAI Conference on Artificial Intelligence (2010)
25. New York State: 2016 Annual Lot New York County (2018). <http://gis.ny.gov/gateway/orthoprogram/lot16/new-york.htm>. Data was retrieved through ArcGIS World Imagery
26. Ngatchou, P., Zarei, A., El-Sharkawi, A.: Pareto multi objective optimization. In: 2005 Proceedings of the 13th International Conference on Intelligent Systems Application to Power Systems, pp. 84–91. IEEE (2005)
27. Ochoa, C.A., Atkins, E.M.: Fail-safe navigation for autonomous urban multicopter flight. In: AIAA Information Systems-AIAA Infotech @ Aerospace. American Institute of Aeronautics and Astronautics (2017). <https://doi.org/10.2514/6.2017-0222>
28. Open NRW: Open Geo Data (2018). <https://www.opengeodata.nrw.de/produkte/geobasis/dom/dom1/>. Accessed 05 Sept 2017
29. OpenStreetMap: planet dump retrieved from <https://planet.osm.org> (2018). <https://www.openstreetmap.org>

30. Paul, S., Hole, F., Ztyek, A., Varela, C.A.: Flight Trajectory Planning for Fixed-Wing Aircraft in Loss of Thrust Emergencies (2017). [arXiv:1711.00716](https://arxiv.org/abs/1711.00716)
31. Prevot, T., Rios, J., Kopardekar, P., III, J.E.R., Johnson, M., Jung, J.: UAS traffic management (UTM) concept of operations to safely enable low altitude flight operations. In: 16th AIAA Aviation Technology, Integration, and Operations Conference. American Institute of Aeronautics and Astronautics (2016). <https://doi.org/10.2514/6.2016-3292>
32. Primatesta, S., Rizzo, A., la Cour-Harbo, A.: Ground risk map for unmanned aircraft in urban environments. *J. Intell. Robot. Syst.* (2019). <https://doi.org/10.1007/s10846-019-01015-z>
33. Saha, M., Sanchez-Ante, G., Latombe, J.C.: Planning multi-goal tours for robot arms. In: 2003 IEEE International Conference on Robotics and Automation (Cat. No.03CH37422). IEEE (2003). <https://doi.org/10.1109/robot.2003.1242179>
34. Samosky, J.T.: SectionView—A System For Interactively Specifying and Visualizing Sections Through Three-Dimensional Medical Image Data. Ph.D. thesis, Massachusetts Institute of Technology, Department of Electrical Engineering and Computer Science (1993)
35. Sankararaman, S.: Towards a computational framework for autonomous decision-making in unmanned aerial Vehicles. In: AIAA Information Systems—AIAA Infotech @ Aerospace. American Institute of Aeronautics and Astronautics (2017). <https://doi.org/10.2514/6.2017-0446>
36. Sarne, D., Manisterski, E., Kraus, S.: Multi-goal economic search using dynamic search structures. *Auton. Agents Multi-Agent Syst.* **21**, 204–236 (2009)
37. Stevenson, J.D., O’Young, S., Rolland, L.: Estimated levels of safety for small unmanned aerial vehicles and risk mitigation strategies. *J. Unmanned Veh. Syst.* **3**(4), 205–221 (2015). <https://www.nrcresearchpress.com/doi/abs/10.1139/juvs-2014-0016>
38. T. Harmsel, A.J., Olson, I.J., Atkins, E.M.: Emergency flight planning for an energy-constrained multicopter. *J. Intell. Robot. Syst.* **85**(1), 145–165 (2017). <https://doi.org/10.1007/s10846-016-0370-z>. License: CC BY 4.0
39. Theodore, C., Rowley, D., Ansar, A., Matthies, L., Goldberg, S., Hubbard, D., Whalley, M.: Flight trials of a rotorcraft unmanned aerial vehicle landing autonomously at unprepared sites. In: Annual Forum Proceedings of American Helicopter Society, vol. 62, p. 1250 (2006)
40. USGS: LIDAR Point Cloud NY CMPG 2013 (2018). [ftp://rockyftp.cr.usgs.gov/vdelivery/Datasets/Staged/Elevation/LPC/Projects/USGS\\_Lidar\\_Point\\_Cloud\\_NY\\_CMPG\\_2013\\_LAS\\_2015/metadata](ftp://rockyftp.cr.usgs.gov/vdelivery/Datasets/Staged/Elevation/LPC/Projects/USGS_Lidar_Point_Cloud_NY_CMPG_2013_LAS_2015/metadata). Accessed 05 Sept 2018
41. USGS: Lidar Point Cloud; Washtenaw County, MI (2018). [ftp://rockyftp.cr.usgs.gov/vdelivery/Datasets/Staged/Elevation/LPC/Projects/MI\\_WashtenawCo\\_2009/laz](ftp://rockyftp.cr.usgs.gov/vdelivery/Datasets/Staged/Elevation/LPC/Projects/MI_WashtenawCo_2009/laz). Accessed 05 Sept 2018
42. Warren, M., Mejias, L., Yang, X., Arain, B., Gonzalez, F., Upcroft, B.: Enabling aircraft emergency landings using active visual site detection. In: Field and Service Robotics, pp. 167–181. Springer (2015)
43. Whyatt, J., Wade, P.: The Douglas-Peucker line simplification algorithm. *Bull. Soc. Univ. Cartogr.* **22**(1), 17–25 (1988)
44. Winnefeld, J.A., Kendall, F.: Unmanned Systems Integrated Roadmap FY 2011–2036. Office of the Secretary of Defense, US (2011)

# Chapter 21

## Reinforcement Learning: An Industrial Perspective



Amit Surana

**Abstract** In this chapter, we discuss potential opportunities and challenges associated with applications of reinforcement learning (RL) in the aerospace domain. In particular, we focus on problems related to sensor resource management, autonomous navigation, advanced manufacturing, maintenance, repair and overhaul operations, and human-machine collaboration. We present two detailed RL case studies related to sensor tasking for aerial surveillance and robot control in an additive manufacturing application which utilizes different flavors of RL including the more recent deep RL framework. Finally, we highlight some ongoing research developments which could address key challenges in deploying RL in the aerospace domain.

### 21.1 Introduction

Reinforcement learning (RL) comes into play when examples of desired behavior or labels are not available as in supervised learning but where it is possible to score examples of behavior according to some performance criterion [71]. Recent breakthroughs in deep reinforcement learning (DRL) has demonstrated scalability of RL to high-dimensional problems [56, 57, 68]. With these advances, practical RL applications in a variety of domains have been emerging, including: robotics and industrial automation [35, 41, 64, 87]; autonomy related applications such as autonomous driving [65]; energy management [29]; communication and networks [52]; resource management [53]; health care [30]; transportation [23]; education [50]; business management [13]; search, recommendation, and online advertising [86]; interactive perception [11]; and text, speech, and dialog systems [45], see [47] for further details.

---

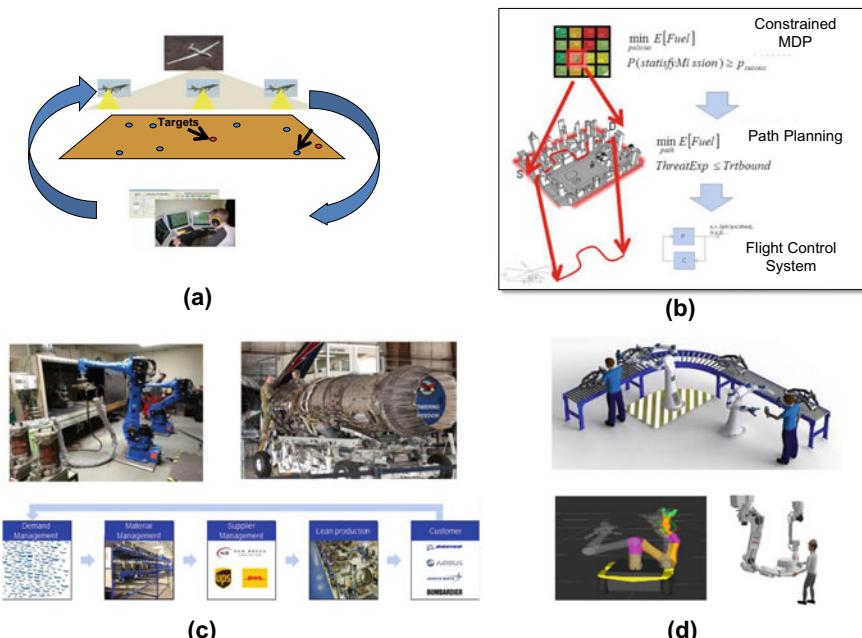
A. Surana (✉)

Raytheon Technologies Research Center, 411 Silver Lane, East Hartford, CT, USA  
e-mail: [amit.surana@rtx.com](mailto:amit.surana@rtx.com)

RL applications in context of aerospace industry have however been limited due to high assurance mission critical requirements. In this chapter we discuss several problems in aerospace applications where RL could be potentially relevant, and the challenges involved. Such applications include sensor management for aerial intelligence, surveillance, and reconnaissance; high-level reasoning in autonomous navigation; control of industrial robots in advanced manufacturing; maintenance, repair, and overhaul operations, and supply chain management. We also present two detailed RL case studies related to sensor tasking for aerial surveillance and robot control in an advanced manufacturing process. These different applications build on a range of RL techniques including classical approximate dynamic programming, constrained Markov Decision Processes, and more recent DRL framework. Finally, we highlight some ongoing research directions which could potentially address the key challenges in deploying RL in the aerospace domain.

## 21.2 RL Applications

In this section we provide an overview of some potential applications of RL in aerospace domain, see Fig. 21.1.



**Fig. 21.1** Use Cases: **a** Sensor Management in ISR, **b** Hierarchical Path Planning, **c** Advanced Manufacturing, and Maintenance, Repair and Overhaul Operations, and **d** Human Robot Collaboration

### 21.2.1 Sensor Management in Intelligence, Surveillance, and Reconnaissance

Recently, there has been an increasing need for automating aerial surveillance using unmanned air vehicles (UAVs) for civilian, homeland security, and military applications [18, 32, 59]. UAVs are equipped with a variety of agile sensors such as electro-optical/infrared cameras, synthetic aperture radar, and motion target indicators, allowing fast redirection and fast switching between different modes. This provides a large number of control options to maintain situational awareness in intelligence, surveillance, and reconnaissance (ISR) mission scenarios, see Fig. 21.1a. Currently, multiple human operators manage these sensor suites which limits their effective utilization in real time especially given a variety of constraints such as sensors may have a limited field of view, there may be multiple targets of interest which need to be viewed, deadlines on gathering timely information, and exposure to threat environments. Sensor control/tasking algorithms are required that maximize the value of information obtained while complying with such constraints and thus aide human operators who can focus on high-level mission objectives. This could further help reduce number of sensor assets and human operators required to maintain situational awareness in ISR missions.

The sensor tasking problems can be posed as a stochastic control problem in which sensor actions are the controls, and the controlled state is the information state capturing uncertainty in the environment, and thus particularly well suited for RL application [33, 37, 81]. We present a partially observed Markov decision process (POMDP) based formulation of sensor tasking in Sect. 21.3, and use a multi-arm bandit problem approximation to solve the problem in a multi-target tracking scenario.

### 21.2.2 High Level Reasoning in Autonomous Navigation

Complex missions for optionally piloted aircrafts such as cargo re-supply and medical evacuation in a cluttered, urban environment are subject to several objectives and constraints. For such applications a mission management system must (1) accept rich mission specifications which capture spatial, temporal, and logical constraints (e.g., expressed as temporal logic formulas) during mission execution, (2) deal with uncertainties in performing actions and their outcomes, (3) handle large partially known cluttered environments, (4) avoid obstacles and respect vehicle dynamic constraints, and (5) tradeoff different objectives such as probability of mission success, mission completion time, threat/weather exposure, fuel consumption, and network connectivity. Moreover, mission contexts are highly dynamic, requiring the mission management system to react and adapt to new situations and contingencies. These requirements lead to an extremely computationally challenging and intractable planning problem.

One way to deal with the complexity is to decompose the problem across different layers, leveraging the differences in spatial and temporal scales of the mission objectives. For instance, authors in [19, 20] propose a planner which is comprised of an upper layer, responsible for the mission level management and human interaction, a middle layer that handles the motion planning, and a bottom layer that generates feasible control actions in form of trajectories or waypoints, see Fig. 21.1b. While different planning approaches can be used in these layers, the high-level mission planning is particularly well suited for formulation in terms of a constrained Markov decision processes (CMDP) [4, 21]. In such a formulation the environment is partitioned into regions, and the high-level planning is abstracted into selecting actions corresponding to the motion primitives to follow while moving between the regions. Given the mission specifications as a temporal logic formula (e.g., via syntactically co-safe LTL), expected/total cost constraints such as threat exposure, and critical resource constraints such as fuel consumption formulated via sample path chance constraints, the CMDP planner determines a policy to maximize mission success probability while satisfying these constraints. Such a policy is obtained by solving a linear program using a dual formulation based on occupation measure [4]. The high-level actions selected by CMDP are then delegated to a low level path planner such as probabilistic road map (PRM) that determines the waypoints while accounting for the obstacles. Other objectives such as threat exposure can also be incorporated at this level via multi-objective formulation [20]. In order to effectively implement such a framework, a flexible, re-usable (i.e., independent of specific application domain), and run-time adaptable knowledge-based architecture has also been proposed in [20]. The numerical studies show that the proposed approach leads to scalable path planning in large urban domains under complex mission specifications, and also enables more efficient contingency management [19, 20].

The CMDP approach for mission planning discussed above is generic and can be potentially applied in many planning domains. However, CMDP is a model-based approach requiring problem dependent expert-driven abstraction. A more systematic approach needs to be developed which can automatically identify appropriate levels of hierarchy and mission objective/constraints assignment at each level. Hierarchical model-free RL approaches [7, 58, 78] combined with framework to incorporate constraints [66] provide promising directions to address this challenge.

### ***21.2.3 Advanced Manufacturing Process Control***

Additive manufacturing (AM) is an advanced manufacturing method of part production wherein a component is built from a layer-by-layer deposition of material. Metallic AM processes such as cold spray and powder bed fusion (PBF) are particularly relevant in aerospace industry. AM techniques such as PBF excel in their ability to create complex internal geometries previously unfeasible for manufacturing with conventional techniques. Other advantages include reduced material waste, part consolidation, and the ability to produce parts without the need for expensive

part-specific tooling. Despite advantages of AM process over conventional manufacturing, lack of part quality control and process repeatability continue to hamper achieving its full potential.

Achieving high levels of quality and repeatability of metallic AM parts is an extremely challenging task due to several factors, such as the high complexity of the underlying physical phenomena and transformations that take place during part production, and the lack of formal mathematical and statistical models needed to control the build process and ensure part quality. Extensive experimentation or round-robin testing are currently employed to determine the optimal process parameter settings that achieve the desired properties for a specific material and additive process. Failure Mode and Effect Analysis tools are employed to a priori ensure all aspects within the manufacturing space are controlled. Significant effort is spent in post build part inspection and characterization. This qualification process can be costly and time consuming and still may not ensure desired repeatability due to various uncertainties and unmodeled disturbances/physics during the process.

To overcome these challenges and ensure a reliable, repeatable, and adaptable AM process, recently emphasis has been placed on in-situ process monitoring and control [72]. The capability of in-situ process monitoring and in-situ metrology for AM technologies has been subject to significant ongoing research activity [24], which makes the concept of feedforward/feedback control in AM viable. Hierarchical control architecture is envisioned in which different process variables (e.g., laser power, scan speed, hatch spacing, the laser-scan paths, etc. in PBF) are manipulated in real time at different time scales based on feedback from measured sensor data. For instance, during layer buildup, laser power, and scan speed can be adapted to create a homogeneous temperature field, which results in better microstructure and mechanical properties. Laser-scan paths/hatch spacing could be changed from layer-to-layer to accommodate for process anomalies. On the other hand for distortion compensation one needs to account for the impact of process variable changes on build up over possibly several layers. Since changes in process variables at current layer affect build up at future layers, and previous layers could get affected during build up of future layers, these control problems have to be formulated over an appropriate time horizon. Moreover, there could be constraints related to safe operating zone or constraints on process variables to prevent defect formation, and bounds on laser power/scan speed. Thus, overall this leads to a high-dimensional constrained optimal control problem over a finite horizon.

Optimal control methods such as repetitive process control (RPC) [80] and iterative learning control (ILC) [17] type techniques are particularly well suited as they can exploit repetitive trials prevalent in manufacturing processes to learn optimal control. Model predictive control (MPC) [3, 26] is another general purpose approach which could be employed to solve the constrained optimal control problem in a receding horizon fashion. However, when dealing with high speed/high-dimensional sensor data as would be commonplace in AM, estimation and real-time optimization could become a bottleneck. Deep reinforcement learning (DRL) methodology provides a powerful alternative of an end-to-end system that can learn a policy offline to map high-dimensional sensor data to control actions eliminating the need for expensive

state estimation and real-time optimization [22, 41, 49, 56, 57, 68]. We present a case study related to application of DRL to cold spray control problem in Sect. 21.4.

### 21.2.4 Maintenance, Repair, and Overhaul Operations

Maintenance, repair, and overhaul (MRO) in the aerospace industry is a complex process that has strict and precise requirements defined by airworthiness authorities to guarantee the safety of passengers and aircrew. MRO operations involve activities such as part inspection, scheduling maintenance and repair actions, and inventory management for spare parts, see Fig. 21.1c.

Manual inspection and non-destructive evaluation techniques are employed for assessing part quality. With the recent advances in computer vision/machine learning there has been a push in automating the inspection process. However, to achieve human level performance a framework such as interactive perception will be required, such that the automated system can reason about what to observe (e.g., look at the part from different viewpoints), and acquire more relevant information during inspection (without needing labeled data) for reducing the uncertainty about the decision (e.g., presence of defects or part wear). Interactive perception can be addressed via DRL formulation [11].

Additionally, a variety of other decision-making under uncertainty type problems arise in MRO operations which can be addressed by RL, e.g., (1) how and when should parts be tested/repaired/replaced to optimize expected future value of maintenance contracts ? (2) how should one adapt to shortfalls and varying degrees of supplier reliability ? and (3) when should one notify downstream customer of supplier under performance? The key challenge in such problems is the high dimensionality, and recent work in DRL and multi-agent DRL could be particularly relevant.

### 21.2.5 Human–Robot Collaboration

The different application scenarios discussed above would require RL agents to interact and collaborate with humans at different levels. For example, for human and robots working together on a manufacturing shop floor (see Fig. 21.1d), RL agents would need to reason about and predict human intent, and guarantee safe operation. The human intent can be represented as a belief state in a POMDP, and the agent could learn a policy conditioned on the belief state to collaborate with the human to jointly achieve the task objective [61]. Learning from human demonstrations through inverse reinforcement learning (IRL) [1, 6, 25, 88], and combining model-free approaches with model-based techniques (e.g., incorporating human expert knowledge) is another promising direction to promote human-like behavior in RL agents. The use of virtual/augmented reality environment could particularly be a cost effective way to generate realistic data from human demonstrations [69].

In autonomy applications such as involving UAVs, humans operators will play a more supervisory role focusing on high-level mission management. Human–machine trust is a key challenge which would require an interdisciplinary research incorporating cognitive sciences/human factors aspects into RL. Ultimately, the RL agents would need to learn and think like people, and support human-like explanation and understanding [39].

## 21.3 Case Study I: Optimal Sensor Tasking

In this section we present an approximate dynamic programming approach for solving a sensor tasking problem in a surveillance scenario. By sensor tasking we refer to the problem of sequentially allocating sensor resources so that uncertainty in the states of dynamically evolving targets is minimized. Sensor resource refers to any sensor asset such as a UAV equipped with a pan-zoom-tilt (PZT) camera. In particular we use Multi-Arm Bandit-based problem approximation which leads to priority rules for sensor allocation based on Gittin’s index. We apply the approach in a simulated multi-target tracking scenario where the number of sensors is significantly less than the number of targets to be tracked, and numerically study the impact of number of sensors and sensing capabilities (e.g., sensor slewing rate) on the tracking performance.

### 21.3.1 Sensor Tasking as a Stochastic Optimal Control Problem

The sensor tasking problem can be formulated as a partially observed Markov decision process (see section [33, 37]) in which sensor actions are the controls, and the controlled state denoted by  $\Phi_t = ((\zeta_t^1)', \dots, (\zeta_t^N)')'$  (where  $'$  denotes vector/matrix transpose) is a joint information state which is composed of information states  $\zeta_t^i, i = 1, \dots, N$  associated with each target. It is a standard procedure to introduce information state to convert the POMDP to a fully observed MDP, see section [10, 33] for details. In multi-target tracking application, a natural information state is the joint multi-target probability density (JMPD)  $P(\mathbf{s}_t | \mathbf{Y}_t)$  which captures the uncertainty about the multi-target state  $\mathbf{s}_t = ((\mathbf{x}_t^1)', \dots, (\mathbf{x}_t^N)')'$ , conditioned on all the previous measurements  $\mathbf{Y}_t = \{\mathbf{y}_t, \dots, \mathbf{y}_0\}$ . In practise conditional probability  $P(\mathbf{s}_t | \mathbf{Y}_t)$  is approximated in terms of particles (as in particle filters) or by mean and covariance (as in Kalman filter). Hence, the information state for each target can be presented as a finite-dimensional vector  $\zeta \in \mathbb{I} \subset \mathbb{R}^M$  (for some appropriate  $M$ , e.g., under Gaussian approximation,  $\zeta = (\mu_x, \mu_y, P_{xx}, P_{xy}, P_{yy})' \in \mathbb{R}^5$ , where  $\mu$  denotes the mean and  $\mathbf{P} = \begin{pmatrix} P_{xx} & P_{xy} \\ P_{yx} & P_{yy} \end{pmatrix}$  denotes the covariance matrix representing the uncertainty in target position  $\mathbf{x} = (x, y)'$ ).

Generically, we will represent the evolution of the joint information state by a stochastic equation,

$$\Phi_{t+1} = \mathbf{f}(\Phi_t, \mathbf{w}_t, a_t), \quad (21.1)$$

where,  $\mathbf{f}$  is the filtering (particle/Kalman) operation,  $\mathbf{w}_t$  is random vector representing how uncertainty in target state actually changes as well as uncertainties in the sensor measurement of the target state with probability density  $p(\mathbf{w}_t)$ , and  $a_t$  is the control action taking values in set  $\mathbb{U}$ . For example, control action can be sensor allocation to the targets, i.e.,  $\mathbb{U} = \{1, \dots, N\}$ . We restrict to a single sensor, extension to incorporate multiple sensors is discussed in Sect. 21.3.2.2. A policy  $\pi$  is the mapping  $\pi : \mathbb{I} \rightarrow \mathbb{U}$  from information space  $\mathbb{I}$  to action space  $\mathbb{U}$ . The sensor tasking problem is to determine the control policy  $\pi$  which maximizes the expected value of the total discounted reward,

$$\max_{\pi \in \Pi} J(\pi) \quad (21.2)$$

where,

$$J(\pi) = \mathbb{E}_{\pi, p_0} \left[ \sum_{t=0}^{\infty} \alpha^t r(\Phi_t, a_t) \right], \quad (21.3)$$

$\alpha$  is the discount factor,  $r$  is the reward,  $p_0$  is the probability density on the initial information state  $\Phi_0$ . The optimization is taken over a set of stationary control policies  $\Pi$ . We will assume that the  $r$  can be decomposed as

$$r(\Phi, a) = \sum_{i=1}^N r^i(\zeta^i, a), \quad (21.4)$$

where  $r^i(\zeta^i, a)$  is the reward associated with the  $i$ -th target. The optimal value function  $J(\pi^*)$  (for optimal policy  $\pi^*$ ) satisfies the Bellman equation which can be solved via dynamic programming (DP) [9, 10]. Commonly used DP approaches include value iteration and policy iteration. Due to the curse of dimensionality associated with the Bellman equation, such direct techniques are seldom useful in applications. Over several decades various problem approximation and approximate dynamic programming approaches have been developed [14, 44], which we explore next.

### 21.3.2 Multi-Arm Bandit Problem Approximation

For resource allocation problems, the multi-arm bandit (MAB) (see [28, 33, 38]) based problem approximation is particularly attractive as it admits a solution via forward induction leading to natural decoupling of the Bellman equation. Forward induction is computationally less intensive than backward induction; an optimal forward induction policy can be obtained by solving  $N$  one-dimensional problems by

computing the Gittin's indices of each bandit process instead of one  $N$ -dimensional problem. In order for the classical MAB solution to be applicable to the sensor tasking problem considered above with one sensor and multiple targets, the following assumptions should be satisfied:

- I. Targets are independent and information state of targets that are not tracked remain frozen. This assumption holds, for example, when the targets are stationary. If information state of the targets evolves slowly compared to the sensor dynamics, then this could also be a good approximation. With this assumption, the information state of each target evolves independently of each other, as [81]

$$\zeta_{t+1}^i = \begin{cases} \mathbf{f}^i(\zeta_t^i, \mathbf{w}_t^i), & a_t = i \\ \zeta_t^i & a_t \neq i \end{cases}, \quad (21.5)$$

where  $\mathbf{f}^i$  is the filtering operation (particle filtering or Kalman filter) and  $\mathbf{w}^i$  is the uncertainty corresponding to the  $i$ -th target, respectively.

- II. Targets which are not tracked do not contribute to the reward. In tracking problems the information state (e.g., the tracking error covariance) is usually not frozen, e.g., prediction error increases when the target is not observed. However, it is always possible to satisfy this assumption by transforming the reward as follows:

$$\tilde{r}^i(\zeta) = \int r^i(\mathbf{f}^i(\zeta, \mathbf{w}^i), i) p_i(\mathbf{w}^i) d\mathbf{w}^i - r^i(\zeta, i). \quad (21.6)$$

Given above assumptions, the solution of the sensor tasking problem (21.2) with a single sensor is given by a policy which is a priority rule [81],

$$a_t = \pi(\Phi_t) = \arg \max_i \nu^i(\zeta_t^i), \quad (21.7)$$

where  $\nu^i$  is the Gittin's index,

$$\nu^i(\zeta) = \inf\{M : J^i(\zeta, M) = M\}, \quad (21.8)$$

with  $J^i$  being the value function for the  $i$ -th target, which satisfies a one-dimensional Bellman equation

$$J^i(\zeta, M) = \max \left( M, \tilde{r}^i(\zeta^i) + \alpha \int J^i(\mathbf{f}^i(\zeta, \mathbf{w}^i), M) p_i(\mathbf{w}^i) d\mathbf{w}^i \right), i = 1, 2, \dots, N. \quad (21.9)$$

Since, the computation of Gitin's index can be decoupled, we drop the superscript  $i$  denoting the target index in what follows.

### 21.3.2.1 Gittin's Index in a Simplified Setting

We next consider a simplified setting, and take the information state of each target to be the error covariance matrix, i.e.,  $\zeta_t = \mathbb{E}[(\mathbf{x}_t - \hat{\mathbf{x}}_t)(\mathbf{x}_t - \hat{\mathbf{x}}_t)']$ , where  $\mathbf{x}_t$  is the target position and  $\hat{\mathbf{x}}_t$  is its estimate obtained using the Kalman filter (under the assumption of linear process model  $\mathbf{F}$  and linear measurement model  $\mathbf{H}$  with Gaussian model and sensor noise with covariance  $\mathbf{Q}$  and  $\mathbf{R}$ , respectively). Additionally, we will assume that if the sensor looks at a target, then the probability of detecting that target is  $\beta$ . Under these assumptions, the evolution of  $\zeta_t$  is determined by the Kalman measurement update equation

$$\zeta_{t+1} = \mathbf{f}(\zeta_t) = \mathbf{F}(\zeta_t^{-1} + \mathbf{H}'\mathbf{R}^{-1}\mathbf{H})^{-1}\mathbf{F}' + \mathbf{Q}, \quad (21.10)$$

if the sensor looks at the target and the target is detected, and otherwise  $\zeta_{t+1} = \zeta_t$  under the frozen assumption (I). We consider a reward function:

$$r(\zeta) = \mathcal{I}(tr(\zeta) \leq \delta), \quad (21.11)$$

where  $tr$  denotes the trace and  $\delta$  is desired upper bound on the mean square error in the target position. The transformed reward (see Eq. (21.6)) becomes

$$\tilde{r}(\zeta) = \beta\mathcal{I}(tr(\zeta) > \delta \cap tr(\mathbf{f}(\zeta)) < \delta), \quad (21.12)$$

where  $\mathcal{I}$  is an indicator function, and an exact analytical expression for Gittin's index can be computed to be [81]

$$\nu(\zeta) = \frac{\kappa^{n(\zeta)} \alpha^{-1}}{1 - \kappa^{n(\zeta)}} \mathcal{I}(n(\zeta) > 0). \quad (21.13)$$

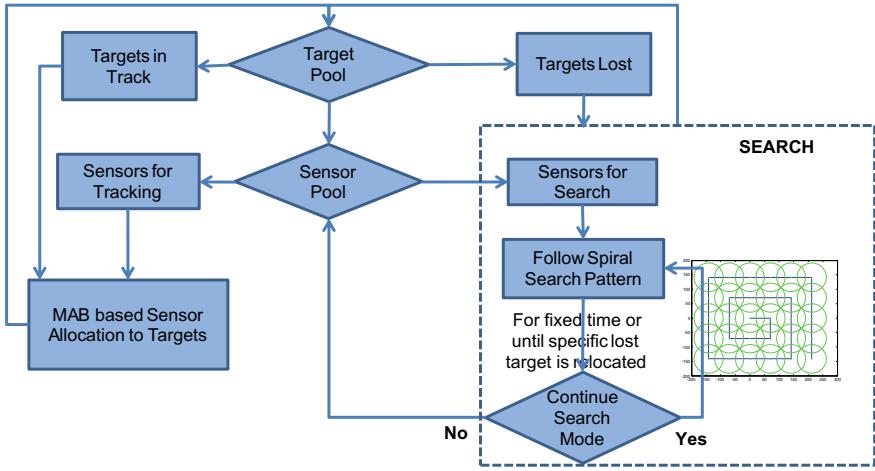
Here  $\kappa = \frac{\alpha\beta}{1-\alpha(1-\beta)}$  and the exponent satisfies

$$n(\zeta) = \min\{m \in \mathbb{Z}_+ : tr(\mathbf{f}^m(\zeta)) \leq \delta\}, \quad (21.14)$$

with  $\mathbb{Z}_+$  being the set of non-negative numbers.

### 21.3.2.2 Tracking with Multiple Sensors Combined with Search

In this section we extend the MAB-based sensor tasking algorithm discussed above to incorporate multiple sensors and a search mode. For multiple sensors the priority rule (21.7) does not remain optimal. There are several extensions described in [33] which deal with this case and other generalizations. We use a greedy heuristic in which the highest priority target (ordered according to their Gittin's index) gets first assigned to a sensor which takes minimum time to be redirected to that target from its current position. Similar process is applied to other targets until all available sensors



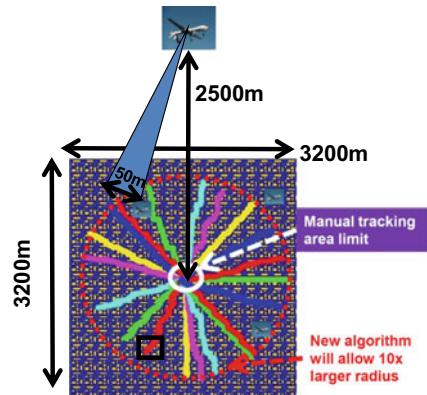
**Fig. 21.2** Scheme for combined tracking and search

have been allocated. Search mode is required in order to recover the lost targets. This situation can arise for example, when sensor to target ratio increases or the target motion model is not an accurate representation (e.g., in case of deceptive targets). Search can be implemented in several ways, e.g., the sensor (UAV or PZT camera) can follow a prescribed path such as a greedy spiral or execute a multi-spectral search [54, 55]. With PZT cameras, search can also be implemented by successively zooming out, and hence increasing the field of view. Figure 21.2 shows the overall logic for combined tracking and search with multiple sensors. The algorithm is summarized below:

1. Execute the current sensor mode, for each sensor.
2. Determine target status, i.e., whether the target is in track or is lost. A target is considered lost, if it cannot be located in neighborhood of its predicted location.
3. For sensors which have lost their target, switch to search mode. For targets which have been relocated, free the sensors allocated to search them. Also, sensors, which have reached the upper bound on search time, are considered free for being allocated for tracking. For all the sensors whose active mode is not search, and all targets, which are currently in track, apply the MAB-based resource allocation algorithm.
4. Repeat steps 1–3.

What constitutes an optimal combination of search and tracking, and how to choose optimal search pattern and upper bound on search time, are potential avenues for future investigation.

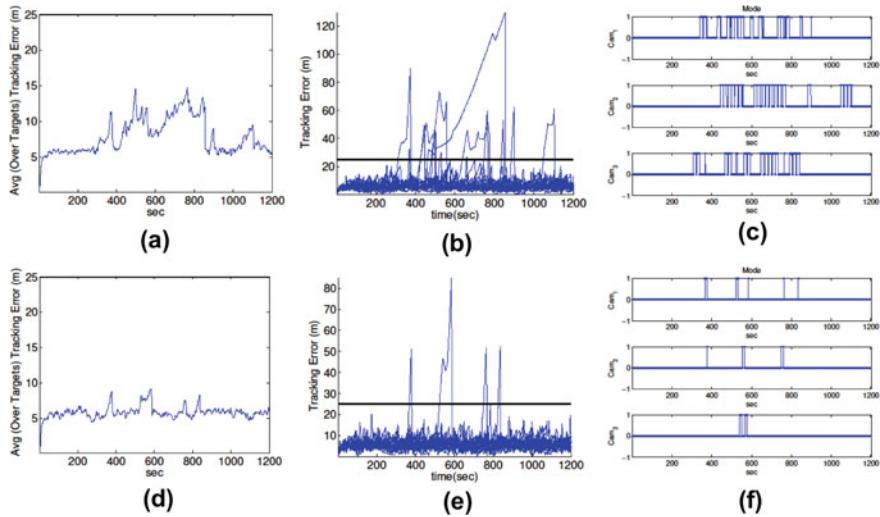
**Fig. 21.3** Target dispersal scenario in an urban domain



### 21.3.3 Numerical Study

In this section, we demonstrate the sensor tasking approach discussed above for tracking multiple targets using electro-optical (EO) PZT cameras mounted on UAVs in an urban domain. Specifically, we consider an urban domain of size 3200 square meters with 21 dispersing agents (see Fig. 21.3). We assume that the UAVs follow a fixed prescribed path at a constant altitude of 2500 m and 40 m/s cruising speed. Each UAV has a PZT camera with following characteristics: maximum slew rate (200 deg/s), field of view 1.14 deg (which corresponds to 50 m diameter at an altitude of 2.5 km), speed of 30 frames per second, and a pointing resolution of around 0.05deg (which correspond to around 1.5 m position error). It is assumed that the target acquisition and recognition processing is done onboard. The sensor tasking algorithm is implemented in a centralized fashion, and the associated communication and processing time is assumed negligible. Given this, the total switching and processing time can vary between 0.35 s to around 0.75 s, which we refer to as the effective slewing rate. The goal is to allocate the PZT cameras such that all the 21 agents remain in track for as much time as possible. This will allow human operators to gather enough clues about each individual target and prioritize them for further action.

For this scenario we used the combined search and tracking algorithm described in the Sect. 21.3.2.2. The target motion model is assumed to be double integrator with acceleration noise in free space and wall following behavior near the buildings. The urban terrain map is assumed to be known a priori and is used in particle filtering for projecting the target motion in the future. Figure 21.4a shows the average tracking performance for this scenario with 3 PZT cameras, while the subplot b shows the tracking error for each individual target. As can be seen, some targets are lost but are eventually recovered as the cameras switch to search mode. Note that a target is considered lost if its tracking error is greater than 25 m (corresponding to sensor's field of view of 50 m diameter as mentioned above). Compared to the baseline in which human operators have to eventually decide top three significant targets to

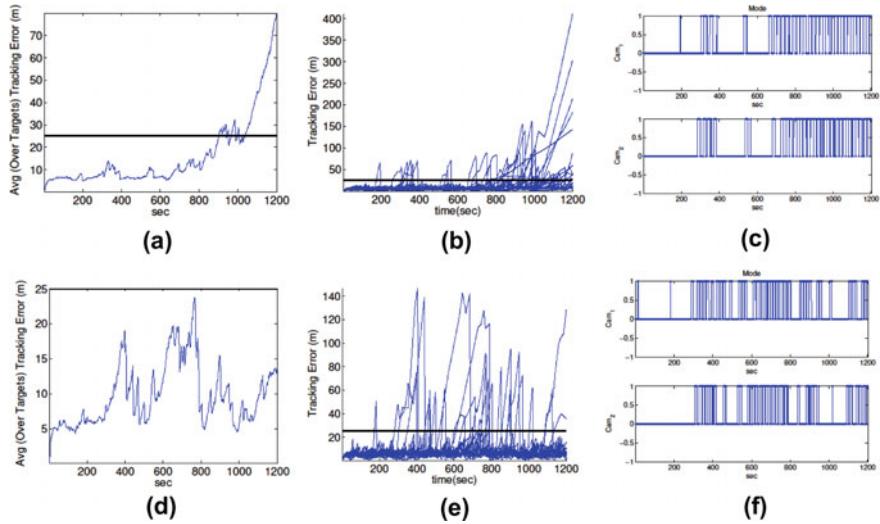


**Fig. 21.4** Target tracking performance with 3 PZT cameras with an effective slewing rate of 0.5 s: **a** Average target tracking error, **b** Individual target tracking error, **c** Camera mode. The subplots d–f show results for effective slewing rate of 0.35 s

track within 2 min, the sensor tasking algorithm allows this time to be extended to 20 min, a 10X fold increase in the performance. The frequency of switching between track and search mode is shown in Fig. 21.4c, where 0 corresponds to track mode and 1 corresponds to search mode, respectively. This switching frequency increases considerably as shown in Fig. 21.5c when using only 2 PZT cameras. Moreover, the tracking performance deteriorates and some targets are lost which are never recovered, see Fig. 21.5a, b. In these experiments, the effective slewing rate for each camera was assumed to be 0.5 s. Figures 21.4d–f and 21.5d–f show similar results but with a faster effective slewing rate of 0.35 s. Note that the overall tracking performance improves, and all the 21 targets remain in track even with 2 PZT cameras. Thus, the sensor tasking algorithm not only allows the targets to remain in track for a longer time, but with fast enough sensors, one can also reduce the number of sensors (and therefore human operators which manage the path of the UAVs carrying the PZT cameras) to achieve similar performance.

## 21.4 Case Study II: Deep Reinforcement Learning for Advanced Manufacturing Control

In this case study we explore the application of guided policy search (GPS) a deep RL framework for feedback control of the robot motion during cold spray manufacturing process. The results are compared to the benchmarks such as iterative linear quadratic



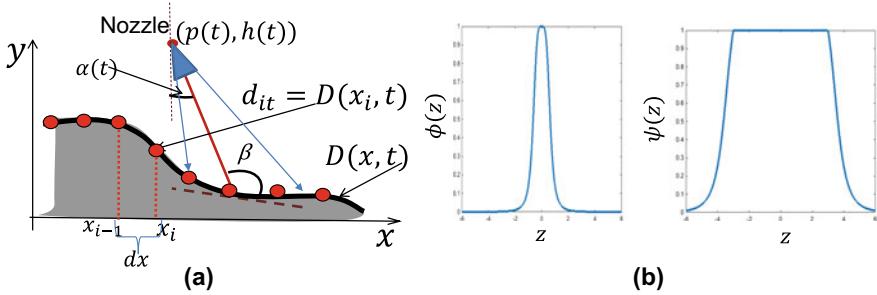
**Fig. 21.5** Target tracking performance with 2 PZT cameras with an effective slewing rate of 0.5 s: **a** Average target tracking error, **b** Individual target tracking error, **c** Camera mode. The subplots d–f show results for effective slewing rate of 0.35 s

regular (iLQR) and model predictive control (MPC), and the generalizability of the framework is also evaluated.

#### 21.4.1 Cold Spray Control Problem

Cold spray (CS) is an additive manufacturing process in which powder particles (typically 10 to 50 micron) are accelerated to very high velocities (200 to 1000 m/s) by a supersonic compressed gas jet at temperatures well below their melting point and then deposited on the surface to build new parts or repair structural damages such as cracks and unwanted indentation [84]. CS is a multi-scale process with complex physics [79], making it challenging to optimize the process parameters which leads to desired thermal/mechanical properties. Additionally, the nozzle motion relative to part/defect geometry needs to be programmed to achieve near net surface deposition. Currently, this is accomplished by manual trial and error, and a conservative approach is taken leading to excess material deposition which is eventually machined to meet the required geometric tolerances. Moreover, the nozzle motion once programmed remains fixed and is not adapted to account for any variations/disturbances during the operation. Such an approach to overbuild could lead to a significant loss of expensive material.

The focus of our work is to develop a framework for feedback control of nozzle motion that minimizes the material wastage while achieving near net surface deposition. While detailed CS models are highly complex, for our control development purposes it suffices to use a reduced order model described in [16] which captures



**Fig. 21.6** Cold spray model

the impact of nozzle motion/spray behavior on the dynamics of deposition process at a macroscopic scale. Let  $p(t)$ ,  $h(t)$  and  $\alpha(t)$  be the spray nozzle's position and orientation as a function of time, respectively, see Fig. 21.6a. Then the surface height profile  $D(x, t)$  ( $x \in \mathbb{R}$ ) evolution due to particle deposition is governed by a nonlinear integro-differential equation,

$$\frac{\partial D(x, t)}{\partial t} = \int_0^T \phi(\tan \beta(t) - \tan \alpha(t)) \psi \left( \frac{\tan \beta(t) - \frac{\partial}{\partial x} D(x, t)}{1 + \tan \beta(t) \frac{\partial}{\partial x} D(x, t)} \right) dt, \quad (21.15)$$

where  $\tan \beta(t) = \frac{x-p(t)}{h(t)-D(x,t)}$  and the functions

$$\phi(z) = 1 - \left( 1 + \frac{\rho}{z^2} \right)^{-1}, \quad \psi(z) = \frac{1}{c} \left( 0.5 + \frac{\text{atan}(-a \max(|z/b|^\kappa, 1))}{\pi} \right),$$

capture the distribution of particles in the spray cone and the nozzle efficiency, respectively (see Fig. 21.6b). We will denote the model parameter vector as  $\mathbf{p} = (\rho, a, b, c, \kappa)'$ . In above we have restricted to a one-dimensional deposition model, as often in applications the part is rotated while the nozzle motion is confined to be along the axis of rotation.

The space–time discretization of Eq. (21.15) leads to a discrete-time nonlinear input–output system

$$\mathbf{s}_{t+1} = \mathbf{f}(\mathbf{s}_t, \mathbf{u}_t; \mathbf{p}), \quad (21.16)$$

where  $\mathbf{s}_t = (\mathbf{d}'_t, p_t, h_t, \alpha_t)'$  is the system state,  $\mathbf{u}_t = (v_t^x, v_t^h, \omega_t)'$  are the control inputs, and

$$\mathbf{f}(\mathbf{s}_t, \mathbf{u}_t, \mathbf{p}) = \begin{pmatrix} d_{1t} + g_1(\mathbf{d}_t, p_t, h_t, \alpha_t; \mathbf{p})dt \\ \vdots \\ d_{N_d t} + g_{N_d}(\mathbf{d}_t, p_t, h_t, \alpha_t; \mathbf{p})dt \\ p_t + v_t^x dt \\ h_t + v_t^h dt \\ \alpha_t + \omega_t dt \end{pmatrix}. \quad (21.17)$$

Here,  $\mathbf{d}_t = (d_{1t} \cdots d_{N_d t})'$  is the vector of discretized surface height profile  $D(x, t)$ , with  $d_{it}$  being the surface heights at the spatial locations  $x_i = x_0 + (i - 1)dx$ ,  $i = 1, \dots, N_d$  and  $N_d$  is the number of discretized cells of size  $dx$ , see Fig. 21.6a. We assume a kinematic model of the nozzle motion, so that the nozzle position/orientation can be controlled by changing its linear  $v_t^x$ ,  $v_t^h$  and angular  $\omega_t$  speeds, respectively. The functions  $g_i$  represent the discretization of the integral term appearing in the Eq. (21.15),

$$g_i(\mathbf{d}_t, p_t, h_t, \alpha_t; \mathbf{p}) = \phi(\tan \theta_{it} - \tan \alpha_t) \psi(\cot \beta_{it}), \quad (21.18)$$

where

$$\tan \theta_{it} = \frac{x_i - p_t}{h_t - d_{it}}, \quad \cot \beta_{it} = \frac{(x_i - p_t) - (h_t - d_{it})S_{it}}{(h_t - d_{it}) + (x_i - p_t)S_{it}}$$

with  $S_{it}$  being the approximation of surface slope at the  $i$ -th grid location.

Given initial condition  $\mathbf{s}_0 = (\mathbf{d}'_0, p_0, h_0, \alpha_0)'$ , the objective is to determine control sequence  $\mathbf{u}_t = (v_t^x, v_t^h, \omega_t)'$ ,  $t = 1 \cdots, T$  such that the final material deposition profile matches a desired profile  $\mathbf{d}_f$ ,

$$\min_{\mathbf{u}_1, \dots, \mathbf{u}_T} \sum_{t=1}^{T-1} c(\mathbf{s}_t, \mathbf{u}_t) + c_e(\mathbf{s}_T), \quad (21.19)$$

where  $c_e(\mathbf{s}_T) = ||\mathbf{d}_T - \mathbf{d}_f||^2$  is the terminal cost, and  $c(\mathbf{s}_t, \mathbf{u}_t) = w_1(v_t^x)^2 + w_2(v_t^h)^2 + w_3\omega_t^2$  is running cost which imposes penalties on nozzle's speed and angular rate. Additionally, bounds can be imposed on velocity and angular rates:  $v_{\min}^x \leq v_t^x \leq v_{\max}^x$ ,  $v_{\min}^h \leq v_t^h \leq v_{\max}^h$  and  $\omega_{\min} \leq \omega_t \leq \omega_{\max}$ , respectively. This nonlinear optimization problem can be solved using model predictive control (MPC), see [16]. However, since  $N_d \gg 1$  to resolve the surface profile sufficiently, such an approach can be computationally demanding for real-time deployment. We explore reinforcement learning, in particular direct policy search methods as discussed next.

### 21.4.2 Guided Policy Search

Policy search-based RL methods are often employed in high-dimensional control applications, since they scale gracefully with dimensionality and offer appealing convergence guarantees [36, 62]. However, it is often necessary to carefully choose a specialized policy class to learn the policy in a reasonable number of iterations without falling into poor local optimal. Most practical applications rely on policy classes that are either hand-engineered and/or domain-specific, neither of which seems adequate for learning the sorts of rich motion repertoires that might be needed, for example, for a robot arm that must execute a range of motion to account for different part geometries in advanced manufacturing application. In order to scale up

optimal control and reinforcement learning to a wide range of tasks, general purpose policies are needed that can represent rich set of behaviors without extensive hand engineering. One way to accomplish this is by using more expressive and flexible policy classes such as large neural networks (NN).

Guided policy search (GPS) [40–43] is one such approach which uses a NN to parameterize the policy and transforms the policy search into a supervised learning problem, where the training set (which “guides” the policy search to regions of high reward) is generated by simple trajectory-centric algorithms. However, naive supervised learning will often fail to produce a good policy, since a small mistake on the part of the policy will put it in states that are not part of the training, causing compounding errors. To avoid this problem, the training data must come from the policy’s own state distribution. Thus, an iterative procedure is required which adapts the trajectories to the policy, alternating between optimizing the policy to match the trajectories, and optimizing the trajectories to minimize cost and match the policy, such that at convergence, they have the same state distribution. One might wonder why the policy obtained from trajectory optimization is not itself a suitable controller. The issue is that such a policy is time varying and only valid around a single trajectory, while the final NN policy is learned from many trajectory optimization solutions in many situations. GPS can thus be viewed as transforming a collection of locally optimal trajectories into a global controller. This controller can adhere to any parameterization, reflecting constraints on computation, or available sensors in partially observed domains. In our application, we will use the end-to-end formulation of GPS [41], which we briefly review.

Consider a classical finite horizon stochastic optimal control problem

$$\min_{\pi_\theta} \mathbb{E}_{\pi_\theta} \left[ \sum_{t=1}^T c(\mathbf{s}_t, \mathbf{u}_t) \right], \quad (21.20)$$

where  $c(\mathbf{s}, \mathbf{u})$  is the cost function, and the expectation  $\mathbb{E}$  is taken under the policy  $\pi_\theta(\mathbf{u}_t | \mathbf{s}_t)$  which is parameterized by  $\theta$ . Let  $q(\mathbf{u}_t | \mathbf{s}_t)$  be a guiding policy, then the problem (21.20) can be reformulated as an equivalent problem:

$$\min_{q, \pi_\theta} \mathbb{E}_q \left[ \sum_{t=1}^T c(\mathbf{s}_t, \mathbf{u}_t) \right], \quad q(\mathbf{u}_t | \mathbf{s}_t) = \pi_\theta(\mathbf{u}_t | \mathbf{s}_t), \quad \forall t, \mathbf{s}_t, \mathbf{u}_t, \quad (21.21)$$

which has an infinite number of constraints. To make the problem tractable one can use the moment matching approach, e.g.,

$$\min_{q, \pi_\theta} \mathbb{E}_q \left[ \sum_{t=1}^T c(\mathbf{s}_t, \mathbf{u}_t) \right], \quad \mathbb{E}_{q(\mathbf{u}_t | \mathbf{s}_t)q(\mathbf{s}_t)} [\mathbf{u}_t] = E_{\pi_\theta(\mathbf{u}_t | \mathbf{s}_t)q(\mathbf{s}_t)} [\mathbf{u}_t], \quad \forall t. \quad (21.22)$$

This constrained problem can be solved by a dual descent method, e.g., Bregman alternative direction method of multipliers (BADMM) Lagrangian formulation leads to

$$\mathcal{L}(q, \pi_\theta) = \sum_{t=1}^T \left[ \mathbb{E}_{q(\mathbf{s}_t, \mathbf{u}_t)}(c(\mathbf{s}_t, \mathbf{u}_t)) + \lambda'_{\mu t} (\mathbb{E}_{\pi_\theta(\mathbf{u}_t | \mathbf{s}_t) q(\mathbf{s}_t)}[\mathbf{u}_t] - \mathbb{E}_{q(\mathbf{s}_t, \mathbf{u}_t)}[\mathbf{u}_t]) + \nu_t \phi_t(q, \theta) \right],$$

where  $\phi_t$  is the Bregmann divergence term. We will take  $\phi_t$  as the KL divergence  $D_{KL}(\pi_\theta | q)$  between the distributions  $q$  and  $\pi_\theta$ . This leads to following iterative optimization scheme,

$$q \leftarrow \arg \min_q \sum_{t=1}^T \left[ \mathbb{E}_{q(\mathbf{s}_t, \mathbf{u}_t)}[c(\mathbf{s}_t, \mathbf{u}_t) - \lambda'_{\mu t} \mathbf{u}_t] + \nu_t \phi_t(q, \theta) \right], \quad (21.23)$$

$$\theta \leftarrow \arg \min_\theta \sum_{t=1}^T \left[ \mathbb{E}_{\pi_\theta(\mathbf{u}_t | \mathbf{s}_t) q(\mathbf{s}_t)}[\lambda'_{\mu t} \mathbf{u}_t] + \nu_t \phi_t(\theta, q) \right], \quad (21.24)$$

$$\lambda'_{\mu t} \leftarrow \lambda'_{\mu t} + \alpha \nu_t (E_{\pi_\theta(\mathbf{u}_t | \mathbf{s}_t) q(\mathbf{s}_t)}[\mathbf{u}_t] - \mathbb{E}_{q(\mathbf{s}_t, \mathbf{u}_t)}[\mathbf{u}_t]), \quad t = 1, \dots, T. \quad (21.25)$$

Note that for each  $t = 1, \dots, T$ ,  $\lambda'_{\mu t}$  is a vector of Lagrangian multiplier with same dimension as the control  $\mathbf{u}_t$ , and  $'$  denotes matrix/vector transpose.  $\alpha$  can be chosen in range  $[0, 1]$ , lower values lead to better numerical stability. The weights  $\nu_t$  are initialized to low values such as 0.01 and incremented based on a schedule which adjusts the KL-divergence penalty to keep the policy and trajectory in agreement by roughly the same amount at all time steps. The above optimization steps can be greatly simplified under Gaussian assumptions:

1.  $\pi_\theta(\mathbf{u}_t | \mathbf{s}_t) \sim \mathcal{N}(\mathbf{u}_t; \boldsymbol{\mu}^\pi(\mathbf{s}_t; \mathbf{w}), \Sigma^\pi)$ , where the policy parameters are  $\theta = (\mathbf{w}, \Sigma^\pi)$ . We will use a neural network (NN) with weights  $\mathbf{w}$  to represent the mean policy  $\boldsymbol{\mu}^\pi(\mathbf{s}_t; \mathbf{w})$ , while  $\Sigma^\pi$  is policy covariance which is assumed to be state independent, though more general formulations are possible [41].
2.  $q = \sum_{i=1}^M q_i$  is mixture of Gaussians with  $q_i(\mathbf{u}_t | \mathbf{s}_t) \sim \mathcal{N}(\mathbf{u}_t; \boldsymbol{\mu}_i^{q_i}(\mathbf{s}_t), \Sigma_t^{q_i})$ ,

where  $\mathcal{N}(\cdot, \mu, \Sigma)$  is a multinomial normal distribution with mean vector  $\mu$  and covariance matrix  $\Sigma$ . With this assumption, the overall GPS algorithm is summarized below. Steps 1–4 are repeated for preselected  $K$  iterations or else if a prescribed convergence criterion is met.

Step 1: Solve (21.23) which under the Gaussian assumption simplifies to

$$\arg \min_{q_i} \sum_{t=1}^T \mathbb{E}_{q_i(\mathbf{s}_t, \mathbf{u}_t)} [c_i(\mathbf{s}_t, \mathbf{u}_t) - \mathcal{H}(q_i(\mathbf{u}_t | \mathbf{s}_t))],$$

where  $c_i(\mathbf{s}_t, \mathbf{u}_t) = \frac{c(\mathbf{s}_t, \mathbf{u}_t)}{\nu_t} - \frac{1}{\nu_t} \mathbf{u}_t' \lambda_{\mu t}^i - \log(\pi_\theta(\mathbf{u}_t | \mathbf{s}_t))$ ,  $\mathcal{H}$  is the standard differential entropy, and each  $q_i$  has a set of Lagrangian multipliers  $\lambda_{\mu t}^i$  associated with it. A local optimal solution of the above problem around a selected nominal trajectory  $\tau_i = \{(\bar{\mathbf{s}}_t^i, \bar{\mathbf{u}}_t^i) : t = 1, \dots, T\}$  can be obtained via a variation of iterative LQR (iLQR) [46],

leading to  $q_i(\mathbf{u}_t | \mathbf{s}_t) = \mathcal{N}(\mathbf{u}_t^i; \boldsymbol{\mu}_t^{q_i}(\mathbf{s}_t), \boldsymbol{\Sigma}_t^{q_i})$  with  $\boldsymbol{\mu}_t^{q_i}(\mathbf{s}_t) = \bar{\mathbf{u}}_t^i + \mathbf{k}_t^i + K_t^i(\mathbf{s}_t - \bar{\mathbf{s}}_t^i)$  and  $\boldsymbol{\Sigma}_t^{q_i} = (Q_{\mathbf{uu}}^i)^{-1}$ . The nominal trajectory can be constructed in a variety of ways, e.g., from demonstrations, using randomized control inputs, or via solution obtained using MPC [85].

Step 2: Generate sample trajectories  $\tau^{ji} = \{(\mathbf{s}_t^{ij}, \mathbf{u}_t^{ij}) : t = 1 : T\}, j = 1, \dots, N$  from the distribution

$$q_i(\{\mathbf{s}_1, \mathbf{u}_1, \dots, \mathbf{s}_T, \mathbf{u}_T\}) = p_0^i(\mathbf{s}_1) \prod_{t=1}^T q_i(\mathbf{u}_t | \mathbf{s}_t) p(\mathbf{s}_{t+1} | \mathbf{s}_t, \mathbf{u}_t), \quad (21.26)$$

induced by each of the guiding distributions  $q_i, i = 1, \dots, M$ . Here,  $p(\mathbf{s}_{t+1} | \mathbf{s}_t, \mathbf{u}_t)$  is the state transition probability for the system dynamics, and  $p_0^i(\mathbf{s}_1), i = 1, \dots, M$  is the initial state distribution.

Step 3: Train the NN representing the policy mean  $\boldsymbol{\mu}^\pi(\mathbf{s}; \mathbf{w})$  with a modified objective (21.24) which under Gaussian assumption becomes

$$\arg \min_{\mathbf{w}} \frac{1}{2N} \sum_{t=1}^T \sum_{i=1}^M \sum_{j=1}^N \left[ \nu_t(\boldsymbol{\mu}_t^{q_i}(\mathbf{s}_t^{ij}) - \boldsymbol{\mu}^\pi(\mathbf{s}_t^{ij}; \mathbf{w}))' (\boldsymbol{\Sigma}_t^{q_i})^{-1} (\boldsymbol{\mu}_t^{q_i}(\mathbf{s}_t^{ij}) - \boldsymbol{\mu}^\pi(\mathbf{s}_t^{ij}; \mathbf{w})) + 2(\boldsymbol{\lambda}_{\mu t}^i)' \boldsymbol{\mu}^\pi(\mathbf{s}_t^{ij}; \mathbf{w}) \right]. \quad (21.27)$$

The policy covariance  $\boldsymbol{\Sigma}^\pi$  can be computed using a closed form expression  $\boldsymbol{\Sigma}^\pi = \left[ \frac{1}{MT} \sum_{i=1}^M \sum_{t=1}^T (\boldsymbol{\Sigma}_t^{q_i})^{-1} \right]^{-1}$ .

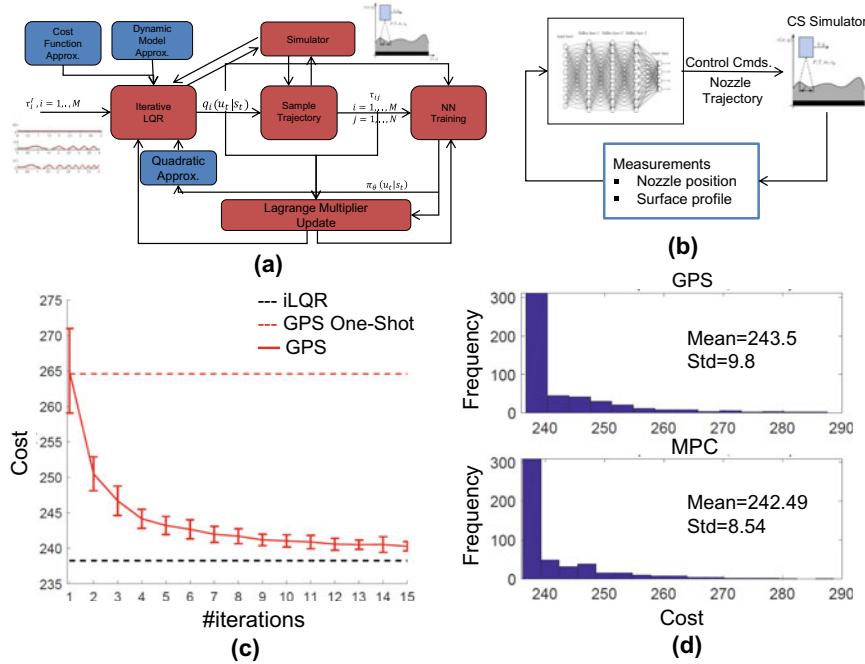
Step 4: Update the Lagrange multipliers

$$\boldsymbol{\lambda}_{\mu t}^i \leftarrow \boldsymbol{\lambda}_{\mu t}^i + \alpha \nu_t \frac{1}{N} \sum_{j=1}^N [\boldsymbol{\mu}^\pi(\mathbf{s}_t^{ij}; \mathbf{w}) - \boldsymbol{\mu}_t^{q_i}(\mathbf{s}_t^{ij})], \quad t = 1, \dots, T; \quad i = 1, \dots, M. \quad (21.28)$$

### 21.4.3 Simulation Results

In this section we compare GPS with iLQR and MPC and evaluate its generalizability to handle different types of surface indentations for repair. For comparative purposes we also consider one-shot GPS approach. By one-shot GPS we mean that Steps 1–3 in the GPS algorithm (see Sect. 21.4.2) are applied only once, i.e., first iLQR is used to generate the guiding distribution with the given cost function without the additional coupling terms related to Lagrange multiplier and KL divergence in (21.23), and similarly the training samples drawn from the so generated guiding distributions are used to train the NN with standard prediction error objective, i.e., without the coupling terms in (21.24).

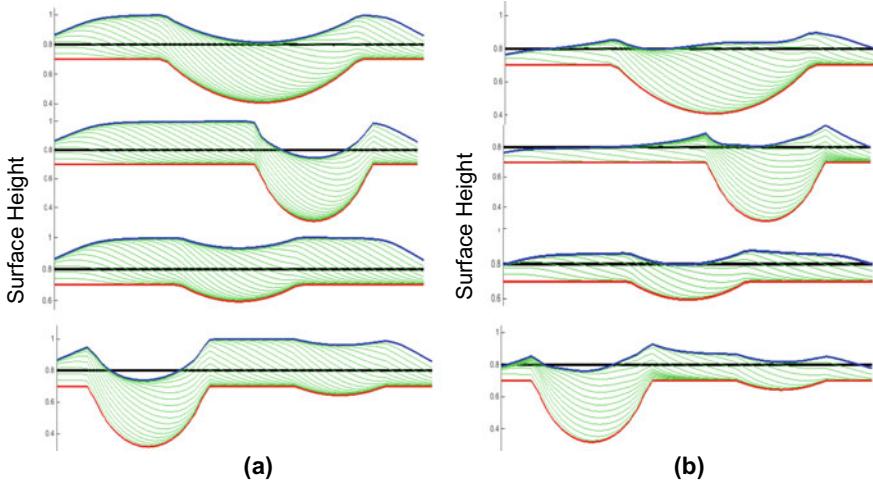
Three trajectory samples, i.e.,  $N = 3$  containing CS surfaces with increasing level of unevenness were selected for generating the guiding samples, see Fig. 21.7a. The



**Fig. 21.7** **a** Schematic of GPS framework, **b** Schematic of closed-loop control, **c** Comparison of one-shot GPS and GPS with iLQR during training, **d** Comparison of GPS-based control performance with MPC

NN used in GPS had a 101-dimensional input layer (corresponding to 100 discretized cells and the nozzle position), hidden layer with 10 units, and a one-dimensional output layer corresponding to the nozzle speed. Figure 21.7c shows the optimal cost achieved by iLQR. Without the coupling and iteration via Lagrange multiplier update (which causes the NN policy and iLQR solution to incrementally match each other), one-shot GPS results in a policy with comparatively sub-optimal costs and with large variance. The GPS algorithm on the other hand shows convergence toward locally optimal iLQR solution, and the variance also reduces with increasing number of iterations. The GPS-based control policy learnt after 15 iterations is compared with the nonlinear MPC-based approach. For this purpose we randomly generate initial surfaces to be repaired and implement feedback control based on the two approaches as shown in Fig. 21.7b. We used the standard nonlinear optimization routine in MATLAB to solve the MPC. Figure 21.7d shows that GPS has similar optimal cost distribution as MPC, and thus demonstrating that GPS-based NN controller can achieve optimal solutions in par with MPC. Moreover, unlike MPC where an optimization problem needs to be solved for each instance, GPS policy is precomputed offline once, and thus offers a computational advantage.

One important requirement for RL policy is that it exhibits generalizability to a broad range of repair geometries. To evaluate this aspect of GPS, few trajectory



**Fig. 21.8** Comparison of surface profile of deposition, resulting from (a) open loop constant speed nozzle motion, and (b) GPS based feedback motion control of nozzle speed. Each case starts with a different location and depth of initial semicircular indentation. The black lines indicate desired surface profile

samples with semicircular indentation surface were chosen, see Fig. 21.8. In CS application, semicircular indentations of various depth, width, and orientation are very common as different types of surface damages are first machined to semicircular indentations before applying CS for repair. Figure 21.8 presents the behavior of GPS over a range of semicircular indentations with single and multiple occurrences. Here, GPS was trained using surfaces with only single indentation with different sizes and locations with the goal of filling these indentations and make the surface nearly flat (denoted by black lines). As can be seen, the GPS-based closed-loop control performs better (Fig. 21.8b) than open-loop constant speed profile (Fig. 21.8a) in achieving desired surface profile while depositing less material. Furthermore, GPS-based controller is able to generalize well for surface with two indentations with different sizes (see last row of surfaces) which is not used during training. The GPS framework has also been experimentally implemented, see [70] for the details.

## 21.5 Future Outlook

While significant progress has been made in RL/DRL, some of the fundamental questions still remain when scaling to complex tasks: how to effectively interact with the environment through, e.g., efficient exploration vs. exploitation and improved sample efficiency; how to learn from experience effectively in problems with long-term credit assignment and sparse reward signals; how can one promote adaptation

and life long learning; and how can one guarantee safety, human–machine trust and promote explainability. Addressing the last challenge is particularly important before RL could become a mainstream approach in aerospace domain where mission critical applications abound.

Recent promising directions that attempt to address above stated challenges include: data efficient exploration and variance reduction techniques [31, 56, 74, 83], multi-task RL [73], hierarchical RL [7, 58, 78], incorporating long-term memory and predictive modeling in RL [82], combined model-free and model-based approaches [63], and incorporating expert knowledge such as via learning from demonstrations [6, 25, 88]. In particular, learning hierarchical structures in control and reinforcement learning has long been recognized as an important research goal, due to its potential to drastically improve generalization and transfer.

In many applications, it is expected that RL agents will be trained offline in simulation environment, and need to exhibit robustness when deployed in real world. Novel approaches for training such as domain randomization will be required which account for differences in simulation and real environment [34, 76]. For life long learning harnessing compositionality and learning-to-learn to rapidly acquire and generalize knowledge to new tasks and situations would become necessary [39, 67, 75]. In order to adapt to changing environment/objectives, safe exploration and learning will be needed [8, 15, 27, 66]. Incorporating constraints in RL [2] is another avenue to guarantee bounded behavior. Safety, trust, and explainability will be particularly important in scenarios where RL agents are expected to work in synergy with humans [5]. Furthermore research is required to scale RL to multi-agent settings [12, 51, 60, 77], and for developing appropriate architectures for deployment including end-to-end learning approaches [41, 57], and distributed/edge computing-based solutions [35, 48].

## References

1. Abbeel, P., Ng, A.Y.: Apprenticeship learning via inverse reinforcement learning. In: Proceedings of the Twenty-First International Conference on Machine Learning, p. 1. ACM, New York (2004)
2. Achiam, J., Held, D., Tamar, A., Abbeel, P.: Constrained policy optimization. In: Proceedings of the 34th International Conference on Machine Learning, vol. 70, pp. 22–31. JMLR. org (2017)
3. Allgöwer, F., Zheng, A.: Nonlinear Model Predictive Control, vol. 26. Birkhäuser, Basel (2012)
4. Altman, E.: Constrained Markov decision processes, vol. 7. CRC Press, Boca Raton (1999)
5. Amershi, S., Cakmak, M., Knox, W.B., Kulesza, T.: Power to the people: the role of humans in interactive machine learning. *AI Mag.* **35**(4), 105–120 (2014)
6. Argall, B.D., Chernova, S., Veloso, M., Browning, B.: A survey of robot learning from demonstration. *Robot. Auton. Syst.* **57**(5), 469–483 (2009)
7. Barto, A.G., Mahadevan, S.: Recent advances in hierarchical reinforcement learning. *Discret. Event Dyn. Syst.* **13**(1–2), 41–77 (2003)
8. Berkenkamp, F., Turchetta, M., Schoellig, A., Krause, A.: Safe model-based reinforcement learning with stability guarantees. In: Advances in Neural Information Processing Systems, pp. 908–918 (2017)

9. Bertsekas, D.P.: Dynamic Programming and Optimal Control, vol. 1. Athena scientific, Belmont, MA (1995)
10. Bertsekas, D.P., Tsitsiklis, J.N.: Neuro-Dynamic Programming, vol. 5. Athena Scientific, Belmont, MA (1996)
11. Bohg, J., Hausman, K., Sankaran, B., Brock, O., Krugic, D., Schaal, S., Sukhatme, G.S.: Interactive perception: leveraging action in perception and perception in action. *IEEE Trans. Robot.* **33**(6), 1273–1291 (2017)
12. Bušoniu, L., Babuška, R., De Schutter, B.: Multi-agent reinforcement learning: an overview. In: *Innovations in Multi-agent Systems and Applications*, vol. 1, pp. 183–221. Springer, Berlin (2010)
13. Cai, Q., Filos-Ratsikas, A., Tang, P., Zhang, Y.: Reinforcement mechanism design for fraudulent behaviour in e-commerce. In: *Thirty-Second AAAI Conference on Artificial Intelligence* (2018)
14. Cassandra, A.R.: Exact and approximate algorithms for partially observable Markov decision processes. PhD thesis, Brown University (1998)
15. Chakrabarty, A., Jha, D.K., Buzzard, G.T., Wang, Y., Vamvoudakis, K.: Safe approximate dynamic programming via kernelized lipschitz estimation. [arXiv:1907.02151](https://arxiv.org/abs/1907.02151) (2019)
16. Chakraborty, A., Shishkin, S., Birnkrant, M.J.: Optimal control of build height utilizing optical profilometry in cold spray deposits. In: *Sensors and Smart Structures Technologies for Civil, Mechanical, and Aerospace Systems 2017*. International Society for Optics and Photonics, vol. 10168, p. 101683H (2017)
17. Yangquan, C., Changyun, W.: *Iterative Learning Control: Convergence, Robustness and Applications*. Springer, Berlin (1999)
18. Cox, T.H., Nagy, C.J., Skoog, M.A., Somers, I.A., Warner, R.: Civil uav capability assessment
19. Ding, X.C., Pinto, A., Surana, A.: Strategic planning under uncertainties via constrained markov decision processes. In: *2013 IEEE International Conference on Robotics and Automation*, pp. 4568–4575. IEEE, Piscataway (2013)
20. Ding, X.D., Englot, B., Pinto, A., Speranzon, A., Surana, A.: Hierarchical multi-objective planning: from mission specifications to contingency management. In: *2014 IEEE International Conference on Robotics and Automation (ICRA)*, pp. 3735–3742. IEEE, Piscataway (2014)
21. Dolgov, D.A., Durfee, E.H.: Stationary deterministic policies for constrained mdps with multiple rewards, costs, and discount factors. In: *IJCAI*, pp. 1326–1331 (2005)
22. Duan, Y., Chen, X., Houthooft, R., Schulman, J., Abbeel, P.: Benchmarking deep reinforcement learning for continuous control. CoRR. [arxiv:abs/1604.06778](https://arxiv.org/abs/abs/1604.06778) (2016)
23. El-Tantawy, Samah., Abdulhai, Baher, Abdelgawad, Hossam: Multiagent reinforcement learning for integrated network of adaptive traffic signal controllers (marlin-atsc): methodology and large-scale application on downtown toronto. *IEEE Trans. Intell. Trans. Syst.* **14**(3), 1140–1150 (2013)
24. Everton, S.K., Hirsch, M., Stravroulakis, P., Leach, R.K., Clare, A.T.: Review of in-situ process monitoring and in-situ metrology for metal additive manufacturing. *Mat. Design* **95**, 431–445 (2016)
25. Finn, C., Levine, S., Abbeel, P.: Guided cost learning: Deep inverse optimal control via policy optimization. In: *International Conference on Machine Learning*, pp. 49–58 (2016)
26. Garcia, C.E., Prett, D.M., Morari, M.: Model predictive control: theory and practicea survey. *Automatica* **25**(3), 335–348 (1989)
27. Garcia, Javier, Fernández, Fernando: A comprehensive survey on safe reinforcement learning. *J. Mach. Learn. Res.* **16**(1), 1437–1480 (2015)
28. Gittins, J., Glazebrook, K., Weber, R.: *Multi-armed bandit allocation indices*. Wiley, Hoboken (2011)
29. Glavic, Mevludin., Fonteneau, Raphaël, Ernst, Damien: Reinforcement learning for electric power system decision and control: past considerations and perspectives. *IFAC-PapersOnLine* **50**(1), 6918–6927 (2017)
30. Gottesman, O., Johansson, F., Komorowski, M., Faisal, A., Sontag, D., Doshi-Velez, F., Celi, L.A.: Guidelines for reinforcement learning in healthcare. *Nat. Med.* **25**(1), 16–18 (2019)

31. Gu, S.S., Lillicrap, T., Turner, R.E., Ghahramani, Z., Schölkopf, B., Levine, S.: Interpolated policy gradient: merging on-policy and off-policy gradient estimation for deep reinforcement learning. In: Advances in Neural Information Processing Systems, pp. 3846–3855 (2017)
32. Haddal, C.C., Gertler, J.: Homeland security: unmanned aerial vehicles and border surveillance. Library of Congress Washington DC Congressional Research Service (2010)
33. Hero, A.O., Castañón, D., Cochran, D., Kastella, K.: Foundations and Applications of Sensor Management. Springer Science & Business Media, Berlin (2007)
34. James, S., Davison, A.J., Johns, E.: Transferring end-to-end visuomotor control from simulation to real world for a multi-stage task. [arXiv:1707.02267](https://arxiv.org/abs/1707.02267) (2017)
35. Kehoe, Ben., Patil, Sachin., Abbeel, Pieter, Goldberg, Ken: A survey of research on cloud robotics and automation. IEEE Trans. Autom. Sci. Eng. **12**(2), 398–409 (2015)
36. Kober, J., Bagnell, J.A., Peters, J.: Reinforcement learning in robotics: a survey. Int. J. Robot. Res. **32**(11), 1238–1274 (2013)
37. Kreucher, C.M.: An information-based approach to sensor resource allocation. PhD thesis, University of Michigan (2005)
38. Krishnamurthy, V., Evans, R.J.: Hidden markov model multiarm bandits: a methodology for beam scheduling in multitarget tracking. IEEE Trans. Signal Process. **49**(12), 2893–2908 (2001)
39. Lake, B.M., Ullman, T.D., Tenenbaum, J.B., Gershman, S.J.: Building machines that learn and think like people. Behav. Brain Sci. **40**, e253 (2017)
40. Levine, S., Abbeel, P.: Learning neural network policies with guided policy search under unknown dynamics. In: Ghahramani, Z., Welling, M., Cortes, C., Lawrence, N.D., Weinberger, K.Q. (eds.), Advances in Neural Information Processing Systems, vol. 27, pp. 1071–1079. Curran Associates, Inc., Red Hook, NY (2014)
41. Levine, Sergey., Finn, Chelsea., Darrell, Trevor, Abbeel, Pieter: End-to-end training of deep visuomotor policies. J. Mach. Learn. Res. **17**(1), 1334–1373 (2016)
42. Levine, S., Koltun, V.: Guided policy search. In: Proceedings of the 30th International Conference on Machine Learning, ICML 2013, Atlanta, GA, USA, 16–21 June 2013, pp. 1–9 (2013)
43. Levine, S., Koltun, V.: Learning complex neural network policies with trajectory optimization. In: Jebara, T., Xing, E.P. (eds.), Proceedings of the 31st International Conference on Machine Learning (ICML-14). JMLR Workshop and Conference Proceedings, pp. 829–837 (2014)
44. Lewis, F.L., Liu, D.: Reinforcement Learning and Approximate Dynamic Programming for Feedback Control, vol. 17. Wiley, Hoboken (2013)
45. Li, J., Monroe, W., Ritter, A., Galley, M., Gao, J., Jurafsky, D.: Deep reinforcement learning for dialogue generation. [arXiv:1606.01541](https://arxiv.org/abs/1606.01541) (2016)
46. Li, Weiwei, Todorov, Emanuel: Iterative linear quadratic regulator design for nonlinear biological movement systems. ICINCO **1**, 222–229 (2004)
47. Li, Y.: Deep reinforcement learning: an overview. [arXiv:1701.07274](https://arxiv.org/abs/1701.07274) (2017)
48. Liang, E., Liaw, R., Moritz, P., Nishihara, R., Fox, R., Goldberg, K., Gonzalez, J.E., Jordan, M.I., Stoica, I.: Rllib: abstractions for distributed reinforcement learning. [arXiv:1712.09381](https://arxiv.org/abs/1712.09381) (2017)
49. Lillicrap, T.P., Hunt, J.J., Pritzel, A., Heess, N., Erez, T., Tassa, Y., Silver, D., Wierstra, D.: Continuous control with deep reinforcement learning. [arXiv:1509.02971](https://arxiv.org/abs/1509.02971) (2015)
50. Liu, Y.-E., Mandel, T., Brunskill, E., Popovic, Z.: Trading off scientific knowledge and user learning with multi-armed bandits. In: EDM, pp. 161–168 (2014)
51. Lowe, R., Wu, Y., Tamar, A., Harb, J., Abbeel, O.P., Mordatch, I.: Multi-agent actor-critic for mixed cooperative-competitive environments. In: Advances in Neural Information Processing Systems, pp. 6379–6390 (2017)
52. Luong, N.C., Hoang, D.T., Gong, S., Niyato, D., Wang, P., Liang, Y.-C., Kim, D.I.: Applications of deep reinforcement learning in communications and networking: a survey. IEEE Communications Surveys and Tutorials (2019)
53. Mao, H., Alizadeh, M., Menache, I., Kandula, S.: Resource management with deep reinforcement learning. In: Proceedings of the 15th ACM Workshop on Hot Topics in Networks, pp. 50–56. ACM, New York (2016)

54. Mathew, George, Mezić, Igor: Metrics for ergodicity and design of ergodic dynamics for multi-agent systems. *Phys. D: Nonlinear Phenom.* **240**(4–5), 432–442 (2011)
55. Mathew, G., Surana, A., Mezić, I.: Uniform coverage control of mobile sensor networks for dynamic target detection. In: 49th IEEE Conference on Decision and Control (CDC), pp. 7292–7299. IEEE, Piscataway (2010)
56. Mnih, V., Badia, A.P., Mirza, M., Graves, A., Lillicrap, T., Harley, T., Silver, D., Kavukcuoglu, K.: Asynchronous methods for deep reinforcement learning. In: International Conference on Machine Learning. pp. 1928–1937 (2016)
57. Mnih, V., Kavukcuoglu, K., Silver, D., Rusu, A.A., Veness, J., Bellemare, M.G., Graves, A., Riedmiller, M., Fidjeland, A.K., Ostrovski, G., et al.: Human-level control through deep reinforcement learning. *Nature* **518**(7540), 529–533 (2015)
58. Nachum, O., Gu, S.S., Lee, H., Levine, S.: Data-efficient hierarchical reinforcement learning. In: Advances in Neural Information Processing Systems, pp. 3303–3313 (2018)
59. U. S. Army UAS Center of Excellence.: U.S. Army roadmap for unmanned aircraft systems, pp. 2010–2035. U.S. Army (2010)
60. Omidshafiei, S., Pazis, J., Amato, C., How, J.P., Vian, J.: Deep decentralized multi-task multi-agent reinforcement learning under partial observability. In: Proceedings of the 34th International Conference on Machine Learning, vol. 70, pp. 2681–2690. JMLR.org (2017)
61. Oshin, O., Nair, B.M., Ding, J., Osborne, R.W., Varma, R., Bernal, E.A., Stramandinoli, F.: Generative and discriminative machine learning models for reactive robot planning in human-robot collaboration (2019)
62. Peters, Jan, Schaal, Stefan: Reinforcement learning of motor skills with policy gradients. *Neural Netw.* **21**(4), 682–697 (2008)
63. Pong, V., Gu, S., Dalal, M., Levine, S.: Temporal difference models: model-free deep rl for model-based control. [arXiv:1802.09081](https://arxiv.org/abs/1802.09081) (2018)
64. Rajeswaran, A., Kumar, V., Gupta, A., Vezzani, G., Schulman, J., Todorov, E., Levine, S.: Learning complex dexterous manipulation with deep reinforcement learning and demonstrations. [arXiv:1709.10087](https://arxiv.org/abs/1709.10087) (2017)
65. Sallab, Ahmad E.L., Abdou, Mohammed., Perot, Etienne., Yogamani, Senthil: Deep reinforcement learning framework for autonomous driving. *Electron. Imaging* **2017**(19), 70–76 (2017)
66. Schulman, J., Levine, S., Moritz, P., Jordan, M.I., Abbeel, P.: Trust region policy optimization. [arXiv:1502.05477](https://arxiv.org/abs/1502.05477) (2015)
67. Silver, D.L., Yang, Q., Li, L.: Lifelong machine learning systems: beyond learning algorithms. In: 2013 AAAI Spring Symposium Series (2013)
68. Silver, David., Hubert, Thomas., Schrittwieser, Julian., Antonoglou, Ioannis., Lai, Matthew., Guez, Arthur., Lanctot, Marc., Sifre, Laurent., Kumaran, Dharshan., Graepel, Thore., et al.: A general reinforcement learning algorithm that masters chess, shogi, and go through self-play. *Science* **362**(6419), 1140–1144 (2018)
69. Stramandinoli, F., Lore, K.G., Peters, J.R., ONeill, P.C., Nair, B.M., Varma, R., Ryde, J.C., Miller, J.T., Reddy, K.K.: Robot learning from human demonstration in virtual reality (2018)
70. Surana, A., Reddy, K., Siopsis, M.: Guided policy search based control of a high dimensional advanced manufacturing process. arxiv preprint, [arXiv:2009.05838](https://arxiv.org/abs/2009.05838) (2020)
71. Sutton, R.S., Barto, A.G.: Reinforcement learning: an introduction. MIT press, Cambridge (2018)
72. Tapia, G., Elwany, A.: A review on process monitoring and control in metal-based additive manufacturing. *J. Manuf. Sci. Eng.* **136**(6), 060801 (2014)
73. Taylor, M.E., Stone, P.: Transfer learning for reinforcement learning domains: a survey. *J. Mach. Learn. Res.* **10**, 1633–1685 (2009)
74. Thomas, P., Brunskill, E.: Data-efficient off-policy policy evaluation for reinforcement learning. In: International Conference on Machine Learning, pp. 2139–2148 (2016)
75. Thrun, S., Pratt, L.: Learning to learn. Springer Science & Business Media, Berlin (2012)
76. Tobin, J., Fong, R., Ray, A., Schneider, J., Zaremba, W., Abbeel, P.: Domain randomization for transferring deep neural networks from simulation to the real world. In: 2017 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS), pp. 23–30. IEEE, Piscataway (2017)

77. Vamvoudakis, K.G., Lewis, F.L., Hudas, G.R.: Multi-agent differential graphical games: Online adaptive learning solution for synchronization with optimality. *Automatica* **48**(8), 1598–1611 (2012)
78. Vezhnevets, A.S., Osindero, S., Schaul, T., Heess, N., Jaderberg, M., Silver, D., Kavukcuoglu, K.: Feudal networks for hierarchical reinforcement learning. In: Proceedings of the 34th International Conference on Machine Learning, vol. 70, pp. 3540–3549. JMLR.org (2017)
79. Wang, X., Feng, F., Klecka, M.A., Mordasky, M.D., Garofano, J.K., El-Wardany, T., Nardi, A., Champagne, V.K.: Characterization and modeling of the bonding process in cold spray additive manufacturing. *Addit. Manuf.* **8**, 149–162 (2015)
80. Wang, Y., Gao, F., Doyle, F.J., III.: Survey on iterative learning control, repetitive control, and run-to-run control. *J. Process Control* **19**(10), 1589–1600 (2009)
81. Washburn, R.B., Schneider, M.K., Fox, J.J.: Stochastic dynamic programming based approaches to sensor resource management. In: Proceedings of the Fifth International Conference on Information Fusion. FUSION 2002 (IEEE Cat. No. 02EX5997), vol. 1, pp. 608–615. IEEE, Annapolis, MD (2002)
82. Wayne, G., Hung, C-C., Amos, D., Mirza, M., Ahuja, A., Grabska-Barwinska, A., Rae, J., Mirowski, P., Leibo, J.Z., Santoro, A., et al.: Unsupervised predictive memory in a goal-directed agent. arxiv preprint, [arXiv:1803.10760](https://arxiv.org/abs/1803.10760) (2018)
83. Wu, C., Rajeswaran, A., Duan, Y., Kumar, V., Bayen, A.M., Kakade, S., Mordatch, I., Abbeel, P.: Variance reduction for policy gradient with action-dependent factorized baselines. [arXiv:1803.07246](https://arxiv.org/abs/1803.07246) (2018)
84. Yin, Shuo., Cavaliere, Pasquale., Aldwell, Barry., Jenkins, Richard., Liao, Hanlin., Li, Wenya, Lupoi, Rocco: Cold spray additive manufacturing and repair: fundamentals and applications. *Addit. Manuf.* **21**, 628–650 (2018)
85. Zhang, T., Kahn, G., Levine, S., Abbeel, P.: Learning deep control policies for autonomous aerial vehicles with mpc-guided policy search. CoRR. [arxiv:abs/1509.06791](https://arxiv.org/abs/abs/1509.06791) (2015)
86. Zhao, X., Xia, L., Tang, J., Yin, D.: Deep reinforcement learning for search, recommendation, and online advertising: a survey by xiangyu zhao, long xia, jiliang tang, and dawei yin with martin vesely as coordinator. ACM SIGWEB Newsletter, vol. 4. Spring, Berlin (2019)D
87. Zhu, H., Gupta, A., Rajeswaran, A., Levine, S., Kumar, V.: Dexterous manipulation with deep reinforcement learning: Efficient, general, and low-cost. In: 2019 International Conference on Robotics and Automation (ICRA), pp. 3651–3657. IEEE, Montreal, QC (2019)
88. Ziebart, B.D., Maas, A., Bagnell, J.A., Dey, A.K.: Maximum entropy inverse reinforcement learning. In: Proceedings of the Twenty-Third AAAI Conference on Artificial Intelligence (2008)

# Chapter 22

## Robust Autonomous Driving with Human in the Loop



Mengzhe Huang, Zhong-Ping Jiang, Michael Malisoff, and Leilei Cui

**Abstract** This chapter presents a reinforcement-learning-based shared control design for semi-autonomous vehicles with a human in the loop. The co-pilot controller and the driver operate together and control the vehicle simultaneously. To address the effects of human driver reaction time, the interconnected human–vehicle system is described by differential-difference equations. Exploiting the real-time measured data, the adaptive optimal shared controller is learned via adaptive dynamic programming, without accurate knowledge of the driver and vehicle models. Adaptivity, near optimality and stability are ensured simultaneously when the data-driven shared steering controller is applied to the human-in-the-loop vehicle system, which can handle the potential parametric variations and uncertainties in the human–vehicle system. The efficacy of the proposed control strategy is validated by proofs and demonstrated by numerical simulations.

### 22.1 Introduction

Driving is more than the operation of throttle control, braking, and steering, due to unanticipated road conditions and potential collision with other vehicles. According to [1], there were an estimated 6,452,000 motor vehicle crashes in the United States in 2017, resulting in 37,133 fatalities and 2,746,000 people injured. In the last decades, autonomous driving and advanced driver-assistance systems (ADAS) have been proposed to reduce drivers’ workloads and to improve transportation safety.

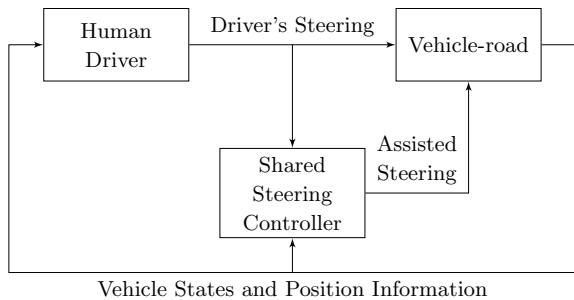
Both academia and industry have worked to improve sensing, computing, and control technologies for autonomous vehicles. Prior research efforts have been focused on the automatic control of a vehicle, where the driver is not engaged in controlling

---

M. Huang · Z.-P. Jiang (✉) · L. Cui  
Department of Electrical and Computer Engineering, Tandon School of Engineering, New York University, Brooklyn, NY 11201, USA  
e-mail: [zjiang@nyu.edu](mailto:zjiang@nyu.edu)

M. Malisoff  
Department of Mathematics, Louisiana State University, Baton Rouge, LA 70803-4918, USA

**Fig. 22.1** Block diagram of shared steering control.  
 © 2019 IEEE. Reprinted, with permission, from [19]



the vehicle. However, the human subject is expected to take over the vehicle control, when the vehicle automation system encounters any unanticipated driving situation. According to [36], this transition from autonomous driving to manual driving is challenging for human drivers, since they can be distracted by non-driving tasks. Recently, shared control has been investigated as a promising framework to integrate drivers' steering and throttle inputs into the control design for ADAS [2, 19, 21, 29, 32, 38, 42], as shown in Fig. 22.1. Essentially, the human driver is constantly involved in the task of driving while the shared controller provides assistance to improve the safety performance.

In order to incorporate the interaction between humans and vehicles, researchers have proposed several driver models to illustrate a human's driving mechanism including perception and control. In [20, 28], game-theoretic modeling strategies are applied to describe a driver's steering and car-following behavior. In [37], the authors suggested a two-point visual control model in which a driver fixates his or her focus on two regions (near and far) in front of the vehicle. After processing the visual information, he or she makes steering decisions so that the vehicle follows the centerline of the lane if there is no obstacle. Exploiting these driver models, several cooperative or shared steering control strategies have been constructed for semi-autonomous vehicles in recent years. Considering diverse driver characteristics, the authors of [8] propose a robust driver-assistance design to improve steering performance during the handover transition. A two-level hierarchical shared control design is developed in [38], including an operational and a tactical part, where the tactical layer aims to reduce the conflict between the driver and the automation system, and the lower level control part assists the driver in lane keeping. In [15], the authors consider a shared control design for connected vehicles to achieve energy-efficient mobility in an urban area by sending driving commands to the drivers.

There are still open problems in the field of shared steering control for semi-autonomous vehicles. In previous studies, e.g., [18, 38], the driver models used to develop shared controllers do not explicitly consider human reaction time. Nevertheless, small delays in the human-in-the-loop vehicle system can greatly affect stability and control performance [13]. In addition, driver reaction time differs from person to person based on age differences [40]. As pointed out in [6], the short- and long-term variations of the driver model, including feedback gains, should not be ignored

and need to be handled in real time. These factors motivate our development of a novel shared control algorithm to guarantee adaptivity, optimality, and stability of the human-in-the-loop vehicular control system subject to human driver reaction delays.

In this paper, we take the driver reaction time into consideration and propose a data-driven control solution to the shared steering problem, employing an adaptive dynamic programming (ADP) method. ADP is a non-model-based control design approach combining ideas from reinforcement learning and optimal control [5]. ADP aims to iteratively find optimal control laws using real-time measurable data without precise knowledge of the system dynamics [7, 12, 22, 23, 26, 33, 41]. Recently, ADP has been applied to solve adaptive optimal control problems for connected and autonomous vehicles (CAVs) [10, 17]. In our previous work [18, 19], robust ADP (RADP) is employed with small-gain theory and an output regulation technique to design a data-driven shared steering assistance system, taking diverse driver behavior into consideration. However, the effect of driver reaction delay had not been addressed.

The main contributions of our work are summarized below:

1. An ADP-based control design is proposed to solve the adaptive optimal shared steering control problem with humans in the loop, taking driver reaction time and unknown driver and vehicle system parameters into consideration;
2. Output regulation techniques are integrated with the data-driven learning-based control design to achieve robust steering performance for semi-autonomous vehicles;
3. Optimality and stability analyses of the shared human–vehicle control system are presented, which give desirable lane-keeping performance.

We explicitly take the driver reaction time into account and formulate the shared steering problem as a control design for a set of differential-difference equations [3, 4, 39]. Then, a data-driven ADP-based control policy is developed by employing real-time data, without precise knowledge of the driver model and the vehicle parameters.

The rest of the study is organized as follows. Section 22.2 describes the mathematical model of an interconnected driver–vehicle system. Section 22.3 presents a model-based method to design an optimal shared controller to achieve lane keeping. Section 22.4 demonstrates a data-driven approach to obtain the approximate optimal shared control policy in the presence of unknown driver and vehicle dynamics. Section 22.5 provides numerical simulations, including an algorithmic implementation and discussions. Section 22.6 gives the conclusion of this paper.

**Notation.** Throughout this study,  $\mathbb{R}$  and  $\mathbb{Z}_+$  denote the sets of real numbers and non-negative integers, respectively,  $\mathbb{C}^-$  stands for the open left-half complex plane,  $\mathbb{C}^+$  stands for the right-half complex plane, including the imaginary axis, and  $|\cdot|$  represents the Euclidean norm for vectors, or the induced matrix norm for matrices. Also,  $\otimes$  indicates the Kronecker product, and  $\text{vec}(A) = [a_1^T, a_2^T, \dots, a_m^T]^T$ , where  $a_i \in \mathbb{R}^n$  are the columns of  $A \in \mathbb{R}^{n \times m}$ . When  $m = n$ ,  $\sigma(A)$  is the complex spectrum (or the set of all eigenvalues) of  $A$ ,  $\det(A)$  is the determinant of  $A$ , and  $I_n$  represents the  $n \times n$  identity matrix. For a symmetric matrix  $P \in \mathbb{R}^{m \times m}$ ,  $\text{vecs}(P) = [p_{11}, 2p_{12}, \dots, 2p_{1m}, p_{22}, 2p_{23}, \dots, 2p_{m-1,m}, p_{mm}]^T \in \mathbb{R}^{\frac{1}{2}m(m+1)}$ . For an arbitrary

column vector  $v \in \mathbb{R}^n$ ,  $\text{vec}(v) = [v_1^2, v_1 v_2, \dots, v_1 v_n, v_2^2, \dots, v_{n-1} v_n, v_n^2]^T \in \mathbb{R}^{\frac{1}{2}n(n+1)}$ , and  $\text{diag}(v_1, \dots, v_n)$  stands for a diagonal matrix with  $v_1, v_2, \dots, v_n$  on the diagonal and zeros elsewhere.

## 22.2 Mathematical Modeling of Human–Vehicle Interaction

We present the shared control framework based on a vehicle model and a driver model with reaction time delay. Specifically, the human-in-the-loop shared control system will be formulated as a set of differential-difference equations, where the delay in the state variable is produced by a driver's reaction time.

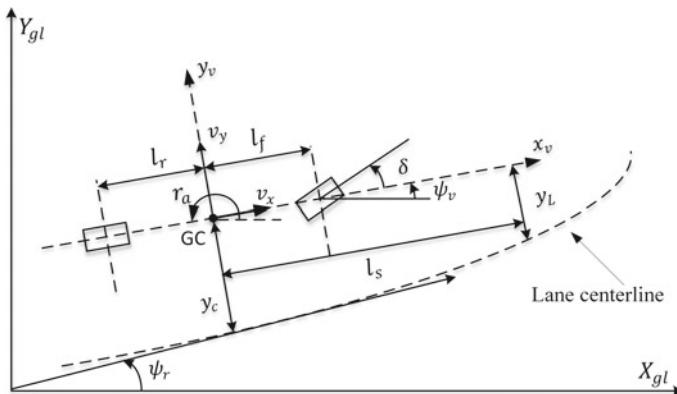
### 22.2.1 Vehicle Lateral Dynamics

The complete vehicle model consists of the lateral vehicle dynamics, the steering column and the vision-position model for the lane-keeping task. According to [34, 38], under the assumptions of small angles and constant longitudinal speed, it can be described by

$$\dot{x}_v(t) = A_v x_v(t) + B_v(u(t) + T_d(t)) + D_v \rho, \quad (22.1a)$$

$$y(t) = C_v x_v(t), \quad (22.1b)$$

where  $x_v = [v_y \ r_a \ \psi_L \ y_L \ \delta \ \dot{\delta}]^T$ ,  $y = y_c$  is the lateral offset at the vehicle center of gravity, and  $\psi_L = \psi_v - \psi_r$  as illustrated in Fig. 22.2. In (22.1), the model assumes



**Fig. 22.2** Vehicle model illustration. © 2019 IEEE. Reprinted, with permission, from [19]

that we have a constant curvature  $\rho$ . In practice, the non-constant road curvature can be approximated by piecewise constant functions as in [19, 31]. In a detailed description,  $v_y$  is the vehicle lateral velocity,  $r_a$  is the yaw rate,  $\psi_L$  is the heading angle error,  $y_L$  is the previewed distance from the lane center at a look-ahead distance  $l_s$ ,  $\delta$  is the steering angle,  $\dot{\delta}$  is the steering angle rate, and  $T_d$  is the driver's torque. See Fig. 22.2 for details. The state vector  $x_v$  can be measured using various sensors including a camera and the inertial measurement unit (IMU). The matrices  $A_v$ ,  $B_v$ ,  $C_v$  and  $D_v$  are expressed as

$$A_v = \begin{bmatrix} a_{11} & a_{12} & 0 & 0 & b_1 & 0 \\ a_{21} & a_{22} & 0 & 0 & b_2 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 \\ 1 & l_s & v_x & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 \\ T_{s1} & T_{s2} & 0 & 0 & T_{s3} & T_{s4} \end{bmatrix}, B_v = \begin{bmatrix} 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ \frac{1}{I_s R_s} \end{bmatrix},$$

$$C_v = [0 \ 0 \ -l_s \ 1 \ 0 \ 0],$$

$$D_v = [0 \ 0 \ -v_x \ 0 \ 0 \ 0]^T,$$

where

$$a_{11} = -\frac{2(C_f + C_r)}{M v_x}, \quad a_{12} = \frac{2(C_r l_r - C_f l_f)}{M v_x} - v_x,$$

$$a_{21} = \frac{2(C_r l_r - C_f l_f)}{I_z v_x}, \quad a_{22} = \frac{-2(C_f l_f^2 + C_r l_r^2)}{I_z v_x},$$

$$b_1 = \frac{2C_f}{M}, \quad b_2 = \frac{2C_f l_f}{I_z}, \quad T_{s1} = \frac{2C_f \eta_t}{I_s R_s^2 v_x},$$

$$T_{s2} = \frac{2C_f l_f \eta_t}{I_s R_s^2 v_x}, \quad T_{s3} = \frac{-2C_f \eta_t}{I_s R_s^2}, \quad T_{s4} = -\frac{B_s}{I_s},$$

and  $l_f$  is the distance from the center of gravity to front axle,  $l_r$  is the distance from the center of gravity to rear axle,  $M$  is the mass of the vehicle,  $\eta_t$  is the tire length contact,  $I_z$  is the vehicle yaw moment of inertia,  $I_s$  is the steering system moment of inertia,  $R_s$  is the steering gear ratio,  $B_s$  is the steering system damping,  $C_f$  is the front cornering stiffness,  $C_r$  is the rear cornering stiffness, and  $v_x$  is the fixed longitudinal velocity. The preceding constants are all positive.

**Lemma 22.1** *The vehicle model  $(A_v, B_v)$  is controllable if and only if*

$$M^2 l_f^2 v_x^2 - 2C_r (M l_f^2 l_r + M l_f l_r^2 - I_z l_f - I_z l_r) \neq 0.$$

**Proof** Let  $\lambda_v$  be the determinant of the controllability matrix of the vehicle model  $(A_v, B_v)$ . It is checkable by MATLAB Symbolic Math Toolbox [30] that

$$\lambda_v = \frac{-64C_f^4 C_r^2 (l_f + l_r)^2 \left( M^2 l_f^2 v_x^2 - 2C_r \left( M l_f^2 l_r + M l_f l_r^2 - I_z l_f - I_z l_r \right) \right)}{I_s^6 I_z^4 M^4 R_s^6 v_x^2}.$$

Under the condition in Lemma 22.1, it follows that  $\lambda_v \neq 0$ , which implies the controllability of the pair  $(A_v, B_v)$ . The proof is thus completed.  $\square$

Here and in the sequel, we assume that  $(A_v, B_v)$  is controllable; see our illustration where the controllability condition is satisfied.

### 22.2.2 Interconnected Human–Vehicle Model

To incorporate the human driver's control, the interaction between the vehicle and the driver needs to be investigated. A proper driver model is key for the design of shared control framework. It is pointed out by Donges [9] and Land and Horwood [25] that there are two common fixating zones for a driver during driving, namely, (1) the near region and (2) the far region. The visually perceived information from the far region accounts for the fact that a driver previews the upcoming road condition, including the road geometry and the preceding vehicle on the road. The information from the near region helps correct the vehicle's current position with respect to the lane center in order to prevent unsafe driving on the road. Later, Salvucci et al. [37] proposed a two-point visual driver model to illustrate that the driver controls the vehicle by looking at two points, namely, (1) the near point and (2) the far point. In particular, the information embedded in these two points can be expressed by two visual angles  $\theta_{near}$  and  $\theta_{far}$  [38], where

$$\theta_{near} = \psi_L + \frac{y_L}{v_x T_p}, \quad (22.2a)$$

$$\theta_{far} = \beta_1 v_y + \beta_2 r_a + \beta_3 \delta, \quad (22.2b)$$

where  $\beta_1 = \tau_a a_{21}$ ,  $\beta_2 = \tau_a + \tau_a^2 a_{22}$ ,  $\beta_3 = \tau_a^2 b_2$ . Here,  $\tau_a$  is the anticipatory time and  $T_p$  is the look-ahead preview time [38]. Based on the definitions of  $\theta_{near}$  and  $\theta_{far}$ , the driver's torque  $T_d$  can be modeled by two parts as follows:

$$T_d(t) = G_c(t) + G_a(t) = K_c \theta_{near}(t - \tau) + K_a \theta_{far}(t - \tau), \quad (22.3)$$

where  $G_c$  and  $G_a$  represent compensatory and anticipatory behavior, respectively.  $K_c$  and  $K_a$  represent the driver's proportional gains for his or her driving actions, and the positive constant  $\tau$  is the driver reaction delay, which represents the amount of time that a driver needs to act on the steering wheel when he or she receives new visual information. By combining (22.2) and (22.3), the driver's steering input is modeled as

$$T_d(t) = K_d x_v(t - \tau), \quad (22.4)$$

where

$$K_d = \left[ K_a \beta_1, K_a \beta_2, K_c, \frac{K_c}{v_x T_p}, K_a \beta_3, 0 \right].$$

By combining (22.1) and (22.4), we have the interconnected human–vehicle system

$$\dot{x}_v(t) = A_v x_v(t) + A_d x_v(t - \tau) + B_v u(t) + D_v \rho, \quad (22.5a)$$

$$y(t) = C_v x_v(t), \quad (22.5b)$$

where  $A_d = B_v K_d$ .

## 22.3 Model-Based Control Design

### 22.3.1 Discretization of Differential-Difference Equations

In this study, we aim to find an approximate optimal control law based on the discretization method proposed in [35]. Let  $m$  be a positive integer. For each integer  $i \in \{0, \dots, m-1\}$ , we consider the following finite-dimensional approximation using Taylor expansion:

$$x_v \left( t - \frac{i\tau}{m} \right) \approx x_v \left( t - \frac{(i+1)\tau}{m} \right) + \frac{\tau}{m} \dot{x}_v \left( t - \frac{(i+1)\tau}{m} \right). \quad (22.6)$$

Then, by (22.6), (22.5) can be approximated as follows:

$$\dot{x}(t) = Ax(t) + Bu(t) + D\rho \quad (22.7a)$$

$$y(t) = Cx(t), \quad (22.7b)$$

where  $x(t) = \left[ x_v^T(t), x_v^T \left( t - \frac{\tau}{m} \right), x_v^T \left( t - \frac{2\tau}{m} \right), \dots, x_v^T \left( t - \tau \right) \right]^T \in \mathbb{R}^{n_x}$ ,  $n_x = 6(m+1)$  and

$$A = \begin{bmatrix} A_v & 0 & \cdots & 0 & A_d \\ \frac{m}{\tau} I - \frac{m}{\tau} I & 0 & \cdots & 0 \\ 0 & \frac{m}{\tau} I & -\frac{m}{\tau} I & \cdots & 0 \\ \vdots & \vdots & \ddots & \ddots & \vdots \\ 0 & 0 & 0 & \frac{m}{\tau} I - \frac{m}{\tau} I & \end{bmatrix}, \quad B = \begin{bmatrix} B_v \\ 0 \\ 0 \\ \vdots \\ 0 \end{bmatrix}, \quad D = \begin{bmatrix} D_v \\ 0 \\ 0 \\ \vdots \\ 0 \end{bmatrix},$$

$$C = [C_v \ 0 \ 0 \ \cdots \ 0].$$

The following lemma ensures that the pair  $(A, B)$  of the augmented system (22.7) is stabilizable.

**Lemma 22.2** *The pair of  $(A, B)$  is stabilizable for  $m \geq 1$ .*

**Proof** By applying the Popov–Belevitch–Hautus (PBH) test from [14, Theorem 14.2], we obtain

$$[sI - A, B] = \begin{bmatrix} sI - A_v & 0 & \cdots & 0 & -A_d & B_v \\ -\frac{m}{\tau}I & (\frac{m}{\tau} + s)I & 0 & \cdots & 0 & 0 \\ 0 & -\frac{m}{\tau}I & (\frac{m}{\tau} + s)I & \cdots & 0 & 0 \\ \vdots & \vdots & \ddots & \ddots & \vdots & \vdots \\ 0 & 0 & 0 & -\frac{m}{\tau}I & (\frac{m}{\tau} + s)I & 0 \end{bmatrix}. \quad (22.8)$$

After applying row and column operations, the rank of  $[sI - A, B]$  becomes

$$\begin{aligned} & \text{rank } [sI - A, B] \\ &= \text{rank} \begin{bmatrix} 0 & 0 & \cdots & 0 & -A_d + q(sI - A_v) & B_v \\ -\frac{m}{\tau}I & 0 & 0 & \cdots & 0 & 0 \\ 0 & -\frac{m}{\tau}I & 0 & \cdots & 0 & 0 \\ \vdots & \vdots & \ddots & \ddots & \vdots & \vdots \\ 0 & 0 & \cdots & -\frac{m}{\tau}I & 0 & 0 \end{bmatrix} \end{aligned} \quad (22.9)$$

for any  $q \neq 0$ . Also, for any  $q \neq 0$ , we have

$$\begin{aligned} & \text{rank } [-A_d + q(sI - A_v), B_v] \\ &= \text{rank} \left( [sI - A_v, B_v] \begin{bmatrix} qI & 0 \\ 0 & 1 \end{bmatrix} \begin{bmatrix} I & 0 \\ -K_d & 1 \end{bmatrix} \right) \\ &= 6, \end{aligned} \quad (22.10)$$

where the second equality holds according to Lemma 22.1. Hence, we have

$$\text{rank } ([sI - A, B]) = n_x, \quad (22.11)$$

which implies the stabilizability. Thus, the proof is completed.  $\square$

**Lemma 22.3** *Consider system (22.7). For  $m \geq 1$ , the transmission zeros condition holds, i.e.,*

$$\text{rank } \begin{bmatrix} A & B \\ C & 0 \end{bmatrix} = n_x + 1. \quad (22.12)$$

**Proof** After row and columns operations, one obtains

$$\begin{aligned} & \text{rank} \begin{bmatrix} A & B \\ C & 0 \end{bmatrix} \\ &= \text{rank} \begin{bmatrix} 0 & 0 & \cdots & A_d + A_v & B_v \\ \frac{m}{\tau} I & 0 & \cdots & 0 & 0 \\ 0 & \ddots & 0 & 0 & 0 \\ 0 & \cdots & \frac{m}{\tau} I & 0 & 0 \\ 0 & \cdots & 0 & C_v & 0 \end{bmatrix}. \end{aligned} \quad (22.13)$$

Observe that (22.13) is of full rank if and only if

$$\begin{bmatrix} A_d + A_v & B_v \\ C_v & 0 \end{bmatrix} = 7.$$

Using the MATLAB Symbolic Math Toolbox [30], we can verify that

$$\det \left( \begin{bmatrix} A_d + A_v & B_v \\ C_v & 0 \end{bmatrix} \right) = \frac{4C_f C_r (l_f + l_r)}{I_s I_z M R_s} > 0,$$

which implies that (22.12) holds. The proof is thus completed.  $\square$

### 22.3.2 Formulation of the Shared Control Problem

The objective of lane keeping is to force the output  $y$  of system (22.1) to zero, that is,  $\lim_{t \rightarrow \infty} y(t) = 0$  from all initial states. To this end, we employ the theory of output regulation to tackle the presence of nonzero curvature  $\rho$  in (22.5). More specifically, we design a feedback controller to achieve asymptotic tracking with disturbance rejection for a reference input while preserving the closed-loop stability [16].

The following theorem provides a model-based design for the shared controller to achieve cooperative lane keeping:

**Theorem 22.1** *The lane-keeping task is solvable by applying the shared controller*

$$u(t) = -Kx(t) + (KX + U)\rho \quad (22.14)$$

*to the system (22.7), where  $K$  is chosen such that  $\sigma(A - BK) \subseteq \mathbb{C}^-$ , and where the constants  $X \in \mathbb{R}^{n_x}$  and  $U \in \mathbb{R}$  satisfy the following regulator equations:*

$$0 = AX + BU + D, \quad (22.15a)$$

$$0 = CX. \quad (22.15b)$$

**Proof** Based on Lemma 22.2, the existence of such a  $K$  is ensured. Also, according to Lemma 22.3, the existence and uniqueness of the solutions to (22.15) is guaranteed. Define  $\bar{x}(t) = x(t) - X\rho$ , and  $\bar{u} = u(t) - U\rho$ . Then, the closed-loop system with the controller (22.14) is

$$\dot{\bar{x}}(t) = (A - BK)\bar{x}(t) \quad (22.16a)$$

$$y(t) = C\bar{x}(t). \quad (22.16b)$$

Since  $\sigma(A - BK) \subseteq \mathbb{C}^-$ , we have  $\lim_{t \rightarrow \infty} \bar{x}(t) = 0$ , so  $\lim_{t \rightarrow \infty} y(t) = 0$ . Thus, the proof is completed.  $\square$

Notice that the controller gain  $K$  is non-unique, therefore providing a degree of freedom to optimize a performance index as shown next.

### 22.3.3 Model-Based Optimal Control Design

The purpose of this section is to find an optimal feedback controller  $K^*$  that satisfies the condition in Theorem 22.1 and that is a minimizer of a suitable cost criterion during the shared driving process. To this end, we introduce the following optimal control problem known as the linear quadratic regulator (LQR) problem:

$$\begin{aligned} & \underset{\bar{u}}{\text{minimize}} \quad \int_0^\infty (\bar{x}^T(t)Q\bar{x}(t) + r\bar{u}^2(t)) dt \\ & \text{subject to} \quad \dot{\bar{x}}(t) = A\bar{x}(t) + B\bar{u}(t) \\ & \qquad \qquad \qquad y(t) = C\bar{x}(t), \end{aligned}$$

where  $Q = Q^T > 0$  with  $(A, Q^{\frac{1}{2}})$  observable and  $r > 0$  is a constant. By linear optimal control theory from [27, Theorem 3.4-2], the gain  $K^*$  of the optimal controller  $\bar{u}(t) = -K^*\bar{x}(t)$  is determined by  $K^* = \frac{1}{r}B^TP^*$ , where  $P^* = P^{*T} > 0$  is the unique solution to the following algebraic Riccati equation:

$$A^T P + PA - \frac{1}{r}PBB^TP + Q = 0. \quad (22.17)$$

Next, we recall the following model-based policy iteration (PI) algorithm from [24] to solve (22.17).

**Lemma 22.4** Let  $P_j$ , for  $j = 0, 1, \dots$ , be the unique positive definite solution of

$$0 = (A - BK_j)^T P_j + P_j(A - BK_j) + Q + rK_j^T K_j, \quad (22.18)$$

where

$$K_{j+1} = \frac{1}{r} B^T P_j, \quad (22.19)$$

and where  $K_0$  is any matrix such that  $\sigma(A - BK_0) \subseteq \mathbb{C}^-$ . Then, the following properties hold for the sequences  $\{P_j\}_{j=0}^\infty$  and  $\{K_j\}_{j=1}^\infty$ , where  $K_0$  satisfies the requirement from Theorem 22.1:

1.  $\sigma(A - BK_j) \subseteq \mathbb{C}^-$ ;
2.  $P^* \leq P_{j+1} \leq P_j$ ;
3.  $\lim_{j \rightarrow \infty} K_j = K^*$  and  $\lim_{j \rightarrow \infty} P_j = P^*$ .

The system matrices  $A$  and  $B$  depend on the human driver parameters in (22.4) and the vehicle parameters in (22.1), which are often not precisely known. As a consequence, the optimal gain  $K^*$  and sub-optimal gains  $K_j$  are not implementable in practice. Thus, it is desired to overcome this obstacle by developing a practically implementable algorithm that does not rely on the knowledge of these parameters, which we do in the next section.

## 22.4 Learning-Based Optimal Control for Cooperative Driving

In this section, we develop a data-driven approach to obtain an estimate of the unknown optimal control gain  $K^*$ , and to acquire the steady-state vectors  $X$  and  $U$  in (22.15), using an ADP method without assuming exact knowledge of the dynamics of the interconnected human–vehicle system.

Motivated by [11], we observe that  $X$  can be expressed by

$$X = Y^1 + \sum_{l=2}^{n_x} \alpha^l Y^l, \quad (22.20)$$

where  $Y^1 = 0_{n_x \times 1}$ , the  $\alpha^l$ 's are real values and  $\{Y^2, \dots, Y^{n_x}\}$  contains a basis for the null space of  $C$ .

By letting  $\bar{x}^l = x - Y^l \rho$ , the system (22.7) can be rewritten as follows:

$$\begin{aligned} \dot{\bar{x}}^l &= Ax + Bu + D\rho \\ &= A_j \bar{x}^l + B(K_j \bar{x}^l + u) + (D + AY^l)\rho, \end{aligned} \quad (22.21)$$

where  $A_j = A - BK_j$  and the  $K_j$ 's satisfy the requirement from Lemma 22.4. Then, for each  $\delta t > 0$ , we have

$$\begin{aligned}
& (\bar{x}^l)^T P_j \bar{x}^l \Big|_t^{t+\delta t} \\
&= \int_t^{t+\delta t} \left[ (\bar{x}^l)^T (A_j^T P_j + P_j A_j) \bar{x}^l + 2(K_j \bar{x}^l + u) B^T P_j \bar{x}^l + 2\rho(D + AY^l)^T P_j \bar{x}^l \right] d\tau \\
&= - \int_t^{t+\delta t} (\bar{x}^l)^T (Q + r K_j^T K_j) \bar{x}^l d\tau + 2 \int_t^{t+\delta t} r(u + K_j \bar{x}^l) K_{j+1} \bar{x}^l d\tau \\
&\quad + 2 \int_t^{t+\delta t} \rho(D + AY^l)^T P_j \bar{x}^l d\tau. \tag{22.22}
\end{aligned}$$

Based on the Kronecker representation, it follows that  $a^T W b = (b^T \otimes a^T) \text{vec}(W)$  and  $(a \otimes b)^T = a^T \otimes b^T$  for all vectors  $a$  and  $b$  and matrices  $W$  such that  $a^T W b$  is defined. For a symmetric matrix  $M$  with an appropriate dimension, we have  $a^T M a = (\text{vecv}(a))^T \text{vecs}(M)$ . Then, for a positive integer  $k \geq 1$ , the following matrices can be defined:

$$\begin{aligned}
\delta_{\bar{x}^l \bar{x}^l} &= [\text{vecv}(\bar{x}^l(t_1)) - \text{vecv}(\bar{x}^l(t_0)), \dots, \text{vecv}(\bar{x}^l(t_k)) - \text{vecv}(\bar{x}^l(t_{k-1}))]^T, \\
\Gamma_{\bar{x}^l \bar{x}^l} &= \left[ \int_{t_0}^{t_1} \bar{x}^l \otimes \bar{x}^l d\tau, \int_{t_1}^{t_2} \bar{x}^l \otimes \bar{x}^l d\tau, \dots, \int_{t_{k-1}}^{t_k} \bar{x}^l \otimes \bar{x}^l d\tau \right]^T, \\
\Gamma_{\bar{x}^l u} &= \left[ \int_{t_0}^{t_1} \bar{x}^l \otimes u d\tau, \int_{t_1}^{t_2} \bar{x}^l \otimes u d\tau, \dots, \int_{t_{k-1}}^{t_k} \bar{x}^l \otimes u d\tau \right]^T, \\
\Gamma_{\bar{x}^l \rho} &= \left[ \int_{t_0}^{t_1} \bar{x}^l \otimes \rho d\tau, \int_{t_1}^{t_2} \bar{x}^l \otimes \rho d\tau, \dots, \int_{t_{k-1}}^{t_k} \bar{x}^l \otimes \rho d\tau \right]^T,
\end{aligned}$$

where  $0 \leq t_0 < t_1 < t_2 \dots < t_{k-1} < t_k$ . For any of the  $K_j$ 's, Eq. (22.22) implies that

$$\Psi_j^l \begin{bmatrix} \text{vecs}(P_j) \\ \text{vec}(K_{j+1}) \\ \text{vec}((D + AY^l)^T P_j) \end{bmatrix} = \Phi_j^l, \tag{22.23}$$

where

$$\begin{aligned}
\Psi_j^l &= \left[ \delta_{\bar{x}^l \bar{x}^l}, -2\Gamma_{\bar{x}^l \bar{x}^l} \left( I_{n_x} \otimes K_j^T r \right) - 2r\Gamma_{\bar{x}^l u}, -2\Gamma_{\bar{x}^l \rho} \right], \\
\Phi_j^l &= -\Gamma_{\bar{x}^l \bar{x}^l} \text{vec}(Q + r K_j^T K_j),
\end{aligned}$$

and  $n_x$  is the dimension of  $x(t)$ . Note that we also employ the mixed-product property of Kronecker product, i.e., for matrices  $\mathcal{A}, \mathcal{B}, \mathcal{C}, \mathcal{D}$  with appropriate dimensions,  $(\mathcal{A} \otimes \mathcal{B})(\mathcal{C} \otimes \mathcal{D}) = (\mathcal{AC}) \otimes (\mathcal{BD})$ .

**Assumption 22.1** There exists a positive integer  $j^*$  such that for all  $j > j^*$ ,  $\Psi_j^l$  has full column rank, for  $l = 1, \dots, n_x$ .

Under Assumption 22.1, (22.23) can be solved by the standard least squares method, i.e.,

$$\begin{bmatrix} \text{vecs}(P_j) \\ \text{vec}(K_{j+1}) \\ \text{vec}((D + AY^l)^T P_j) \end{bmatrix} = [(\Psi_j^l)^T \Psi_j^l]^{-1} (\Psi_j^l)^T \Phi_j^l. \tag{22.24}$$

Note that  $D$  can be computed from (22.24), since  $Y^1 = 0$ . In addition,  $B$  can be determined by  $B = rP_j^{-1}K_{j+1}^T$  from (22.19) and  $AY^l$  can be determined for  $l = 2, \dots, n_x$ . Hence, from (22.15a),  $X$ ,  $U$  and scalars  $\alpha^l$  can be determined by

$$\mathcal{S}(Y) + BU = -D, \quad (22.25)$$

where  $\mathcal{S}(Y) = \sum_{l=2}^{n_x} \alpha^l AY^l$ . In particular, (22.20) and (22.25) can be written as

$$\begin{bmatrix} AY^2 & \cdots & AY^{n_x} & 0 & B \\ Y^2 & \cdots & Y^{n_x} & -I_{n_x} & 0 \end{bmatrix} \begin{bmatrix} \alpha_2 \\ \vdots \\ \alpha_{n_x} \\ X \\ U \end{bmatrix} = \begin{bmatrix} -D \\ 0 \end{bmatrix}. \quad (22.26)$$

Observe that (22.26) is equivalent to the output regulator Eq. (22.15), which has a unique solution for  $X$  and  $U$ .

The algorithm for designing the data-driven shared steering controller is given as follows, where the constant  $\gamma$  that is used as a criterion to stop the algorithm can be selected by the user. See Fig. 22.3 for an illustration.

---

**Algorithm 22.1 PI-Based ADP Algorithm**


---

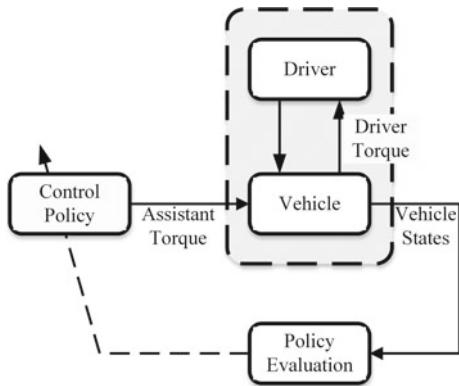
- 1: Select a  $K_0$  such that  $\sigma(A - BK_0) \subseteq \mathbb{C}^-$  and a sufficiently small threshold  $\gamma > 0$ .
  - 2: Let the driver control the vehicle with exploration noise i.e.,  $u(t) = \eta(t)$  on the time interval  $[t_0, t_k]$  to collect real-time data until Assumption 22.1 is satisfied. Compute  $\Psi_0^l$  and  $\Phi_0^l$  for  $l = 2, \dots, n_x$ . Let  $j \leftarrow 0$ .
  - 3: **repeat**
  - 4:     Determine  $P_j$  and  $K_{j+1}$  from (22.24);
  - 5:      $j \leftarrow j + 1$ ;
  - 6: **until**  $|P_j - P_{j-1}| < \gamma$  and  $j > 1$
  - 7:  $j^* \leftarrow j$ .
  - 8: Obtain  $X$  and  $U$  from (22.26).
  - 9: Update controller  $u(t) = -K_{j^*}x(t) + (K_{j^*}X + U)\rho$ .
- 

**Lemma 22.5** Under Assumption 22.1, the obtained sequences  $\{P_j\}_{j=0}^\infty$  and  $\{K_j\}_{j=1}^\infty$  from (22.24) satisfy

$$\lim_{j \rightarrow \infty} P_j = P^* \text{ and } \lim_{j \rightarrow \infty} K_j = K^*.$$

**Proof** Given any one of the  $K_j$ 's, let  $P_j = P_j^T$  be the unique solution to (22.18), and  $\Lambda_j = \text{vec}((D + AY^l)^T P_j)$ . Then, the updated controller  $K_{j+1}$  is determined by  $K_{j+1} = \frac{1}{r}B^T P_j$ . From (22.22), we know  $P_j$ ,  $K_{j+1}$  and  $\Lambda_j$  are the solutions to (22.24). In addition,  $P_j$ ,  $K_{j+1}$  and  $\Lambda_j$  are unique under Assumption 22.1. Hence, we conclude that solving (22.24) is equivalent to solving (22.18) and (22.19). Therefore,

**Fig. 22.3** Schematic illustration of an ADP-based control design for the interconnected human–vehicle system



by Lemma 22.4, the convergence of  $P_j$  and  $K_j$  follows immediately, so our data-driven algorithm can approximate an optimal control policy  $K^*$  that achieves desired lane-keeping performance.  $\square$

**Theorem 22.2** Consider the interconnected driver–vehicle system (22.7). Let the shared steering controller  $u(t) = -K_{j^*}x(t) + (K_{j^*}X + U)\rho$  be the result obtained from Algorithm 22.1. Then, the lane-keeping error converges to zero, i.e.,  $\lim_{t \rightarrow \infty} y(t) = 0$ .

**Proof** By Lemma 22.4, it follows that  $\sigma(A - BK_{j^*}) \subseteq \mathbb{C}^-$ . By applying the co-pilot controller  $K_{j^*}$  learned from real-time data, the closed-loop error dynamics satisfies

$$\dot{\bar{x}} = (A - BK_{j^*})\bar{x}. \quad (22.27)$$

Thus,  $\lim_{t \rightarrow \infty} \bar{x}(t) = 0$ . Based on Theorem 22.1, we conclude that  $\lim_{t \rightarrow \infty} y(t) = 0$ . The proof is completed.  $\square$

## 22.5 Numerical Results

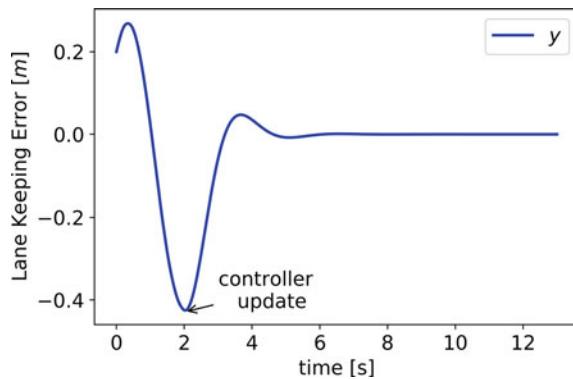
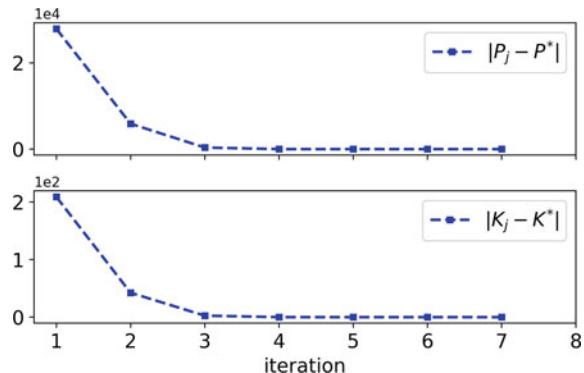
In this section, several simulation results are presented to show the efficacy of our ADP-based shared control algorithm. The longitudinal velocity  $v_x$  is fixed at 15 (m/s). Other numerical values for the simulation are shown in Table 22.1.

### 22.5.1 Algorithmic Implementation

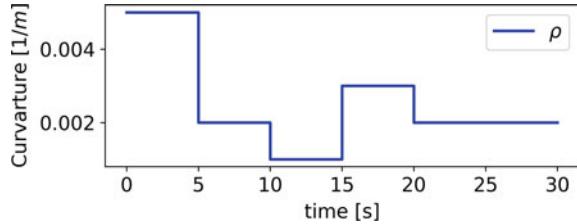
In the learning phase, the curvature is  $\rho = 0.005$ . We initiate Algorithm 22.1 to obtain the shared steering controller. The real-time data collection phase is from  $t = 0$  [s]

**Table 22.1** Numerical values in simulations

Parameters	Numerical values	Parameters	Numerical values
$l_f$	1.0065 (m)	$I_z$	2454 ( $\text{kg} \cdot \text{m}^2$ )
$l_r$	1.4625 (m)	$I_s$	0.05 ( $\text{kg} \cdot \text{m}^2$ )
$M$	1500 (kg)	$R_s$	16
$l_s$	5 (m)	$B_s$	5.73
$\eta_t$	0.185 (m)	$C_r$	56636 (N/rad)
$C_f$	47135 (N/rad)	$T_P$	0.3 (s)
$K_a$	20	$K_c$	80
$\tau_a$	0.5 (s)	$\gamma$	$10^{-5}$
$\tau$	0.25 (s)		

**Fig. 22.4** Lane-keeping error during the learning phase**Fig. 22.5** Convergence of  $P_j$ ,  $K_j$  to  $P^*$  and  $K^*$ , respectively

**Fig. 22.6** Road curvature profile



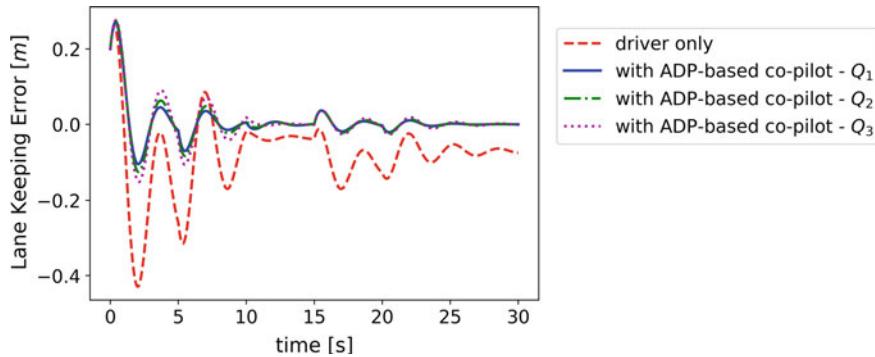
to  $t = 2$  [s] as shown in Fig. 22.4, during which the driver controls the vehicle and an exploration noise is applied. At time  $t = 2$  [s], Assumption 22.1 is satisfied. Then,  $P_j$  and  $K_{j+1}$  are computed, and their convergence to  $P^*$  and  $K^*$  is achieved after 7 iterations as presented in Fig. 22.5. After the learned feedback gain  $K_7$  is obtained,  $X$  and  $U$  can be determined immediately from (22.26). The controller is updated once  $X$  and  $U$  are computed. Then, we observe that the lane-keeping error converges to zero after the update.

### 22.5.2 Comparisons and Discussions for ADP-Based Shared Control Design

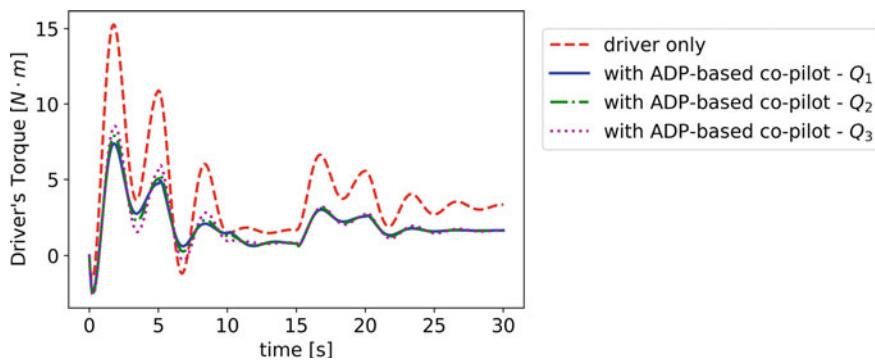
In this simulation, we evaluate the learned ADP-based shared controller on a non-constant road with curvature profile shown in Fig. 22.6. The velocity is fixed at 15 (m/s). We adopt different choices of the design parameters, i.e., the weighting matrices  $Q$  and  $r$ . In particular, we use the design parameters  $r = 1$ ,  $Q_1 = \text{diag}(1, 1, 1, 100, 1, 1, \dots, 1)$ ,  $Q_2 = \text{diag}(1, 1, 1, 50, 1, 1, \dots, 1)$ , and  $Q_3 = \text{diag}(1, 1, 1, 10, 1, 1, \dots, 1)$ .

Figure 22.7 shows the performance comparison of the lane-keeping error during the driving process with driver-only control (i.e., without co-pilot controller, meaning:  $u = 0$ ) and ADP-based shared control strategies. From Fig. 22.7, the improved lane-keeping performance is evident, where the overshoots are reduced greatly compared to the driver-only control. For  $Q_1$ ,  $Q_2$  and  $Q_3$ , the lane-keeping error is consistently smaller than the manual control situation, which shows the effectiveness of our ADP-based shared control framework.

Figure 22.8 presents the trajectories of the driver's torque during the process with and without the ADP-based co-pilot. It shows that the driver steering behavior changes due to the different selections for the design parameters. Notably, the ADP-based co-pilot using  $Q_3$  has the minimal influence on the driver's torque behavior while the co-pilot based on  $Q_1$  has an evident impact on the driver. All the ADP-based co-pilots reduce the workload of the driver, because the magnitude of the driver's torque decreases. Finally, Fig. 22.9 demonstrates the assistant torque applied by our ADP-based co-pilots with different  $Q$  values.

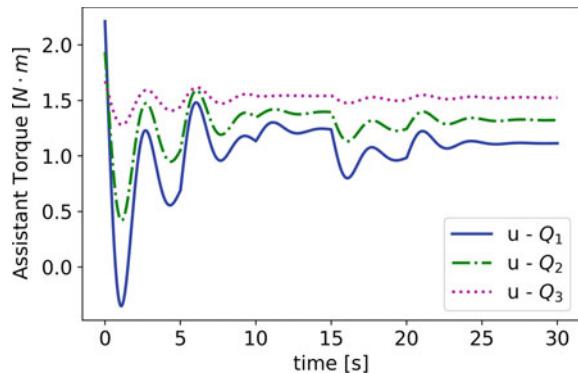


**Fig. 22.7** Performance comparison between driver-only manual control and ADP-based shared control



**Fig. 22.8** Driver's torque during the driving process

**Fig. 22.9** Assistant torque during the driving process



## 22.6 Conclusions and Future Work

We proposed a data-driven shared steering control design using an ADP-based algorithm, considering the presence of driver reaction delay. The shared steering problem is formulated as a control design for a set of differential-difference equations. In order to simplify the design procedure, a discretization technique is applied to transform the delay problem into a finite-dimensional control problem. An ADP-based control algorithm is then proposed to obtain the approximate optimal shared control policy for the interconnected driver–vehicle system, without exact knowledge of their dynamics. Our convergence proofs demonstrate the efficacy of our ADP-based shared control method. Our future work will be focused on data-driven control design for nonlinear driver and vehicle models, taking into account the driver reaction time. In addition, we aim to develop adaptive and learning-based control algorithms for multiple semi-autonomous vehicles to achieve more complex maneuvers.

**Acknowledgements** This work has been supported partly by the National Science Foundation under Grants DMS-2009644 and EPCN-1903781.

## References

1. Police-reported motor vehicle traffic crashes in 2017. Technical report, National Highway Traffic Safety Administration, 2019. Report No. DOT HS 812 696. <https://crashstats.nhtsa.dot.gov/Api/Public/ViewPublication/812696>
2. Abbink, D.A., Carlson, T., Mulder, M., de Winter, J.C.F., Aminravan, F., Gibo, T.L., Boer, E.R.: A topology of shared control systems-finding common ground in diversity. *IEEE Trans. Hum.-Mach. Syst.* **48**(5), 509–525 (2018)
3. Bellman, R.: On the existence and boundedness of solutions of non-linear differential-difference equations. *Ann. Math.* **50**(2), 347–355 (1949)
4. Bellman, R., Danskin, J.M.: A Survey of the Mathematical Theory of Time-Lag, Retarded Control, and Hereditary Processes. RAND Corporation, Santa Monica (1954)
5. Bertsekas, D.: Reinforcement Learning and Optimal Control. Athena Scientific, Belmont (2019)
6. Best, M.C.: Real-time characterisation of driver steering behaviour. *Veh. Syst. Dyn.* **57**(1), 64–85 (2019)
7. Bian, T., Jiang, Z.P.: Value iteration and adaptive dynamic programming for data-driven adaptive optimal control design. *Automatica* **71**, 348–360 (2016)
8. Chen, Y., Zhang, X., Wang, J.: Robust vehicle driver assistance control for handover scenarios considering driving performances. *IEEE Trans. Syst. Man Cybern.: Syst.* (2019). doi:<https://doi.org/10.1109/TSMC.2019.2931484>
9. Donges, E.: A two-level model of driver steering behavior. *Hum. Factor* **20**, 691–707 (1978)
10. Gao, W., Gao, J., Ozbay, K., Jiang, Z.P.: Reinforcement-learning-based cooperative adaptive cruise control of buses in the Lincoln tunnel corridor with time-varying topology. *IEEE Trans. Intell. Trans. Syst.* **20**(10), 3796–3805 (2019)
11. Gao, W., Jiang, Z.P.: Adaptive dynamic programming and adaptive optimal output regulation of linear systems. *IEEE Trans. Autom. Control* **61**(12), 4164–4169 (2016)
12. Gao, W., Jiang, Z.P.: Learning-based adaptive optimal tracking control of strict-feedback non-linear systems. *IEEE Trans. Neural Netw. Learn. Syst.* **29**(6), 2614–2624 (2018)

13. Hale, J.K., Lunel, S.M.V.: Effects of small delays on stability and control. In: Operator Theory and Analysis: Advances and Applications, vol. 122, pp. 275–301. Birkhäuser, Basel (2001)
14. Hespanha, J.P.: Linear Systems Theory. Princeton University Press, Princeton (2009)
15. HomChaudhuri, B., Pisu, P.: A control strategy for driver specific driver assistant system to improve fuel economy of connected vehicles in urban roads. In: Proceedings of American Control Conference, pp. 5557–5562 (2019)
16. Huang, J.: Nonlinear Output Regulation: Theory and Applications. SIAM, Philadelphia (2004)
17. Huang, M., Gao, W., Jiang, Z.P.: Connected cruise control with delayed feedback and disturbance: an adaptive dynamic programming approach. *Int. J. Adapt. Control Signal Process.* **33**, 356–370 (2019)
18. Huang, M., Gao, W., Wang, Y., Jiang, Z.P.: Data-driven shared steering control design for lane keeping. In: IFAC Conference on Modelling, Identification and Control of Nonlinear Systems, pp. 155–160 (2018)
19. Huang, M., Gao, W., Wang, Y., Jiang, Z.P.: Data-driven shared steering control of semi-autonomous vehicles. *IEEE Trans. Hum.-Mach. Syst.* **49**(4), 350–361 (2019)
20. Ji, X., Yang, K., Na, X., Lv, C., Liu, Y.: Shared steering torque control for lane change assistance: a stochastic game-theoretic approach. *IEEE Trans. Ind. Electron.* **66**(4), 3093–3105 (2019)
21. Jiang, J., Astolfi, A.: Shared-control for a rear-wheel drive car: dynamic environments and disturbance rejection. *IEEE Trans. Hum.-Mach. Syst.* **47**(5), 723–734 (2017)
22. Jiang, Y., Jiang, Z.P.: Robust Adaptive Dynamic Programming. Wiley-IEEE Press, Hoboken (2017)
23. Kiumarsi, B., Vamvoudakis, K.G., Modares, H., Lewis, F.L.: Optimal and autonomous control using reinforcement learning: a survey. *IEEE Trans. Neural Netw. Learn. Syst.* **29**(6), 2042–2062 (2018)
24. Kleinman, D.: On an iterative technique for Riccati equation computations. *IEEE Trans. Autom. Control* **13**(1), 114–115 (1968)
25. Land, M., Horwood, J.: Which parts of the road guide steering? *Nature* **377**, 339–340 (1995)
26. Lewis, F.L., Vrabie, D.: Reinforcement learning and adaptive dynamic programming for feedback control. *IEEE Circuits Syst. Mag.* **9**(3), 32–50 (2009)
27. Lewis, F.L., Vrabie, D.L., Syrmos, V.L.: Optimal Control. Wiley, Hoboken (2012)
28. Li, N., Oyler, D.W., Zhang, M., Yildiz, Y., Kolmanovsky, I., Girard, A.R.: Game theoretic modeling of driver and vehicle interactions for verification and validation of autonomous vehicle control systems. *IEEE Trans. Control Syst. Technol.* **26**(5), 1782–1797 (2018)
29. Lu, Y., Bi, L., Li, H.: Model predictive-based shared control for brain-controlled driving. *IEEE Trans. Intell. Trans. Syst.* (2019). <https://doi.org/10.1109/TITS.2019.2897356>
30. MATLAB. 9.6.0.1174912 (R2019a). The MathWorks Inc., Natick, Massachusetts (2019)
31. Nguyen, A., Sentouh, C., Popieul, J.: Sensor reduction for driver-automation shared steering control via an adaptive authority allocation strategy. *IEEE/ASME Trans. Mechatr.* **23**(1), 5–16 (2018)
32. Okamoto, K., Itti, L., Tsotras, P.: Vision-based autonomous path following using a human driver control model with reliable input-feature value estimation. *IEEE Trans. Intell. Veh.* **4**(3), 497–506 (2019)
33. Pang, B., Bian, T., Jiang, Z.P.: Adaptive dynamic programming for finite-horizon optimal control of linear time-varying discrete-time systems. *Control Theory Technol.* **17**(1), 73–84 (2019)
34. Rajamani, R.: Vehicle Dynamics and Control. Springer, Boston (2011)
35. Ross, D., Flügge-Lotz, I.: An optimal control problem for systems with differential-difference equation dynamics. *SIAM J. Control* **7**(4), 609–623 (1969)
36. Russell, H.E.B., Harbott, L.K., Nisky, I., Pan, S., Okamura, A.M., Gerdes, J.C.: Motor learning affects car-to-driver handover in automated vehicles. *Sci. Robot.* **1**(1) (2016). Article. no. eaah5682
37. Salvucci, D.D., Gray, R.: A two-point visual control model of steering. *Perception* **33**(10), 1233–1248 (2004)

38. Sentouh, C., Nguyen, A., Benloucif, M.A., Popieul, J.: Driver-automation cooperation oriented approach for shared control of lane keeping assist systems. *IEEE Trans. Control Syst. Technol.* **27**(5), 1962–1978 (2019)
39. Tsien, H.S.: *Engineering Cybernetics*. McGraw-Hill, New York (1954)
40. Wang, B., Abe, M., Kano, Y.: Influence of driver's reaction time and gain on driver-vehicle system performance with rear wheel steering control systems: part of a study on vehicle control suitable for the aged driver. *JSAE Rev.* **23**(1), 75–82 (2002)
41. Wang, D., He, H., Liu, D.: Adaptive critic nonlinear robust control: a survey. *IEEE Trans. Cybern.* **47**(10), 3429–3451 (2017)
42. Zhang, H., Zhao, W., Wang, J.: Fault-tolerant control for electric vehicles with independently driven in-wheel motors considering individual driver steering characteristics. *IEEE Trans. Veh. Technol.* **68**(5), 4527–4536 (2019)

# Chapter 23

## Decision-Making for Complex Systems Subjected to Uncertainties—A Probability Density Function Control Approach



Aiping Wang and Hong Wang

### 23.1 Introduction

Decision-making, or optimization, has been a subject of study for many years. For complex systems, such as transportation systems, manufacturing, and power grids, the subject is of particular important in the sense that optimization needs to be performed during both the system design and operation. In general, these complex systems are composed of many components (i.e., subsystems or agents) and these components work in a collaborative way so as to achieve a desired operational efficiency and product quality. This means that the optimization for the operation of these systems is particularly important in terms of achieving optimized operational effects. With the ever increased degree of uncertainties, operational optimization or decision-making is facing challenges in terms of realizing an optimal operation in the presence of uncertainties and how the impact of uncertainties onto the system operational quality can be minimized.

For example, in transportation system operation, since a number of vehicles on the road for any duration is random, control of such systems needs to ensure that traffic flows over the concerned area are made as smooth as possible with minimized energy consumption [11]. For power grid, it is important to perform optimization so that the system maintains its quality (frequency, voltage levels, and power flow balance) in the presence of uncertainties caused by changes in renewable energy sources and loads [5]. For manufacturing, product quality and process efficiency need to be optimized so as to realize an operational optimal status in the presence of

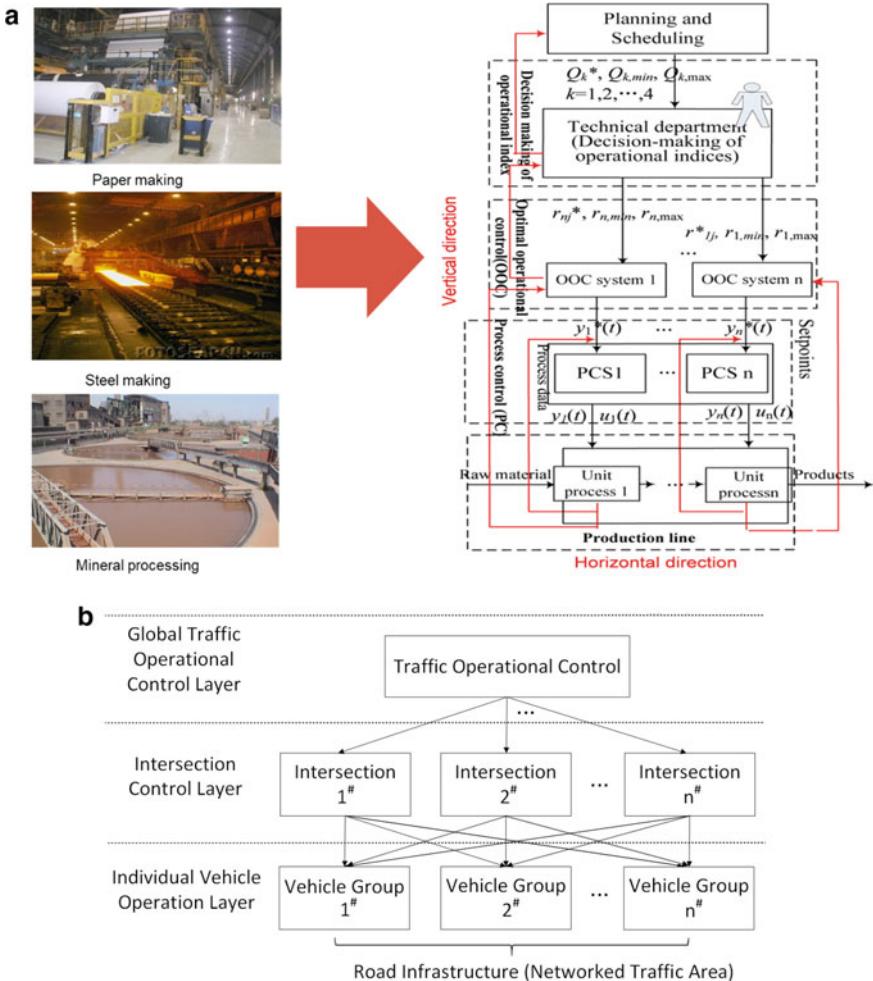
---

A. Wang  
Bozhou University and Anhui University, Anhui, China  
e-mail: [apwang401@126.com](mailto:apwang401@126.com)

H. Wang (✉)  
Oak Ridge National Laboratory, Oak Ridge, TN, US  
e-mail: [wangh6@ornl.gov](mailto:wangh6@ornl.gov)

uncertainties possibly caused by either the variations of raw material compositions or the unexpected changes of operational environment [1, 10]. This objective needs to be met even if when unexpected faults occur in the system, where fault diagnosis and tolerant controls are employed [8]. Once the system design is complete, a common set of operational features of these complex systems are as follows:

- (1) The system is of multiple scales, where the system operation is managed and controlled by a multiple layered structure (Fig. 23.1);
- (2) The system operation is both time and event driven;



**Fig. 23.1** (a) The multi-layered operational structure of complex industrial processes. (b) The multi-layered operational structure of transportation systems of urban areas

- (3) The system operation involves human operators at various layers, leading to human-in-the-loop scenarios where decision-making is performed by both human operators and control systems.

For example, the operation of manufacturing systems involves raw material supplies chain, material processing lines, and after-sales services. Examples are steel making, mineral processing, power systems, car manufacturing, papermaking, and petro-chemical plants. In the production phase there are a lot of control systems working collaboratively to fulfill the required production. These control systems have multiple layered interface with the on-site human operators at different levels. Indeed, the whole system works horizontally through material conversion and processing along production lines and vertically through the integration of different layers of operation control, planning and scheduling as well as operational management [1, 7]. Such a structure is shown in Fig. 23.1. During the operation of these manufacturing processes, the ultimate purpose is to achieve optimized product quality and production efficiency. This means that at least there are several operational performance indexes that need to be optimized at the same time, namely, the product quality indexes, runnability, and costs. This constitutes a multi-objective optimization problem. Indeed, it is well-known that once the production infrastructure structure is fixed, it is the integration of control systems and human operators that would play a vital role to realize the optimality of the operation. Assuming that the control loop layer has been well designed and realized by distributed control systems (DCSs), the final stage of decision variables would be a set of set-points applied to these control loops along the concerned production line. This presents two challenges, (1) how optimization can be performed so as to obtain an optimal set of set-points to control loops and (2) how the impact of non-zero tracking errors in control loops would affect reversely the optimality.

With the multiple layered structure as shown in Fig. 23.1, optimization would generally take place in each layer and at different time scales. For example, the optimization for the industrial process at the control loop layer would be realized in line with the closed-loop sampling rate of the concerned closed-loop systems using the knowledge of the process. On the other hand, the operational control layer would require certain degree of the involvement of human operators who help to fine tune the set-points to the control loops. Finally, the optimization at the planning and scheduling level would require a longer sampling time and involve significant assistance of human operators for the decision-making. Decision-making or optimization for complex systems subjected to uncertainties require us to present uncertainties as random variables and look at the system as a large-scale dynamic stochastic system. The scope is therefore in line with stochastic optimization, where most of the approaches nowadays aim at minimizing the mean value of the concerned objective function without considering the impact of uncertainties possibly caused by human operators in the decision-making phase. It is therefore important to address the following issues:

- (1) Development of generical modeling and optimization framework for complex systems subjected to uncertainties by making a full use of probability and

statistic measures for the formulation of optimization problems and their solutions;

- (2) Establishment of assessment tools that can be used to analyze the system optimality when a full realization of the obtained optimal decision cannot be achieved in the case that system operation is subjected to dynamic transient caused by control systems operation.

In this chapter, the modeling and system analysis tools together with optimal decision-making using probability density function (PDF) control theory (Wang, 1999) will be described. This summarizes what the authors have studied in the past years and includes system integration using  $\{\max, +\}$  algebra, learning and modeling using probability density function shaping, human-in-the-loop uncertainties modeling using brain-computer interface (BCI), and the square impact principle (SIP) that reveals the key mathematical representation for the integration of information technology (IT) with system infrastructure design and probability density function shaping-based optimization. Note that the materials come from some of our published papers and report (see the references lists).

## **23.2 Integrated Modeling Perspectives—Ordinary Algebra Versus $\{\text{Max}, +\}$ Algebra**

Optimization requires a good understanding of system behavior. This is achieved via effective modeling of the process to be optimized. In this context, the modeling should consist of the following:

- (1) Modeling of the objective functions that reveal the relationship between the objectives and the decision variables and other variables in the system;
- (2) Modeling of constraints which are the relationships between decision variables and system dynamics.

Since the system is subjected to uncertainties, both objective function models and constraint models are stochastic in nature and in particular constraints models are represented by stochastic differential equation under ordinary algebra sense. In this section, process level modeling that uses ordinary algebra bases such as differential equations and managerial level modeling that can use  $\{\max, +\}$  algebra will be described. In particular the integration of ordinary algebra with  $\{\max, +\}$  algebra for multi-scale modeling for both objective functions and constraints will be discussed, where the purpose of using  $\{\max, +\}$  algebra-based modeling is to simplify the mathematical representation of the system in particular for the planning and scheduling layers as seen in Fig. 23.1.

### 23.2.1 Process Level Modeling via Ordinary Algebra Systems

The purpose of process level modeling is to establish mathematical models that describe the dynamics at system component level. For example, for industrial processes in Fig. 23.1, the process level modeling means to model the dynamics of process units along the production line and for transportation systems, the process level modeling is to obtain models of vehicles movement in terms of speed and position profile. This level of modeling has been very much focused on modeling the state and output dynamics of the system components and the use of ordinary algebra has enabled good modeling exercises leading to some algebra and differential equations. Good examples are the state space models and *Ito* stochastic differential equation models that capture the dynamics of the system and the impact of random inputs and disturbances to the system [25], and thus lay foundations for optimal control and system optimization. In this context, if there are uncertainties presented in the system, stochastic differential equations should be generally used.

Let  $u(t) \in R^m$  be the control input or the vector composed a set of decision variables,  $x(t) \in R^n$  as the state vector and  $y(t) \in R^l$  as the measured output vector, then at the process level modeling the system dynamics can be generally expressed using ordinary algebra means in the following way

$$\begin{aligned} dx &= f(x, u)dt + g(x, u)dw, \\ y &= h(x, u) + v \end{aligned} \tag{23.1}$$

where  $w(t) \in R^n$  is a Brownian motion and  $v(t) \in R^l$  is a random process (noise or uncertainty),  $\{f, g, h\}$  are generic functions that represent the dynamics and characteristics of the system obtained using the first principles or data-driven approaches. When these functions are unknown, data-driven modeling or reinforced learning such as neural network-based learning can be used to obtain their features [12]. Sometimes combination of physical modeling with data-driven modeling needs to be used. This belongs to the gray-box modeling area and will not be discussed here.

In formulating the problem of optimization that uses  $u(t)$  as a decision vector, this equation is used as part of the constraints together with other constraints such as inequalities and probability constraints for the system state vector.

### 23.2.2 $\{\text{Max}, +\}$ Algebra-Based Modeling

At the upper level of the system such as the planning and scheduling layer for industrial processes in Fig. 23.1, there are a lot of nonlinear operations and *min-max* problem to be solved. This level of system can still be modeled using ordinary algebra concept. However, it would lead to complicated models that are difficult to be used for the system optimization. As such, one can use  $\{\max, +\}$  algebra as the basis

to formulate the models in a much simplified format. At least some linear models using  $\{\max, +\}$  operators are easier to handle than the nonlinear models with many *mini-max* operations.

Different from ordinary algebra, for any given real numbers  $a$  and  $b$ ,  $\{\max, +\}$  algebra defines algebraic summation and multiplication operations in the following way:

$$\begin{aligned} a \oplus b &= \max\{a, b\}, \\ a \otimes b &= a + b \end{aligned} \quad (23.2)$$

where  $\oplus$  is the summation operation in  $\{\max, +\}$  algebra and  $\otimes$  is a multiplication operator in  $\{\max, +\}$  algebra sense [15]. It can be seen that  $\{\max, +\}$  algebra absorbs the nonlinear operation at its basic two operators level and thus leads to a simple format of model representation for systems that involve heavily some nonlinear dynamics and operations.

Using  $\{\max, +\}$  algebra, the planning and scheduling layer can be generally represented in a discrete-time scenarios as follows [2, 19]:

$$x(k+1) = [(A \otimes x(k)) \oplus (B \otimes u(k))] \oplus w(k), \quad (23.3)$$

where  $k = 1, 2, \dots$  are the sampling numbers,  $x(k) \in R^n$  is the state vector,  $u(k) \in R^m$  is the decision vector,  $A$  and  $B$  are parameter matrices, respectively.

It can therefore be seen that the integration of the modeling would mean to combine (23.1) and (23.3) and use them both whenever necessary in the constraints modeling for the system optimization together with other types of constraints such as inequalities and probability constraints.

To conclude, it can be seen that  $\{\max, +\}$  algebra has “funny” operators that transfer nonlinear systems into possible linear ones at least in its format. As such, complex system modeling for manufacturing systems shown in Fig. 23.1 can therefore have the following three layers:

1. Enterprises level and system level decision-making with a combination of manufacturing execution systems (MES) and enterprise resources planning (EPR);
2. Planning and scheduling layer with  $\{\max, +\}$  algebra-based models as alternatives as given in Eq. (23.3);
3. Process layer modeling with stochastic state space model as shown in Eq. (23.1).

In terms of decision-making or optimization, denote  $J$  as the objective function vector with each of its components representing a single performance index item such as product quality or energy consumption, then a comprehensive formulation of optimization problem can be mathematically expressed as follows:

$$\begin{aligned}
& \min_{u_1, u_2} J(x_1, x_2, u_1, u_2, w_1, w_2) \\
& s.t. \\
& dx_1 = f(x_1, u_1)dt + g(x_1, u_1)dw_1 \text{ (Process level)} \\
& x_2(k+1) = [(A \otimes x_2(k)) \oplus (B \otimes u_2(k))] \oplus w_2(k) \text{ (Planning and scheduling)} \\
& M_1 \leq l(x_1, x_2, u_1, u_2) \leq M_2,
\end{aligned} \tag{23.4}$$

where  $l(\cdot)$  is a vector function that defines the inequality constraints and  $\{M_1, M_2\}$  are the lower and upper limits for the inequality constraints. In Eq. (23.4),  $\{x_1, x_2, u_1, u_2, w_1, w_2\}$  are sub-vectors of appropriate dimensions that represent the system state, input, and noise vectors. It can be seen that optimization problem (23.4) is a stochastic optimization problem, where the PDF of the performance function  $J$  should be ultimately optimized. This will be described in the following sections. In particular, the process level constraint is an *Ito* stochastic differential equation, its solution is the dynamical progression of the state vector PDF denoted by  $\gamma_{x_1}(\tau, u_1)$ , which can be obtained from the following Fokker–Planck equation [23].

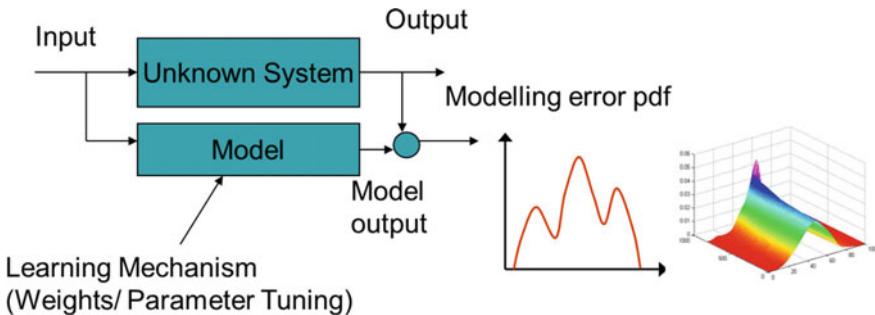
$$\begin{aligned}
\frac{\partial}{\partial t} \gamma_{x_1}(u_1, \tau) &= -\frac{\partial}{\partial \tau} [f(\tau, u_1) \gamma_{x_1}(u_1, \tau)] + \frac{\partial^2}{\partial \tau^2} [D(\tau, u_1) \gamma_{x_1}(u_1, \tau)] \\
D(\tau, u_1) &= \frac{1}{2} g^2(\tau, u_1).
\end{aligned}$$

By solving the above partial differential equation, the PDF of  $x_1(t)$  can be used to formulate the PDF of the performance function  $J$ . In this context, the process level constraint is in fact a partial differential equation that generates the dynamical evolution of the PDF of  $x_1(t)$ .

Indeed, this is a complicated optimization issue yet it gives a relative full picture on the scope and the complex of the problem. In addition, model uncertainties should also be considered in both the objective performance function  $J$  and the constraints in practice.

### 23.2.3 Learning Under Uncertainties–PDF Shaping of Modeling Error-Based Approach

When the system is subjected to uncertainties, its modeling or learning should be made adaptive in an online way, where data continuously collected from the system will be used to tune the models. This is achieved via data-driven modeling or reinforced learning using neural networks or other types of learning tools [12]. In general, for this type of modeling and learning exercises, the structure is shown in Fig. 23.2, where a model with parameters or neural network weights are connected in parallel



**Fig.23. 2** Modeling error PDF shaping based online learning and modeling

to the system. When the information on the input and output data are collected, they are used to tune model parameters (or neural network weights) so that the modeling error is minimized. In this case, the well-known sum-of-squared-error (SSE) has been used as a criterion for the modeling error. In general, if the SSE of the modeling error is small then one can conclude that a good model has been obtained.

However, SSE is a numerical measure of variance for the modeling error and this is a good measure if the modeling error is a Gaussian process. If the modeling error is non-Gaussian then the SSE would not generally represent a full picture of modeling error distribution. In this context, other criteria for modeling error should be considered. Indeed, for non-Gaussian modeling error, one can use either the entropy or the PDF of modeling error to represent its distribution, and the purpose of online parameters tuning is to either make the entropy of the modeling error as small as possible because minimized entropy would mean that the uncertainties and the randomness embedded in the modeling error are small, or alternatively, one can use PDF shaping of the modeling error technique to tune the model parameters. In this case, the idea is to tune the model parameters or neural network weights so that the modeling error PDF is made to follow a narrowly distributed and zero mean Gaussian PDF as shown in Fig. 23.2 [9]. This is because a narrowly distributed Gaussian PDF means that the modeling error has less randomness and uncertainties, and zero mean says that the modeling error is symmetrically distributed closed to the zero value.

For example, let us assume that the system to be learned is given by the following NARMAX form [6]

$$y(k+1) = f(y(k), y(k-1), \dots, u(k), u(k-1), \dots, u(k-m), w(k)), \quad (23.5)$$

where  $u(k)$  is the input sequence,  $y(k)$  is the output sequence, and  $w(k)$  is a random noise, and  $k (= 1, 2, \dots)$  is the sample numbers. It is assumed that the input and output are both measurable and can therefore be used to learn unknown system feature denoted by function  $f(\dots)$ . To simplify the following presentation, we have assumed that the system given in Eq. (23.5) is of a single-input and single-output system.

To learn the above system at the current sample number  $k$  using the input and output data, the following parametrical model is used

$$y_m(k+1) = g(\varphi(k), \theta) \quad (23.6)$$

$$\varphi(k) = [y(k), y(k-1), \dots, u(k), u(k-1), \dots, u(k-m)], \quad (23.7)$$

where  $g(\dots)$  is a known structured function and  $\theta$  is the tuning parameter vector or neural network weights. For example, in neural network-based learning,  $g(\dots)$  would be a structured multiple layered neural network and  $\theta$  groups all the weights and biases to be trained.

The objective of learning is to tune the model parameter vector  $\theta$  so that the modeling error defined as

$$e(k) = y(k) - y_m(k) \quad (23.8)$$

is made as small as possible. For this purpose, let the PDF of the modeling error is given by  $\gamma_e(\tau, \theta, k)$ , then for the modeling exercises the tuning of the model parameter vector  $\theta$  at sample time  $k$  can be obtained by solving the following optimization problem

$$\min_{\theta(k)} \left\{ \int_a^b [\gamma_e(\tau, \theta, k) - \delta(\tau)]^2 d\tau \right\}, \forall e(k) \in [a, b], \quad (23.9)$$

where  $\delta(\tau)$  is the well-known impulse function defined as follows:

$$\delta(\tau) = \begin{cases} +\infty, & \tau = 0 \\ 0, & \text{otherwise} \end{cases}$$

In Eq. (23.9),  $a$  and  $b$  are the lower and upper bounds of the estimated modeling error. The above integral reflects the distance measure between the modeling error PDF and the impulse function which is regarded as zero mean and zero variance Gaussian PDF.

In terms of minimized entropy modeling, the tuning of the model parameter vector should be obtained by solving the following optimization problem:

$$\min_{\theta(k)} \left\{ - \int_a^b \gamma_e(\tau, k) \log \gamma_e(\tau, k) d\tau + \left[ \int_a^b \tau \gamma_e(\tau, k) d\tau \right]^2 \right\}, \quad (23.10)$$

where the first term is modeling error entropy and the second term is the mean value of the modeling error.

By transferring learning and modeling into the above optimization problem, the obtained models are much reliable and general, and the modeling error performance covers the SSE-based learning as when the modeling error obeys a Gaussian PDF, solving optimization problems in (23.9) or (23.10) is equivalent to solving the modeling and learning using SSE criteria.

Modeling using modeling error PDF shaping can therefore be combined with any reinforced learning algorithms or neural network training to form an alternative yet comprehensive modeling approach applicable to practical systems [12, 16, 33]. This method provides a generic tool in dealing with non-Gaussian systems modeling and therefore would provide an effective solution as potentially a key component for artificial intelligence. Of course, solutions to the optimization problems (23.9) and (23.10) are not always easy and it requires a good estimate of the range of the modeling error before it can be used. This means that model structure denoted by function  $g(\dots)$  in Eq. (23.6) needs to be carefully selected and sometimes compromises between the complexity of the model structure vs the speed of convergence of parameter tuning is required.

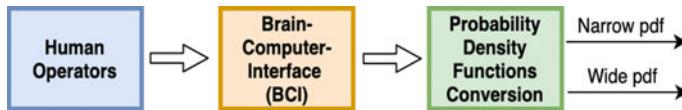
In addition, the above method is described for single-input and single-output systems. For multi-inputs and multi-outputs systems, the modeling error is a vector, and the joint modeling error PDF among each component of the modeling error vector should be shaped in selecting the model parameter tuning rules. This can increase the computational load dramatically and sometimes high performance computing has to be used to learn complex systems online.

### 23.3 Human-in-the-Loop Consideration: Impact of Uncertainties in Decision-Making Phase

As shown in Fig. 23.1, human operators are involved in various layers in the system during its operation. They participate in decision-making and at the same time present uncertainties in the decision-making phase. This presents new challenges for the optimization where human-in-the-loop aspects need to be considered, and in particular the effect of uncertainties of human operators in the decision-making phase needs to be quantitatively analyzed. This would allow us to assess the optimality effect and the related robustness of the optimal operation of the concerned complex systems when they are subjected to unexpected variations in both process level operation and in human operators decision-making phase.

For manufacturing processes, it is therefore imperative to study how the decision-making of human operator is integrated with the actual process information so as to optimize production performance in terms of enhanced product quality and reduced costs (e.g., raw materials and energy usage, and effluent discharge).

At present, the plant level operational control and management have been largely solved [10], where a group of manipulated (decision) process variables can be generally formulated from the models of the process to optimize the process operation



**Fig. 23.3** Obtaining PDFs of EEG signals to distinguish experienced and non-experienced operators

together with the decisions made by the human operators represented as rule-based reasoning, etc. In general, the decision-making phase of human operators contains uncertainties in the time-horizon in terms of observation, use of their knowledge, and decision realization. Since these uncertainties will affect the effect of the decision-making, there is a challenging issue here on how the negative effect of these uncertainties on the optimization can be minimized. This constitutes one of the key issues yet to be solved for the knowledge automation for industrial processes and other complex systems control, which is a new concept nowadays following the launch of industrial 4.0 in Germany.

A preliminary study by Hong Wang (Zhang, Wang, Wang and Wu, 2013) has shown that the uncertainties of human operator's decision-making can be somehow measured through electroencephalogram (EEG) (i.e., brain signals) via brain-computer interface (BCI) as shown in Fig. 23.3, where the brain signals of inexperienced operators would generally exhibit large portion of uncertainties and randomness which lead to a widely distributed probability density functions. On the other hand, the EEG signals of experienced operators would lead to narrowly distributed probability density functions. It is therefore imperative that such a measure be used by well-defined process variables in the minimization of the impact of uncertainties embedded in human operator's decision-making. This forms the main objective of the described study in this section. In particular, our study has over the past years used some methods from different disciplines including techniques from BCI, EEG signal processing, knowledge engineering, data mining, and decision science [17, 18, 26] to look into the following aspects in a logical order:

- (1) Investigate the ways to quantify knowledge quality and randomness using signals collected from the brain of on-site human operators via EEG;
- (2) Develop a method on performance function modeling that can describe the relationships between uncertainties of human decision variables and the well-defined process variables so as to form a comprehensive set-up of optimization problem.
- (3) Explore the framework that can be used to realize the risk-minimization of human decision-making by adjusting a group of well-defined process variables.

The first aspect has been achieved where a set of BCI-based information has been collected and analyzed in order to define the uncertainties treatment to be described in the following section, by using the equivalence between optimization and feedback closed-loop control systems design. The second aspect has also been completed where a novel stochastic optimization solution is obtained that uses probability density function as a means to quantify the randomness of the knowledge

extraction in the optimization phase and shows how such a PDF can be used to design optimization algorithms. These will be described in detail in the following sections.

In specific, a novel sufficient condition has been derived that guarantees that the decision variable sequence would converge to a crisp value in solving the stochastic optimization problem that is subjected to both decision-making uncertainties and process disturbances.

## 23.4 Optimization Under Uncertainties Impacts

To describe the formulation and solution for decision-making or optimization for systems subjected to uncertainties, a general framework will be discussed here so as to accommodate and present the uncertainties injections from both the process and the human operators into the right part of the closed loop system. As it has been described, these uncertainties come from two channels, namely, the uncertainties in the process to be optimized and the uncertainties caused by the human-in-the-loop during decision-making phase. In this section, we will first describe a framework that shows the equivalence between closed-loop feedback control system design and optimization. Then the points of uncertainties injections to the optimization problem will be given. This leads to the formulation of the relevant stochastic optimization problem, where PDF shaping-based technique will be used. In the following sub-sections simplified optimization problem formulation and PDF shaping-based optimization will be described. The materials come from our report in 2016 [32].

### 23.4.1 *Formulation of Optimization as a Feedback Control Design Problem—Optimization is a Special Case of Feedback Control System Design*

Optimization problems stated in (23.4) has been a subject of study for many years and is widely required in many areas. In general, a simple optimization can be formulated as solving the following problem:

$$\min_u J(u) \quad (23.11)$$

$$s.t. f(u) = 0, \quad (23.12)$$

where  $J$  in Eq. (23.11) is a performance function to be optimized and  $u$  is a vector that groups a set of decision variables to realize such an optimization. In the optimization  $u$  should also satisfy the constraints defined by the second Eq. (23.12).

Without the loss of generality, it is assumed that  $J$  is always positive (for example, it represents the energy cost of the concerned system operation). In this context, the purpose of the optimization can be interpreted as to select an optimal vector of  $u$  so that  $J$  is made as close as possible to zero. Since in many cases the actual optimization algorithm is realized in a form of recursion for  $u$ , if we denote  $u_k$  as the value of  $u$  at sample number  $k$ , then the following recursive optimization algorithm is generally used

$$u_k = \pi(u_{k-1}, J(u_{k-1}), \theta), \quad (23.13)$$

where  $\pi(\cdot, \cdot, \cdot)$  is a functional operator that represents the designed optimization algorithm and  $\theta$  is a group of learning rates. For example, when there is no constraint and  $J$  is differentiable and is defined on a compact set, the well-known gradient descent algorithm that minimizes  $J$  recursively is given by

$$u_k = u_{k-1} - \theta \frac{\partial J(u_{k-1})}{\partial u_{k-1}} \quad k = 1, 2, \dots \quad (23.14)$$

This means that the functional operator  $\pi(\dots)$  of the selected optimization algorithm (23.14) is given by

$$\pi(u_{k-1}, J(u_{k-1}), \theta) = u_{k-1} - \theta \frac{\partial J(u_{k-1})}{\partial u}. \quad (23.15)$$

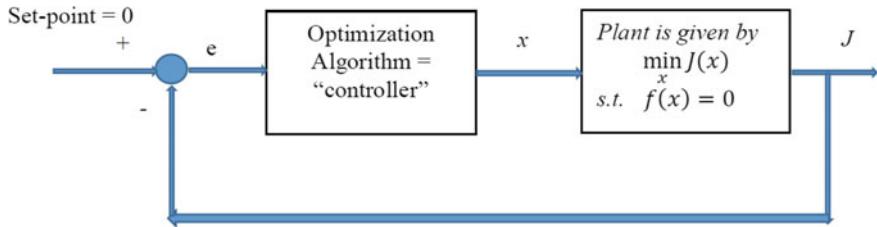
Taking the optimization algorithm as a “control” algorithm, this means that to optimize  $J$  subjected to the required constraints one needs to design a “controller” that can ensure *ideally* that the following tracking error converges to zero.

$$\lim_{k \rightarrow +\infty} e_k = \lim_{k \rightarrow +\infty} [0 - J(u_k)] = 0. \quad (23.16)$$

By taking  $J$  as the output of the “plant” to be controlled and  $u$  as the control input applied to the “plant”, the following “closed-loop control system” in Fig. 23.4 can be readily obtained to represent the optimization procedure [32].

Since  $J$  is always positive and the set-point to the above closed-loop control system is always zero, the optimization can be regarded as a special case of the feedback control system design, where the controller design (i.e., the optimization algorithm) is to realize a good tracking of  $J$  with respect to a zero set-point. This reveals a very important fact which says that any control design method can be considered as a possible candidate in the design of an optimization algorithm. This observation has therefore enlarged the scope of the optimization and at least the following facts are true:

- (1) Multi-objective optimization can be treated as a multi-input and multi-output (MIMO) feedback controller design problem;



**Fig. 23.4** Treating the optimization as a closed-loop feedback control problem

(2) The rather rich stability and robustness analysis tools (such as Lyapunov stability theory and small gain theory) developed in control theory can be directly employed to analyze the performance of the concerned optimization in terms of its optimality and robustness.

For example, the convergence and dynamic performance of the tracking error given in Eq. (23.16) can be readily assessed using Lyapunov stability theory or Popov’s absolute stability theory.

Indeed, the above observations have enlarged the flexibility of selecting and developing optimization algorithms for a given problem and can enormously help for structured optimization, where the algorithm structure can be imposed before the beginning of the optimization. For example, one can select a PID-based optimization algorithm to first close the loop for a given optimization problem as shown in Fig. 23.4, then the PID gains can be regarded as a group of “learning rates” to be selected to guarantee the stability (convergence) of the optimization algorithm and the best tracking performance for the tracking error in Eq. (23.16).

For the gradient descent algorithm in (23.15), if we denote  $z^{-1}$  as the unit delay operator as in the discrete-time control system design, then the functionality of the “controller” is given by

$$u_k = \frac{-\theta}{1 - z^{-1}} \left[ \frac{\partial J(u_{k-1})}{\partial u_{k-1}} \right].$$

If a PI structure is used with the performance function as a direct feedback signal, we have the following solution to the optimization problem:

$$u_k = K_p(0 - J(u_{k-1})) + K_I \rho_k$$

$$\rho_k = \rho_{k-1} + h(0 - J(u_{k-1})),$$

where  $h > 0$  is the sampling length, and  $\{K_p, K_I\} = \theta$  are the “PI” gains to be selected so that the closed-loop control system shown in Fig. 23.4 is stable and the closed-loop system has a good tracking performance albeit  $J$  does not always approach the zero set-point. We can also control the gradient by using  $\frac{\partial J(u_{k-1})}{\partial u_{k-1}}$  as the

feedback signal to the “controller” with a PI structure. In this case, the following optimization algorithm can be formulated:

$$u_k = K_p \left( 0 - \frac{\partial J(u_{k-1})}{\partial u_{k-1}} \right) + K_I \rho_k$$

$$\rho_k = \rho_{k-1} + h \left( 0 - \frac{\partial J(u_{k-1})}{\partial u_{k-1}} \right).$$

In comparison with the direct performance function feedback, the above gradient feedback aims at achieving a zero gradient when  $k \rightarrow +\infty$ . Since in general  $J(u)$  is a static nonlinear function and the optimization algorithm is in a recursive form, the search for a proper optimization algorithm would mean that one needs to select a dynamic “controller” that can control a static nonlinear plant well. This has further indicated that optimization is a special case of closed-loop control system design.

### 23.4.1.1 Source of Uncertainties in Decision-Making Phase

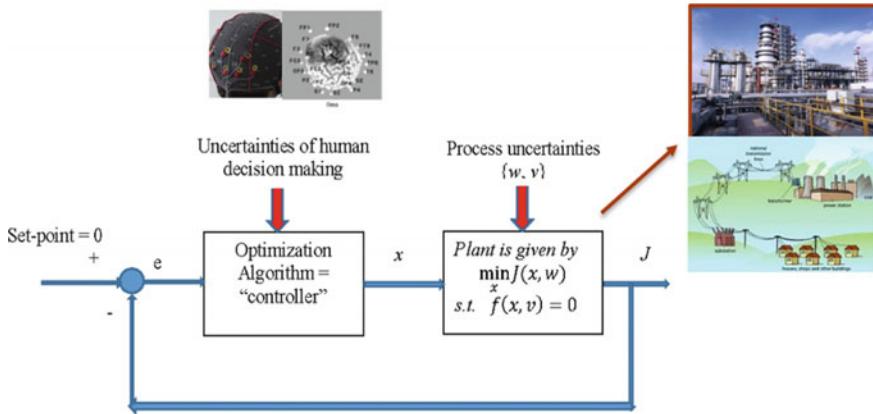
For complex dynamical processes, human operators at different layers in Fig. 23.1 generally participate in the optimization or decision making. For example, in the operation control layer the operators can be involved in the tuning of the set-points to the control loops in an intuitive way using their operational experience. In this decision-making phase there are some uncertainties involved. For example, an experienced operator would exhibit less uncertainties in his/her mind when making the decision. Such uncertainties can be represented as a disturbance to the “controller” in Fig. 23.4. Of course the “plant” is generally subjected to some degree of uncertainties in terms of the uncertainties embedded in both the performance function and the constraints. Indeed, the effect of uncertainties to the plant has been well studied and is generally treated as a robust optimization problem, where the objective of the optimization is to select a proper set of decision variables  $u$  so that the performance index  $J$  is minimized when it is subjected to uncertainties.

In this context, the realistic optimization problem can be further represented as in Fig. 23.5, where two types of uncertainties—one to the “controller” and the other to the “plant” are included. The optimization problem can therefore be generalized to solve the following problem:

$$\min_u J(u, w) \quad (23.17)$$

$$\text{s.t. } f(u, v) = 0, \quad (23.18)$$

where  $w$  and  $v$  are the uncertainties embedded in the performance function and the constraints and are assumed to be independent random processes whose probability density functions are used to represent their characteristics (see Fig. 23.5). The



**Fig. 23.5** The optimization framework that incorporates uncertainties from both the process and the human decision-making phase

purpose of the optimization is therefore to select a good  $u$  so that  $J$  is made to follow the zero set-point as close as possible in the presence of random noises  $w$  and  $v$ , and the uncertainties in the “controller” as characterized by the PDF of the BCI signals described in Sect. 23.3.

In comparison with standard controller design, the uncertainties embedded in the controller are a new phenomenon as in general for the closed-loop control design and implementation there are no uncertainties for the controller. This presents a new challenge where uncertainties such as the randomness of human decision-making need to be considered in the optimization phase. Indeed, when the constraint is given by

$$f(u, v) = f(u + v) = 0$$

then  $v$  would represent the uncertainties and randomness in the human decision-making phase whose PDFs can be obtained using EEG signal as shown in Fig. 23.3 in Sect. 23.3.

Since the uncertainties in the decision-making phase of human operators can be characterized using the signals from BCI, they can be quantitatively represented as a set of random processes which can be eventually expressed by the PDFs of the BCI signals after the subtraction of the signals themselves from their mean values. In this context, the generalized optimization structure in Fig. 23.5 can be obtained which presents a framework of optimization when it is subjected to uncertainties from both the process and the optimization phase.

To summarize, this framework reveals that the actual task of the optimization should be to perform a robust control design problem that obtains a “good”  $u$  so that  $J$  is made to be as close as possible to zero when the closed-loop system is subjected to uncertainties both in the “controller” and in the “plant”.

## 23.5 A Generalized Framework for Decision-Making Using PDF Shaping Approach

Since the process performance function  $J$  is subjected to the uncertainties as represented by  $w$  in Eqs. (23.17), (23.18),  $J$  is a random process and its optimization should be performed so that the mean value of  $J$  is minimized while the randomness of the optimized  $J$  is minimized. This is a PDF shaping problem and the stochastic distribution control theory originated by Wang, 1999; [14, 28] can be directly applied so that the PDF of  $J$  is moved as close as possible to the left and its spread area is made as narrow as possible.

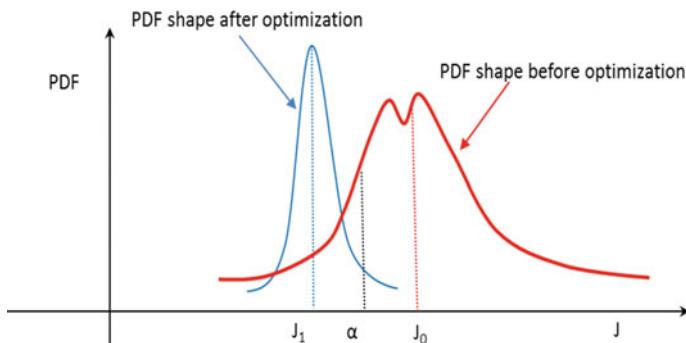
This is shown in Fig. 23.6 and forms the main contents to be described in this section. The analysis of optimality effect will be given in Sect. 23.6.

### 23.5.1 PDF Shaping for the Performance Function

Denote the PDF of  $J$  as  $\gamma_J(u, \tau)$  where  $\tau \in [a, b]$  is the definition variable that defines  $\gamma_J(u, \tau)$ ,  $a$  and  $b$  are two positive known numbers that stand for the definition interval of this PDF. Then, to realize the above purpose of optimization given in Eqs. (23.17) and (23.18), we need to minimize the following induced functional distance (Wang 1999)

$$\pi_0(u) = \|\delta(\tau - a) - \gamma_J(u, \tau)\| = \sqrt{\int_a^b [\delta(\tau - a) - \gamma_J(u, \tau)]^2 d\tau} = \min \quad (23.19)$$

subjected to the constraint expressed in (23.18), where  $\delta(\tau - a)$  is an impulse PDF function defined as follows:



**Fig. 23.6** PDFs of the performance function before and after the optimization

$$\delta(\tau - a) = \begin{cases} +\infty, & \tau = a \\ 0, & \text{otherwise} \end{cases}. \quad (23.20)$$

In fact, this  $\delta$ -function can be regarded as a specialized Gaussian PDF with its mean at point  $a$  and with zero variance. Without loss of generality, one can just minimize the following subjected to constraint (23.18):

$$\pi(u) = \int_a^b [\delta(\tau - a) - \gamma_J(u, \tau)]^2 d\tau = \min. \quad (23.21)$$

This presents a much generalized approach for stochastic optimization and the existing rather rich results on mean value and variance-based stochastic optimization become a special case of solving (23.21).

However, to use this approach the PDF of the performance function is required. This can be carried out in several ways. For example, if the PDF of the uncertainty  $w$  in (23.17) is known and denoted as  $\gamma_w(\tau)$  and function  $J(u, w)$  is monotonic with respect to  $w$ , then using the PDF formulation of a function of a random variable in probability theory, the PDF of the performance index  $J$  can be readily obtained as follows:

$$\gamma_J(u, \tau) = \gamma_w(J^{-1}(u, \tau)) \left| \frac{d}{d\tau} (J^{-1}(u, \tau)) \right|,$$

where  $J^{-1}$  is the inverse function of  $J$  with respect to  $w$ .

When the performance function  $J$  is not monotonic or is a vector function representing multiple objective optimization, a similar formulation for its PDF can also been carried out [4].

### 23.5.2 Dealing with the Constraint

While solving (23.19) or (23.21) is a straightforward stochastic distribution control problem (Wang, 1999), dealing with the constraint  $f(u, v) = 0$  at the same time is not easy as  $v$  is a random process. Satisfying such a constraint would mean that  $u$  should be a random process applied to the actual process. Here we will try to transfer this constraint into another PDF shaping problem albeit in general we can use the mean value of the optimized  $u$  as the actual decision variable to be applied to the process (see the next subsection).

For this purpose, we denote the PDF of  $f(u, v)$  as  $\gamma_f(u, \varphi)$ , where  $\varphi \in [c, d]$  is a definition variable for  $\gamma_f(u, \varphi)$  with known  $c < 0$  and  $d > 0$ , then to realize  $f(u, v) = 0$ , we need to make sure that in PDF sense the following

$$\gamma_f(u, \varphi) = \delta(\varphi), \forall \varphi \in [c, d] \quad (23.22)$$

is satisfied for the selected  $u$ . This means that the PDF of the constraint function should be equal to an impulse function as expressed in Sect. 23.2.3. Again, in line with the PDF shaping for the performance function, this can be realized “somehow” by selecting  $u$  that minimizes the following functional distance

$$\varepsilon_0(u) = \left| \left| \delta(\varphi) - \gamma_f(u, \varphi) \right| \right| = \sqrt{\int_c^d [\delta(\varphi) - \gamma_f(u, \varphi)]^2 d\varphi} \quad (23.23)$$

or to select  $u$  that simply minimizes the following:

$$\varepsilon(u) = \int_c^d [\delta(\varphi) - \gamma_f(u, \varphi)]^2 d\varphi.$$

As a result, the actual optimization becomes finding  $u$  so that both  $\pi(u)$  and  $\varepsilon(u)$  defined in the above equations are minimized at least and at the same time. Let  $u_k (k = 1, 2, \dots)$  be the sequence of the decision variable at sample number  $k$ , then to obtain the optimal  $u_{opt}$  that optimizes  $J$  subjected to  $f(u, v) = 0$ , we need to select  $u_k$  so that the following performance function is minimized:

$$J_\sigma(u) = \pi(u_k) + \sum_{j=1}^k \varepsilon(u_j), \quad k = 1, 2, 3, \dots \quad (23.24)$$

If the above can be minimized (i.e., a necessary condition) for  $k$  going to infinite, the constraint  $f(u, v) = 0$  can be strictly guaranteed. This is because when  $k$  goes to infinite we have

$$\sum_{j=1}^{+\infty} \varepsilon(u_j) < +\infty \quad (23.25)$$

which indicates that

$$\lim_{k \rightarrow +\infty} \varepsilon(u_k) = \lim_{k \rightarrow +\infty} \int_c^d [\delta(\varphi) - \gamma_f(u_k, \varphi)]^2 d\varphi = 0 \quad (23.26)$$

and

$$\lim_{k \rightarrow +\infty} \gamma_f(u_k, \varphi) = \delta(\varphi), \quad \forall \varphi \in [c, d]. \quad (23.27)$$

As a result, the optimal decision variable that minimizes  $J_\sigma(u)$  in Eq. (23.24) can be made to converge to its true optimal value as follows:

$$\lim_{k \rightarrow +\infty} u_k = u_{opt}. \quad (23.28)$$

This leads to the following theorem that states a necessary condition for solving the stochastic optimization problem given in (23.17), (23.18).

**Theorem 23.1** (The Discrete-time Case) *A necessary condition for solving the optimization problem (23.17), (23.18) is that there is a sequence of decision variable  $\{u_k\}$  for  $k = 1, 2, 3, \dots$  so that the following inequality holds:*

$$\sum_{k=1}^{+\infty} \varepsilon(u_k) < +\infty,$$

where it has been denoted that

$$\varepsilon(u_k) = \int_c^d [\delta(\varphi) - \gamma_f(u_k, \varphi)]^2 d\varphi.$$

When  $\Delta u = u_k - u_{k-1}$  is very small, a continuous-time format of Theorem 23.1 can also be obtained for minimizing Eq. (23.24) to give

$$J_\sigma(u) = \pi(u_t) + \int_0^t \int_c^d [\delta(\varphi) - \gamma_f(u(\tau), \varphi)]^2 d\varphi d\tau. \quad (23.29)$$

In this case we need to find a profile  $u(\tau)$  for  $\tau \in [0, t]$  that can minimize (23.29) for any  $t > 0$ . This means that we need to solve the following:

$$\min_{u(\tau), \tau \in [0, t]} \left\{ \pi(u_t) + \int_0^t \int_c^d [\delta(\varphi) - \gamma_f(u(\tau), \varphi)]^2 d\varphi d\tau \right\} \forall t \in [0, +\infty).$$

This is an induced optimization problem for solving (23.17), (23.18). If it is solvable for any  $t > 0$ , then it can be concluded that

$$\int_0^t \int_c^d [\delta(\varphi) - \gamma_f(u(\tau), \varphi)]^2 d\varphi d\tau < +\infty, \text{ for } \forall t > 0$$

which indicates that

$$\lim_{t \rightarrow +\infty} \gamma_f(u(t), \varphi) = \delta(\varphi), \text{ for } \forall \varphi \in [c, d]$$

$$\lim_{t \rightarrow +\infty} u(t) = u_{opt}. \quad (23.30)$$

Along with the above formulation, the following theorem can also be readily obtained for the continuous-time case.

**Theorem 23.2** (The Continuous-time Case) *A necessary condition for solving the optimization problem (23.17), (23.18) is that there is a function  $u(\tau)$  for  $\tau \in [0, t]$*

so that the following inequality holds:

$$\int_0^t \int_c^d [\delta(\varphi) - \gamma_f(u(\tau), \varphi)]^2 d\varphi d\tau < +\infty, \text{ for } \forall t > 0.$$

### 23.5.3 Dealing with Dynamic Constraint

Very often the solution to the following optimal control problem is needed for control systems research

$$\begin{aligned} \min_{\{u_1, u_2, \dots, u_N\}} J &= \sum_{k=1}^N L(x_k, u_k) \\ s.t. x_{k+1} &= f(x_k, u_k, w_k), \end{aligned} \quad (23.31)$$

where  $x_k \in R^n$  is the measurable state vector,  $u_k \in R^m$  is the control input vector,  $L(\dots)$  is the term that constitutes performance index, and  $f(\dots)$  is a known nonlinear vector function representing the system dynamics,  $w_k$  is a random process that represents either the disturbances or the uncertainties to the system and  $N$  is an integer that stands for the time-horizon for the optimization to take place.

The purpose is to select the optimal control input sequence  $\{u_1, u_2, \dots, u_N\}$  so that the above minimization can be realized. Since the dynamics of the system is subjected to a random process  $\{w_k\}$ , the state vector is also a random process. Therefore, the performance function  $J$  is a random variable and is related to the control input sequence. This is in line with the problem set-up of Eqs. (23.17) and (23.18). To solve the above problem using PDF shaping-based approaches, one needs to formulate the probability density function of  $J(u_1, u_2, \dots, u_N | X_N)$  for a given observation sequence

$$X_N = \{x_0, x_1, \dots, x_N\}.$$

Assuming that the PDF of the noise sequence  $\{w_k\}$  is known and is denoted as  $\gamma_w(\tau)$ , then this can be achieved by using the following procedure:

- Using the constraint equation in (23.31) and PDF  $\gamma_w(\tau)$  to obtain the joint PDF of  $X_N$  per denoted as  $\gamma_X(\tau)$ ,
- Using the obtained joint PDF  $\gamma_X(\tau)$  together with the characteristics of function  $L(\dots)$  to obtain the PDF of  $J$ .

This allows us to use the PDF shaping-based optimization to perform the required optimization task as described in this section. Of course, one can also use Monte

Carlo simulation to obtain the PDF of the constraint Eq. (23.31) and then follow the above procedure to formulate the PDF of  $J$ .

### 23.5.4 A Total Probabilistic Solution

Since constraint (23.18) indicates that in general  $u$  is also a random variable, another way to solve (23.17), (23.18) is to find an optimal probability density function for the decision variable  $u$  so that the probability density function of the performance function  $J$  is made as left and as narrow as possible as shown in Fig. 23.6.

Denote the optimal probability density function of  $u$  as  $\gamma_u(\tau)$ ,  $\tau \in [e, f]$  for a known  $e$  and  $f$ , then with this PDF one can find a value of  $u$  to apply to the system. For example, the optimal and applicable  $u_{opt}$  can be calculated from

$$u_{opt} \approx \int_e^f \tau \gamma_u(\tau) d\tau.$$

Since such an approximated approach requires the optimal solution of the PDF of  $u$ , it is referred to as the total probabilistic solution for (23.17), (23.18). Note that equation in the above is only an approximation because in practice  $\gamma_u(\tau)$  cannot be directly applied to the system. This constitutes a marked disadvantage of this type of solution.

#### 23.5.4.1 Relations to Chance Constrained Optimization

The PDF shaping-based solution to problem (23.17), (23.18) can also be used to solve chance constrained optimization problem [13, 36] studied in optimal power flow design for power grid in the sense that we need to find an optimal  $u$  for the following problem:

$$\min_u J(u, w) \quad (23.32)$$

$$s.t. \text{Prob}\{f(u, v) = 0\} \geq 1 - \beta \quad (23.33)$$

where  $\beta \approx 0$  is a very small and pre-specified positive number. In the existing chance constrained optimization, the above can be further relaxed into the following problem:

$$\min_u E_w\{J(u, w)\} \quad (23.34)$$

$$s.t. \text{Prob}\{f(u, v) \leq 0\} \geq 1 - \beta = \alpha, \quad (23.35)$$

where  $E_w\{\cdot\}$  is the mathematical expectation operator over random variable  $w$ . Indeed, this is a standard chance constrained optimization problem where the mean of the performance function is minimized. In PDF shaping sense, this means again that we need to find a good  $u$  so that the PDF of  $J$  is moved as left and as narrow as possible together with the PDF of  $f(u, v)$  satisfying constraints (23.35). As a result, the formulation and necessary conditions stated in Theorems 23.1 and 23.2 can all be applied to seek the solutions to the above chance constrained optimization problem.

Of course, in this context the rather rich algorithms developed in stochastic distribution control (Wang, 1999; [14, 21, 28, 31] can again be used to solve chance constrained optimization. For example, using the B-spline expression for  $\gamma_f(u, \varphi)$ , we have

$$\gamma_f(u, \varphi) = \sum_{i=1}^N \theta_i(u) B_i(\varphi), \forall \varphi \in [c, d], \quad (23.36)$$

where  $B_i(\varphi)$ ,  $\{i = 1, 2, \dots, N\}$  are the set of pre-specified basis functions defined on a known interval  $[c, d]$ , and  $\theta_i(u)$ ,  $\{i = 1, 2, \dots, N\}$  are B-spline expansion weights that are related to  $u$ . Then the probability constraint (23.33) or (23.35) is equivalent to

$$\int_c^0 \gamma_f(u, \varphi) d\varphi = \sum_{i=1}^N \theta_i(u) \int_c^0 B_i(\varphi) d\varphi \geq \alpha. \quad (23.37)$$

If we denote  $\gamma_w(\tau)$  as the PDF of  $w$ , where without loss of generality it has assumed that  $\tau \in [a, b]$ , then it can be obtained that

$$E_w\{J(u, w)\} = \int_a^b J(u, \tau) \gamma_w(\tau) d\tau = J_1(u). \quad (23.38)$$

In this case the chance constrained optimization (23.32), (23.33) can be transferred into the solution to the following deterministic nonlinear optimization problem:

$$\min_u J_1(u) \quad (23.39)$$

$$s.t. \sum_{i=1}^N \theta_i(u) \int_c^0 B_i(\varphi) d\varphi \geq \alpha. \quad (23.40)$$

As a result, PDF shaping-based approach for stochastic optimization provides a generalized framework that takes any existing stochastic optimization approaches as a special case.

### 23.5.5 Uncertainties in Performance Function and Constraints

Optimization problem (23.17), (23.18) assumes perfect models in performance function  $J$  and the constraint,  $f(u, v) = 0$ . This can be restrictive for some practical cases when both of these two functions are subjected to model uncertainties. In this case, what should actually be solved is the following optimization problem:

$$\min_u \{J(u, w) + \Delta J\} \quad (23.41)$$

$$s.t. f(u, v) + \Delta f = 0, \quad (23.42)$$

where  $\{\Delta J, \Delta f\}$  represent the model uncertainties embedded in the performance function and constraint models, respectively. Without loss of generality, it is assumed that these uncertainties are bounded by known real numbers as in the following inequalities:

$$M_1 \leq \Delta J \leq M_2 \quad (23.43)$$

$$M_3 \leq \Delta f \leq M_4, \quad (23.44)$$

where  $M_i$  ( $i = 1, 2, 3, 4$ ) are pre-specified. In this case, the optimization can still be solved using PDF shaping-based approach as described in this section. Using the information of the lower and upper bounds in Eqs. (23.43) and (23.44), we can obtain the relevant boundary PDFs for both performance function  $J$  and constraint model  $f(u, v)$  as follows using the PDF of these two functions obtained in Sects. 23.5.1 and 23.5.2 as denoted by  $\gamma_J(u, \tau)$  and  $\gamma_f(u, \tau)$ . This means that  $\gamma_J(u, \tau - M_1)$  is the PDF of lower boundary performance function  $J(u, w) + M_1$  and  $\gamma_J(u, \tau - M_2)$  represents the PDF of  $J(u, w) + M_2$ . As a result, in dealing with the PDF shaping of the performance function, we need to select the decision variable  $u$  so that both  $\gamma_J(u, \tau - M_1)$  and  $\gamma_J(u, \tau - M_2)$  are made as left and as narrow as possible. This indicates that one needs to minimize the following index

$$\pi_\Delta(u) = \int_{a-M_1}^{b+M_2} \{[\delta(\tau - a + M_1) - \gamma_J(u, \tau - M_1)]^2 + [\delta(\tau - a + M_1) - \gamma_J(u, \tau - M_2)]^2\} d\tau. \quad (23.45)$$

To deal with the constraints, we need to introduce the following instant-time index:

$$\varepsilon_\Delta(u) = \int_{-\infty}^{+\infty} \{[\delta(\varphi) - \gamma_f(u, \varphi - M_3)]^2 + [\delta(\varphi) - \gamma_f(u, \varphi - M_4)]^2\} d\varphi, \quad (23.46)$$

where  $\gamma_f(u, \varphi - M_3)$  is the PDF of  $f(u, v) + M_3$ , and  $\gamma_f(u, \varphi - M_4)$  is the PDF of  $f(u, v) + M_4$ , respectively.

Using the same formulation as in Sects. 23.5.1 and 23.5.2, in the discrete-time format we need to select the decision variable sequence  $\{u_k\}$  so that the following deterministic optimization is achieved

$$\min_{\{u_k\}} \left\{ \pi_\Delta(u_k) + \sum_{k=1}^K \varepsilon_\Delta(u_j) \right\}. \quad (23.47)$$

This clearly solves the problem as stated in Eqs. (23.41) and (23.42). In fact, solution to Eq. (23.47) constitutes a robust optimization effect in terms of probability density function shaping (Wang, 1999). If such a solution exists for any  $k \rightarrow +\infty$ , then the interval constraint represented in Eq. (23.42) can be automatically realized as it has been described in Sect 23.5.2.

## 23.6 System Analysis: Square Impact Principle as a Mathematical Principle for Integrated IT with Infrastructure Design

For the manufacturing system shown in Fig. 23.1, it is a well-known fact that once the structure of the production operation is determined, the status of the plant-wide operation and ultimately the optimality are determined by a number of controlled variables (i.e., the variables available to control and manage the process operation). These variables are controlled through control loops via distributed control systems (DCSs) and the control performance would have a significant impact on the whole production operation. Therefore, it can be seen that these controlled variables play unique role in realizing plant-wide optimization and improved sustainability as they are the only group of decision variables that can be optimized.

For industrial processes where trustable mathematical models can be established such as chemical processes, model-based operational optimization and control can be readily applied. In this context, real-time optimization (RTO) uses traditional feedback strategy to form the required control methods for optimal operation. Such methods would select the set-points for the controlled variables that correspond to an economically optimal steady state for industrial processes. By adjusting relevant set-points of controlled variables and ensuring that the controlled variables can follow their set-points, the whole process can be made to operate near the economically optimal steady state [1, 7].

However, since there are various tracking errors in the loop control level due to consistent variations of the process dynamics during its operation, the actual optimality is affected by the tracking error of the control loops. This means that in reality there are two issues that need to be considered, namely,

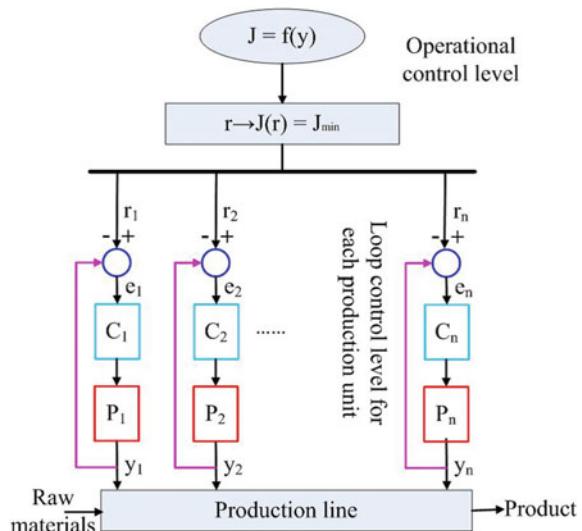
- (1) The solution to optimization problem given in Eqs. (23.17), (23.18) need to be obtained;
- (2) Once the optimal decision variables are obtained, what would happen if their realization in practice cannot be accurately made due to the inaccurate tracking of the control systems output with respect to these optimal decision variables (i.e., optimal set-points)?

This means that once the optimal set-points are obtained, the actual optimality of the process operation shown in Fig. 23.1 needs to be quantitatively analyzed. This forms the main purpose of this section, where through some simple mathematical formulations a novel principle, namely, the square impact principle (SIP), will be described that shows the fact that the tracking errors at the loop control level affect the optimality of the processes in a square amplitude way. In this section, the SIP developed by the authors will be described and the materials come from our published work in [30].

### 23.6.1 Description of Operational Optimal Control

To simply the representation for the system shown in Fig. 23.1, let us consider only a two-layered structure of the plant-wide operational control system as shown in Fig. 23.7, where a performance function  $J$  (say energy) is required to be minimized. Such a performance would be a function of actual controlled variables composed vector denoted as  $y = y(k) = [y_1(k), y_2(k), \dots, y_n(k)]^T \in R^n$  to give

**Fig. 23.7** A simple two-layered operational control scheme



$$J = f(y) = f(y_1, y_2, \dots, y_n), \quad (23.48)$$

where  $f(\dots)$  is a known nonlinear function and is assumed a differentiable function of controlled variable vector group in  $y$ . Each controlled variable is controlled in a closed-loop way and these closed-loop control systems constitute loop control layer as shown in Fig. 23.7 [34].

To minimize  $J$ , a set of optimal set-points  $r_i(k)(i = 1, 2, \dots, n)$  can be obtained so that at the optimal status we have

$$\frac{\partial J}{\partial y} = \left[ \frac{\partial f(y)}{\partial y} \right]_{y=r} = 0, \quad (23.49)$$

where  $r = r(k) = [r_1(k), r_2(k), \dots, r_n(k)]^T \in R^n$  are the optimal set-point vector for the controlled variable vector  $y$  to follow. This means that the minimized performance function satisfies

$$J_{min} = f(r(k)). \quad (23.50)$$

Since the above operational performance function is in fact functions of controlled variables of involved production equipment and units rather than their set-points which can only be practically selected to optimize the performance functions, it is necessary to analyze actual optimality of operational optimal control when there are nonzero tracking errors between the controlled variables grouped in  $y$  and their optimal set-points in  $r$ . In particular, it is necessary to explore how the loop tracking errors and disturbances would deteriorate the optimality of the performance function  $J$  that has been optimized using the above simple method.

### 23.6.2 Square Impact Principle (SIP): Infrastructure Versus Control Performance

To analyze the impact of the tracking errors and uncertainties to the optimality, let us consider the discretized-time case and define the tracking error of each control loop in Fig. 23.7 as

$$e_i(k) = r_i(k) - y_i(k), (i = 1, 2, \dots, n)$$

$$e(k) = [e_1(k), e_2(k), \dots, e_n(k)]^T \in R^n. \quad (23.51)$$

Then the performance function can be expressed in terms of the tracking error vector as

$$J = J(y(k)) = f(y(k) - r(k) + r(k)) = f(r(k) - e(k)). \quad (23.52)$$

Assuming that these errors are reasonably small (i.e.,  $\|e\| \ll 1$ ), then the linearization of the performance function around the optimal set-points  $r$  is ready to be expressed as

$$J(y(k)) = J(r) + \left[ \frac{\partial f}{\partial y} \right]_{y=r} e(k) + \frac{1}{2} e^T(k) \left[ \frac{\partial^2 f}{\partial y^2} \right]_{y=r} e(k). \quad (23.53)$$

Since the set-point is optimal, the second term in the above equation equals to zero because at the optimal set-point we have Eq. (23.49). This indicates that the performance function in Eq. (23.53) can be well approximated as

$$J(y(k)) = J_{min} + \frac{1}{2} e^T(k) \left[ \frac{\partial^2 f}{\partial y^2} \right]_{y=r} e(k). \quad (23.54)$$

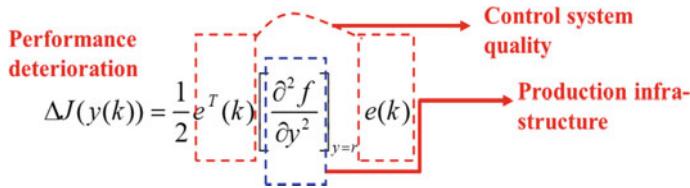
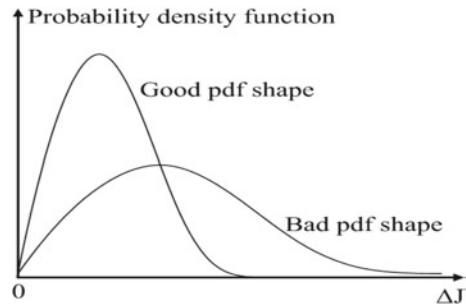
This equation reveals the fact that if the tracking error is not zero (or the output of the control loops are not tracking their set-points accurately), then the deterioration of the performance function is proportional to the square of the tracking error. This forms the square impact principle and the above equation can be further diluted into the following form:

$$\Delta J(y(k)) = J(y(k)) - J_{min} \sim O(\|e(k)\|^2), \quad (23.55)$$

where  $\Delta J$  is the performance deterioration caused by the inaccurate tracking of  $y(k)$  with respect to its set-point  $r(k)$ .

From the above analysis it can be seen that there are two groups of factors that can affect or deteriorate the optimality. Apart from the loop tracking errors that affect the optimality of the performance function in a square magnitude way, the infrastructure of the production system, which is related to  $\left[ \frac{\partial^2 f}{\partial y^2} \right]_{y=r}$ , will also affect the performance deterioration. This is simply because the way that the performance function is related to controlled variables is determined by the production infrastructure. This can be clearly shown by examining term  $\left[ \frac{\partial^2 f}{\partial y^2} \right]_{y=r}$  which reveals how  $y$  affects  $f$  once the production infrastructure design is completed. In this context, term  $\frac{1}{2} e^T(k) \left[ \frac{\partial^2 f}{\partial y^2} \right]_{y=r} e(k)$  says how the control quality, which is related to IT strategy in terms of control, communication, and computing, is coupled with the system infrastructure as shown in Fig. 23.8. Therefore, this relationship reveals the need for the integrated design of processes with control systems. From the infrastructure design perspective, we would hope that the term given by  $\left[ \frac{\partial^2 f}{\partial y^2} \right]_{y=r}$  is made as small as possible near the optimal point of the system operation.

As the system is also subjected to expected uncertainties and randomness, the performance deterioration  $\Delta J(y(k))$  in Eq. (23.55) is also affected by random inputs.

**Fig. 23.8** Impacts on performance deterioration**Fig. 23.9** The probability density function of performance deterioration

This indicates that  $\Delta J$  is a random variable. Therefore, to have a minimized degree of deterioration, we need to select the decision variables (i.e., set-points) so that the probability density function of  $\Delta J$  is made as left and as narrow as possible.

This is again in line with the probability density function shaping-based optimization described in Sect. 23.5 and is illustrated in Fig. 23.9.

It can be seen that the minimization of performance deterioration from its optimized point constitutes an extra layer of stochastic optimization,  $\Delta J$  evaluated at optimal set-point  $r(t)$  is an added performance function that needs to be considered in the optimization phase. This means that in general, multi-objective stochastic optimization should be considered even if the original performance function for the concerned system is a scale function. Moreover, optimizing the performance function together with its deterioration around its optimal points would provide an effective optimal solution that is robust with respect to unexpected variations of the process operation.

To minimize the deterioration of the performance function, sometimes the fine tuning of the optimal set-point needs to be made. This has been detailed in our paper [30].

## 23.7 Conclusions

This chapter has summarized the work that the authors carried out over the past years on the decision-making and optimization for systems subjected to uncertainties, where a number of issues have been addressed. In particular, the challenges on the modeling, human-in-the-loop, equivalence between feedback control design and optimization, PDF shaping-based optimization, and optimality analysis have been given. In terms of modeling, both process level modeling and planning and scheduling level modeling have been described. It has been shown that for high-level modeling  $\{\max, +\}$  algebra-based modeling can provide a simple format in formulating optimization problem. As for the human-in-the-loop cases, BCI generated signals can be used to model the level of maturity of the on-site operators in their decision-making phase and can therefore provide a quantitative measure of the uncertainties embedded in the optimization phase.

With the help of the unified framework that says that the optimization is a special case for general feedback control system design, the uncertainties caused by human-in-the-loop decision-making can be regarded as disturbances to the controller. This leads to a generic formulation of stochastic optimization problem where both the performance functions and the constraints are represented as stochastic equations as given in Eqs. (23.17), (23.18). Using the PDF shaping-based approach, the stochastic optimization has been transferred into the controls of PDF shape for the performance functions, and the purpose of optimization is to obtain optimal decision variables so that the PDF of the performance functions are moved as left and as narrow as possible in their definition domain. It has also been shown that the well-studied chance constrained optimization becomes a special case for the PDF shaping-based stochastic optimization.

The uncertainties considered in this chapter are represented by a set of random variables injected to the system as a group of inputs in both performance and constraint equations. In some cases, these uncertainties also present in other forms such as model uncertainties, etc. In this case, robust optimization methods [3] should be combined and integrated with the PDF shaping-based optimization. Using the equivalence between optimization and closed-loop feedback control design in Sect. 23.4.1, robustness of the optimization can be analyzed using well-developed tools in robust control design theory. This leads to reliable optimization effect applicable to an even wider range of complex systems. This belongs to one of the activities for the study in the future. Also, in some cases uncertainties cannot be measured by PDFs as probability is just a *Lebesgue* measure [24]. When the uncertainties cannot be measured in probability sense, other measures need to be considered for the measure in the performance function and the constraints.

**Acknowledgements** Part of materials presented here reflect the work of the second author when he was a chair professor in the University of Manchester (UK) before he moved to USA in 2016. The support from the University of Manchester is therefore gratefully acknowledged. In addition, this manuscript has been co-authored by UT-Battelle, LLC under Contract No. DE-AC05-00OR22725 with the U.S. Department of Energy. The United States Government retains and the publisher, by

accepting the article for publication, acknowledges that the United States Government retains a nonexclusive, paid-up, irrevocable, world-wide license to publish or reproduce the published form of this manuscript, or allow others to do so, for United States Government purposes. The Department of Energy will provide public access to these results of federally sponsored research in accordance with the DOE Public Access Plan (<https://energy.gov/downloads/doe-publicaccess-plan>).

## References

1. Adetola, V., Guay, M.: Integration of real-time optimization and model predictive control. *J. Process Control* **20**, 125–133 (2010)
2. Afshar, P., Wang, H.: Multiobjective meta-heuristic product scheduling for multi-machine manufacturing systems. In: Proceedings of the 50th IEEE Conference on Decision and Control and European Control Conference (CDC-ECC), pp. 1405–1410 (2011)
3. Ben-Tal, A., El Ghaoui, L., Nemirovsk, A.: Robust Optimization. Princeton Series in Applied Mathematics. Princeton University Press, Princeton, pp. 9–16 (2009)
4. Billingsley, P.: Probability and Measure. Wiley, New York, Toronto, London (1979)
5. Blaabjerg, F., Teodorescu, R., Liserre, M., Timbus, A.V.: Overview of control and grid synchronization for distributed power generation systems. *IEEE Trans. Ind. Elect.* **53**, 1398–1409 (2006)
6. Chen, S., Billings, S.A.: Representations of non-linear systems: the NARMAX model. *Int. J. Control* **49**, 1013–1032 (1989)
7. Darby, M.L., Nikolaou, M., Jones, J., Nicholson, D.: RTO: An overview and assessment of current practice. *J. Process Control* **21**, 874–884 (2011)
8. Davoodi, M., Meskin, N., Khorasani, K.: Integrated fault diagnosis and control design of linear complex systems. *IET Digital Library* (2018)
9. Ding, J., Chai, T.Y., Wang, H.: Off-line modelling for product quality prediction of mineral processing using modelling error PDF shaping and entropy minimization. *IEEE Trans. Neural Networks* **22**, 408–419 (2011)
10. Engell, S.: Feedback control for optimal process operation. *J. Process Control* **17**, 203–219 (2007)
11. Ezell, S.: Intelligent transportation systems. In: The Information Technology & Innovation Foundation (ITIF), Washington, DC, USA (2010)
12. Lewis, F.L., Liu, D., (eds.): Reinforcement learning and approximate dynamic programming for feedback control. Computational Intelligence Series. John Wiley/IEEE Press, New York
13. Geletu, A., Li, P.: Recent developments in computational approaches to optimization under uncertainty and application in process systems engineering. *Chem. Bio. Eng. Rev.* **1**, 170–190 (2014)
14. Guo, L., Wang, H.: Stochastic distribution control systems design: a convex optimization approach. Springer, London (2010)
15. Heidergott, B., Olsder, G.J., van der Woude, J.: Max plus at work. Princeton University Press, Princeton, NJ (2006)
16. Kaelbling, L.P., Littman, M.L., Moore, A.W.: Reinforcement learning: a survey. *J. Artif. Intell. Res.* **4**, 237–285 (1996)
17. Kehoe, B., Patil, S., Abbeel, P., Goldberg, K.: A survey of research on cloud robotics and automation. *IEEE Trans. Autom. Sci. Eng.: Spec. Issue Cloud Robot. Autom.* **12**, 1–11 (2015)
18. Mason, S.G., Birch, G.E.: A brain-controlled switch for asynchronous control applications. *IEEE Trans. Biomed. Eng.* **47**, 1297–1307 (2000)
19. McEneaney, W.: Max-plus methods for nonlinear control and estimation. Birkhäuser, Berlin (2006)

20. Liu, Y., Wang, H., Guo, L.: Observer-based feedback controller design for a class of stochastic systems with non-Gaussian variables. *IEEE Trans. Autom. Control* **60**, 1445–1450 (2014)
21. Ren, M.F., Zhang, J.H., Wang, H.: Non-Gaussian system control and filtering. Chinese Science Publishers, Beijing (2016)
22. Ren, M.F., Zhang, J.H.: Wang H (2013) Minimized tracking error randomness control for nonlinear multivariate and non-Gaussian systems using the generalized density evolution equation. *IEEE Trans. Autom. Control* **59**, 2486–2490 (2014)
23. Risken, H.: The Fokker–Planck Equation: Methods of Solutions and Applications, 2nd edn. Springer Series in Synergetics. Springer, Berlin (1989)
24. Royden, H.L.: Real analysis, 3rd edn. Macmillan, New York (1988)
25. Slavík, A.: Generalized differential equations: differentiability of solutions with respect to initial conditions and parameters. *J. Math. Anal. Appl.* **402**(1), 261–274 (2013)
26. Wairagkar, M., Daly, I., Hayashi, Y., Nasuto, S.J.: Autocorrelation based EEG dynamics depicting motor intention. In: Proceedings of the 6th International Brain-Computer Interface Meeting, Organized by the BCI Society (2016)
27. Wang, H.: Bounded dynamic stochastic distributions—modelling and control. Springer, London (1999a)
28. Wang, H., Afshar, P.: ILC-based fixed-structure controller design for output PDF shaping in stochastic systems using LMI techniques. *IEEE Trans. Autom. Control* **54**, 760–773 (2009)
29. Wang, H.: Robust control of the output probability density functions for multivariable stochastic systems with guaranteed stability. *IEEE Trans. Autom. Control* **41**, 2103–2107 (1999b)
30. Wang, A.P., Zhou, P., Wang, H.: Performance analysis for operational optimal control for complex industrial processes under small loop control errors. In: Proceedings of the 2014 International Conference on Advanced Mechatronic Systems, vol. 1, pp. 159–164 (2014)
31. Wang, H., Zhang, J.H.: Bounded stochastic distribution control for pseudo ARMAX systems. *IEEE Trans. Autom. Control* **46**, 486–490 (2001)
32. Wang, H., Wang., S.B., Fan, R., Zhang, Z.F.: Minimizing uncertainties impact in decision making with an applicability study for economic power dispatch. PNNL Report. [www.pnnl.gov](http://www.pnnl.gov) (2016)
33. Williams, R.J.: A class of gradient-estimating algorithms for reinforcement learning in neural networks. In: Proceedings of the IEEE First International Conference on Neural Networks, San Diego, CA (1987)
34. Yin, L., Wang, H., Guo, L., Zhang, H.: Data-driven pareto-DE-based intelligent optimal operational control for stochastic processes. In: *IEEE Transactions on Systems, Man and Cybernetics—Systems* (Accepted) (2019)
35. Zhang, C., Wang, H., Wang, H., Wu, M.H.: EEG-based expert system using complexity measures and probability density function control in alpha sub-band. *Integr Comput.-Aided Eng.* **20**, 391–405 (2013)
36. Zhang, H., Li, P.: Chance constrained programming for optimal power flow under uncertainty. *IEEE Trans. Power Syst.* **26**, 2417–2424 (2011)

**Part VI**

**Multi-Disciplinary Connections**

## Chapter 24

# A Hybrid Dynamical Systems Perspective on Reinforcement Learning for Cyber-Physical Systems: Vistas, Open Problems, and Challenges



Jorge I. Poveda and Andrew R. Teel

**Abstract** The next generation of autonomous cyber-physical systems will integrate a variety of heterogeneous computation, communication, and control algorithms. This integration will lead to closed-loop systems with highly intertwined interactions between the digital world and the physical world. For these systems, designing robust and optimal data-driven control algorithms necessitates fundamental breakthroughs at the intersection of different areas such as adaptive and learning-based control, optimal and robust control, hybrid dynamical systems theory, and network control, to name just a few. Motivated by this necessity, control techniques inspired by ideas of reinforcement learning have emerged as a promising paradigm that could potentially integrate most of the key desirable features. However, while significant results in reinforcement learning have been developed during the last decades, the literature is still missing a systematic framework for the design and analysis of reinforcement learning-based controllers that can safely and systematically integrate the intrinsic continuous-time and discrete-time dynamics that emerge in cyber-physical systems. Motivated by this limitation, and by recent theoretical frameworks developed for the analysis of hybrid systems, in this chapter we explore some vistas and open problems that could potentially be addressed by merging tools from reinforcement learning and hybrid dynamical systems theory, and which could have significant implications for the development of the next generation of autonomous cyber-physical systems.

### 24.1 Introduction

Cyber-physical systems are complex dynamical systems that integrate physics-based dynamics and digital-based dynamics induced by communication, computation, and control algorithms. Recent technological advances in the areas of sensing, com-

---

J. I. Poveda (✉)  
University of Colorado, Boulder, CO 80309, USA  
e-mail: [jorge.poveda@colorado.edu](mailto:jorge.poveda@colorado.edu)

A. R. Teel  
University of California, Santa Barbara, CA 93106, USA  
e-mail: [teel@ucsb.edu](mailto:teel@ucsb.edu)

munication and actuation have enabled the development of a new generation of cyber-physical systems in different engineering domains [1], including autonomous vehicles [2], intelligent water distribution systems [3], smart transportation systems [4], manufacturing systems [5], and the smart power grid [6], to name just a few. A common feature that emerges in all these systems is a strong interaction between continuous-time dynamics, usually modeled by differential equations, and discrete-time dynamics, usually modeled by difference equations. These interactions lead to complex mathematical models described by hybrid dynamical systems (HDSs), which have been the subject of extensive research during the last years in different disciplines such as control theory [1], biology [7], and computer science [8]. One of the main challenges that emerges in the analysis, control, and optimization of cyber-physical systems is the high complexity of the mathematical models that describe their dynamic behavior. This complexity often leads to gray-box and black-box characterizations, where only certain qualitative properties of the system are known a priori, precluding the implementation of traditional model-driven controllers such as linear quadratic regulators (LQRs) and model-predictive control (MPC).

On the other hand, there have been significant recent efforts in the development of model-free and data-driven algorithms that aim to control and optimize the performance of a dynamical system by using minimal information of its mathematical model, and by exploiting massive amounts of data that are now available in several engineering domains; see, for instance, [9–12], and references therein. Along these lines, reinforcement learning (RL) has emerged as a promising paradigm that lies at the intersection of traditional control theory and modern approaches based on machine learning, neural networks, and artificial intelligence. The main idea behind RL is to use evaluative feedback from the environment in order to take appropriate actions that optimize a particular performance measure. This evaluative feedback is usually based on a reward function generated by the environment in response to a particular input or action by the “agent” or control system. In contrast to traditional evolutionary algorithms, RL aims to find a policy (or control law), that maps states to inputs, i.e., a feedback law. Therefore, in this regard, RL is more powerful than traditional evolutionary algorithms, as it exploits the structure of the environment in order to find a suitable controller [13]. Comprehensive introductions to RL are presented in the books [13–16], and recent survey papers from a feedback controls perspective are presented in [17, 18]. Connections between RL, MPC, and duality approaches of LQR have been recently studied in [19, 20], respectively.

In this chapter, we will focus on a particular class of infinite horizon RL problems that are closely related to the setting of approximate optimal control and approximate adaptive dynamic programming (ADP). These algorithms, usually classified in the literature as *neuro-adaptive optimal controllers* [16, 17, 21], aim to find in real time a solution to the Hamilton–Jacobi–Bellman equation by using output measurements of the plant under control without sacrificing desirable stability properties for the closed-loop system. From the perspective of control theory, ADP merges tools from optimal control theory and adaptive control theory, as well as universal approximation properties of neural networks. Thus, RL algorithms based on ADP combine the advantages of optimal control and adaptive control, and can be studied by well-

established analytical tools from linear and nonlinear control theory. On the other hand, it is natural to expect that RL algorithms based on ADP also inherit some of the intrinsic limitations of optimal and adaptive controllers, which include the requirement of a persistence of excitation (PE) condition that is in general difficult to guarantee a priori, the assumption of existence of initial admissible stabilizing policies, the requirements of certain affine-in-the-input structures in the dynamics of the plant, and the high dimension of the vector of hyper-parameters that is used to parameterize the value function and the feedback policy. Therefore, most of the recent work in neuro-adaptive optimal control has been focused on addressing some of these limitations by using strategies based on concurrent learning [22], state-following kernels [23], multi-layer neural networks [24], and dynamic identifiers [25].

While RL algorithms based on ADP and neuro-adaptive ideas have been extensively studied in discrete time and in continuous time, a general theory for hybrid dynamical systems has remained elusive. This challenge is mainly related to several problems that remain open in the areas of hybrid optimal control and hybrid adaptive control. Results for hybrid optimal control can be found in [26–29], and more recently in [30]. Hybrid adaptive algorithms have been recently developed in the context of output regulation [31], non-smooth and hybrid extremum seeking control [32–35], and iterative learning [36]. Hybrid approaches based on least-squares with covariance resetting are also now standard in adaptive control. A recent survey paper on learning-based hybrid controllers is presented in [37].

Motivated by the previous background, this chapter aims to introduce readers to the mathematical framework of HDS [38], and to identify potential connections and open research problems that integrate RL and HDS, which could have a significant impact on the development of the next generation of autonomous cyber-physical systems. The mathematical framework described in this chapter differentiates from other approaches in different ways. In particular, we consider hybrid dynamical systems that can be characterized by dynamic hybrid inclusions, i.e., differential and difference inclusions. This formulation allows us to model a broader class of dynamic behaviors by using simpler mathematical models that can be rigorously analyzed via Lyapunov tools and invariance principles. Second, we consider HDS that may not generate unique solutions from a given initial condition. Third, under mild assumptions on the equations that describe the system, the solutions of the HDS exhibit a sequential compactness property and a semi-continuity property with respect to initial conditions. These properties are relevant for the study of perturbed HDS, which usually emerge in RL due to approximation errors generated by neural networks or time-scale separations. We also discuss connections with recent approaches developed for RL in the context of policy gradient, as well as some illustrative open problems that could benefit from using tools from hybrid dynamical systems theory.

The rest of this chapter is organized as follows: Section 24.2 introduces readers to the theory of HDS. Section 24.3 reviews the basic ideas behind policy gradient in RL and actor–critic approaches based on optimal control and adaptive control. Section 24.4 draws some connections between HDS and existing algorithms and architectures

used in RL, as well as some potential extensions that incorporate HDS to improve the performance of the system. Finally, Sect. 24.5 ends with some conclusions.

## 24.2 Hybrid Dynamical Systems

Hybrid dynamical systems (HDSs) are dynamical systems that combine continuous-time dynamics, usually modeled by ordinary differential equations (ODEs), and discrete-time dynamics, usually modeled by difference equations or recursions. Thus, HDS provide suitable mathematical representations for the physics-based dynamics and the digital-based dynamics that emerge in cyber-physical systems. In this chapter, we will follow the framework of [38], where a HDS is represented by the following two equations:

$$x \in C, \quad \dot{x} = f(x), \quad (24.1a)$$

$$x \in D, \quad x^+ = g(x), \quad (24.1b)$$

where  $x \in \mathbb{R}^n$  is the state of the system, which can model positions, velocities, voltages, traffic flow, logic states, etc.;  $f : \mathbb{R}^n \rightarrow \mathbb{R}^n$  is called the flow map, and it characterizes the continuous-time dynamics of the system;  $g : \mathbb{R}^n \rightarrow \mathbb{R}^n$  is called the jump map, and it describes the discrete-time dynamics of the system;  $C \subset \mathbb{R}^n$  is called the flow set, and it describes the points in the space where the system evolves according to the ODE  $\dot{x} = f(x)$ ; and  $D \subset \mathbb{R}^n$  is called the jump set, and it describes the points in the space where the system evolves according to the recursion  $x^+ = g(x)$ . Due to the generality of the model, HDS of the form (24.1) can be used to characterize the dynamic behavior of different classes of physical systems, digital systems, cyber-physical systems, dynamic algorithms, feedback control mechanisms, etc. In the following, we present some illustrative examples:

**Example 24.1** (*The Bouncing Ball*) A simple mechanical system that can be modeled by Eq. (24.1) is a point-mass ball that bounces vertically on a horizontal surface. The dynamics of this system can be characterized by the following hybrid model:

$$\begin{aligned} x \in C := \{x \in \mathbb{R}^2 : x_1 \geq 0\}, \quad & \begin{bmatrix} \dot{x}_1 \\ \dot{x}_2 \end{bmatrix} = f(x) := \begin{bmatrix} x_2 \\ -\gamma \end{bmatrix}, \\ x \in D := \{x \in \mathbb{R}^2 : x_1 = 0, x_2 \leq 0\}, \quad & \begin{bmatrix} x_1^+ \\ x_2^+ \end{bmatrix} = g(x) := \begin{bmatrix} 0 \\ -\lambda x_2 \end{bmatrix}, \end{aligned}$$

where  $x_1$  represents the height above the surface, and  $x_2$  represents the vertical velocity. The constant  $\gamma > 0$  corresponds to the acceleration due to gravity, and the parameter  $\lambda \in (0, 1)$  is the dissipation coefficient. It can be shown that every solution

of this system converges to the point  $\mathcal{A} = [0, 0]^\top$ , and that the limiting behavior is Zeno.<sup>1</sup>  $\square$

**Example 24.2** (*Dynamic Accelerated Optimization Algorithms*) Hybrid systems can be used to model recursive optimization algorithms. For instance, consider the optimization problem

$$\min_{x_1 \in \mathbb{R}^n} \phi(x_1), \quad (24.2)$$

where the function  $\phi : \mathbb{R}^n \rightarrow \mathbb{R}$  is  $\mu$ -strongly convex and has a Lipschitz gradient and bounded level sets. Let  $\mathcal{A} := \operatorname{argmin}_{x_1 \in \mathbb{R}^n} \phi(x_1)$ . Then, the following hybrid dynamical system with state  $x = (x_1, x_2, x_3)$  models a suitable accelerated optimization algorithm:

$$x \in C := \{x \in \mathbb{R}^3 : x_3 \in [T_{\min}, T_{\max}]\}, \quad \begin{bmatrix} \dot{x}_1 \\ \dot{x}_2 \\ \dot{x}_3 \end{bmatrix} = f(x) := \begin{bmatrix} \frac{2}{x_3}(x_2 - x_1) \\ -2kx_3 \nabla \phi(x_1) \\ \frac{1}{2} \end{bmatrix}, \quad (24.3a)$$

$$x \in D := \{x \in \mathbb{R}^3 : x_3 = T_{\max}\}, \quad \begin{bmatrix} x_1^+ \\ x_2^+ \\ x_3^+ \end{bmatrix} = g(x) := \begin{bmatrix} x_1 \\ x_1 \\ T_{\min} \end{bmatrix}, \quad (24.3b)$$

where  $k > 0$  is a tunable gain. If the parameters of the algorithm satisfy the conditions  $0 < T_{\min} < T_{\max}$  and  $T_{\max}^2 - T_{\min}^2 \geq \frac{1}{2\mu k}$ , it can be shown that the solutions  $x_1$  of the HDS (24.3) will converge to  $\mathcal{A}$  exponentially fast [39]. A related hybrid accelerated optimization algorithm is presented in [40].  $\square$

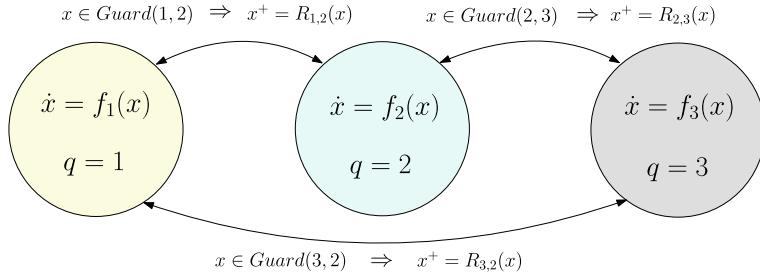
**Example 24.3** (*Sampled-Data Control of Dynamical Systems*) Feedback control systems are usually implemented in a sampled-data structure, where a discrete-time controller is interconnected with a continuous-time plant via a zero-order hold mechanism (ZOH) and a sampling device. The closed-loop system can then be modeled as a HDS of the form (24.1) by defining  $\theta$  as the state of the plant, and  $u$  as the state of the controller. The sampler and the ZOH mechanism can be represented by a periodic clock with state  $\tau$ , which is reset every time that a measurement and an update of the controller occurs. In its simpler form, a sampled-data closed-loop control system can then be written as a HDS with states  $x_1 = \theta$ ,  $x_2 = u$ , and  $x_3 = \tau$ , and dynamics:

$$x \in C := \{x \in \mathbb{R}^3 : x_3 \in [0, T_s]\}, \quad \begin{bmatrix} \dot{x}_1 \\ \dot{x}_2 \\ \dot{x}_3 \end{bmatrix} = f(x) := \begin{bmatrix} f_p(x_1, x_2) \\ 0 \\ 1 \end{bmatrix}, \quad (24.4a)$$

$$x \in D := \{x \in \mathbb{R}^3 : x_3 = T_s\}, \quad \begin{bmatrix} x_1^+ \\ x_2^+ \\ x_3^+ \end{bmatrix} = g(x) := \begin{bmatrix} x_1 \\ g_c(x_1, x_2) \\ 0 \end{bmatrix}, \quad (24.4b)$$

---

<sup>1</sup>A solution is said to be Zeno if it has an infinite number of jumps in a finite interval of time.



**Fig. 24.1** A hybrid automaton with three logic modes. Transitions between modes occur whenever the guard conditions are satisfied. The discrete updates of the state are characterized by the reset maps

where \$T\_s > 0\$ is the sampling period of the clock of the ZOH/sampling mechanism. The HDS (24.4) describes a system with jumps happening only when the clock satisfies the condition \$x\_3 = T\_s\$. Each jump resets the clock to zero and triggers an update of the control input via the mapping \$g\_c(x\_1, x\_2)\$, while keeping the state of the plant constant. On the other hand, during flows, the state of the controller remains constant and the plant evolves according to the ODE \$\dot{x}\_1 = f\_p(x\_1, x\_2)\$. This model can be extended and generalized to capture asynchronous sampling devices [41], delays [42], output measurements, etc.  $\square$

**Example 24.4 (The Hybrid Automaton)** The hybrid automaton has been extensively studied in the literature of computer science, learning theory, and hybrid systems. The standard hybrid automaton describes a system that can evolve according to a finite number of ‘‘modes’’, represented by a logic state \$q \in Q := \{1, 2, 3, \dots, \bar{q}\}\$, with \$\bar{q} \in \mathbb{Z}\_{>0}\$. In each mode, the dynamics of the system can evolve according to the constrained differential equation

$$x \in C_q, \quad \dot{x} = f_q(x).$$

Whenever a particular guard condition \$\text{Guard}(q, q')\$ is satisfied, the state \$q\$ is switched to a different value \$q' \in Q\$, and the state \$x\$ is updated according to some reset rule \$x^+ = R\_{q,q'}(x)\$, see Fig. 24.1. As shown in [38, Sect. 1.4.2], the hybrid automaton can be modeled as a HDS of the form (24.1). Since in these types of systems it is common to have multiple possible switching rules for \$q\$, the hybrid automata usually lack the uniqueness of solutions property. It turns out that the model (24.1) is particularly well-suited to capture this behavior.  $\square$

**Example 24.5 (Global Stabilization of a Point on Compact Smooth Manifolds)** The combination of continuous-time and discrete-time dynamics provides HDS with the potential to solve stabilization, optimization, and estimation problems that cannot be solved using standard or discretized ODEs. For instance, consider the problem of global robust stabilization of a point \$x^\* \in \mathbb{S}^1\$ on the unit circle

$\mathbb{S}^1 = \{x \in \mathbb{R}^2 : x_1^2 + x_2^2 = 1\}$ . Here, by robust we mean stabilization that is “insensitive” to arbitrarily small additive disturbances in states and dynamics of the system, e.g., noisy measurements, numerical approximations, etc. By “global”, we mean that the stabilization of  $x^*$  must be achieved from all initial conditions in  $\mathbb{S}^1$ . Since any smooth compact manifold without boundary is not contractible [43, 189, Definition B.13. and Lemma 2.1.9] and since the basin of attraction of an asymptotically stable equilibrium point of any differential equation with a continuous right-hand side is contractible [43, 189, Theorem 2.1.8], it is impossible to achieve global asymptotic stability of an equilibrium in smooth compact manifolds without boundary using smooth feedback of the form

$$x \in \mathbb{S}^1, \quad \dot{x} = f(x).$$

Moreover, discontinuous feedback generates systems that are not robust to small noise [43]. Similar obstructions emerge in obstacle avoidance problems [44, 45].

On the other hand, the robust global stabilization problem on  $\mathbb{S}^1$  can be solved by using a hybrid algorithm. One approach is to consider the logic-based hybrid dynamics

$$x \in \mathbb{S}^1, \quad \dot{x} = u_q(x) S x, \quad S := \begin{bmatrix} 0 & -1 \\ 1 & 0 \end{bmatrix}, \quad (24.5a)$$

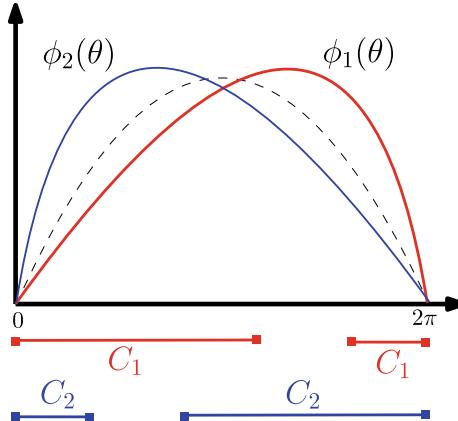
$$q \in \{1, 2\}, \quad \dot{q} = 0, \quad (24.5b)$$

where  $u_q : \mathbb{S}^n \rightarrow \mathbb{R}$  is a switched feedback controller defined as

$$u_q(x) = -\langle \nabla \phi_q(x), S x \rangle,$$

with  $\phi_1$  and  $\phi_2$  being two potential fields having a single maximum  $\tilde{x}_q \in \mathbb{S}^1$  and a unique common minimizer at the point  $x = x^*$ . By an appropriate construction of the potentials, and by using a hysteresis-based switching rule that toggles the logic state to  $q^+ = 3 - q$ , the resulting hybrid feedback controller emulates a hybrid gradient descent that always moves in the direction that minimizes its current potential field. Figure 24.2 illustrates this idea in polar coordinates  $\theta \in [0, 2\pi)$ , for the case when  $x^* = [1, 0]$ , which corresponds to  $\theta^* = 0$ . The jump sets and the flow sets are constructed such that whenever the state  $\theta$  is in a neighborhood of the maximizer  $\tilde{x}_q$  of  $\phi_q$ , the state  $q$  is switched, and the dynamics (24.5a) steer the state  $x$  away from  $\tilde{x}_q$  and toward  $x^*$ . Robust global stabilization can then be established via hybrid Lyapunov functions [43, 46].  $\square$

The previous examples illustrate the versatility of the modeling framework described by Eq.(24.1). In the following, we review some extensions and basic structural properties of HDS.



**Fig. 24.2** Illustration of synergistic Lyapunov functions for the global regulation of the point  $x^* = [1, 0]$ , corresponding to  $\theta^* = 0$  in polar coordinates. The sets  $C_i$  qualitatively describe the points in the space where the controller implements a gradient flow based on the function  $\phi_i$

### 24.2.1 Non-uniqueness of Solutions and Set-Valued Dynamics

While we have not formalized yet the notion of *solutions* for HDS of the form (24.1), it is important to note that, under any reasonable definition of solutions, whenever the intersection of the sets  $C$  and  $D$  is not empty, the HDS (24.1) may not generate unique solutions from initial conditions  $x_0$  satisfying  $x_0 \in C \cap D$ . This marks a clear departure from the traditional literature in non-hybrid nonlinear dynamical systems, where the property of uniqueness of solutions is either assumed a priori, or guaranteed by imposing structural regularity assumptions on the data of the system.

On the other hand, working with dynamical systems that generate non-unique solutions can be harnessed to model families of functions that satisfy a particular common qualitative behavior. The following example illustrates this idea:

**Example 24.6** (*Average Dwell-Time Constraint*) Consider a HDS with states  $(x, q) \in \mathbb{R}^2$ , and dynamics of the form

$$(x, q) \in [0, N_0] \times Q, \quad \dot{x} \in \left[0, \frac{1}{\tau_D}\right], \quad \dot{q} = 0, \quad (24.6a)$$

$$(x, q) \in [1, N_0] \times Q, \quad x^+ = x - 1, \quad q^+ \in Q, \quad (24.6b)$$

where  $N_0 \geq 1$ ,  $\tau_D > 0$ , and  $Q = \{1, 2, 3, \dots, \bar{q}\}$ . This hybrid system can generate infinitely many solutions by flowing at different rates. Moreover, when  $N_0 > 1$ , there exist initial conditions that generate solutions that can evolve either by flowing or jumping. However, every solution of the HDS (24.6) satisfies the following property for any pair of times  $t > s$ :

$$\# \text{ of jumps between times } t \text{ and } s \leq \frac{1}{\tau_D}(t-s) + N_0. \quad (24.7)$$

The property described by Eq. (24.7) is called an *average dwell-time constraint* [47], and it is relevant for the stability analysis of switching systems. Indeed, it can be shown that every switching signal  $q : \mathbb{R}_{\geq 0} \rightarrow Q$  that satisfies the bound (24.7) can be generated by the HDS (24.6). Therefore, the stability analysis of switching systems under switching signals that satisfy an average dwell-time constraint can be carried out by interconnecting the switching system with the hybrid dynamics (24.6).  $\square$

Non-uniqueness of solutions in dynamical systems can also emerge when studying systems with a discontinuous right-hand side, even if the system is not hybrid. In particular, ODEs with a discontinuous right-hand side may fail to have solutions in the sense of Carathéodory from some particular initial conditions. On the other hand, it is possible to consider generalized notions of solutions defined in the sense of Krasovskii or Filippov. Krasovskii solutions are particularly useful to capture the emerging behavior of the system under small vanishing disturbances. For discontinuous ODEs  $\dot{x} = f(x)$  constrained to evolve in a closed set  $C \subset \mathbb{R}^n$ , a Krasovskii solution  $x : \text{dom}(x) \rightarrow \mathbb{R}^n$  is defined as any absolutely continuous function that satisfies for almost all  $t \in \text{dom}(x)$  the inclusions

$$x(t) \in C \quad \text{and} \quad \dot{x}(t) \in \hat{F}(x(t)) := \bigcap_{\delta>0} \overline{\text{co}} f((x + \delta\mathbb{B}) \cap C), \quad (24.8)$$

where  $\overline{\text{co}}(\cdot)$  stands for the closed convex hull, i.e., the mapping  $\hat{F}$  is given by

$$\bigcap_{\delta>0} \overline{\text{co}} f((x + \delta\mathbb{B}) \cap C) := \left\{ v \in \mathbb{R}^n : v = \lim_{i \rightarrow \infty} \sum_{k=1}^{n+1} \lambda_{ik} f_{ik}, \lambda_{ik} \in [0, 1], \sum_{k=1}^{n+1} \lambda_{i,k} = 1, f_{i,k} = f(x_{ik}), C \ni x_{ik} \rightarrow x \text{ as } i \rightarrow \infty \right\}.$$

Similarly, for discontinuous difference equations  $x^+ = g(x)$  constrained to evolve in a closed set  $D \subset \mathbb{R}^n$ , a Krasovskii solution is defined as any function (or sequence, to be more precise)  $x : \text{dom}(x) \mapsto \mathbb{R}^n$  that satisfies for all  $j \in \text{dom}(x)$  the inclusions

$$x(j) \in D \quad \text{and} \quad x(j+1) \in \hat{G}(x(j)) := \bigcap_{\delta>0} \overline{g((x + \delta\mathbb{B}) \cap D)}, \quad (24.9)$$

where the set-valued mapping  $\hat{G}$  is given by

$$\bigcap_{\delta>0} \overline{g((x + \delta\mathbb{B}) \cap D)} := \left\{ v \in \mathbb{R}^n : v = \lim_{i \rightarrow \infty} g_i, g_i = g(x_i), D \ni x_i \rightarrow x \text{ as } i \rightarrow \infty \right\}.$$

We will call the set-valued mappings  $\hat{F}$  and  $\hat{G}$  defined in (24.8) and (24.9), respectively, as the *Krasovskii regularizations* of the mappings  $f$  and  $g$ , respectively. This

regularization will be applied to the HDS (24.1) whenever the flow map and the jump map are discontinuous.

**Example 24.7 (Krasovskii Regularization of HDS)** Consider the following HDS:

$$\begin{aligned} x \in C &:= \{x \in \mathbb{R}^2 : x_1 \in \mathbb{R}, x_2 \in [0, T_s]\}, & \begin{bmatrix} \dot{x}_1 = -\text{sign}(x) \\ \dot{x}_2 = 1 \end{bmatrix}, \\ x \in D &:= \{x \in \mathbb{R}^2 : x_1 \in \mathbb{R}, x_2 = T_s\}, & \begin{bmatrix} x_1^+ = x - \alpha \text{sign}(x) \\ x_2^+ = 0 \end{bmatrix}, \end{aligned}$$

where  $\alpha \in (0, 1)$  is a small positive number,  $T_s > 0$  is a sampling period, and the function  $x \mapsto \text{sign}(x)$  is defined as

$$\text{sign}(x) = \begin{cases} 1 & \text{if } x > 0 \\ 0 & \text{if } x = 0 \\ -1 & \text{if } x < 0 \end{cases}.$$

The Krasovskii regularization of the HDS is given by the following hybrid dynamics

$$\begin{aligned} x \in C &:= \{x \in \mathbb{R}^2 : x_1 \in \mathbb{R}, x_2 \in [0, T_s]\}, & \begin{bmatrix} \dot{x}_1 \in \begin{cases} -1 & \text{if } x > 0 \\ [-1, 1] & \text{if } x = 0 \\ 1 & \text{if } x < 0 \end{cases} \\ \dot{x}_2 = 1 \end{bmatrix}, \\ x \in D &:= \{x \in \mathbb{R}^2 : x_1 \in \mathbb{R}, x_2 = T_s\}, & \begin{bmatrix} x_1^+ = x - \alpha \in \begin{cases} 1 & \text{if } x > 0 \\ \{-1, 0, 1\} & \text{if } x = 0 \\ -1 & \text{if } x < 0 \end{cases} \\ x_2^+ = 0 \end{bmatrix}. \end{aligned}$$

Since the resulting continuous-time dynamics and discrete-time dynamics are set-valued, the solutions of the regularized system are in general not unique.  $\square$

We finish this subsection by linking the Krasovskii regularizations (24.8) and (24.9) with a fundamental robustness result for dynamical systems. In particular, we consider the perturbed set-valued hybrid dynamical system

$$x + e \in C, \quad \dot{x} \in F(x + e), \tag{24.10a}$$

$$x + e \in D, \quad x^+ \in G(x + e), \tag{24.10b}$$

where  $e$  is a small possibly time-varying disturbance uniformly bounded by a small constant  $\varepsilon > 0$ . The perturbed model (24.10) is useful to define the so-called *Hermes solutions* [38, Chap. 4]. Loosely speaking, Hermes solutions are solutions generated by (24.10) under shrinking disturbances  $e$ , and therefore they capture the behavior of a nominal HDS under arbitrarily small additive disturbances acting on the states. It turns out that when the flow map and the jump map are locally bounded, Hermes solutions are equivalent to Krasovskii solutions [38, Theorem 4.17]. Thus, Krasovskii

solutions play an important role in the robustness analysis of HDS since they predict the emerging behavior of the nominal system under vanishing additive perturbations.

### 24.2.2 *Hybrid Time Domains and Solutions of Hybrid Dynamical Systems*

We are now ready to define the notion of *solution* for a HDS. Since set-valued dynamics emerge naturally in hybrid systems, we will consider the general model given by the inclusions

$$x \in C, \quad \dot{x} \in F(x), \tag{24.11a}$$

$$x \in D, \quad x^+ \in G(x), \tag{24.11b}$$

where  $F : \mathbb{R}^n \rightrightarrows \mathbb{R}^n$  and  $G : \mathbb{R}^n \rightrightarrows \mathbb{R}^n$  are set-valued mappings. This setting generalizes the single-valued case (24.1), which can be written as (24.11) by an appropriate choice of  $F$  and  $G$ ; see [38, Chap. 2]. Solutions  $x$  of (24.11) are parameterized by two temporal indexes:  $t \in \mathbb{R}_{\geq 0}$  and  $j \in \mathbb{Z}_{\geq 0}$ . The continuous-time index  $t$  increases continuously whenever the system flows according to (24.11a). The discrete-time variable  $j$  increases by one every time that the system jumps according to (24.11b). Therefore, we say that the solutions of (24.11) are defined on *hybrid time domains*. This dual parameterization is needed in order to define an appropriate notion of “closeness” between solutions of (24.11), which, in turn, is instrumental to establish certain semi-continuity and sequential compactness properties that will emerge under mild assumptions on the data  $\mathcal{H} = \{C, F, D, G\}$ .

**Example 24.8** (*Uniform Convergence and Continuity of Solutions With Respect to Initial Conditions*) In the standard theory of differential equations and inclusions, the notions of uniform distance and uniform convergence are used to analyze closeness between solutions. This is the case because, for these types of systems, solutions are indeed continuous functions. However, for HDS these notions are problematic. As a matter of fact, consider the hybrid model describing the bouncing ball of Example 24.1, and suppose that the solutions of the system are defined as piece-wise continuous functions parameterized only by  $t$ . Let  $\gamma = 1$  and  $\lambda = 1/2$ , and consider a solution with initial condition  $x_0 = (1 + \delta, 0)$ , where  $\delta \in [0, 1]$ . We denote this solution by  $x_\delta = (x_{1,\delta}, x_{2,\delta})$  since it depends on the value of  $\delta$ . Then, for all  $t \in [0, 2]$ , the velocity  $x_{2,\delta}$  of the system satisfies the equation

$$x_{2,\delta}(t) = \begin{cases} -t, & t \in [0, \sqrt{2(1+\delta)}] \\ -t + 3\sqrt{(1+\delta)/2}, & t \in [\sqrt{2(1+\delta)}, 2] \end{cases}.$$

It follows that the uniform distance between two solutions  $x_{2,0}(t)$  and  $x_{2,\delta}(t)$  is given by  $\sup_{t \in [0,2]} |x_{2,0}(t) - x_{2,\delta}(t)| \geq 3\sqrt{2}/2$ . This distance does not decrease as  $\delta \rightarrow 0$ , which shows that whenever the uniform distance is used, the velocity does not depend

continuously on initial conditions. This contradicts the intuition (and the physical evidence) that two solutions of the bouncing ball that are dropped from close initial conditions are indeed close to each other on compact time domains.  $\square$

In order to establish a semi-continuity property with respect to initial conditions for the solutions of the HDS (24.11), we use the notion of hybrid time domains. A set  $E \subset \mathbb{R}_{\geq 0} \times \mathbb{Z}_{\geq 0}$  is called a *compact hybrid time domain* if  $E = \cup_{j=0}^{J-1}([t_j, t_{j+1}], j)$  for some finite sequence of times  $0 = t_0 \leq t_1 \dots \leq t_J$ . The set  $E$  is a *hybrid time domain* if for all  $(T, J) \in E$ , the set  $E \cap ([0, T] \times \{0, \dots, J\})$  is a compact hybrid time domain.

By using the notion of hybrid time domains we can introduce the notion of *hybrid arcs*. A function  $x : \text{dom}(x) \mapsto \mathbb{R}^n$  is a *hybrid arc* if  $\text{dom}(x)$  is a hybrid time domain and if for each  $j$  such that the interval  $I_j := \{t : (t, j) \in \text{dom}(x)\}$  has nonempty interior the function  $t \mapsto x(t, j)$  is absolutely continuous on each compact subinterval of  $I_j$ . A hybrid arc  $x$  is a *solution* to (24.11) if  $x(0, 0) \in \bar{C} \cup D$ , and the following two conditions hold:

1. For each  $j \in \mathbb{Z}_{\geq 0}$  such that  $I_j$  has nonempty interior:  $x(t, j) \in C$  for all  $t \in \text{int}(I_j)$ , and  $\dot{x}(t, j) \in F(x(t, j))$  for almost all  $t \in I_j$ .
2. Foreach  $(t, j) \in \text{dom}(x)$  such that  $(t, j+1) \in \text{dom}(x)$ :  $x(t, j) \in D$  and  $x(t, j+1) \in G(x(t, j))$ .

Solutions that have an unbounded time domain are called *complete*. Figure 24.3 shows different hybrid time domains associated to different types of hybrid solutions: periodic, average dwell-time, Zeno, eventually continuous, eventually discrete, and a solution with finite escape time.

### 24.2.3 Graphical Convergence, Basic Assumptions and Sequential Compactness

Working with hybrid time domains facilitates the analysis of solutions of HDS (24.11) via the notion of graphical convergence. The *graph* of a hybrid arc is defined as

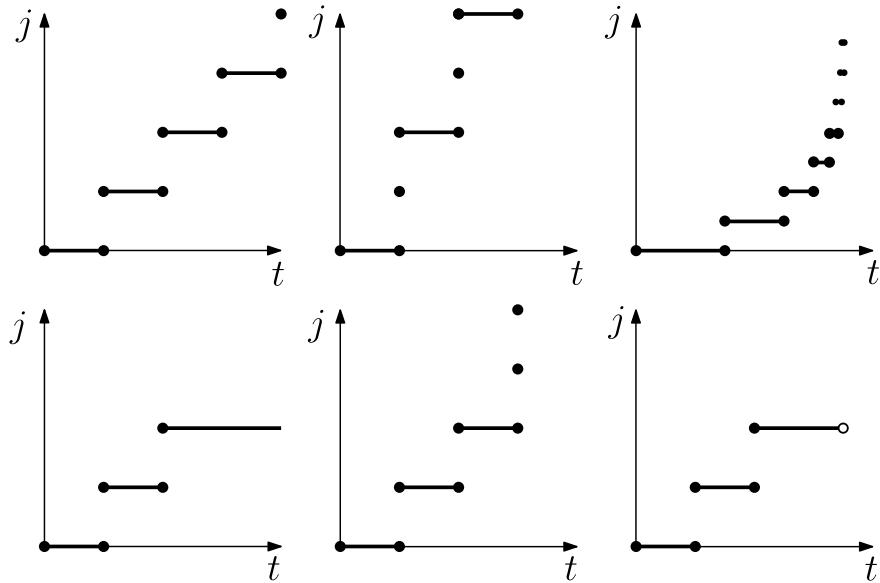
$$\text{gph}(x) = \{(t, j, z) : (t, j) \in \text{dom}(x), z = x(t, j)\} \subset \mathbb{R}^{n+2}.$$

A sequence of hybrid arcs  $\{x_i\}_{i=1}^{\infty}$  converges graphically if the sequence  $\{\text{gph}(x_i)\}_{i=1}^{\infty}$  of graphs converges in the sense of set convergence.<sup>2</sup> Graphical convergence can also be equivalently characterized using the notion of  $(\tau, \varepsilon)$ -closeness. In particular, two hybrid arcs  $x$  and  $y$  are  $(\tau, \varepsilon)$ -close if the following conditions are satisfied:

1. For each  $(t, j) \in \text{dom}(x)$  with  $t + j \leq \tau$  there exists  $s \in \mathbb{R}_{\geq 0}$  such that  $(s, j) \in \text{dom}(y)$ ,  $|t - s| \leq \varepsilon$ , and

---

<sup>2</sup>A sequence of sets converges if its inner limit and its outer limit exist and are equal. See [38, Definition 5.1].



**Fig. 24.3** Different hybrid time domains generated by different types of solutions. Upper row, from left to right: Periodic solution, average dwell-time solution, Zeno solution. Bottom row, from left to right: eventually continuous solution, eventually discrete solution, solution with finite escape time

$$|x(t, j) - y(s, j)| \leq \varepsilon.$$

2. For each  $(t, j) \in \text{dom}(y)$  with  $t + j \leq \tau$  there exists  $s \in \mathbb{R}_{\geq 0}$  such that  $(s, j) \in \text{dom}(x)$ ,  $|t - s| \leq \varepsilon$ , and

$$|y(t, j) - x(s, j)| \leq \varepsilon.$$

Using the notion of  $(\tau, \varepsilon)$ -closeness it can be shown that a locally eventually bounded sequence of hybrid arcs  $\{x_i\}_{i=1}^{\infty}$  converges graphically to  $x$  if and only if, for every  $\tau \geq 0$  and  $\varepsilon > 0$ , there exists  $i_0$  such that, for all  $i > i_0$ , the hybrid arcs  $x_i$  and  $x$  are  $(\tau, \varepsilon)$ -close.

**Example 24.9** (*Graphical Convergence and Semi-Continuity of Solutions With Respect to Initial Conditions*) Consider again the solution considered in Example 24.8. Using hybrid time domains to parameterize the trajectories we obtain the solutions

$$x_{2,\delta}(t) = \begin{cases} -t, & t \in [0, \sqrt{2(1+\delta)}], \quad j = 0, \\ -t + 3\sqrt{(1+\delta)/2}, & t \in [\sqrt{2(1+\delta)}, 2], \quad j = 1. \end{cases}$$

It can now be shown that the hybrid arcs  $x_{2,\delta}$  and  $x_{2,0}$  are  $(\tau, \varepsilon)$ -close with any  $\tau \geq 0$  and  $\varepsilon = 3\sqrt{2}(\sqrt{1+\delta} - 1)/2$ . Therefore,  $x_{2,\delta}$  converges graphically, as  $\delta \rightarrow 0$ , to  $x_{2,0}$ .  $\square$

In order to use the notion of graphical convergence to establish suitable semi-continuity properties of solutions of (24.11) with respect to the initial conditions, certain regularity conditions must be satisfied by the HDS. These conditions can be guaranteed if the following Basic Assumptions hold:

The Basic Assumptions are the following conditions on the data  $(C, F, D, G)$ :

1. The sets  $C$  and  $D$  are closed.
2. The set-valued mapping  $F : \mathbb{R}^n \rightrightarrows \mathbb{R}^n$  is outer semicontinuous, locally bounded on  $C$ , and such that  $F(x)$  is nonempty and convex for each  $x \in C$ .
3. The set-valued mapping  $G : \mathbb{R}^n \rightrightarrows \mathbb{R}^n$  is outer semicontinuous, locally bounded on  $D$ , and such that  $G(x)$  is nonempty for each  $x \in D$ .

HDS that satisfy the Basic Conditions generate solutions that satisfy a sequential compactness property. That is, for every sequence of solutions  $\{x_i\}_{i=1}^\infty$  that is locally eventually bounded there exists a convergent subsequence that converges graphically to a solution of the HDS. This sequential compactness result is instrumental to establish an outer semicontinuous dependence of solutions on initial conditions. Namely, if a HDS satisfies the Basic Assumptions, and  $x_0 \in \mathbb{R}^n$  is an initial condition that generates complete or bounded solutions, then for every  $\tau \geq 0$  and  $\varepsilon > 0$ , there exists  $\delta > 0$  such that, for each solution  $x'$  of the HDS that satisfies  $|x'(0, 0) - x_0| \leq \delta$ , there exists a solution  $x$  to the HDS with  $x(0, 0) = x_0$  such that  $x'$  and  $x$  are  $(\tau, \varepsilon)$ -close.

#### 24.2.4 Stability and Robustness

The standard boundedness, stability, and convergence notions studied in continuous-time [48] and discrete-time [49] dynamical systems can also be extended to hybrid dynamical systems. In the context of reinforcement learning, the most commonly used properties are (uniform) asymptotic stability (UGAS), exponential stability (UGES), and uniform ultimate boundedness (UUB). The solutions of a HDS are said to be uniformly ultimately bounded (UUB) if there are no finite escape times and there exists  $M > 0$  such that for each  $\Delta > 0$  there exists  $T > 0$  such that for every solution  $x$  with  $x(0, 0) \in \Delta \mathbb{B}$  either  $t + j < T$  for all  $(t, j) \in \text{dom}(x)$  or  $x(t, j) \in M \mathbb{B}$  for all  $(t, j) \in \text{dom}(x)$  satisfying  $t + j \geq T$ .

Given a point  $x \in \mathbb{R}^n$  and a compact set  $\mathcal{A} \subset \mathbb{R}^n$ , we define the distance from  $x$  to  $\mathcal{A}$  as  $|x|_{\mathcal{A}} := \min_{y \in \mathcal{A}} |x - y|$ . Then, the set  $\mathcal{A}$  is said to be UGAS for a HDS (24.11) if there exists a class  $\mathcal{KL}$  function<sup>3</sup>  $\beta$  such that for all initial conditions  $x(0, 0) \in \mathbb{R}^n$

---

<sup>3</sup>A function  $\beta : \mathbb{R}_{\geq 0} \times \mathbb{R}_{\geq 0} \rightarrow \mathbb{R}_{\geq 0}$  is of class  $\mathcal{KL}$  if it is nondecreasing in its first argument, nonincreasing in its second argument,  $\lim_{r \rightarrow 0^+} \beta(r, s) = 0$  for each  $s \in \mathbb{R}_{\geq 0}$ , and  $\lim_{s \rightarrow \infty} \beta(r, s) = 0$  for each  $r \in \mathbb{R}_{\geq 0}$ .

all solutions of the HDS satisfies the bound

$$|x(t, j)|_{\mathcal{A}} \leq \beta(|x(0, 0)|_{\mathcal{A}}, t + j), \quad \forall (t, j) \in \text{dom}(x). \quad (24.12)$$

If the  $\mathcal{KL}$  function has the form  $\beta(r, s) = c_1 r \exp(-c_2 s)$  for some  $(c_1, c_2) \in \mathbb{R}_{>0}^2$ , then the set  $\mathcal{A}$  is said to be uniformly globally exponential stable (UGES). Similar characterizations exists for finite-time stability and fixed-time stability. It is important to note that the definitions of UGAS do not insist that every solution must have an unbounded time domain, since the bound (24.12) needs to hold only for points  $(t, j) \in \text{dom}(x)$ . However, if (24.12) holds and the solution  $x$  is complete, then  $x$  must converge to  $\mathcal{A}$ . Because of this reason, the property described by the bound (24.12) is also sometimes referred to as uniform global pre-asymptotic stability.

Similar to the continuous-time and discrete-time case, stability properties in HDS can also be characterized in terms of Lyapunov functions. Indeed, if there exists a continuously differentiable function  $V : \mathbb{R}^n \rightarrow \mathbb{R}_{\geq 0}$ , class  $\mathcal{K}_\infty$  functions  $\alpha_1(\cdot), \alpha(\cdot)$ , and a positive definite function  $\rho(\cdot)$ , such that

$$\alpha_1(|x|_{\mathcal{A}}) \leq V(x) \leq \alpha_2(|x|_{\mathcal{A}}), \quad \forall x \in C \cup D \cup G(D), \quad (24.13a)$$

$$\langle \nabla V(x), f \rangle \leq -\rho(|x|_{\mathcal{A}}), \quad \forall x \in C, f \in F(x), \quad (24.13b)$$

$$V(g(x)) - V(x) \leq -\rho(|x|_{\mathcal{A}}), \quad \forall x \in D, g \in G(x), \quad (24.13c)$$

then, the set  $\mathcal{A}$  is UGAS. Similar characterizations exist for UGES, and for UUB in terms of a property called recurrence [50].

**Example 24.10 (UGES of Hybrid Optimization Dynamics)** Consider again the hybrid optimization dynamics of Example 24.2, and suppose that the cost function has a Lipschitz gradient and it is  $\mu$ -strongly convex. Let  $\mathcal{A} := \{x^*\} \times \{x^*\} \times [T_{\min}, T_{\max}]$  and  $\phi^* = \min_{x_1 \in \mathbb{R}^n} \phi(x_1)$ . Let  $x = (x_1, x_2, x_3) \in \mathbb{R}^{2n+1}$ , and consider the Lyapunov function

$$V(x) = \frac{1}{4}|x_1 - x_2|^2 + \frac{1}{4}|x_2 - x^*|^2 + kx_3^2 (\phi(x_1) - \phi^*),$$

which is radially unbounded and positive definite with respect to the set  $\mathcal{A}$ . Moreover, if  $0 < T_{\min} < T_{\max}$  and  $T_{\max}^2 - T_{\min}^2 \geq \frac{1}{2\mu k}$  there exists positive constants  $(\lambda_1, \lambda_2)$  such that

$$\langle \nabla V(x), \dot{x} \rangle \leq -\lambda_1 V(x), \quad \forall x \in C, \quad (24.14a)$$

$$V(g(x)) - V(x) \leq -\lambda_2 V(x), \quad \forall x \in D. \quad (24.14b)$$

It then follows that the set  $\mathcal{A}$  is UGAS. Indeed, since in this case the Lyapunov function satisfies  $c_1|x|_{\mathcal{A}}^2 \leq V(x) \leq c_2|x|_{\mathcal{A}}^2$ , for some  $c_1, c_2 > 0$  the set  $\mathcal{A}$  is actually UGES [39].  $\square$

We finish this section by reviewing a fundamental robustness result for HDS that satisfy the Basic Assumptions. In order to state the result, we consider an “inflated” system

$$x \in C_\delta, \quad \dot{x} \in F_\delta(x) \quad (24.15a)$$

$$x \in D_\delta, \quad x^+ \in G_\delta(x), \quad (24.15b)$$

where the data  $(C_\delta, F_\delta, D_\delta, G_\delta)$  is defined as follows:

$$\begin{aligned} C_\delta &:= \{x \in \mathbb{R}^n : x + \delta\mathbb{B} \cap C \neq \emptyset\}, \\ F_\delta(x) &:= \overline{\text{co}} \ F((x + \delta\mathbb{B}) \cap C) + \delta\mathbb{B}, \quad \forall x \in C_\delta, \\ D_\delta &:= \{x \in \mathbb{R}^n : x + \delta\mathbb{B} \cap D \neq \emptyset\}, \\ G_\delta(x) &:= \{v : v \in g + \delta\mathbb{B}, g \in G((x + \delta\mathbb{B}) \cap D)\}, \quad \forall x \in D_\delta. \end{aligned}$$

System (24.15) can be seen as a perturbed version of the nominal dynamics (24.11), where the  $\delta$ -inflation can capture additive small bounded disturbances acting on the states and dynamics of the system, as well as approximation errors, uncertainty in the dynamics, etc. One of the key features of the HDS (24.11) satisfying the Basic Assumptions is that UGAS properties of compact attractors are preserved for the perturbed version (24.15) provided  $\delta$  is sufficiently small, i.e., if  $\mathcal{A}$  is UGAS for system (24.11), then for each compact set of initial conditions  $K \subset \mathbb{R}^n$  and each  $\varepsilon > 0$  there exists a  $\delta^* > 0$  such that for all  $\delta \in (0, \delta^*)$  every solution of system (24.15) with  $x(0, 0) \in K$  satisfies the bound

$$|x(t, j)|_{\mathcal{A}} \leq \beta(|x(0, 0)|_{\mathcal{A}}, t + j) + \varepsilon, \quad \forall (t, j) \in \text{dom}(x).$$

Similar characterizations can be obtained for input-to-state-stability (ISS) properties. Extensions to stochastic hybrid systems that combine diffusions and discrete-time random variables can be found in [51].

## 24.3 Reinforcement Learning via Dynamic Policy Gradient

In order to draw some potential connections between HDS and RL, we will focus on a class of policy gradient-based RL algorithms called actor–critic, which combine tools from adaptive and optimal control. These algorithms aim to find approximate solutions to the Hamilton–Jacobi–Bellman (HJB) equation related to an infinite horizon optimal control problem. In particular, we will consider the framework presented in [21, 25, 52], which borrows ideas from traditional policy iteration algorithms [14, 53], but which are applicable to continuous-time dynamical systems with a continuous spaces of actions.

To simplify our presentation, we focus on nonlinear dynamical systems affine on the input, given by

$$\dot{x} = f(x) + g(x)u, \quad (24.16)$$

where  $x \in \mathbb{R}^n$  is the state,  $u \in \mathbb{R}^m$  is the input, and the functions  $f : \mathbb{R}^n \rightarrow \mathbb{R}^n$  and  $g : \mathbb{R}^n \rightarrow \mathbb{R}^{n \times m}$  are assumed to be Lipschitz continuous. Moreover, we will assume

that  $f(0) = 0$ . Some of these assumptions can be relaxed in order to handle non-smooth systems.

In the literature of RL, the dynamics (24.16) describe the *environment*, and the input  $u$  is also called the *action*. The goal in RL is to maximize an accumulated long-term reward  $r(x, u)$  that depends on the state of the system  $x$  and the input  $u$ . This reward can be written as

$$r(x, u) = Q(x) + u^\top R u, \quad (24.17)$$

where  $Q : \mathbb{R}^n \rightarrow \mathbb{R}_{\geq 0}$  is positive for any  $x \neq 0$ , and satisfies  $Q(0) = 0$ , and  $R \in \mathbb{R}^{m \times m}$  is a symmetric positive definite matrix. By using the reward function (24.17), we can define the following accumulated cost functional:

$$J(x_0, u) := \int_0^\infty r(x(\tau), u(\tau)) d\tau, \quad (24.18)$$

where  $x_0 = x(0)$ , and where the function  $x$  that appears at the right-hand side of (24.18) is the solution of (24.16) from the initial condition  $x_0$  under the input  $u : \mathbb{R}_{\geq 0} \rightarrow \mathbb{R}^m$ . The main goal in RL is to find an *admissible feedback* law  $u = \alpha(x)$  that guarantees asymptotic stability of the origin for system (24.16), with a basin of attraction containing a pre-defined compact set  $K$ , while simultaneously minimizing the cost (24.18). Other objectives such as tracking or synchronization in multi-agent systems can be incorporated under suitable modifications of (24.18).

The solution of the RL problem is characterized in terms of *admissible controllers*. A feedback law is said to be *admissible* in a compact set  $K \subset \mathbb{R}^n$  and with respect to the cost (24.18) if: (a) the mapping  $x \mapsto \alpha(x)$  is continuous; (b)  $\alpha(0) = 0$ ; (c) for every  $x_0 \in K$  the control law  $u = \alpha(x)$  stabilizes the origin for system (24.16); and (d) the value (24.18) generated by  $u = \alpha(x)$  is finite for every  $x_0 \in K$ . The optimal *value function* associated to the optimal control problem is then defined as

$$V^*(x(t)) := \inf_{u(\tau): \tau \in \mathbb{R}_{\geq t}} \int_t^\infty r(x(\tau), u(\tau)) d\tau. \quad (24.19)$$

Assuming that an optimal admissible controller exists such that  $V^*$  is finite, the optimal value function can be characterized in terms of the HJB equation:

$$0 = \min_u (r(x, u) + \nabla V^*(x)^\top (f(x) + g(x)u)), \quad \forall x \in K, \quad V(0) = 0. \quad (24.20)$$

Whenever Eq.(24.20) admits a  $C^1$  solution  $V^*$ , it characterizes a sufficient and necessary condition for optimality. Note that Eq.(24.20) can be written as

$$0 = \min_u H(x, u, -\nabla V^*(x)),$$

where the Hamiltonian  $H$  is defined as  $H(x, u, p) := \langle p, \dot{x} \rangle - r(x, u)$ . For a given  $V^*$ , and under the structure of (24.17), it is easy to see that the controller that

minimizes the Hamiltonian is given by

$$u^* = \arg \min_u H(x, u, -\nabla V^*(x)) = -\frac{1}{2} R^{-1} g(x)^\top \nabla V^*(x). \quad (24.21)$$

Substituting Eq. (24.21) in Eq. (24.20), we obtain the coupled HJB equation:

$$0 = Q(x) + \nabla V^*(x)^\top f(x) - \frac{1}{4} \nabla V^*(x)^\top g(x) R^{-1} g(x)^\top \nabla V^*(x), \quad V^*(0) = 0. \quad (24.22)$$

If the plant dynamics (24.16) are linear, and if the function  $Q(x)$  is quadratic, Eq. (24.22) corresponds to the standard Riccati equation, which can be solved under different controllability, observability, detectability, or stabilizability conditions on the system, see [54, 55]. For linear systems, there exist several efficient numerical methods for the solution of the Riccati equation [56–58]. However, finding explicit solutions of Eq. (24.22) for nonlinear systems is hopeless in all but a few special cases. This difficulty has motivated the development of different iterative techniques that start with approximations  $\hat{V}$  and  $\hat{u}$ , and iteratively refine the approximations aiming to converge to the input and the value  $(u^*, V^*)$  that satisfy (24.22). One of the most common approaches that follows this philosophy is policy iteration (PI).

### 24.3.1 Asynchronous Policy Iteration

In asynchronous PI, two different steps are recursively executed. First, in the evaluation step, given an initial admissible control law  $\alpha^{i-1}(x)$ , the *generalized HJB equation*, given by

$$0 = r(x, \alpha^{i-1}(x)) + \nabla V^i(x)^\top (f(x) + g(x)\alpha^{i-1}(x)), \quad \forall x \in K, \quad V^{i+1}(0) = 0, \quad (24.23)$$

is solved for the function  $V^i(x)$ . This function aims to serve as an evaluation of  $\alpha^{i-1}$ . Once the function  $V^i$  has been obtained, the control law is updated as  $\alpha^i = -\frac{1}{2} R^{-1} g(x)^\top \nabla V^i(x)$ . Using this new feedback law, Eq. (24.23) is solved again for  $V^{i+1}(x)$ , which, in turn, is used again to update the controller to generate  $\alpha^{i+1}$ . These steps are repeated sequentially until the pair of sequences  $(\{V^i\}, \{\alpha^i\})_{i=1}^\infty$  eventually converges. As shown in [59], under mild assumptions on the reward function (24.17), and the dynamics (24.16), the limits of these sequences exist, and they correspond to the optimal value  $V^*$  and the optimal input  $u^*$ .

While the synchronous PI algorithm has been used with great success in different applications, guaranteeing stability of the closed-loop system during the learning phase is always a challenge. This difficulty motivated the development of *synchronous* PI techniques, where the plant dynamics, the evaluation step, and the updates of the controller take place simultaneously while preserving the stability properties of the closed-loop system.

### 24.3.2 Synchronous Policy Iteration: Online Training of Actor–Critic Structures

In order to simultaneously solve the HJB Eq. (24.20) while controlling the dynamics (24.16) in a stable manner, an online actor–critic-based RL algorithm was presented in [21]. In this approach, the value function is parameterized as follows:

$$V(x) = W^\top \phi(x) + \varepsilon(x), \quad \forall x \in K, \quad (24.24)$$

where  $W \in \mathbb{R}^p$  is a vector of weights,  $\phi : \mathbb{R}^n \rightarrow \mathbb{R}^p$  is a vector of basis functions, and  $\varepsilon(x)$  is the approximation error. By the universal approximation theorem [60], any smooth function  $V$  can be approximated as (24.24) on compact sets  $K$  and with any  $e$ -precision, i.e., there exists a sufficiently large  $p$  such that  $\sup_{x \in K} |\varepsilon(x)| \leq e$ . Naturally, this approximation property extends to the gradient of  $V$ , i.e.,

$$\nabla V(x) = \nabla \phi(x)^\top W + \nabla \varepsilon(x), \quad \forall x \in K.$$

In particular, as  $p \rightarrow \infty$  the approximation errors  $\varepsilon$  and  $\nabla \varepsilon$  converge to zero, uniformly on  $K$ . By exploiting this approximation property, one can work with estimations of  $V$  and  $\nabla V$ , given by

$$\hat{V}(x) = \hat{W}_c^\top \phi(x), \quad \text{and} \quad \nabla \hat{V}(x) = \nabla \phi(x)^\top \hat{W}_c, \quad (24.25)$$

where  $\hat{W}_c$  is an estimation of the true weights  $W$ . For the value function, the estimation process is performed by a subsystem called the *critic*. Using (24.25), the approximation error in the gradient of the value function can be written as

$$\rho(x, \hat{W}_c) := \nabla V(x) - \nabla \hat{V}(x) = \nabla \phi(x)^\top (W - \hat{W}_c) + \nabla \varepsilon(x),$$

and the generalized HJB Eq. (24.23) can be written as

$$0 = r(x, u) + \left( \nabla \hat{V}(x) + \rho(x, \hat{W}_c) \right)^\top (f(x) + g(x)u), \quad \forall x \in K, \quad V(0) = 0. \quad (24.26)$$

The *Bellman error* is then defined as  $\delta(x, \hat{W}_c) := -\rho(x, \hat{W}_c)^\top (f(x) + g(x)u)$ , and Eq. (24.26) becomes

$$\delta(x, \hat{W}_c) = r(x, u) + \hat{W}_c^\top \nabla \phi(x)(f(x) + g(x)u), \quad \forall x \in K, \quad V(0) = 0. \quad (24.27)$$

By defining  $w := \nabla \phi(x)(f(x) + g(x)u)$  as a regressor vector, Eq. (24.27) can be written in compact form as

$$\delta(x, \hat{W}_c) = r(x, u) + \hat{W}_c^\top w, \quad (24.28)$$

which illustrates one of the main advantages of the parameterization (24.25): the right-hand side of (24.28) is linear in  $\hat{W}_c$ . Based on this, a suitable approach to minimize the parameter estimation error  $\tilde{W}_c = W - \hat{W}_c$  is to minimize the Bellman error.

### 24.3.2.1 Learning Dynamics for the Critic

In order to minimize the Bellman error in (24.28) for a fixed input  $u$ , the estimate  $\hat{W}_c$  can be updated by using learning dynamics based on parameter estimation algorithms used in adaptive control, e.g., [61, Chap. 2], [62, Chap. 4]. For instance, one could consider minimizing the quadratic Bellman error  $J(\delta) := \frac{1}{2}\delta(x, \hat{W}_c)^2$  by using normalized gradient descent, i.e.,

$$\hat{W}_c \in C := \mathbb{R}^p, \quad \dot{\hat{W}}_c = f_c(x, w, \hat{W}_c) := -\gamma \frac{w}{(1 + w^\top w)^2} \delta(x, \hat{W}_c). \quad (24.29)$$

Indeed, by exploiting the linear structure of (24.28), and standard results for adaptive control, it can be shown that if the signal  $w$  is *persistently exciting* (PE), i.e., if there exist positive constants  $(\alpha_0, \alpha_1, T)$  such that

$$\alpha_1 I \geq \int_{t_0}^{t_0+T} w(t) w(t)^\top dt \geq \alpha_0 I, \quad \forall t_0 \geq 0,$$

then the estimation error  $\tilde{W}_c = W_c - \hat{W}_c$  will converge exponentially fast to a neighborhood of zero, where the size of the neighborhood is proportional to the approximation error  $\varepsilon(x)$  induced by the parameterization (24.24). Other learning dynamics that could be considered for  $\hat{W}_c$  include gradient descent with projection, or least-squares dynamics with forgetting factor or covariance resetting.

### 24.3.2.2 Learning Dynamics for the Actor

In order to guarantee stability of the nonlinear plant dynamics (24.16), a suitable feedback law is needed. By (24.21), the optimal controller that minimizes the Hamiltonian can be written as

$$\hat{u} := -\frac{1}{2} R^{-1} g(x)^\top \nabla \phi(x)^\top \hat{W}_a, \quad (24.30)$$

where  $\nabla \phi^\top \hat{W}_a$  is an approximation of the unknown gradient  $\nabla V$  performed by a different subsystem called the *actor*. Since the basis functions  $\phi$  are fixed, the actor also updates its estimate  $\hat{W}_a$  aiming to minimize the Bellman error. Using different estimates of  $W$  for the actor and the critic allows to obtain a Bellman error that is linear in  $\hat{W}_c$ , and quadratic in  $\hat{W}_a$ , which facilitates the stability analysis of the closed-loop system. Indeed, using (24.30) as input in (24.16) and (24.17), the Bellman error in the closed-loop system becomes

$$\begin{aligned}\delta(x, \hat{W}_c, \hat{W}_a) = & Q(x) + \frac{1}{4} \hat{W}_a^\top \nabla \phi(x) g(x) R^{-1} g(x)^\top \nabla \phi(x)^\top \hat{W}_a \\ & + \hat{W}_c^\top \nabla \phi(x) f(x) - \frac{1}{2} \hat{W}_c^\top \nabla \phi(x) g(x) R^{-1} g(x)^\top \nabla \phi(x)^\top \hat{W}_a,\end{aligned}\quad (24.31)$$

and its partial derivative with respect to  $\hat{W}_a$  satisfies  $\frac{\partial \delta(x, \hat{W}_c, \hat{W}_a)}{\partial \hat{W}_a} = \frac{1}{2} G(x)(\hat{W}_a - \hat{W}_c)$ , where

$$G(x) = \nabla \phi(x) g(x) R^{-1} g(x)^\top \nabla \phi(x)^\top. \quad (24.32)$$

Based on this, one can consider the following learning dynamics for the actor [25]:

$$\dot{\hat{W}}_a = f_a(x, \hat{W}_a, \hat{W}_c) := -\frac{\gamma_1 G(x)(\hat{W}_a - \hat{W}_c)\delta(x, \hat{W}_c, \hat{W}_a)}{\sqrt{1 + w^\top w}} - \gamma_2(\hat{W}_a - \hat{W}_c), \quad (24.33)$$

where  $\gamma_1 > 0$  and  $\gamma_2 > 0$ , and where the last term is added to guarantee stability of the closed-loop system, see [25]. Alternatively, the actor could also consider the learning dynamics

$$\dot{\hat{W}}_a = f'_a(x, \hat{W}_a, \hat{W}_c) := \gamma_1 G(x) \hat{W}_a w^\top \hat{W}_c - \gamma_2 (\hat{W}_a - \hat{W}_c) - \gamma_3 \hat{W}_a, \quad (24.34)$$

with  $\gamma_i > 0$ , for all  $i \in \{1, 2, 3\}$ , which are designed based on the Lyapunov analysis of the closed-loop system.

### 24.3.2.3 Closed-Loop System and Extensions to Other Settings

Once the dynamics of the actor and the critic have been designed, the stability properties of the closed-loop system, which combines the plant dynamics (24.16), the critic learning dynamics, and the actor learning dynamics, can be studied using Lyapunov tools. In particular, the closed-loop can be written as the continuous-time dynamical system of the form

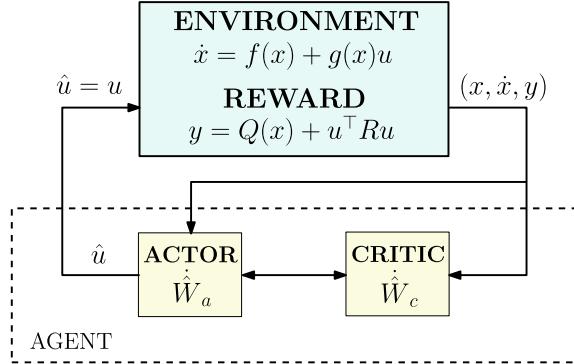
$$\dot{x} = f(x) + g(x)\hat{u}, \quad (24.35a)$$

$$\dot{\hat{W}}_c = f_c(x, \hat{W}_a, \hat{W}_c) \quad (24.35b)$$

$$\dot{\hat{W}}_a = f_a(x, \hat{W}_a, \hat{W}_c), \quad (24.35c)$$

where  $\hat{u}$  is defined as in (24.30), and where the functions  $f_c$  and  $f_a$  characterize the learning dynamics of the critic and the actor, respectively.

For instance, for a system implementing the critic dynamics (24.29), and the actor dynamics (24.34), the function  $V(x) = V^*(x) + \frac{1}{2} \tilde{W}_c^\top \Gamma^{-1} \tilde{W}_c + \frac{1}{2} \tilde{W}_a^\top \tilde{W}_a$  is a suitable Lyapunov function to establish uniform ultimate boundedness for the overall state  $z = (x, \hat{W}_c, \hat{W}_a)$ , with an ultimate bound being proportional to the approxi-



**Fig. 24.4** Continuous-time model of a closed-loop architecture where a dynamical system interacts with a RL-based algorithm that implements a dynamic critic and a dynamic actor

mation error induced by the neural network. Since the dynamics of the critic and the actor are based on parameter estimation dynamics used in adaptive control, convergence to a neighborhood of the true parameters requires the satisfaction of a PE condition along the trajectories of the system. In practice, this condition is induced by adding a dithering signal to the input. Once the parameters  $\hat{W}_c$  and  $\hat{W}_a$  have converged to a neighborhood of their true values, the dither signal is turned off (note that removing the excitation also removes the uniformity in the convergence). Recent approaches have exploited ideas from concurrent learning in order to dispense with the PE assumption [22]. Figure 24.4 shows a scheme illustrating the closed-loop interconnection between the different components of the system.

If the dynamics of the plant (i.e., environment) evolve in discrete time rather than in continuous time, it is still possible to construct an equivalent set of equations of the form (35) by using Bellman's principle of optimality, and by modeling the dynamics of the critic and the actor as recursions. This approach is pursued in [63–65], to name just a few examples. In this case, the closed-loop system can be written as

$$x^+ = f(x) + g(x)\hat{u}, \quad (24.36)$$

$$\hat{W}_c^+ = \hat{W}_c + \alpha f_c(x, \hat{W}_a, \hat{W}_c) \quad (24.37)$$

$$\hat{W}_a^+ = \hat{W}_a + \alpha f_a(x, \hat{W}_a, \hat{W}_c), \quad (24.38)$$

where  $\alpha > 0$  is a small discretization step size. The stability properties of this system can be analyzed using Lyapunov conditions of the form (24.13a) and (24.13c), see [66, 67].

#### 24.3.2.4 Extensions to Other Settings

Since the adaptive actor–critic approach relies on an underlying well-posed infinite horizon optimal control problem, it is natural to expect that similar optimal regulation or tracking problems can also be addressed by the RL actor–critic formalism. For instance, when the plant dynamics are given by the perturbed system  $\dot{x} = f(x) + g(x)u + \kappa(x)d$ , with  $d$  acting as a disturbance, an optimal and robust feedback law  $u = \alpha(x)$  can be obtained by solving the following min-max problem

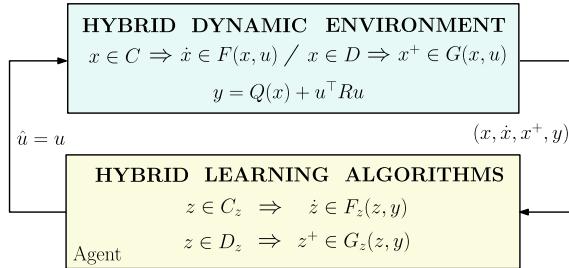
$$\min_u \max_d \int_0^\infty (Q(x) + u^\top Ru - \gamma^2|d|^2)dt,$$

which can be modeled as a zero-sum game between the controller and the disturbance. When a solution to this problem exists, it can be characterized in game-theoretic terms as a *Nash equilibrium*. This Nash equilibrium can be found in real time by implementing a synchronous policy iteration algorithm that incorporates three neural networks, see [68]. The combination of game-theoretic tools and RL-based algorithms can also be exploited to analyze synchronization, coordination, and competition problems in multi-agent networked systems [69, 70].

We finish this section by emphasizing that the actor–critic approach shown in Fig. 24.4 is, by nature, a model-based approach. This is the case because the learning dynamics of the actor and the critic need information about the right-hand side of the plant dynamics (24.16). In order to alleviate this problem, one can consider integral methods, dynamic identifiers, or Q-learning-based approaches, see [25, 71–73].

## 24.4 Reinforcement Learning in Hybrid Dynamical Systems

The main ideas behind the actor–critic approach presented in the previous section are rooted in well-established theoretical results in optimal control, adaptive control, and stability theory of continuous-time and discrete-time nonlinear systems. Nevertheless, a general extension of the actor–critic approach to problems that have nontrivial interactions between continuous-time dynamics and discrete-time dynamics remains absent. Some of the reasons behind this limitation include a lack of general and tractable frameworks for the solution of infinite horizon hybrid optimal control problems using ideas from dynamic programming, and the fairly recent development of analytical tools for nonlinear hybrid dynamical systems that parallel the results developed for purely continuous-time systems and purely discrete-time systems. Motivated by this background, in this section we discuss some potential avenues where we expect to see novel interactions between RL and HDS during the next years. In some cases, such as in sampled-data and event-triggered RL, these interactions are already emerging [72, 74, 75]. However, several domains of hybrid RL remain unexplored. While a general hybrid RL framework should incorporate hybrid dynamics in the environment and in the algorithms, in order to organize our discussion we will differentiate between RL architectures where the hybrid dynam-



**Fig. 24.5** An utopic scheme of hybrid reinforcement learning (HRL) for cyber-physical systems. A dynamic plant with continuous-time dynamics and discrete-time dynamics interacts with a hybrid dynamic controller or algorithm. The discrete-time dynamics can model communication protocols, switching optimization algorithms, dynamic logic states, etc. The closed-loop system can be modeled using the formalism of Sect. 24.2

ics emerge exclusively in the environment, and settings where the hybrid dynamics emerge exclusively as part of the learning algorithm designed to solve the approximate optimal control problem (24.18). Figure 24.5 shows an utopic scheme of a hybrid RL problem, where a dynamic plant, with possibly hybrid dynamics, interacts with a hybrid RL algorithm. We expect that different implementations of RL algorithms in cyber-physical systems will be captured by this scheme. We also stress that the interaction described in Fig. 24.4 is a subset of the framework described by Fig. 24.5.

### 24.4.1 Hybrid Learning Algorithms

Online learning algorithms that combine continuous-time dynamics and discrete-time dynamics have the potential to solve problems that otherwise cannot be solved by using standard continuous-time or discrete-time approaches. For classic stabilization problems, this potential has already been exploited in [44, 76–79], to name just a few examples where different hybrid control mechanisms have been presented. Hybrid algorithms can also model realistic scenarios that emerge in practical implementations where computational devices make the presence of discrete-time dynamics unavoidable. In this section, we describe some potential directions where hybrid algorithms or architectures could be used to efficiently solve the adaptive optimal control problem described in Sect. 24.3.

#### 24.4.1.1 Sampled-Data and Event-Triggered Architectures

In practice, most physical systems modeled by ODEs are interconnected with feedback controllers via sampled-data systems that incorporate a zero-order hold device

and a sampling mechanism. As discussed in Example 24.3 of Sect. 24.2, sampled-data systems are naturally modeled as hybrid dynamical systems. In the actor–critic approach, a classic periodic sampled-data structure will update the discrete-time dynamics of the actor and the critic only when the clock of the sampling mechanism triggers the controller. Sampled-data systems can be designed either by emulation or by direct design [80–82]. In the emulation approach, stability properties are first established for an ideal continuous-time closed-loop system. After this, by using suitable discretization techniques and sufficiently small step sizes, the sampled-data structured is analyzed. For controllers that are not designed based on emulation techniques, hybrid Lyapunov functions provide a useful tool for the analysis and design of sampled-data systems, see [38, Example 3.21]. Different techniques for the analysis of sampled-data systems via emulation in nonlinear and adaptive control are presented in [83–85]. Optimal sample-data systems are studied in detail in [86]. Applications to actor–critic methods with periodic sampling can be found in [87].

Sampled-data systems that are not periodic have received significant attention during the last years due to their ability for saving computational resources and/or improving the transient performance of the system. In a class of non-periodic sampled-data systems, called *event-triggered*, the control action is updated only when a particular event occurs in the closed-loop system. This event is usually expressed in terms of a triggering signal that is being continuously or periodically monitored. Event-triggered methods can be used to either improve the transient performance of the system by triggering the controller at faster rates [41], or to minimize the computational and communication resources required by the algorithm by triggering the controller only when a particular tracking or regulation error is sufficiently large [88]. As shown in [41, 89, 90], event-triggered methods can be modeled as HDS of the form (24.11) provided the data  $(C, F, D, G)$  satisfies the Basic Assumptions. When stability properties such as UGAS, UGES, or UUB are established for the closed-loop system, stability results for the inflated system (24.15) follow directly, which establishes positive margins of robustness with respect to measurements on the triggering signals. In the context of synchronous actor–critic methods, different types of model-based and model-free event-triggered algorithms have been recently developed in [72, 74, 91], which are analyzed as impulsive systems, a subclass of HDS of the form (24.11).

#### 24.4.1.2 Hybrid Coordination of Multi-agent Reinforcement Learning Algorithms

In the context of multi-agent systems (MASs) with limited information, RL-based controllers based on sampled-data structures may need additional coordination and synchronization mechanisms in order to guarantee a robust behavior for the overall MAS. In fact, as shown in [92], even for simple linear systems, lack of perfect coordination in multi-agent sampled-data systems with identical clock dynamics can lead to unstable behaviors under arbitrarily small disturbances. As shown in [92, 93], HDS of the form (24.11) are well-suited for the design and analysis of distributed coor-

dination algorithms. An example of a synchronization and coordination mechanism for model-free optimization algorithms in networked dynamical systems is presented in [94]. We envision that similar results will be developed for sampled-data actor–critic algorithms in MAS when there exist individual admissible controllers  $\alpha_i(x)$  that preclude finite escape times during any period where the clocks of the sampling mechanisms of the agents are not synchronized. The development of related asynchronous and robust sampled-data actor–critic algorithms for coordination and control of continuous-time multi-agent systems is still an active area of research.

#### 24.4.1.3 Hybrid Optimization and Estimation Algorithms

Learning and estimation algorithms that combine continuous-time dynamics and discrete-time dynamics have been used in system identification and adaptive control since the early 80s and 90s, see, for instance, [61, p. 63], [62, Sect. 4.6]. One of the most well-known hybrid learning dynamics corresponds to the least-squares method with covariance resetting. In the context of actor–critic RL, this hybrid learning algorithm has been successfully used to update the weights of the critic by using the following hybrid dynamics [25, Chap. 3]:

$$C = \left\{ \hat{W}_c \in \mathbb{R}^p, P \in \mathbb{R}^{p \times p} : \lambda_{\min}(P(t)) \geq \bar{\lambda} \right\}, \quad \begin{cases} \dot{\hat{W}}_c = -k_c P \frac{w}{1 + \gamma w^\top P w} \delta, \\ \dot{P} = -k_c P \frac{ww^\top}{1 + \gamma w^\top P w} P, \end{cases}$$

$$D = \left\{ \hat{W}_c \in \mathbb{R}^p, P \in \mathbb{R}^{p \times p} : \lambda_{\min}(P(t)) \leq \underline{\lambda} \right\}, \quad \begin{cases} \dot{\hat{W}}_c^+ = \hat{W}_c^+, \\ P^+ = \gamma I, \end{cases}$$

where  $\underline{\lambda} < \bar{\lambda} < \gamma$  are tunable parameters. These dynamics reset the covariance matrix  $P$  whenever its inverse is sufficiently close to becoming singular, avoiding the so-called covariance wind-up problem.

The idea of resetting some of the states of a dynamical system in order to improve transient performance has recently received significant attention in the context of reset control [95, 96]. As a matter of fact, these ideas can also be exploited to improve transient performance in optimization and learning algorithms. For instance, in [97] a hybrid heavy-ball algorithm has shown to guarantee global convergence for a broader class of cost functions compared to the traditional heavy-ball dynamics. Similarly, in [98, 99] centralized and distributed hybrid regularizations of Nesterov’s ODE have shown to guarantee suitable robustness properties and semi-acceleration for a class of convex optimization problems. A robust Hamiltonian-based hybrid algorithm (HHA) with fast rates of convergence was also developed in [40]. The HHA achieves acceleration by synergistically merging momentum and special structures of the flow set and the jump set. The resulting behavior exhibits a rate of convergence that, under discretization via symplectic integrators, can compete with some of the fastest known discrete-time optimization algorithms in the literature, e.g., [100]. Hybrid optimization algorithms can also be used to overcome lack of con-

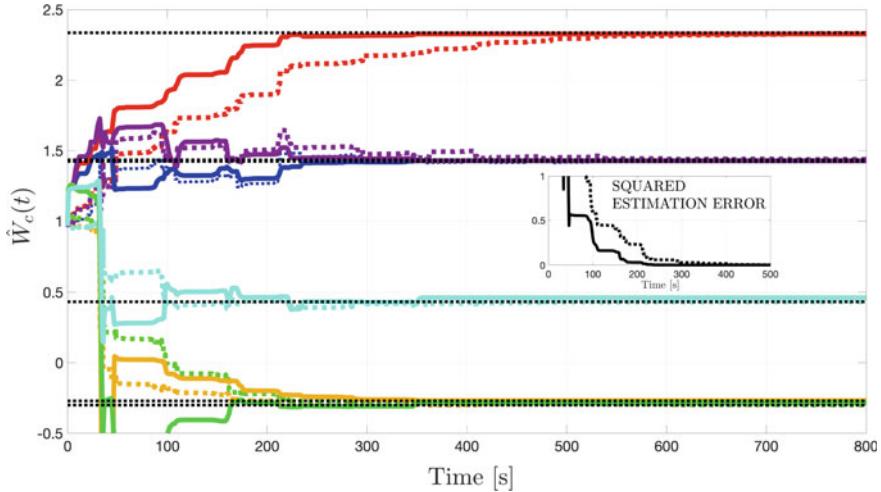
vexity in optimization problems. For instance, in [46] it was shown that a hybrid gradient descent algorithm with a hysteresis mechanism can guarantee convergence to a global minimizer in a class of smooth cost functions defined on the unit circle. Hybrid optimization algorithms have also been developed for optimization under non-convex safety constraints [45], as well as under adversarial settings with persistent disturbances using time–ratio constraints [33]. Finally, hybrid optimization algorithms that incorporate randomness have been shown to guarantee global convergence to global minimizers in a class of non-convex optimization problems defined on smooth compact manifolds [101, 102].

The previous discussion illustrates the existence of different exciting research opportunities for the development of robust and efficient policy-gradient methods that incorporate hybrid dynamics. Since the learning dynamics of the actor and the critic are usually designed as gradient systems, it is natural to expect that they also inherit the same limitations that are present in standard gradient descent optimization dynamics, i.e., slow rates of convergence, lack of global convergence in non-convex optimization problems, etc. Indeed, these limitations have been observed in existing actor–critic approaches. We expect that a synergistic incorporation of hybrid dynamics in the existing actor–critic RL algorithms could potentially improve the performance of the closed-loop system, making the algorithms suitable for real-time implementations in cyber-physical systems where fast rates of convergence are critical, e.g., autonomous vehicles, smart transportation systems, etc. Promising numerical results are shown in Figs. 24.6, 24.7, and 24.8. In these experiments the actor–critic RL approach developed in [21] is compared with a similar algorithm that uses hybrid dynamics to accelerate the training of the critic. As shown in Figs. 24.6 and 24.7 with solid lines, the hybrid dynamics induce a faster convergence of the weights  $\hat{W}_c$  and  $\hat{W}_a$ , which is evidenced in the total squared estimation errors  $|\tilde{W}_c|^2$  and  $|\tilde{W}_a|^2$  shown in the insets. Both simulations were run using the same parameters and excitation signals considered in [21]. In our case, the amplitude of the excitation signal is turned off once the total square estimation error is less than  $1 \times 10^{-4}$ . For the standard actor–critic approach, this event occurs approximately after 720 s, whereas for the hybrid approach this occurs after approximately 350 s. The resulting evolution of the states of the plant is shown in Fig. 24.8. As it can be observed, the hybrid approach reduces the duration of the training phase by almost half.

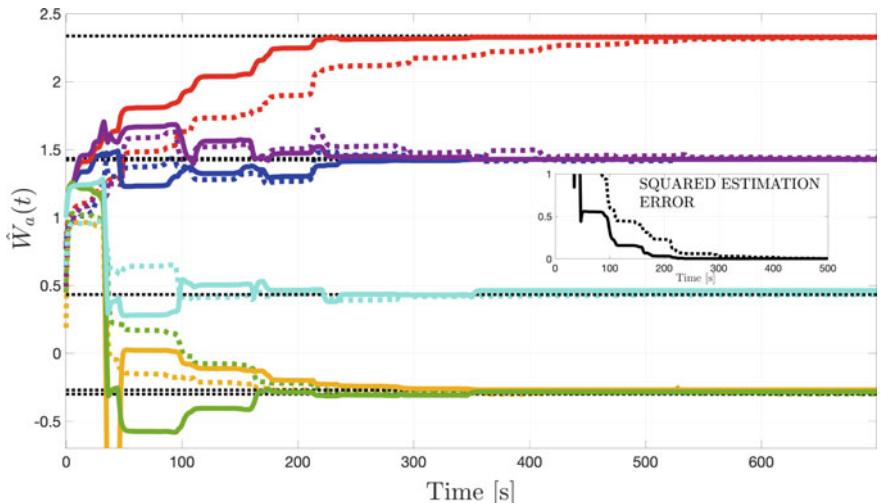
We finish this subsection by emphasizing that the combination of hybrid techniques with random inputs could also have a significant impact in the convergence properties of RL algorithms with multi-layer neural networks, i.e., deep RL, where the convexity of the Bellman error with respect to the weights of the actor is lost in most of the cases.

#### 24.4.2 Hybrid Dynamic Environments

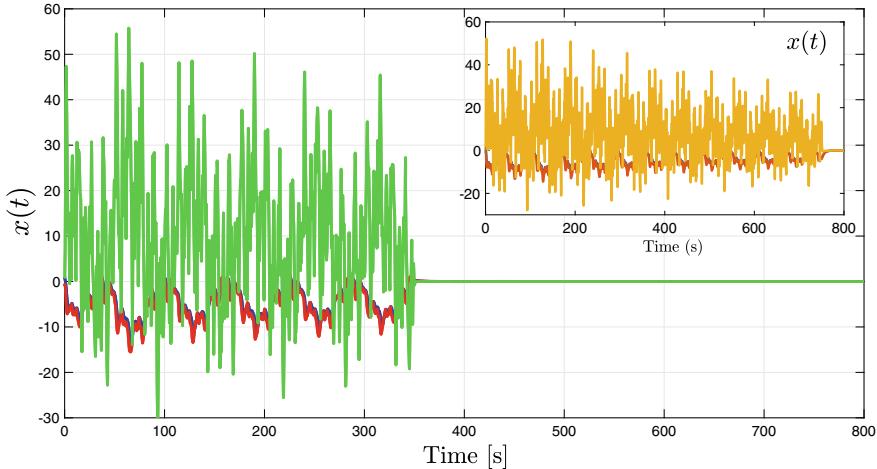
While there has been significant recent progress in the advancement of hybrid algorithms for classic estimation, optimization, and control, the development of a com-



**Fig. 24.6** Evolution in time of the weights  $\hat{W}_c$  of the critic, updated via a hybrid learning mechanism (solid curve) and the standard gradient descent mechanism considered in [21] (dashed curve), using the same initial conditions and excitation signal. The states of the hybrid dynamics converge to their true value after approximately 360 s



**Fig. 24.7** Evolution in time of the weights  $\hat{W}_a$  of the actor, with (solid line) and without (dashed line) hybrid dynamics in the critic, using the same initial conditions and excitation signal. Since the dynamics of the actor and the critic are coupled, the fast convergence of the critic also improves the transient performance of the actor



**Fig. 24.8** Evolution in time of the states of the plant (24.16) under an actor–critic with hybrid dynamics in the critic. Since the input is excited to guarantee the PE condition, the states oscillate during an initial transient phase. The inset shows the standard non-hybrid case. In both simulations, the excitation signal is turned off after the estimation squared error is below  $1 \times 10^{-4}$

prehensive framework for hybrid optimal control is still at its infancy, particularly for infinite horizon problems. Since, as shown in Sect. 24.3, the standard actor–critic architectures for continuous-time systems and discrete-time systems exploit the optimality conditions induced by the HJB Eq. (24.20), in order to develop suitable actor–critic RL techniques for hybrid plants, fundamental preliminary results for hybrid optimal control and hybrid dynamic programming need to be developed. To illustrate some of the potential challenges that emerge in optimal hybrid control, we can consider the following HDS with linear flow map, linear jump map, and periodic jumps triggered by the clock  $\tau$ :

$$\tau \in [0, T], \quad \begin{cases} \dot{x} = Ax + Bu_F, \\ \dot{\tau} = 1, \end{cases} \quad (24.39a)$$

$$\tau = T, \quad \begin{cases} x^+ = Ex + Fu_J, \\ \tau^+ = 0, \end{cases} \quad (24.39b)$$

where  $T > 0$  characterizes the period of the jumps. Systems of this form have been studied in [103, 104]. In order to eliminate any dependence on the initial condition of the clock, we can set  $\tau(0, 0) = 0$ , which will generate the hybrid time domains shown in the upper left corner of Fig. 24.3, which corresponds to  $\cup_{j \in \mathbb{Z}_{\geq 0}} [jT, (j+1)T]$ . For this HDS, we can consider the following quadratic cost:

$$\begin{aligned} J(x_0, u_F, u_J) = & \int_0^\infty (x(t, j)^\top Q_F x(t, j) + u_F(t, j)^\top R_F u_F(t, j)) dt \\ & + \sum_{k=1}^{\infty} x(t_k, j-1)^\top Q_J x(t_k, j-1) + u_J(t, j)^\top R_J u_J(t, j), \end{aligned}$$

where  $x_0 = x(0, 0)$ , the matrices  $Q_F$  and  $Q_J$  are symmetric and positive semidefinite, and the matrices  $R_F$ ,  $R_J$  are symmetric and positive definite. As shown in [103, 104], under some stabilizability and detectability assumptions, system (24.39) admits optimal stabilizing controllers  $u_F^*$  for the flows and  $u_J^*$  for the jumps, given by

$$u_F^*(t, j) = -R_F^{-1} B' K(\tau(t, j)) x(t, j), \quad u_J^*(j) = -(R_J + F' K(0) F)^{-1} F' K(0) E x(t, j),$$

where  $K(\tau)$  is a state-dependent matrix that solves a particular hybrid Riccati equation. The structure of this controller suggests that even for a HDS with time-invariant dynamics and time-invariant cost, the optimal infinite horizon controller generates state-dependent optimal gains  $K$ . This fact already highlights some potential challenges for the direct extension of the actor–critic approach of Sect. 24.3 to the setting of hybrid systems since the discontinuities of  $\tau$  and  $x$  could be problematic for the approximation of the controller via standard neural networks. Moreover, for HDS of the form (24.39) the hybrid Riccati equation is actually a dynamic equation, even though the time horizon of the cost is infinite. One would expect that for HDS with nonlinear flow maps and jump maps the complexity of the problem would increase dramatically, even if the dynamics have the affine structure of Eq. (24.16).

On the other hand, infinite horizon formulations for nonlinear hybrid dynamic programming have been studied in [26, 27, 105, 106], to name just a few. In [26], the authors presented a general framework for controlled hybrid dynamical systems with a nonlinear infinite horizon discounted cost. Under certain conditions on the data of the hybrid system, existence and optimality characterizations for the controllers were provided, as well as sufficient optimality conditions expressed in terms of value functions satisfying a set of quasi-variational inequalities (QVI). Continuity and uniqueness properties of the solutions of these QVIs were later studied in [105, 106]. Harnessing these results for the development of novel hybrid RL techniques remains an open problem.

Finally, recent results in the context of finite horizon hybrid dynamic programming have been presented in [30] for a class of hybrid optimal control problems with a pre-determined sequence of switchings. A finite horizon LQR problem for multi-modal hybrid systems is studied in [107] in terms of a multi-modal jumping differential Riccati equation. Optimal control problems in event-driven systems were studied in [29] in the context of manufacturing systems. A common approach to address hybrid optimal control problems is to discretize the state space. This approach was studied in [108] using tools from convex dynamic programming. A family of finite horizon hybrid optimal control problems were also studied in [28] using a Hamiltonian approach, as well as in [109] for systems with a continuous space partitioned by switching manifolds. Optimal control of dynamical systems

with switching logic states and discretized flows is studied in [110, 111]. Finite horizon hybrid optimal control is intrinsically related to hybrid model-predictive control. Potential connections between MPC and RL are discussed in [19].

## 24.5 Conclusions

In this chapter, we have reviewed some of the main principles behind hybrid dynamical systems and actor–critic reinforcement learning methods for approximate optimal control. We have shown that hybrid systems provide a powerful framework for modeling and analyzing a broad class of systems, including cyber-physical systems, control mechanisms, optimization and estimation algorithms, and simple physical systems such as a bouncing ball. Thus, the development of reinforcement learning techniques for these types of systems could have significant impact in accelerating the deployment of the next generation of autonomous cyber-physical systems. By using the analytical framework of Sect. 24.2, we discussed different potential research directions where hybrid systems and reinforcement learning could be synergistically combined in order to generate closed-loop systems with high performance and desirable robustness properties. We illustrated this idea in Sect. 24.4 by numerical results that highlight the potential advantages of using hybrid algorithms to solve RL problems in the context of actor–critic techniques. We anticipate that some of these research directions will be actively pursued during the next years. Likewise, several RL problems that incorporate hybrid dynamics in the environment remain open. Promising applications of hybrid RL in cyber-physical systems include the smart-grid, autonomous vehicles, and intelligent transportation systems, to name just a few.

**Acknowledgements** The first author would like to thank John Hauser, Ashutosh Trivedi, and Fabio Somenzi for fruitful conversations about optimal control, dynamic programming, and reinforcement learning from the controls and computer science perspective. The first author acknowledges support from NSF via the grant CNS-1947613. The second author acknowledges support from AFOSR via the grant FA9550-18-1-0246.

## References

1. Lamnabhi-Lagarrigue, F., Annaswamy, A., Engel, S., Isaksson, A., Khargonekar, P., Murray, R., Nijmeijer, H., Samad, T., Tilbury, D., den Hof, P.V.: Systems & control for the future of humanity, research agenda: Current and future roles, impact and grand challenges. *Ann Rev Control* **43**, 1–64 (2017)
2. Xue, M., Wang, W., Roy, S.: Security concepts for the dynamics of autonomous vehicle networks. *Automatica* **50**(3), 852–857 (2014)
3. Ocampo-Martinez, C., Puig, V., Cembrano, G., Quevedo, J.: Application of predictive control strategies to the management of complex networks in the urban water cycle. *IEEE Control Syst. Mag.* **33**(1), 15–41 (2003)

4. Nie, Y., Wang, X., Cheng, K.: Multi-area self-adaptive pricing control in smart city with EV user participation. *IEEE Trans. Intell. Transport. Syst.* **99**, 1–9 (2017)
5. Pepyne, D.L., Cassandras, C.G.: Control of hybrid systems in manufacturing. *Proceed. IEEE* **88**(7), 1108–1122 (2000)
6. Allgöwer, F., Borges de Sousa, J., Kapinski, Mosterman, P., Oehlerking, J., Panciatici, P., Prandini, M., Rajhans, A., Tabuada, P., Wenzelburger, P.: Position paper on the challenges posed by modern applications to cyber-physical systems theory. *Nonlinear Anal.: Hybrid Syst.* **34**, 147–165 (2019). <https://doi.org/10.1016/j.nahs.2019.05.007>
7. Passino, K.: Biomimicry for Optimization, Control, and Automation. Springer, Berlin (2016)
8. Alur, R., Forejt, V., Moarref, S., Trivedi, A.: Safe schedulability of bounded-rate multi-mode systems. In: Proceedings of the 16th International Conference on Hybrid Systems: Computation and Control, pp. 243–252 (2013)
9. Hou, Z., Wang, Z.: From model-based control to data-driven control: survey, classification and perspective. *Inf. Sci.* **235**(20), 3–35 (2013)
10. Tao, G.: Multivariable adaptive control: a survey. *Automatica* **50**, 2737–2764 (2014)
11. Kim, J.W., Park, B.J., Yoo, H., Lee, J.H., Lee, J.M.: Deep reinforcement learning based finite-horizon optimal tracking control for nonlinear system. *IFAC-PapersOnLine* **51**(25), 257–262 (2018). <https://doi.org/10.1016/j.ifacol.2018.11.115>
12. Ravanbakhsh, H., Sankaranarayanan, S.: Learning control Lyapunov functions from counterexamples and demonstrations. *Auton. Robots* **43**(2), 275–307 (2019)
13. Bertsekas, D.: Reinforcement Learning and Optimal Control. Athena Scientific, Nashua (2019)
14. Sutton, R.S., Barto, A.G.: Reinforcement Learning: An Introduction. MIT Press, Cambridge (1998)
15. Vrabie, D., Vamvoudakis, K.G., Lewis, F.L.: Optimal Adaptive Control and Differential Games by Reinforcement Learning Principles. IET Press (2012)
16. Lewis, F.L., Liu, D.: Reinforcement Learning and Approximate Dynamic Programming for Feedback Control. Computational Intelligence Series. John Wiley/IEEE Press, Hoboken (2012)
17. Kiumarsi, B., Vamvoudakis, K.: Optimal and autonomous control using reinforcement learning: a survey. *IEEE Trans. Neural Netw. Learn. Syst.* **29**(6), 2042–2062 (2018)
18. Recht, B.: A tour of reinforcement learning: the view from continuous control. *Ann. Rev. Control Robot. Auton. Syst.* **2**, 253–279 (2019)
19. Görres, D.: Relations between Model predictive control and reinforcement learning. *IFAC-PapersOnLine* **50**(1), 4920–4928 (2017). <https://doi.org/10.1016/j.ifacol.2017.08.747>
20. Lee, D., Hu, J.: Primal-dual Q-learning framework for LQR design. *IEEE Trans. Autom. Control* **64**(9), 3756–3763 (2018)
21. Vamvoudakis, K.G., Lewis, F.L.: Online actor-critic algorithm to solve the continuous-time infinite horizon optimal control problem. *Automatica* **46**(5), 878–888 (2010)
22. Kamalapurkar, R., Klotz, J.R., Dixon, W.E.: Concurrent learning-based approximate feedback-nash equilibrium solution of n-player nonzero-sum differential games. *IEEE/CAA J. Autom. Sinica* **1**, 239–247 (2014)
23. Kamalapurkar, R., Rosenfeld, J.A., Dixon, W.E.: Efficient model-based reinforcement learning for approximate online optimal control. *Automatica* **74**, 247–258 (2016). <https://doi.org/10.1016/j.automatica.2016.08.004>
24. Wang, Y., Velswamy, K., Huang, B.: A novel approach to feedback control with deep reinforcement learning. *IFAC-PapersOnLine* **51**(18), 31–36 (2018). <https://doi.org/10.1016/j.ifacol.2018.09.241>
25. Kamalapurkar, R., Walters, P., Rosenfeld, J., Dixon, W.: Reinforcement Learning for Optimal Feedback Control: A Lyapunov Based Approach. Springer, Berlin (2018)
26. Branicky, M.S., Borkar, V.S., Mitter, S.K.: A unified framework of hybrid control: model and optimal control theory. *IEEE Trans. Autom. Control* **43**(1), 31–45 (1998)
27. Bensoussan, A., Menaldi, J.L.: Hybrid control and dynamic programming. *Dyn. Contin. Discrete Impulsive Syst.* **3**(4), 395–442 (1997)

28. Shaikh, M.S., Caines, P.E.: On the hybrid optimal control problem: theory and algorithms. *IEEE Trans. Autom. Control* **52**(9), 1587–1603 (2007)
29. Cassandras, C.G., Pepyne, D.L., Wardi, Y.: Optimal control of a class of hybrid systems. *IEEE Trans. Autom. Control* **46**(3), 398–415 (2001)
30. Pakniyat, A.: Optimal control of deterministic and stochastic hybrid systems: theory and applications. Ph.D. Dissertation, McGill University (2016)
31. Forte, F., Marconi, L., Teel, A.R.: Robust nonlinear regulation: continuous-time internal models and hybrid identifiers. *IEEE Trans. Autom. Control* **62**(7), 3136–3151 (2017)
32. Poveda, J.I., Krstić, M.: Fixed-time gradient-based extremum seeking. *Amer. Control Conf.* 2838–2843 (2020)
33. Poveda, J.I., Teel, A.R.: A framework for a class of hybrid extremum seeking controllers with dynamic inclusions. *Automatica* **76** (2017)
34. Kutadinata, R.J., Moase, W., Manzie, C.: Extremum-seeking in singularly perturbed hybrid systems. *IEEE Trans. Autom. Control* **62**(6), 3014–3020 (2017)
35. Poveda, J.I., Kutadinata, R., Manzie, C., Nešić, D., Teel, A.R., Liao, C.: Hybrid extremum seeking for black-box optimization in hybrid plants: an analytical framework. In: 57th IEEE Conference on Decision and Control, pp. 2235–2240 (2018)
36. Owens, D.H.: Iterative Learning Control: An Optimization Paradigm. Springer, London (2015)
37. Poveda, J.I., Benosman, M., Teel, A.R.: Hybrid online learning control in networked multiagent systems: a survey. *Int. J. Adapt. Control Signal Process.* **33**(2), 228–261 (2019)
38. Goebel, R., Sanfelice, R.G., Teel, A.R.: Hybrid dynamical systems: modeling, stability, and robustness. Princeton University Press, Princeton (2012)
39. Poveda, J.I., Li, N.: Robust hybrid zero-order optimization algorithms with acceleration via averaging in continuous time. *Automatica*, **123**, 2021, 109361
40. Teel, A.R., Poveda, J.I., Le, J.: First-order optimization algorithms with resets and hamiltonian flows. In: 58th IEEE Conference on Decision and Control, pp. 5838–5843 (2019)
41. Poveda, J.I., Teel, A.R.: A robust event-triggered approach for fast sampled-data extremization and learning. *IEEE Trans. Autom. Control* **62**(10) (2017)
42. Liu, J., Teel, A.R.: Lyapunov-based sufficient conditions for stability of hybrid systems with memory. *IEEE Trans. Autom. Control* **61**(4), 1057–1062 (2016)
43. Mayhew, C.G.: Hybrid control for topologically constrained systems. Ph.D Dissertation, University of California, Santa Barbara (2010)
44. Sanfelice, R.G., Messina, M.J., Tuna, S.E., Teel, A.R.: Robust hybrid controllers for continuous-time systems with applications to obstacle avoidance and regulation to disconnected set of points. In: Proceedings of American Control Conference, pp. 3352–3357 (2006)
45. Poveda, J.I., Benosman, M., Sanfelice, R.G., Teel, A.R.: A hybrid adaptive feedback law for robust obstacle avoidance and coordination in multiple vehicle systems. In: Proceedings of American Control Conference, pp. 616–621 (2018)
46. Strizic, T., Poveda, J.I., Teel, A.R.: Hybrid gradient descent for robust global optimization on the circle. In: 56th IEEE Conference on Decision and Control, pp. 2985–2990 (2017)
47. Hespanha, J.P., Morse, A.S.: Stabilization of switched systems with average dwell-time. In: 38th IEEE Conference on Decision and Control, pp. 2655–2660 (1999)
48. Vidyasagar, M.: Nonlinear Systems Analysis. Prentice Hall, Upper Saddle River (1993)
49. Jakubczyk, B., Sontag, E.D.: Controllability of nonlinear discrete-time systems: a Lie-algebraic approach. *SIAM J. Control Opt.* **28**(1), 1–33 (1990)
50. Subbaraman, A., Teel, A.R.: On the equivalence between global recurrence and the existence of a smooth Lyapunov function for hybrid systems. *Syst. & Control Lett.* **88**, 54–61 (2016)
51. Teel, A.R., Subbaraman, A., Sferlaza, A.: Stability analysis for stochastic hybrid systems: a survey. *Automatica* **50**(10), 2435–2456 (2014)
52. Vamvoudakis, K.G., Fantini-Miranda, M., Hespanha, J.P.: Asymptotically stable adaptive-optimal control algorithm with saturating actuators and relaxed persistence of excitation. *IEEE Trans. Neural Netw. Learn. Syst.* **27**(11), 2386–2398 (2016)

53. Barto, A.G., Sutton, R.S., Anderson, C.W.: Neuronlike adaptive elements that can solve difficult learning control problems. *IEEE Trans. Syst. Man Cybern.* **SMC-13**(5), 834–846 (1983)
54. Lewis, F., Syrmos, V.: *Optimal Control*. Wiley, Boston (1995)
55. Hespanha, J.P.: *Linear Systems Theory*. Princeton University Press, Princeton (2009)
56. Press, W.H., Teukolsky, S.A., Vetterling, W.T., Flannery, B.P.: *Numerical Recipes: The Art of Scientific Computing*, 3rd edn. Cambridge University Press, Cambridge (2007)
57. Bhaya, A., Kaszkurewicz, E.: *Control perspectives on numerical algorithms and matrix problems*. SIAM (2006)
58. Nocedal, J., Wright, S.: *Numerical Optimization*. Springer, Berlin (1999)
59. Saridis, G.N., Lee, C.S.: An approximation theory of optimal control for trainable manipulators. *IEEE Trans. Syst. Man Cybern.* **9**(3), 152–159 (1979)
60. Hornik, K., Stinchcombe, S., White, H.: Universal approximation of an unknown mapping and its derivatives using multilayer feedforward networks. *Neural Netw.* **3**, 551–560 (1990)
61. Sastry, S., Bodson, M.: *Adaptive Control: Stability, Convergence, and Robustness*. Prentice-Hall, Englewood Cliffs (1989)
62. Ioannou, P.A., Sun, J.: *Robust Adaptive Control*. Dover Publications Inc., Mineola (2012)
63. Prokhorov, D.V., Wunsch, D.C.: Adaptive critic designs. *IEEE Trans. Neural Netw.* **8**(5), 997–1007 (1997)
64. Abouheaf, M.I., Lewis, F.L., Vamvoudakis, K.G., Haesaert, S., Babuska, R.: Multi-agent discrete-time graphical games and reinforcement learning solutions. *Automatica* **50**(12), 3038–3053 (2014)
65. Yang, Q., Vance, J.B., Jagannathan, S.: Control of nonaffine nonlinear discrete-time systems using reinforcement learning-based linearly parameterized neural networks. *IEEE Trans. Syst. Man Cybern. Part B Cybern.* **38**(4), 994–1001 (2008)
66. Yang, X., Liu, D., Wang, D., Wei, Q.: Discrete-time online learning control for a class of unknown nonaffine nonlinear systems using reinforcement learning. *Neural Netw.* **55**, 30–41 (2014). <https://doi.org/10.1016/j.neunet.2014.03.008>
67. Yang, Q., Jagannathan, S.: Reinforcement learning controller design for affine nonlinear discrete-time systems using online approximators. *IEEE Trans. Syst. Man Cybern. Part B: Cybern.* **42**(2), 377–390 (2012)
68. Vamvoudakis, K., Lewis, F.L.: Online solution of nonlinear two-player zero-sum games using synchronous policy iteration. *Int. J. Robust Nonlinear Control* **22**, 1460–1483 (2011)
69. Vamvoudakis, K.G., Lewis, F.L.: Multi-player non-zero-sum games: Online adaptive learning solution of coupled Hamilton–Jacobi equations. *Automatica* **47**(8), 1556–1569 (2011). <https://doi.org/10.1016/j.automatica.2011.03.005>
70. Kanellopoulos, A., Vamvoudakis, K.G.: Non-equilibrium dynamic games and cyber-physical security: a cognitive hierarchy approach. *Syst. Control Lett.* **125**, 59–66 (2019). <https://doi.org/10.1016/j.sysconle.2019.01.008>
71. Vamvoudakis, K.G.: *Model-Free Learning of Nash Games With Applications to Network Security*. Elsevier Inc., Amsterdam (2016). <https://doi.org/10.1016/B978-0-12-805246-4.00010-0>
72. Vamvoudakis, K.G., Ferraz, H.: Model-free event-triggered control algorithm for continuous-time linear systems with optimal performance. *Automatica* **87**, 412–420 (2018). <https://doi.org/10.1016/j.automatica.2017.03.013>
73. Chen, C., Modares, H., Xie, K., Lewis, F.L., Wan, Y., Xie, S.: Reinforcement learning-based adaptive optimal exponential tracking control of linear systems with unknown dynamics. *IEEE Trans. Autom. Control* **64**(11), 4423–4438 (2019)
74. Vamvoudakis, K.G.: Event-triggered optimal adaptive control algorithm for continuous-time nonlinear systems. *IEEE/CAA J. Autom. Sinica* **1**(3), 282–293 (2014)
75. Su, H., Zhang, H., Sun, S., Cai, Y.: Integral reinforcement learning-based online adaptive event-triggered control for non-zero-sum games of partially unknown nonlinear systems. *Neurocomputing* (2019). <https://doi.org/10.1016/j.neucom.2019.09.088>

76. Prieur, C., Teel, A.R.: Uniting local and global output feedback controllers. *IEEE Trans. Autom. Control* **56**(7), 1636–1649 (2011)
77. Prieur, C., Goebel, R., Teel, A.R.: Hybrid feedback control and robust stabilization of nonlinear systems. *IEEE Trans. Autom. Control* **52**(11), 2103–2117 (2007)
78. Mayhew, C.G., Sanfelice, R.G., Sheng, J., Arcak, M., Teel, A.R.: Quaternion-based hybrid feedback for robust global attitude synchronization. *IEEE Trans. Autom. Control* **57**(8), 2122–2127 (2012)
79. Nešić, D., Teel, A.R., Zaccarian, L.: Stability and performance of SISO control systems with first-order reset elements. *IEEE Trans. Autom. Control* **56**(11), 2567–2582 (2011)
80. Nešić, D., Teel, A.R., Kokotović, P.V.: Sufficient conditions for stabilization of sampled-data nonlinear systems via discrete-time approximations. *Syst. & Control Lett.* **38**, 259–270 (1999)
81. Nešić, D., Teel, A.R.: A framework for stabilization of nonlinear sampled-data systems based on their approximate discrete-time models. *IEEE Trans. Autom. Control* **49**(7), 1103–1121 (2004)
82. Nešić, D., Teel, A.R., Carnevale, D.: Explicit computation of the sampling period in emulation of controllers for nonlinear sampled-data systems. *IEEE Transactions on Automatic and Control* **54**(3), 619–624 (2009)
83. Khong, S.Z., Nešić, D., Tan, Y., Manzie, C.: Unified framework for sampled-data extremum seeking control: global optimisation and multi-unit systems. *Automatica* **49**, 2720–2733 (2013)
84. Chien, C.: A sampled-data iterative learning control using fuzzy network design. *Int. J. Control* **73**, 902–913 (2000)
85. Bai, E.W., Fu, L.C., Sastry, S.S.: Averaging analysis for discrete time and sampled data adaptive systems. *IEEE Trans. Circuits Syst.* **35**(2) (1988)
86. Chen, T., Francis, B.: Optimal Sampled-Data Control Systems. Springer, Berlin (1995)
87. Vrabie, D., Lewis, F.: Neural network approach to continuous-time direct adaptive optimal control for partially unknown nonlinear systems. *Neural Netw.* **22**, 237–246 (2009)
88. Tabuada, P.: Event-triggered real-time scheduling of stabilizing control tasks. *IEEE Trans. Autom. Control* **52**, 1680–1685 (2007)
89. Postoyan, R., Tabuada, P., Nešić, D., Anta, A.: A framework for the event-triggered stabilization of nonlinear systems. *IEEE Trans. Autom. Control* **60**(4), 982–996 (2015)
90. Heemels, W.P.M.H., Donkers, M.C.F., Teel, A.R.: Periodic event-triggered control for linear systems. *IEEE Trans. Autom. Control* **58**, 847–861 (2013)
91. Narayanan, V., Jagannathan, S.: Event-triggered distributed control of nonlinear interconnected systems using online reinforcement learning with exploration. *IEEE Trans. Cybern.* **48**(9), 2510–2519 (2018)
92. Poveda, J.I., Teel, A.R.: Hybrid mechanisms for robust synchronization and coordination of multi-agent networked sampled-data systems. *Automatica* **99**, 41–53 (2019)
93. Persis, C.D., Postoyan, R.: A Lyapunov redesign of coordination algorithms for cyber-physical systems. *IEEE Transactions on Automatic and Control* **62**(2), 808–823 (2017)
94. Poveda, J., Teel, A.: A hybrid systems approach for distributed nonsmooth optimization in asynchronous multi-agent sampled-data systems. *IFAC-PapersOnLine* **49**(18) (2016)
95. Nešić, D., Teel, A.R., Zaccarian, L.: Stability and performance of SISO control systems with first-order reset elements. *IEEE Trans. Autom. Control* **56**(11), 2567–2582 (2011)
96. Prieur, C., Queinnec, I., Tarbouriech, S., Zaccarian, L.: Analysis and Synthesis of Reset Control Systems. Now Foundations and Trends (2018)
97. Hustig-Schultz, D., Sanfelice, R.G.: A robust hybrid heavy ball algorithm for optimization with high performance. Amer. Control Conf. (2019). To appear
98. Poveda, J.I., Li, N.: Inducing uniform asymptotic stability in non-autonomous accelerated optimization dynamics via hybrid regularization. In: 58th IEEE Conference on Decision and Control, pp. 3000–3005 (2019)
99. Ochoa, D., Poveda, J.I., Uribe, C., Quijano, N.: Robust accelerated optimization on networks via distributed restarting of Nesterov's-like ODE. *IEEE Control Syst. Lett.* **5**(1) (2021)

100. Socy, B.V., Freeman, R.A., Lynch, K.M.: The fastest known globally convergent first-order method for minimizing strongly convex functions. *IEEE Control Syst. Lett.* **2**(1), 49–54 (2018)
101. Baradaran, M., Poveda, J.I., Teel, A.R.: Stochastic hybrid inclusions applied to global almost sure optimization on manifolds. In: *IEEE 57th Conference on Decision and Control*, pp. 6538–6543 (2018)
102. Baradaran, M., Poveda, J.I., Teel, A.R.: Global optimization on the sphere: a stochastic hybrid systems approach. In: *Proceedings of the 10th IFAC Symposium on Nonlinear Control Systems* (2019). To appear
103. Possieri, C., Teel, A.R.: LQ optimal control for a class of hybrid systems. In: *IEEE 55th Conference on Decision and Control*, pp. 604–609 (2016)
104. Carnevale, D., Galeani, S., Sassano, M.: A linear quadratic approach to linear time invariant stabilization for a class of hybrid systems. In: *Proceedings of the 22nd Mediterranean Conference on Control and Automation*, pp. 545–550 (2014)
105. Dharmatti, S., Ramaswamy, M.: Hybrid control systems and viscosity solutions. *SIAM J. Control Opt.* **44**(4), 1259–1288 (2005)
106. Barles, G., Dharmatti, S., Ramaswamy, M.: Unbounded viscosity solutions of hybrid control systems. *ESAIM: Control Opt. Cal. Var.* **16**(1), 176–193 (2010)
107. De Carolis, G., Saccon, A.: On linear quadratic optimal control for time-varying multimodal linear systems with time-triggered jumps. *IEEE Control Syst. Lett.* **4**(1), 217–222 (2020)
108. Hedlund, S., Rantzer, A.: Convex dynamic programming for hybrid systems. *IEEE Trans. Autom. Control* **47**(9), 1536–1540 (2002)
109. Passenberg, B., Caines, P.E., Leibold, M., Stursberg, O., Buss, M.: Optimal control for hybrid systems with partitioned state space. *IEEE Trans. Autom. Control* **58**(8), 2131–2136 (2013)
110. Bemporad, A., Giorgetti, N.: Logic-based solution methods for optimal control of hybrid systems. *IEEE Trans. Autom. Control* **51**(6), 963–976 (2006)
111. Chen, H.: Optimal control and reinforcement learning of switched systems. Ph.D. Dissertation, The Ohio State University (2018)

# Chapter 25

## The Role of Systems Biology, Neuroscience, and Thermodynamics in Network Control and Learning



Wassim M. Haddad

**Abstract** Recent technological advances in communications and computation have spurred a broad interest in control of networks and control over networks. Network systems involve distributed decision-making for coordination of networks of dynamic agents and address a broad area of applications including cooperative control of unmanned air vehicles, microsatellite clusters, mobile robotics, battle space management, and congestion control in communication networks. To address the problem of autonomy, complexity, and adaptation for control and coordination of multiagent network systems, in this paper we look to systems biology, neurophysiology, and thermodynamics for inspiration in developing innovative architectures for control and learning.

### 25.1 Introduction

Due to advances in embedded computational resources over the last several years, a considerable research effort has been devoted to the control of networks and control over networks [1–19]. Network systems involve distributed decision-making for coordination of networks of dynamic agents involving information flow enabling enhanced operational effectiveness via cooperative control in autonomous systems. These dynamical network systems cover a very broad spectrum of applications including cooperative control of unmanned air vehicles (UAV's) and autonomous underwater vehicles (AUV's) for combat, surveillance, and reconnaissance [20], distributed reconfigurable sensor networks for managing power levels of wireless networks [21], air and ground transportation systems for air traffic control and payload transport and traffic management [22], swarms of air and space vehicle formations for command and control between heterogeneous air and space assets [10, 14], and congestion control in communication networks for routing the flow of information through a network [5].

---

W. M. Haddad (✉)

School of Aerospace Engineering, Georgia Institute of Technology, 30332-0150 Atlanta,  
GA, Georgia

e-mail: [wm.haddad@aerospace.gatech.edu](mailto:wm.haddad@aerospace.gatech.edu)

To enable the applications for these complex large-scale dynamical systems, cooperative control tasks such as formation control, rendezvous, flocking, cyclic pursuit, cohesion, separation, alignment, and consensus need to be developed [9, 13, 15, 18, 23–26]. To realize these tasks, individual agents need to share information of the system objectives as well as the dynamical network. In particular, in many applications involving multiagent systems, groups of agents are required to agree on certain quantities of interest. Information consensus over dynamic information-exchange topologies guarantees agreement between agents for a given coordination task. Distributed consensus algorithms involve neighbor-to-neighbor interaction between agents wherein agents update their information state based on the information states of the neighboring agents. A unique feature of the closed-loop dynamics under any control algorithm that achieves consensus in a dynamical network is the existence of a continuum of equilibria representing a state of consensus. Under such dynamics, the limiting consensus state achieved is not determined completely by the dynamics, but depends on the initial system state as well.

In systems possessing a continuum of equilibria, *semistability*, and not asymptotic stability is the relevant notion of stability [27–29]. Semistability is the property whereby every trajectory that starts in a neighborhood of a Lyapunov stable equilibrium converges to a (possibly different) Lyapunov stable equilibrium. Semistability thus implies Lyapunov stability and is implied by asymptotic stability. From a practical viewpoint, it is not sufficient to only guarantee that a network converges to a state of consensus since steady-state convergence is not sufficient to guarantee that small perturbations from the limiting state will lead to only small transient excursions from a state of consensus. It is also necessary to guarantee that the equilibrium states representing consensus are Lyapunov stable, and consequently, semistable. Finally, since multiagent network systems can alternate between multiple semistable states under changing system inputs and system parameters, it is also necessary to address system *multistability* for network systems.

Multistability is the property whereby the solutions of a dynamical system can alternate between two or more mutually exclusive Lyapunov stable and convergent states under slowly changing inputs or system parameters. In particular, multistable systems give rise to the existence of multiple (isolated and/or a continuum of) Lyapunov stable equilibria involving a quasistatic-like behavior between these multiple semistable steady states [30–32].

Modern complex large-scale dynamical systems can additionally give rise to systems which have nonsmooth dynamics. Examples include variable structure systems where control inputs switch discontinuously between extreme input values to generate minimum-time or minimum-fuel system trajectories. Discontinuities are also intentionally designed to achieve hierarchical system stabilization [33, 34]. In particular, the complexity of modern controlled large-scale dynamical systems is further exacerbated by the use of hierarchical embedded control subsystems within the feedback control system, that is, abstract decision-making units performing logical checks that identify system mode operation and specify the continuous-variable sub-controller to be activated. Such systems typically possess a multiechelon hierarchical hybrid decentralized control architecture characterized by continuous-time dynamics

at the lower levels of the hierarchy and discrete-time dynamics at the higher levels of the hierarchy. The lower level units directly interact with the dynamical system to be controlled while the higher level units receive information from the lower level units as inputs and provide (possibly discrete) output commands which serve to coordinate and reconcile the (sometimes competing) actions of the lower level units [35].

The hierarchical controller organization reduces processor cost and controller complexity by breaking up the processing task into relatively small pieces and decomposing the fast and slow control functions. Typically, the higher level units perform logical checks that determine system mode operation, whereas the lower level units execute continuous-variable commands for a given system mode of operation. Due to their multiechelon hierarchical structure, hybrid dynamical systems are capable of simultaneously exhibiting continuous-time dynamics, discrete-time dynamics, logic commands, discrete events, and resetting events. Such systems include dynamical switching systems [34, 36, 37], nonsmooth impact systems [38, 39], biological systems [40], sampled-data systems [41], discrete-event systems [42], intelligent vehicle/highway systems [43], constrained mechanical systems [38], flight control systems [44], and command and control for future battlefields, to cite but a few examples. The mathematical descriptions of many of these systems can be characterized by hybrid systems and impulsive differential equations with discontinuous right-hand sides [35, 40, 45–51].

To enable the autonomous operation for multiagent systems, the development of functional algorithms for agent coordination and control is needed. In particular, control algorithms need to address agent interactions, cooperative and non-cooperative control, task assignments, and resource allocations. To realize these tasks, appropriate sensory and cognitive capabilities such as adaptation, learning, decision-making, and agreement (or consensus) on the agent and multiagent levels are required. The common approach for addressing the autonomous operation of multiagent systems is using distributed control algorithms involving neighbor-to-neighbor interaction between agents wherein agents update their information state based on the information states of the neighboring agents. Since most multiagent network systems are highly interconnected and mutually interdependent, both physically and through a multitude of information and communication networks, these systems are characterized by high-dimensional, large-scale interconnected dynamical systems.

To develop distributed methods for control and coordination of autonomous multiagent systems, many researchers have looked to autonomous *swarm* systems appearing in nature for inspiration [1, 2, 6, 52–54]. These systems necessitate the development of relatively simple autonomous agents that are inherently distributed, self-organized, and truly scalable. Scalability follows from the fact that such systems do not involve centralized control and communication architectures. In addition, these systems should be inherently robust to individual agent failures, unplanned task assignment changes, and environmental changes.

In this paper, we highlight the role systems biology, neuroscience, and thermodynamics can play in the control and learning of network systems. Specifically, we explore synergies of cooperative control, learning, dynamical thermodynamics, and neuroscience in developing highly adaptive bioinspired autonomous control meth-

ods for self-organizing teams of multiagent vehicles under sparse sensing. In self-organizing systems, the connection between the local subsystem interactions and the globally complex system behavior is often elusive. In nature, these systems are known as dissipative systems and consume energy and matter while maintaining their stable structure by dissipating entropy to the environment. A common phenomenon among these systems is that they evolve in accordance with the laws of (nonequilibrium) thermodynamics [55, 56]. Dynamical thermodynamics involves open interconnected dynamical systems that exchange matter and energy with their environment in accordance with the first law (conservation of energy) and the second law (nonconservation of entropy) of thermodynamics.

The network-based control schemes implemented by the central nervous system for controlling biological aggregations of complex autonomous systems differs drastically from standard control system approaches appearing in the control engineering literature. Specifically, the control scheme is event-triggered, wherein a neuron sends a voltage spike signal (action potential) when the voltage across the cell membrane reaches a certain threshold. In addition, the form of the action potential carries no information. These neurocontrol architectures comprise inherent parallel processing, distributed cellular mechanisms of adaptation and robustness, distributed memory, and low energy consumption making them very attractive for controlling complex, large-scale network autonomous systems.

In light of the above, it seems both natural and appropriate to postulate the following paradigm for nonlinear analysis and control law design of complex large-scale dynamical network systems and multiagent autonomous systems: Develop a unified neuroinspired thermodynamic analysis and control systems design framework for hybrid hierarchical nonlinear large-scale dynamical network systems and multiagent systems in the face of a specified level of modeling uncertainty. This paradigm can provide a rigorous foundation for developing a unified network thermodynamic system analysis and synthesis framework for systems that combine continuous-time and discrete-time dynamics possessing hybrid, hierarchical, and feedback structures. Correspondingly, the main goal of this paper is to present ideas toward the development of analysis and hierarchical hybrid nonlinear control design tools for nonlinear hybrid dynamical systems and multiagent systems which support this paradigm. The proposed paradigm provides the basis for control system partitioning/embedding and develops concepts of thermodynamic-based control and neuroinspired stabilization for complex, large-scale multiagent networked systems.

The contents of this paper are as follows. In Sect. 25.2, we address the problem of network system complexity for multiagent systems with changing system parameters and switching topologies. Specifically, we propose a generalization of the concepts of energy, entropy, and temperature to undirected and directed networks and use hybrid thermodynamic principles to explore the design of distributed consensus control algorithms for static and dynamic networked systems in the face of system uncertainty and intermittent communication. Here the zeroth, first, and second laws for hybrid thermodynamics [57] are used to design distributed consensus controllers that cause networked dynamical systems to emulate natural thermodynamic behavior involving inherent robustness network system guarantees. In particular,

the proposed hybrid controller architecture involves intermittent exchange of state information between agents guaranteeing that the closed-loop dynamical network is semistable to an equipartitioned equilibrium state representing a state of information consensus consistent with hybrid thermodynamic principles [56].

To capture network system uncertainty and communication uncertainty between the agents in a network, wherein the evolution of each link of the network communication topology follows a Markov process for modeling unknown communication noise and attenuations, in Sect. 25.3 we highlight an extension of deterministic semistability and finite time semistability to the stochastic setting in order to develop almost sure consensus protocols for multiagent systems with nonlinear stochastic dynamics. Specifically, using the stochastic semistability and stochastic finite time semistability frameworks developed in [58] we formulate a design paradigm for distributed asymptotic and finite time consensus control protocols for nonlinear undirected and directed dynamical networks with stochastic communication uncertainty. The proposed controller architectures are predicated on the recently developed notion of stochastic dynamical thermodynamics [56] and result in controller architectures involving the exchange of generalized charge or energy state information between agents that guarantee that the closed-loop dynamical network is consistent with stochastic thermodynamic principles.

In Sect. 25.4, we highlight the role systems biology, neuroscience, and thermodynamics can play in the control of dynamical network systems. In particular, we draw on the fundamental principles from dynamical thermodynamics and neuroscience to propose control system architectures for complex large-scale, interconnected systems that are capable of adaptive learning. More specifically, we discuss how dynamical neuroscience and dynamical thermodynamics can be integrated by embedding thermodynamic state notions (e.g., entropy, energy, free energy, chemical potential) within a distributed control system design framework to directly address large-scale swarm networks. This can allow for the development of the next generation of agile, highly adaptive autonomous micro-air vehicles that can utilize mechanisms from systems biology and dynamical thermodynamics to process information from distributed sensors.

Since multiagent systems can involve information laws governed by nodal dynamics and rerouting strategies that can be modified to minimize waiting times and optimize system throughput, in Sect. 25.5 we outline a stochastic nonlinear optimal control framework for large-scale complex networks. Nonlinear controllers may be required for a variety of reasons. These include system nonlinearities, state and control variable constraints (such as saturation and quantization), nonquadratic cost functionals, and structured plant uncertainty. In accordance with the goal of designing practically implementable nonlinear controllers, we extend the classical stochastic Hamilton–Jacobi Bellman optimal control framework to address optimal finite time coordination in multiagent systems as well as develop connections between the newly developed notion of stochastic dissipativity [59], optimal control, and stability margins for stochastic nonlinear feedback regulators. In addition, we discuss the role of universal (asymptotic and finite time) optimal feedback controllers for nonlinear stochastic systems that possess guaranteed gain, sector, and disk margins for

addressing robust stabilization problems involving both stochastic and deterministic uncertainty with averaged and worst-case performance criteria. Furthermore, we discuss how reinforcement learning can provide model-free learning controllers for optimal, thermodynamic-based control architectures of network systems.

In Sect. 25.6, we discuss a network thermodynamic framework for addressing consensus problems for swarm dynamics. Specifically, we explore the development of distributed boundary controller architectures involving the exchange of information between uniformly distributed swarms over an  $n$ -dimensional (not necessarily Euclidian) space that guarantees that the closed-loop system is consistent with continuum thermodynamic principles [56]. Since the closed-loop system satisfies thermodynamic principles, robustness to individual agent failures and unplanned individual agent behavior is automatically guaranteed. Finally, in Sect. 25.7, we show how the theories of dynamical thermodynamics, information theory, and nonlinear dynamical systems can be merged to develop a unified nonlinear stabilization framework with a priori achievable system performance guarantees using connections between Bode integrals, Shannon entropy, and Clausius entropy.

## 25.2 Large-Scale Networks and Hybrid Thermodynamics

As discussed in the introduction, modern complex multiagent dynamical systems are highly interconnected and mutually interdependent, both physically and through a multitude of information and communication networks. Distributed decision-making for coordination of networks of dynamic agents involving information flow can be naturally captured by graph-theoretic notions. These dynamical network systems cover a very broad spectrum of applications. A key application area is cooperative control of air and space vehicle formations using distributed and decentralized controller architectures. Distributed control refers to a control architecture wherein the control is distributed via multiple computational units that are interconnected through information and communication networks, whereas decentralized control refers to a control architecture wherein local decisions are based only on local information. Vehicle formations are typically dynamically decoupled, that is, the motion of a given agent or vehicle does not directly affect the motion of the other agents or vehicles. The multiagent system is coupled via the task which the agents or vehicles are required to perform.

In many applications involving multiagent systems, groups of agents are required to agree on certain quantities of interest. For example, in a group of autonomous vehicles this property might be a common heading or a shared communication frequency. Moreover, it is important to develop information consensus protocols for networks of dynamic agents wherein a unique feature of the closed-loop dynamics under any control algorithm that achieves consensus is the existence of a continuum of equilibria representing a state of equipartitioning or *consensus*. Under such dynamics, the limiting consensus state achieved is not determined completely by the dynamics but depends on the initial system state as well. As noted earlier, for

such systems possessing a continuum of equilibria, semistability [27–29], and not asymptotic stability, is the relevant notion of stability.

Alternatively, in other applications of multiagent systems, groups of agents are required to achieve and maintain a prescribed geometric shape. This *formation* problem includes *flocking* [11, 18] and *cyclic pursuit* [15], wherein parallel and circular formations of vehicles are sought. For formation control of multiple vehicles, *cohesion*, *separation*, and *alignment* constraints are typically required for individual agent steering, which describe how a given vehicle maneuvers based on the positions and velocities of nearby agents. Specifically, cohesion refers to a steering rule wherein a given vehicle attempts to move toward the average position of local vehicles, separation refers to collision avoidance with nearby vehicles, whereas alignment refers to velocity matching with nearby vehicles.

Using graph-theoretic notions, in [60] the authors developed control algorithms for addressing consensus problems for nonlinear multiagent dynamical systems with fixed communication topologies. The proposed controller architectures were predicated on the recently developed notion of dynamical thermodynamics [56, 61] resulting in controller architectures involving the exchange of information between agents that guarantee that the closed-loop dynamical network is consistent with basic thermodynamic principles.

A key extension to the framework presented in [60] is to develop a unified *hybrid* thermodynamic control framework for addressing consensus, flocking, and cyclic pursuit problems for multiagent dynamical systems with fixed and dynamic (i.e., switching) topologies. Specifically, building on the work of [60], hybrid distributed and decentralized controller architectures for multiagent coordination can be developed based on the hybrid thermodynamic principles presented in [56, 57]. A unique feature of the proposed framework is that the controller architectures are hybrid, and hence, the overall closed-loop dynamics under these controller algorithms achieving consensus, flocking, or cyclic pursuit will possess discontinuous flows since they will combine logical switchings with continuous dynamics, leading to impulsive differential equations [35, 40, 45, 48, 62–64]. The proposed controllers can involve undirected and directed communication graphs to accommodate for a full range of possible graph information topologies without limitations of bidirectional communication.

To present the key ideas for developing a thermodynamic-based hybrid control architecture for information consensus consider the hybrid dynamical system

$$\dot{x}(t) = f_c(x(t)), \quad x(0) = x_0, \quad x(t) \notin \mathcal{Z}, \quad (25.1)$$

$$\Delta x(t) = f_d(x(t)), \quad x(t) \in \mathcal{Z}, \quad (25.2)$$

where, for every  $t \geq 0$ ,  $x(t) \in \mathcal{D} \subseteq \mathbb{R}^n$ ,  $\mathcal{D}$  is an open set with  $0 \in \mathcal{D}$ ,  $\Delta x(t) \triangleq x(t^+) - x(t)$ , where  $x(t^+) \triangleq x(t) + f_d(x(t)) = \lim_{\varepsilon \rightarrow 0^+} x(t + \varepsilon)$ ,  $f_c : \mathcal{D} \rightarrow \mathbb{R}^n$  is Lipschitz continuous and satisfies  $f_c(0) = 0$ ,  $f_d : \mathcal{D} \rightarrow \mathbb{R}^n$  is continuous, and  $\mathcal{Z} \subset \mathcal{D}$  is the *resetting set*. Note that  $x_e \in \mathcal{D}$  is an equilibrium point of (25.1) and (25.2) if and only if  $f_c(x_e) = 0$  and  $f_d(x_e) = 0$ . We refer to the differential equation (25.1)

as the *continuous-time dynamics*, and we refer to the difference Eq. (25.2) as the *resetting law*.

A function  $x : \mathcal{I}_{x_0} \rightarrow \mathcal{D}$  is a *solution* to the impulsive dynamical system (25.1) and (25.2) on the interval  $\mathcal{I}_{x_0} \subseteq \mathbb{R}$  with initial condition  $x(0) = x_0$ , where  $\mathcal{I}_{x_0}$  denotes the maximal interval of existence of a solution to (25.1) and (25.2), if  $x(\cdot)$  is left-continuous and  $x(t)$  satisfies (25.1) and (25.2) for all  $t \in \mathcal{I}_{x_0}$ . For further discussion on solutions to impulsive differential equations, see [35, 40, 47, 48, 64, 65]. For convenience, we use the notation  $s(t, x_0)$  to denote the solution  $x(t)$  of (25.1) and (25.2) at time  $t \geq 0$  with initial condition  $x(0) = x_0$ .

To ensure well-posedness of the solutions to (25.1) and (25.2), we make the following additional assumptions [35].

**Assumption 25.1** If  $x \in \overline{\mathcal{Z}} \setminus \mathcal{Z}$ , then there exists  $\varepsilon > 0$  such that, for all  $0 < \delta < \varepsilon$ ,  $\psi(\delta, x) \notin \mathcal{Z}$ , where  $\psi(\cdot, \cdot)$  denotes the solution to the continuous-time dynamics (25.1).

**Assumption 25.2** If  $x \in \mathcal{Z}$ , then  $x + f_d(x) \notin \mathcal{Z}$ .

Assumption 25.1 ensures that if a trajectory reaches the closure  $\overline{\mathcal{Z}}$  of  $\mathcal{Z}$  at a point that does not belong to  $\mathcal{Z}$ , then the trajectory must be directed away from  $\mathcal{Z}$ ; that is, a trajectory cannot enter  $\mathcal{Z}$  through a point that belongs to the closure of  $\mathcal{Z}$  but not to  $\mathcal{Z}$ . Furthermore, Assumption 25.2 ensures that when a trajectory intersects the resetting set  $\mathcal{Z}$ , it instantaneously exits  $\mathcal{Z}$ . Finally, we note that if  $x_0 \in \mathcal{Z}$ , then the system initially resets to  $x_0^+ = x_0 + f_d(x_0) \notin \mathcal{Z}$ , which serves as the initial condition for the continuous-time dynamics (25.1).

For a particular closed-loop trajectory  $x(t)$ , we let  $t_k \triangleq \tau_k(x_0)$  denote the  $k$ th instant of time at which  $x(t)$  intersects  $\mathcal{Z}$ , and we call the times  $t_k$  the *resetting times*. Thus, the trajectory of the closed-loop system (25.1) and (25.2) from the initial condition  $x(0) = x_0$  is given by  $\psi(t, x_0)$  for  $0 < t \leq t_1$ . If the trajectory reaches a state  $x_1 \triangleq x(t_1)$  satisfying  $x_1 \in \mathcal{Z}$ , then the state is instantaneously transferred to  $x_1^+ \triangleq x_1 + f_d(x_1)$  according to the resetting law (25.2). The trajectory  $x(t)$ ,  $t_1 < t \leq t_2$ , is then given by  $\psi(t - t_1, x_1^+)$ , and so on. Our convention here is that the solution  $x(t)$  of (25.1) and (25.2) is left continuous, that is, it is continuous everywhere except at the resetting times  $t_k$ , and  $x_k \triangleq x(t_k) = \lim_{\varepsilon \rightarrow 0^+} x(t_k - \varepsilon)$  and  $x_k^+ \triangleq x(t_k) + f_d(x(t_k)) = \lim_{\varepsilon \rightarrow 0^+} x(t_k + \varepsilon)$  for  $k = 1, 2, \dots$ .

It follows from Assumptions 25.1 and 25.2 that for a particular initial condition, the resetting times  $t_k = \tau_k(x_0)$  are distinct and well defined [35]. Since the resetting set  $\mathcal{Z}$  is a subset of the state space and is independent of time, impulsive dynamical systems of the form (25.1) and (25.2) are time-invariant systems. These systems are called *state-dependent impulsive dynamical systems* [35]. Since the resetting times are well defined and distinct, and since the solution to (25.1) exists and is unique, it follows that the solution of the impulsive dynamical system (25.1) and (25.2) also exists and is unique over a forward time interval. For details on the existence and uniqueness of solutions of impulsive dynamical systems in forward time see [40, 47, 48].

The information consensus problem we consider is a dynamic graph involving the trajectories of the dynamical network characterized by the multiagent impulsive

dynamical system  $\mathcal{G}_i$  given by

$$\dot{x}_i(t) = u_{ci}(t), \quad x_i(0) = x_{i0}, \quad x_i(t) \notin \mathcal{Z}_i, \quad i = 1, \dots, q, \quad (25.3)$$

$$\Delta x_i(t) = u_{di}(t), \quad x_i(t) \in \mathcal{Z}_i, \quad (25.4)$$

where, for every  $t \geq 0$ ,  $x_i(t) \in \mathbb{R}$  denotes the information state of the  $i$ th agent and  $u_{ci}(t) \in \mathbb{R}$  and  $u_{di}(t) \in \mathbb{R}$ , respectively, denote the continuous and discrete information control inputs of the  $i$ th agent associated with the controller resetting set  $\mathcal{Z}_i \subset \mathbb{R}$ ,  $i \in \{1, \dots, q\}$ . For simplicity of exposition, we limit our attention to static controller architectures; dynamic controller architectures are discussed in the end of this section.

The hybrid consensus protocol is given by

$$u_{ci}(t) = \sum_{j=1, i \neq j}^q \phi_{cij}(x_i(t), x_j(t)), \quad (25.5)$$

$$u_{di}(t) = \sum_{j=1, i \neq j}^q \phi_{dij}(x_i(t), x_j(t)), \quad (25.6)$$

where  $\phi_{cij}(\cdot, \cdot)$  is locally Lipschitz continuous,  $\phi_{dij}(\cdot, \cdot)$  is continuous,  $\phi_{cij}(x_i, x_j) = -\phi_{cji}(x_j, x_i)$ , and  $\phi_{dij}(x_i, x_j) = -\phi_{dji}(x_j, x_i)$  for all  $i, j = 1, \dots, q$ ,  $i \neq j$ . In this case, the closed-loop system (25.3)–(25.6) is given by

$$\dot{x}_i(t) = \sum_{j=1, i \neq j}^q \phi_{cij}(x_i(t), x_j(t)), \quad x_i(0) = x_{i0}, \quad x_i(t) \notin \mathcal{Z}_i, \quad i = 1, \dots, q, \quad (25.7)$$

$$\Delta x_i(t) = \sum_{j=1, i \neq j}^q \phi_{dij}(x_i(t), x_j(t)), \quad x_i(t) \in \mathcal{Z}_i, \quad (25.8)$$

or, equivalently, in vector form

$$\dot{x}(t) = f_c(x(t)), \quad x(0) = x_0, \quad x(t) \notin \mathcal{Z}, \quad (25.9)$$

$$\Delta x(t) = f_d(x(t)), \quad x(t) \in \mathcal{Z}, \quad (25.10)$$

where  $x(t) \triangleq [x_1(t), \dots, x_q(t)]^\top \in \mathbb{R}^q$ ,  $f_c(x(t)) \triangleq [f_{c,1}(x(t)), \dots, f_{c,q}(x(t))]^\top \in \mathbb{R}^q$ ,  $f_d(x(t)) \triangleq [f_{d_1}(x(t)), \dots, f_{d_q}(x(t))]^\top \in \mathbb{R}^q$ ,  $\mathcal{Z} \triangleq \bigcup_{i=1}^q \{x \in \mathbb{R}^q : x_i \in \mathcal{Z}_i\}$ , and, for  $i, j = 1, \dots, q$ ,

$$f_{ci}(x(t)) = \sum_{j=1, i \neq j}^q \phi_{cij}(x_i(t), x_j(t)), \quad (25.11)$$

$$f_{di}(x(t)) = \sum_{j=1, i \neq j}^q \phi_{dij}(x_i(t), x_j(t)). \quad (25.12)$$

Note that  $\mathcal{G}$  given by (25.7) and (25.8) describe an interconnected network where information states are updated using a distributed hybrid controller involving neighbor-to-neighbor interaction between agents. Furthermore, note that although our results can be directly extended to the case where (25.3) and (25.4) describe the hybrid dynamics of an aggregate system with an aggregate state vector  $x(t) = [x_1(t), \dots, x_q(t)]^T \in \mathbb{R}^{Nq}$ , where  $x_i(t) \in \mathbb{R}^N$ ,  $u_{ci}(t) \in \mathbb{R}^N$  and  $u_{di}(t) \in \mathbb{R}^N$ ,  $i = 1, \dots, q$ , by using Kronecker algebra, for simplicity of exposition we focus on individual agent states evolving in  $\mathbb{R}$  (i.e.,  $N = 1$ )

In order to define the resetting set  $\mathcal{Z}$  associated with  $\mathcal{G}$ , we require some additional notation. Let  $O_i$  denote the set of all agents with information flowing out of the  $i$ th agent, let  $I_i$  denote the set of all agents receiving information from the  $i$ th agent, and let  $\mathcal{V} = \{1, \dots, q\}$  denote the set of vertices representing the connections of the hybrid dynamical network. Furthermore, define the local resetting set  $\mathcal{Z}_i$  by

$$\mathcal{Z}_i \triangleq \left\{ x_i \in \mathbb{R} : \sum_{j \in O_i} \phi_{cij}(x_i, x_j)(x_i - x_j) - \sum_{j \in I_i} \phi_{cij}(x_i, x_j)(x_i - x_j) = 0 \right. \\ \left. \text{and } x_i \neq x_j, \quad j \in O_i \cup I_i \right\}, \quad i = 1, \dots, q \quad (25.13)$$

with

$$\mathcal{Z} \triangleq \bigcup_{i=1}^q \{x \in \mathbb{R}^q : x_i \in \mathcal{Z}_i\}. \quad (25.14)$$

It follows from Theorem 5.2 of [57] that (25.13) implies that if the time rate of change of the difference in entropies of the connected input information flow and output information flow between the agents is zero and consensus is not reached, then a resetting occurs.

To further ensure a thermodynamically consistent information flow model, we make the following assumptions on the information flow functions  $\phi_{cij}(\cdot, \cdot)$ ,  $i, j = 1, \dots, q$ , between state resettings:

**Assumption 25.3** The connectivity matrix  $C \in \mathbb{R}^{q \times q}$  associated with the hybrid multiagent dynamical system  $\mathcal{G}$  is defined by

$$C_{(i,j)} = \begin{cases} 0, & \text{if } \phi_{cij}(x_i(t), x_j(t)) \equiv 0, \\ 1, & \text{otherwise,} \end{cases} \quad i \neq j, \quad i, j = 1, \dots, q, \quad t \geq 0 \quad (25.15)$$

and

$$C_{(i,i)} = - \sum_{k=1, k \neq i}^q C_{(k,i)}, \quad i = j, \quad i = 1, \dots, q, \quad (25.16)$$

with  $\text{rank } C = q - 1$ , and for  $C_{(i,j)} = 1$ ,  $i \neq j$ ,  $\phi_{cij}(x_i(t), x_j(t)) = 0$  if and only if  $x_i(t) = x_j(t)$  for all  $x(t) \notin \mathcal{Z}$ ,  $t \geq 0$ .

**Assumption 25.4** For  $i, j = 1, \dots, q$ ,  $[x_i(t) - x_j(t)]\phi_{cij}(x_i(t), x_j(t)) \leq 0$ ,  $x(t) \notin \mathcal{Z}$ ,  $t \geq 0$ .

Furthermore, across resettings the information difference must satisfy the following assumption:

**Assumption 25.5** For  $i, j = 1, \dots, q$ ,  $[x_i(t_{k+1}) - x_j(t_{k+1})][x_i(t_k) - x_j(t_k)] \geq 0$  for all  $x_i(t_k) \neq x_j(t_k)$ ,  $x(t_k) \in \mathcal{Z}$ ,  $k \in \mathbb{Z}_+$ , where  $\mathbb{Z}_+$  denotes the set of non-negative integers.

The condition  $\phi_{cij}(x_i(t), x_j(t)) = 0$  if and only if  $x_i(t) = x_j(t)$ ,  $i \neq j$ , for all  $x(t) \notin \mathcal{Z}$  implies that agents  $\mathcal{G}_i$  and  $\mathcal{G}_j$  are *connected*, and hence, can share information; alternatively  $\phi_{cij}(x_i, x_j) \equiv 0$  implies that agents  $\mathcal{G}_i$  and  $\mathcal{G}_j$  are *disconnected*, and hence, cannot share information.

Assumption 25.3 implies that if the energies or information in the connected agents  $\mathcal{G}_i$  and  $\mathcal{G}_j$  are equal, then energy or information exchange between these agents is not possible. This statement is reminiscent of the *zeroth law of thermodynamics*, which postulates that temperature equality is a necessary and sufficient condition for thermal equilibrium. Furthermore, if  $C = C^T$  and  $\text{rank } C = q - 1$ , then it follows that the connectivity matrix  $C$  is irreducible, which implies that for any pair of agents  $\mathcal{G}_i$  and  $\mathcal{G}_j$ ,  $i \neq j$ , of  $\mathcal{G}$  there exists a sequence of information connectors (information arcs) of  $\mathcal{G}$  that connect  $\mathcal{G}_i$  and  $\mathcal{G}_j$ .

Assumption 25.4 implies that energy or information flows from more energetic or information rich agents to less energetic or information poor agents and is reminiscent of the *second law of thermodynamics*, which states that heat (i.e., energy in transition) must flow in the direction of lower temperatures. Finally, Assumption 25.5 implies that for any pair of connected agents  $\mathcal{G}_i$  and  $\mathcal{G}_j$ ,  $i \neq j$ , the energy or information difference between consecutive jumps is monotonic.

The following key assumption and definition are needed for the main result of this section:

**Assumption 25.6** Consider the impulsive dynamical system (25.1) and (25.2), and let  $s(t, x_0)$ ,  $t \geq 0$ , denote the solution to (25.1) and (25.2) with initial condition  $x_0$ . Then, for every  $x_0 \notin \mathcal{Z}$  and every  $\varepsilon > 0$  and  $t \neq t_k$ , there exists  $\delta(\varepsilon, x_0, t) > 0$  such that if  $\|x_0 - z\| < \delta(\varepsilon, x_0, t)$ ,  $z \in \mathcal{D}$ , then  $\|s(t, x_0) - s(t, z)\| < \varepsilon$ .

Assumption 25.6 is a weakened version of the quasi-continuous dependence assumption given in [35, 64] and is a generalization of the standard continuous dependence property for dynamical systems with continuous flows to dynamical systems with left-continuous flows. For the hybrid consensus protocol (25.7) and

(25.8) with  $\mathcal{Z}_i$  given by (25.13), a straightforward, but lengthy, calculation shows that Assumptions 25.1, 25.2, and 25.6 hold.

The following definition for semistability is needed. Recall that for addressing the stability of an impulsive dynamical system the usual stability definitions are valid [35]. For the statement of the next definition and theorem  $\mathcal{B}_\varepsilon(x)$  denotes the open ball centered at  $x$  with radius  $\varepsilon$  and  $\mathbf{e} \in \mathbb{R}^q$  denotes the ones vector of order  $q$ , that is,  $\mathbf{e} \triangleq [1, \dots, 1]^T$ .

**Definition 25.1** An equilibrium solution  $x(t) \equiv x_e \in \mathbb{R}^n$  to (25.1) and (25.2) is *semistable* if it is Lyapunov stable and there exists  $\delta > 0$  such that if  $x_0 \in \mathcal{B}_\delta(x_e)$ , then  $\lim_{t \rightarrow \infty} x(t)$  exists and corresponds to a Lyapunov stable equilibrium point. An equilibrium point  $x_e \in \mathbb{R}^n$  is a *globally semistable equilibrium* if it is Lyapunov stable and, for every  $x_0 \in \mathbb{R}^n$ ,  $\lim_{t \rightarrow \infty} x(t)$  exists and corresponds to a Lyapunov stable equilibrium point.

**Theorem 25.1** ([66]) Consider the closed-loop hybrid multiagent dynamical system  $\mathcal{G}$  given by (25.9) and (25.10) with resetting set  $\mathcal{Z}$  given by (25.14). Then, for every  $\alpha \geq 0$ ,  $\alpha\mathbf{e}$  is a semistable equilibrium state of  $\mathcal{G}$ . Furthermore,  $x(t) \rightarrow \frac{1}{q}\mathbf{e}\mathbf{e}^T x(0)$  as  $t \rightarrow \infty$  and  $\frac{1}{q}\mathbf{e}\mathbf{e}^T x(0)$  is a semistable equilibrium state.

Next, we provide explicit connections of the proposed thermodynamic-based control architecture developed in this section with the recently developed notion of hybrid thermodynamics [57]. To develop these connections the following hybrid definition of entropy is needed.

**Definition 25.2** For the distributed hybrid consensus control protocol  $\mathcal{G}$  given by (25.9) and (25.10), a function  $\mathcal{S} : \mathbb{R}^q \rightarrow \mathbb{R}$  satisfying

$$\mathcal{S}(x(T)) \geq \mathcal{S}(x(t_1)), \quad t_1 \leq t_k < T, \quad k \in \mathbb{Z}_+, \quad (25.17)$$

is called an *entropy* function of  $\mathcal{G}$ .

The next result gives necessary and sufficient conditions for establishing the existence of a hybrid entropy function of  $\mathcal{G}$  over an interval  $t \in (t_k, t_{k+1}]$  involving the consecutive resetting times  $t_k$  and  $t_{k+1}$ ,  $k \in \mathbb{Z}_+$ .

**Theorem 25.2** ([66]) Consider the closed-loop hybrid multiagent dynamical system  $\mathcal{G}$  given by (25.9) and (25.10), and assume Assumptions 25.3, 25.4, and 25.5 hold. Then a function  $\mathcal{S} : \mathbb{R}^q \rightarrow \mathbb{R}$  is an entropy function of  $\mathcal{G}$  if and only if

$$\mathcal{S}(x(\hat{t})) \geq \mathcal{S}(x(t)), \quad t_k < t \leq \hat{t} \leq t_{k+1}, \quad (25.18)$$

$$\mathcal{S}(x(t_k^+)) \geq \mathcal{S}(x(t_k)), \quad k \in \mathbb{Z}_+. \quad (25.19)$$

The next theorem establishes the existence of a continuously differentiable entropy function for the closed-loop hybrid multiagent dynamical system  $\mathcal{G}$  given by (25.9) and (25.10).

**Theorem 25.3** ([66]) Consider the closed-loop hybrid multiagent dynamical system  $\mathcal{G}$  given by (25.9) and (25.10), and assume Assumptions 25.4 and 25.5 hold. Then the function  $\mathcal{S} : \mathbb{R}^q \rightarrow \mathbb{R}$  given by

$$\mathcal{S}(x) = \mathbf{e}^T \log_e(c\mathbf{e} + x) - q \log_e c, \quad (25.20)$$

where  $\log_e(c\mathbf{e} + x)$  denotes the vector natural logarithm given by  $[\log_e(c + x_1), \dots, \log_e(c + x_q)]^T$  and  $c > \|x\|_\infty$ , is a continuously differentiable entropy function of  $\mathcal{G}$ . In addition,

$$\dot{\mathcal{S}}(x(t)) \geq 0, \quad x(t) \notin \mathcal{Z}, \quad t_k < t < t_{k+1}, \quad (25.21)$$

$$\Delta \mathcal{S}(x(t_k)) \geq 0, \quad x(t_k) \in \mathcal{Z}, \quad k \in \mathbb{Z}_+. \quad (25.22)$$

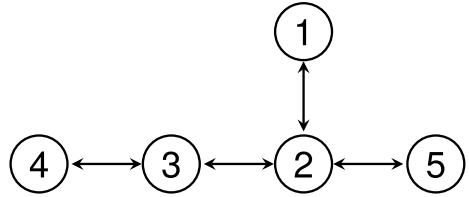
Note that it follows from Theorem 25.2 that the entropy function given by (25.20) satisfies (25.17) as an equality for an equilibrium process and as a strict inequality for a nonequilibrium process. The entropy expression given by (25.20) is identical in form to the Boltzmann entropy for statistical thermodynamics [56]. Due to the fact that the entropy is indeterminate to the extent of an additive constant, we can place the constant  $q \log_e c$  to zero by taking  $c = 1$ . Note that  $\mathcal{S}(x)$  given by (25.20) achieves a maximum when all the information states  $x_i, i = 1, \dots, q$ , are equal [55, 56]. Inequality (25.17) is a generalization of Clausius' inequality for equilibrium and nonequilibrium thermodynamics as well as reversible and irreversible thermodynamics as applied to adiabatically isolated hybrid large-scale thermodynamic systems involving discontinuous phase transitions. For details, see [57].

Next, we demonstrate the proposed distributed hybrid consensus framework on a set of aircrafts achieving asymptotic pitch rate consensus. Specifically, consider the multiagent system comprised of the controlled longitudinal motion of five Boeing 747 airplanes [67] linearized at an altitude of 40 kft and a velocity of 774 ft/s given by

$$\dot{z}_i(t) = Az_i(t) + B\delta_i(t), \quad z_i(0) = z_{i_0}, \quad i = 1, \dots, 5, \quad t \geq 0, \quad (25.23)$$

where  $z_i(t) = [v_{x_i}(t), v_{z_i}(t), q_i(t), \theta_{e_i}(t)]^T \in \mathbb{R}^4$ ,  $t \geq 0$ , is state vector of agent  $i \in \{1, \dots, 5\}$ , with  $v_{x_i}(t)$ ,  $t \geq 0$ , representing the  $x$ -body-axis component of the velocity of the airplane center of mass with respect to the reference axes (in ft/s),  $v_{z_i}(t)$ ,  $t \geq 0$ , representing the  $z$ -body-axis component of the velocity of the airplane center of mass with respect to the reference axes (in ft/s),  $q_i(t)$ ,  $t \geq 0$ , representing the  $y$ -body-axis component of the angular velocity of the airplane (pitch rate) with respect to the reference axes (in crad/s),  $\theta_{e_i}(t)$ ,  $t \geq 0$ , representing the pitch Euler angle of the airplane body axes with respect to the reference axes (in crad),  $\delta(t)$ ,  $t \geq 0$ , representing the elevator control input (in crad), and

**Fig. 25.1** Aircraft communication topology



$$A = \begin{bmatrix} -0.003 & 0.039 & 0 & -0.332 \\ -0.065 & -0.319 & 7.74 & 0 \\ 0.020 & -0.101 & -0.429 & 0 \\ 0 & 0 & 1 & 0 \end{bmatrix}, \quad B = \begin{bmatrix} 0.010 \\ -0.180 \\ -1.16 \\ 0 \end{bmatrix}. \quad (25.24)$$

Here we use a two-level control hierarchy comprised of a lower level controller for command following and a higher level hybrid consensus controller for pitch rate consensus with a communication topology as shown in Fig. 25.1. To address the lower level controller design, let  $x_i(t)$ ,  $i = 1, \dots, 5$ ,  $t \geq 0$ , denote a command generated by (25.7) and (25.8) (i.e., the guidance command) and let  $s_i(t)$ ,  $i = 1, \dots, 5$ ,  $t \geq 0$ , denote the integrator state satisfying

$$\dot{s}_i(t) = Ez_i(t) - x_i(t), \quad s_i(0) = s_{i_0}, \quad i = 1, \dots, 5, \quad t \geq 0, \quad (25.25)$$

where  $E = [0, 0, 1, 0]$ . Now, defining the augmented state  $\hat{z}(t) \triangleq [z^T(t), s_i(t)]^T$ , (25.23) and (25.25) give

$$\dot{\hat{z}}_i(t) = \hat{A}\hat{z}_i(t) + \hat{B}_1\delta_i(t) + \hat{B}_2x_i(t), \quad \hat{z}_i(0) = \hat{z}_{i_0}, \quad i = 1, \dots, 5, \quad t \geq 0, \quad (25.26)$$

where

$$\hat{A} \triangleq \begin{bmatrix} A & 0 \\ E & 0 \end{bmatrix}, \quad \hat{B}_1 \triangleq \begin{bmatrix} B \\ 0 \end{bmatrix}, \quad \hat{B}_2 \triangleq \begin{bmatrix} 0 \\ -1 \end{bmatrix}. \quad (25.27)$$

Furthermore, let the elevator control input be given by

$$\delta(t) = -K\hat{z}(t), \quad (25.28)$$

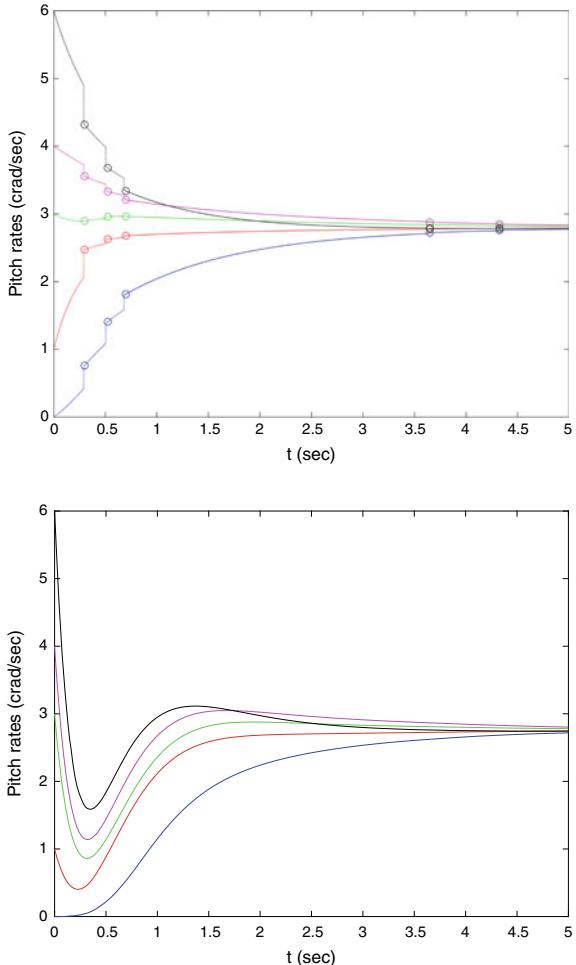
where

$$K = [-0.0157, 0.0831, -4.7557, -0.1400, -9.8603],$$

which is designed based on an optimal linear-quadratic regulator.

For the higher level hybrid consensus controller design, we use (25.7) with  $\phi_{cij}(x(t)) = x_j(t) - x_i(t)$ ,  $i, j \in \{1, \dots, 5\}$ ,  $i \neq j$ , and (25.8) with  $\phi_{dij}(x(t)) = (x_j(t) - x_i(t))/5$ ,  $i, j \in \{1, \dots, 5\}$ ,  $i \neq j$ , to generate  $x(t)$ ,  $t \geq 0$ , that has a direct effect on the lower level controller design to achieve pitch rate consensus. For our simulation we set  $x_1(0) = q_1(0) = 0$ ,  $x_2(0) = q_2(0) = 1$ ,  $x_3(0) = q_3(0) = 3$ ,  $x_4(0) =$

**Fig. 25.2** Closed-loop command signal  $x_i(t)$  (top) and pitch rate  $q_i(t)$  (bottom) trajectories for the hybrid consensus protocol;  $x_1(t)$  in blue,  $x_2(t)$  in red,  $x_3(t)$  in green,  $x_4(t)$  in magenta, and  $x_5(t)$  in black



$q_4(0) = 4$ ,  $x_5(0) = q_5(0) = 6$  and all other initial conditions to zero. Figure 25.2 presents the command signal and pitch rate trajectories of each aircraft versus time. Figure 25.3 shows the system entropy versus time.

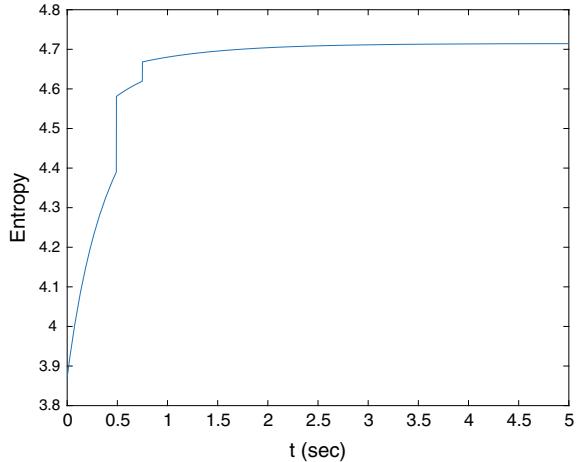
Finally, to present the key ideas for extending this framework to hybrid *dynamic* protocols, consider the continuous-time nonlinear dynamical system given by

$$\dot{x}_p(t) = f_p(x_p(t), u(t)), \quad x_p(0) = x_{p0}, \quad t \geq 0, \quad (25.29)$$

$$y(t) = h_p(x_p(t)), \quad (25.30)$$

where, for every  $t \geq 0$ ,  $x_p(t) \in \mathcal{D}_p \subseteq \mathbb{R}^{n_p}$ ,  $\mathcal{D}_p$  is an open set,  $u(t) \in \mathbb{R}^m$ ,  $f_p : \mathcal{D}_p \times \mathbb{R}^m \rightarrow \mathbb{R}^{n_p}$  is smooth (i.e., infinitely differentiable) on  $\mathcal{D}_p \times \mathbb{R}^m$ , and  $h_p : \mathcal{D}_p \rightarrow \mathbb{R}^l$  is smooth. Furthermore, consider the hybrid (i.e., resetting) dynamic controller given

**Fig. 25.3** System entropy versus time



by

$$\dot{x}_c(t) = f_{cc}(x_c(t), y(t)), \quad x_c(0) = x_{c0}, \quad (x_c(t), y(t)) \notin \mathcal{Z}_c, \quad (25.31)$$

$$\Delta x_c(t) = f_{dc}(x_c(t), y(t)), \quad (x_c(t), y(t)) \in \mathcal{Z}_c, \quad (25.32)$$

$$u(t) = h_{cc}(x_c(t), y(t)), \quad (25.33)$$

where, for every  $t \geq 0$ ,  $x_c(t) \in \mathcal{D}_c \subseteq \mathbb{R}^{n_c}$ ,  $\mathcal{D}_c$  is an open set,  $\Delta x_c(t) \triangleq x_c(t^+) - x_c(t)$ , where  $x_c(t^+) \triangleq x_c(t) + f_{dc}(x_c(t), y(t)) = \lim_{\varepsilon \rightarrow 0^+} x_c(t + \varepsilon)$ ,  $(x_c(t), y(t)) \in \mathcal{Z}_c$ ,  $f_{cc} : \mathcal{D}_c \times \mathbb{R}^l \rightarrow \mathbb{R}^{n_c}$  is smooth on  $\mathcal{D}_c \times \mathbb{R}^l$ ,  $h_{cc} : \mathcal{D}_c \times \mathbb{R}^l \rightarrow \mathbb{R}^m$  is smooth,  $f_{dc} : \mathcal{D}_c \times \mathbb{R}^l \rightarrow \mathbb{R}^{n_c}$  is continuous, and  $\mathcal{Z}_c \subset \mathcal{D}_c \times \mathbb{R}^l$  is the *resetting set*. Note that, for generality, we allow the hybrid dynamic controller to be of fixed dimension  $n_c$ , which may be less than the plant order  $n_p$ .

The equations of motion for the closed-loop dynamical system (25.29)–(25.33) have the form (25.1) and (25.2) with

$$x \triangleq \begin{bmatrix} x_p \\ x_c \end{bmatrix} \in \mathbb{R}^n, \quad f_c(x) \triangleq \begin{bmatrix} f_p(x_p, h_{cc}(x_c, h_p(x_p))) \\ f_{cc}(x_c, h_p(x_p)) \end{bmatrix}, \quad f_d(x) \triangleq \begin{bmatrix} 0 \\ f_{dc}(x_c, h_p(x_p)) \end{bmatrix}, \quad (25.34)$$

and  $\mathcal{Z} \triangleq \{x \in \mathcal{D} : (x_c, h_p(x_p)) \in \mathcal{Z}_c\}$ , with  $n \triangleq n_p + n_c$  and  $\mathcal{D} \triangleq \mathcal{D}_p \times \mathcal{D}_c$ . Note that although the closed-loop state vector consists of plant states and controller states, it is clear from (25.34) that only those states associated with the controller are reset.

As shown in [68], this hybrid dynamic controller architecture can be used to develop a unified framework for addressing *general* nonlinear dynamical systems of the form (25.29) and (25.30) for achieving consensus, flocking, and cyclic pursuit. Specifically, hybrid distributed and decentralized controller architectures for multiagent coordination can be developed. In contrast to virtually all of the existing results in the literature on control of networks, the proposed controllers will be dynamic compensators. Another unique feature of this framework is that the con-

troller architectures are hybrid, and hence, the overall closed-loop dynamics under these controller algorithms achieving consensus, flocking, or cyclic pursuit will possess discontinuous flows since they will combine logical switchings with continuous dynamics, leading to impulsive differential equations [35, 40, 45, 48, 62–64].

The proposed controllers can use undirected and directed graphs to accommodate for a full range of possible graph information topologies without limitations of bidirectional communication. Furthermore, finite time consensus protocols for addressing finite time coordination, as well as addressing disturbance rejection and robustness extensions can also be developed (see Sect. 25.3). In addition, asynchronism, system time-delays, and dynamic (i.e., switching) network topologies for addressing possible information asynchrony between agents, message transmission and processing delays, and communication link failures and communication dropouts can be addressed.

### 25.3 Multiagent Systems with Uncertain Interagent Communication

To capture network system uncertainty and communication uncertainty between the agents in a network, almost sure consensus protocols for multiagent systems with nonlinear stochastic dynamics can be formulated using a stochastic thermodynamics formulation [56, 69]. Specifically, the notions of deterministic semistability and finite time semistability to nonlinear stochastic dynamical systems that possess a continuum of equilibrium solutions can be extended to develop almost sure *asymptotic* and *finite time* convergence as well as stochastic Lyapunov stability properties.

For deterministic dynamical systems, the authors in [25, 27–29] developed a unified stability analysis framework for systems having a continuum of equilibria. Since, as noted earlier, every neighborhood of a nonisolated equilibrium contains another equilibrium, a nonisolated equilibrium cannot be asymptotically stable nor finite time stable. Hence, asymptotic and finite time stability are not the appropriate notions of stability for systems having a continuum of equilibria. Two notions that are of particular relevance to such systems are convergence and semistability. Convergence is the property whereby every system solution converges (asymptotically or in finite time) to a limit point that may depend on the system initial condition. Semistability (resp., finite time semistability) is the additional requirement that all solutions converge asymptotically (resp., in finite time) to limit points that are Lyapunov stable. Semistability (resp., finite time semistability) for an equilibrium thus implies Lyapunov stability and is implied by asymptotic (resp., finite time) stability.

It is important to note that semistability is not merely equivalent to asymptotic stability of the set of equilibria. Indeed, it is possible for a trajectory to converge to the set of equilibria without converging to any one equilibrium point [27]. Conversely, semistability does not imply that the equilibrium set is asymptotically stable in any accepted sense. This is because the stability of sets is defined in terms of distance

(especially in case of noncompact sets), and it is possible to construct examples in which the dynamical system is semistable, but the domain of semistability contains no  $\varepsilon$ -neighborhood (defined in terms of the distance) of the (noncompact) equilibrium set, thus ruling out asymptotic stability of the equilibrium set. Hence, semistability and set stability of the equilibrium set are independent notions.

The theories of semistability and finite time semistability for deterministic dynamical systems developed in [25, 27–29] were extended in [58] to develop a rigorous framework for *stochastic semistability* and *stochastic finite time semistability*. In particular, [58] presents new Lyapunov theorems as well as the first converse Lyapunov theorem for stochastic semistability, which holds with a continuous Lyapunov function whose infinitesimal generator decreases along the stochastic dynamical system sample trajectories and is such that the Lyapunov function satisfies inequalities involving the average distance to the set of equilibria.

Next, [58] establishes stochastic finite time semistability theory. In particular, we develop the notions of finite time convergence in probability and finite time semistability in probability for nonlinear stochastic dynamical systems driven by Markov diffusion processes. Furthermore, the continuity of a settling-time operator is established and a sufficient Lyapunov stability theorem for finite time semistability in probability is developed. Specifically, almost sure finite time convergence and stochastic Lyapunov stability properties are developed to address almost sure finite time semistability requiring that the sample trajectories of a nonlinear stochastic dynamical system converge almost surely in finite time to a set of equilibrium solutions, wherein every equilibrium solution in the set is almost surely Lyapunov stable.

Even though convergence, semistability, finite time semistability, and optimality for *deterministic* multiagent network systems involving cooperative control tasks such as formation control, rendezvous, flocking, cyclic pursuit, and consensus have received considerable attention in the literature (see, for example, [1–5, 7–10, 10–15, 17, 18, 20–26, 60, 70–76]), stochastic multiagent networks have not been as plethoraically developed; notable contributions include [77–81]. These contributions address asymptotic convergence [79], time-varying network topologies [78], communication delays [81], asynchronous switchings [80], and optimality [77]; however, with the exception of [58], none of the aforementioned references address the problems of stochastic semistability and stochastic finite time semistability.

As noted in the Introduction, a unique feature of the closed-loop dynamics under any control algorithm that achieves consensus in a dynamical network is the existence of a continuum of equilibria representing a state of consensus. Under such dynamics, the limiting consensus state achieved is not determined completely by the dynamics but depends on the initial system state as well. Thus, from a practical viewpoint, it is not sufficient for a nonlinear control protocol to only guarantee that a network converges to a state of consensus since steady-state convergence is not sufficient to guarantee that small perturbations from the limiting state will lead to only small transient excursions from a state of consensus. It is also necessary to guarantee that the equilibrium states representing consensus are Lyapunov stable, and consequently, semistable.

To capture network system uncertainty and communication uncertainty between the agents in a network, wherein the evolution of each link of the network communication topology follows a Markov process for modeling unknown communication noise and attenuations, stochastic semistability and finite time semistability results can be used to develop almost sure consensus protocols for multiagent systems with nonlinear stochastic dynamics. Specifically, stochastic semistability and stochastic finite time semistability frameworks can be used to design distributed asymptotic and finite time consensus control protocols for nonlinear bidirectional dynamical networks with stochastic communication uncertainty. The proposed controller architectures can be predicated on the recently developed notion of stochastic dynamical thermodynamics [56, 69] resulting in controller architectures involving the exchange of generalized charge or energy state information between agents that guarantee that the closed-loop dynamical network is consistent with stochastic thermodynamic principles.

To elucidate the basis of this approach, we require some additional notation. Specifically, we denote by  $\mathcal{G}$  a stochastic dynamical system generating a stochastic process  $x : [0, \infty) \times \Omega \rightarrow \mathbb{R}^n$  on a complete probability space  $(\Omega, \mathcal{F}, \mathbb{P})$ , where  $\Omega$  denotes the sample space,  $\mathcal{F}$  denotes a  $\sigma$ -algebra of subsets of  $\Omega$ , and  $\mathbb{P}$  defines a probability measure on  $\mathcal{F}$ . Moreover, all inequalities and equalities involving random process on  $(\Omega, \mathcal{F}, \mathbb{P})$  are to be understood to hold  $\mathbb{P}$ -almost surely (a.s.), and with  $\mathbb{E}^{x_0}[\cdot]$  denoting expectation with respect to the classical Wiener measure  $\mathbb{P}^{x_0}$  and  $\mathcal{H}_n^{\mathcal{D}}$  denoting the Hilbert space of random variables  $x \in \mathbb{R}^n$  with finite average power induced by  $\mathcal{D}$ .

Consider the nonlinear stochastic dynamical system  $\mathcal{G}$  given by

$$dx(t) = f(x(t))dt + D(x(t))dw(t), \quad x(t_0) \stackrel{\text{a.s.}}{=} x_0, \quad t \geq t_0, \quad (25.35)$$

where, for every  $t \geq t_0$ ,  $x(t) \in \mathcal{H}_n^{\mathcal{D}}$  is a  $\mathcal{F}_t$ -measurable random state vector,  $x(t_0) \in \mathcal{H}_n^{x_0}$ ,  $\mathcal{D} \subseteq \mathbb{R}^n$  is an open set with  $0 \in \mathcal{D}$ ,  $w(t)$  is a  $d$ -dimensional independent standard Wiener process (i.e., Brownian motion) defined on a complete filtered probability space  $(\Omega, \{\mathcal{F}_t\}_{t \geq t_0}, \mathbb{P})$ ,  $x(t_0)$  is independent of  $(w(t) - w(t_0))$ ,  $t \geq t_0$ , and  $f : \mathcal{D} \rightarrow \mathbb{R}^n$  and  $D : \mathcal{D} \rightarrow \mathbb{R}^{n \times d}$  are continuous functions and satisfy  $f(x_e) = 0$  and  $D(x_e) = 0$  for some  $x_e \in \mathcal{D}$ . An *equilibrium point* of (25.35) is a point  $x_e \in \mathcal{D}$  such that  $f(x_e) = 0$  and  $D(x_e) = 0$ . It is easy to see that  $x_e$  is an equilibrium point of (25.35) if and only if the constant stochastic process  $x(\cdot) \stackrel{\text{a.s.}}{=} x_e$  is a solution of (25.35). We denote the set of equilibrium points of (25.35) by  $\mathcal{E} \triangleq \{\omega \in \Omega : x(t, \omega) = x_e\} = \{x_e \in \mathcal{D} : f(x_e) = 0 \text{ and } D(x_e) = 0\}$ . Here, we assume that all maximal pathwise solutions to (25.35) in  $(\Omega, \{\mathcal{F}_t\}_{t \geq t_0}, \mathbb{P}^{x_0})$  exist on  $[t_0, \infty)$ , and hence, we assume that (25.35) is *forward complete*.

The following definition introduces the notion of stochastic semistability.

**Definition 25.3** An equilibrium solution  $x(t) \stackrel{\text{a.s.}}{=} x_e \in \mathcal{E}$  of (25.35) is *stochastically semistable* if the following statements hold:

- (i) For every  $\varepsilon > 0$ ,  $\lim_{x_0 \rightarrow x_e} \mathbb{P}^{x_0} (\sup_{0 \leq t < \infty} \|x(t) - x_e\| > \varepsilon) = 0$ . Equivalently, for every  $\varepsilon > 0$  and  $\rho \in (0, 1)$ , there exists  $\delta = \delta(\varepsilon, \rho) > 0$  such that, for all  $x_0 \in \mathcal{B}_\delta(x_e)$ ,

$$\mathbb{P}^{x_0} \left( \sup_{0 \leq t < \infty} \|x(t) - x_e\| > \varepsilon \right) \leq \rho.$$

- (ii)  $\lim_{\text{dist}(x_0, \mathcal{E}) \rightarrow 0} \mathbb{P}^{x_0} (\lim_{t \rightarrow \infty} \text{dist}(x(t), \mathcal{E}) = 0) = 1$ . Equivalently, for every  $\rho \in (0, 1)$ , there exists  $\delta = \delta(\rho) > 0$  such that if  $\text{dist}(x_0, \mathcal{E}) \leq \delta$ , then  $\mathbb{P}^{x_0} (\lim_{t \rightarrow \infty} \text{dist}(x(t), \mathcal{E}) = 0) \geq 1 - \rho$ .

The dynamical system (25.35) is *stochastically semistable* if every equilibrium solution of (25.35) is stochastically semistable. Finally, the dynamical system (25.35) is *globally stochastically semistable* if (i) holds and  $\mathbb{P}^{x_0} (\lim_{t \rightarrow \infty} \text{dist}(x(t), \mathcal{E}) = 0) = 1$  for all  $x_0 \in \mathbb{R}^n$ .

Note that if  $x(t) \stackrel{\text{a.s.}}{\equiv} x_e \in \mathcal{E}$  only satisfies (i) in Definition 25.3, then the equilibrium solution  $x(t) \stackrel{\text{a.s.}}{\equiv} x_e \in \mathcal{E}$  of (25.35) is Lyapunov stable in probability.

The notion of stochastic finite time semistability involves finite time almost sure convergence along with stochastic semistability. For this definition,  $s^x$  represents a sample path trajectory of  $\mathcal{G}$  with initial condition  $x$ .

**Definition 25.4** An equilibrium solution  $x(t) \stackrel{\text{a.s.}}{\equiv} x_e \in \mathcal{E}$  of (25.35) is (*globally*) *stochastically finite time semistable* if there exists an operator  $T : \mathcal{H}_n \rightarrow \mathcal{H}_1^{[0, \infty)}$ , called the *stochastic settling-time operator*, such that the following statements hold:

- (i) *Finite time convergence in probability.* For every  $x(0) \in \mathcal{H}_n \setminus \mathcal{E}$ ,  $s^{x(0)}(t)$  is defined on  $[0, T(x(0))]$ ,  $s^{x(0)}(t) \in \mathcal{H}_n \setminus \mathcal{E}$  for all  $t \in [0, T(x(0))]$ , and

$$\mathbb{P}^{x_0} \left( \lim_{t \rightarrow T(x(0))} \text{dist}(s^{x(0)}(t), \mathcal{E}) = 0 \right) = 1.$$

- (ii) *Lyapunov stability in probability.* For every  $\varepsilon > 0$ ,

$$\lim_{x_0 \rightarrow x_e} \mathbb{P}^{x_0} \left( \sup_{0 \leq t < \infty} \|s^{x(0)}(t) - x_e\| > \varepsilon \right) = 0.$$

Equivalently, for every  $\varepsilon > 0$  and  $\rho \in (0, 1)$ , there exist  $\delta = \delta(\varepsilon, \rho) > 0$  such that, for all  $x_0 \in \mathcal{B}_\delta(x_e)$ ,  $\mathbb{P}^{x_0} (\sup_{0 \leq t < \infty} \|s^{x(0)}(t) - x_e\| > \varepsilon) \leq \rho$ .

- (iii) *Finiteness of the stochastic settling-time operator.* For every  $x \in \mathcal{H}_n \setminus \mathcal{E}$  the stochastic settling-time operator  $T(x)$  exists and is finite with probability one, that is,  $\mathbb{E}^x [T(x)] < \infty$ .

The dynamical system (25.35) is (*globally*) *stochastically finite time semistable* if every equilibrium solution of (25.35) is globally stochastically finite time semistable.

The following additional notion and definitions are needed. Specifically, let  $\mathfrak{G}(C) = (\mathcal{V}, \mathcal{E})$  be a *directed graph* (or digraph) denoting the dynamical network

(or dynamic graph) with the set of *nodes* (or vertices)  $\mathcal{V} = \{1, \dots, q\}$  involving a finite nonempty set denoting the agents, the set of edges  $\mathcal{E} \subseteq \mathcal{V} \times \mathcal{V}$  involving a set of ordered pairs denoting the direction of information flow, and a *connectivity matrix*  $C \in \mathbb{R}^{q \times q}$  such that  $C_{(i,j)} = 1$ ,  $i, j = 1, \dots, q$ , if  $(j, i) \in \mathcal{E}$ , while  $C_{(i,j)} = 0$  if  $(j, i) \notin \mathcal{E}$ . The edge  $(j, i) \in \mathcal{E}$  denotes that agent  $j$  can obtain information from agent  $i$ , but not necessarily vice versa. Moreover, we assume  $C_{(i,i)} = 0$  for all  $i \in \mathcal{V}$ . A *graph* or *undirected graph*  $\mathfrak{G}$  associated with the connectivity matrix  $C \in \mathbb{R}^{q \times q}$  is a directed graph for which the *arc set* is symmetric, that is,  $C = C^T$ . Weighted graphs can also be considered here; however, since this extension does not alter any of the conceptual results in the paper we do not consider this extension for simplicity of exposition.

To address the consensus problem with communication uncertainty, consider  $q$  continuous-time agents with dynamics

$$dx_i(t) = u_i(t)dt + \text{row}_i(D(x(t)))dw(t), \quad i = 1, \dots, q, \quad x_i(0) \stackrel{\text{a.s.}}{=} x_{i0}, \quad t \geq 0, \quad (25.36)$$

where  $q \geq 2$  is the number of agents in the network with a communication graph topology  $\mathfrak{G}(C)$ ,  $D(x)dw$ , where  $D(x) = [\text{row}_1(D(x)), \dots, \text{row}_q(D(x))]^T : \mathbb{R}^q \rightarrow \mathbb{R}^q \times \mathbb{R}^d$ , captures probabilistic variations in the information transfer rates between agents, and, for every  $i \in \{1, \dots, q\}$ ,  $x_i(t) \in \mathcal{H}_1$  denotes the information state of the  $i$ th agent and  $u_i(t) \in \mathcal{H}_1$  denotes the control input of the  $i$ th agent. For a general distributed control architecture resulting in a network consensus action corresponding to an underlying conservation law, we assume  $\mathbf{e}_q^T D(x) = 0$ ,  $x \in \mathbb{R}^q$ , where  $\mathbf{e}_q \triangleq [1, \dots, 1]^T \in \mathbb{R}^q$ , and where the agent state  $x_i(t) \in \mathcal{H}_1$  denotes the generalized charge (i.e., Nöether charge or simply charge) state and the control input  $u_i(t) \in \mathcal{H}_1$  denotes the conserved current input for all  $t \geq 0$ .

The nonlinear consensus protocol is given by

$$u_i(t) = \sum_{j=1, j \neq i}^q C_{(i,j)} [\sigma_{ij}(x_j(t)) - \sigma_{ji}(x_i(t))], \quad (25.37)$$

where  $\sigma_{ij}(\cdot)$ ,  $i, j \in \{1, \dots, q\}$ ,  $i \neq j$ , are Lipschitz continuous. Here we assume that the control process  $u_i(\cdot)$  in (25.37) is restricted to a class of admissible control protocols consisting of measurable functions adapted to the filtration  $\{\mathcal{F}_t\}_{t \geq 0}$  such that, for every  $i \in \{1, \dots, q\}$ ,  $u_i(\cdot) \in \mathcal{H}_1$ ,  $t \geq 0$ , and, for all  $t \geq s$ ,  $w_i(t) - w_i(s)$  is independent of  $u_i(\tau)$ ,  $w_i(\tau)$ ,  $\tau \leq s$ , and  $x_i(0)$ , and hence,  $u_i(\cdot)$  is nonanticipative. Furthermore, we assume  $u_i(\cdot)$  takes values in a compact metrizable set, and hence, it follows from Theorem 2.2.4 of [82] that there exists a unique pathwise solution to (25.36) and (25.37) in  $(\Omega, \{\mathcal{F}_t\}_{t \geq 0}, \mathbb{P}^{x_{i0}})$  for every  $i \in \{1, \dots, q\}$ . Finally, note that the closed-loop system (25.36) and (25.37) is given by

$$\begin{aligned} dx_i(t) = \sum_{j=1, j \neq i}^q C_{(i,j)} [\sigma_{ij}(x_j(t)) - \sigma_{ji}(x_i(t))] dt + \text{row}_i(D(x(t))) dw(t), \\ i = 1, \dots, q, \quad x_i(0) \stackrel{\text{a.s.}}{=} x_{i0}, \quad t \geq 0. \end{aligned} \quad (25.38)$$

Equation (25.38) represents the collective dynamics of  $q$  agents which interact by exchanging charge. The coefficients scaling the functions  $\sigma_{ij}(\cdot)$ ,  $i, j \in \{1, \dots, q\}$ ,  $i \neq j$ , appearing in (25.38) represent the topology of the charge exchange between the agents. More specifically, given  $i, j \in \{1, \dots, q\}$ ,  $i \neq j$ , a coefficient of  $C_{(i,j)} = 1$  denotes that subsystem  $j$  receives charge from subsystem  $i$ , and a coefficient of zero denotes that subsystem  $i$  and  $j$  are disconnected, and hence, cannot share any charge.

As in Sect. 25.2, our results can be directly extended to the case where (25.36) and (25.37) describe the dynamics of an aggregate multiagent system with an aggregate state vector  $x(t) = [x_1^T(t), \dots, x_q^T(t)]^T \in \mathcal{H}_{Nq}$ , where  $x_i(t) \in \mathcal{H}_N$  and  $u_i(t) \in \mathcal{H}_N$ ,  $i = 1, \dots, q$ , by using Kronecker algebra. Here, for simplicity of exposition, we focus on individual agent states evolving in  $\mathcal{H}_1$  (i.e.,  $N = 1$ ).

Next, note that since

$$\mathbf{e}_q^T dx(t) = \mathbf{e}_q^T f(x(t)) dt + \mathbf{e}_q^T D(x(t)) dw(t) = 0, \quad x(0) \stackrel{\text{a.s.}}{=} x_0, \quad t \geq 0, \quad (25.39)$$

it follows that  $\sum_{i=1}^q dx_i(t) \stackrel{\text{a.s.}}{=} 0$ ,  $t \geq 0$ , which implies that the total system charge is conserved, and hence, the controlled network satisfies an underlying conservation law. Now, it follows from Nöther's theorem [83] that to every conservation law there corresponds a symmetry. To show this for our multiagent network, the following definition and assumptions are needed.

**Definition 25.5** ([84]) A directed graph  $\mathfrak{G}(C)$  is *strongly connected* if for every ordered pair of vertices  $(i, j)$ ,  $i \neq j$ , there exists a *path* (i.e., a sequence of arcs) leading from  $i$  to  $j$ .

Recall that the connectivity matrix  $C \in \mathbb{R}^{q \times q}$  is *irreducible*, that is, there does not exist a permutation matrix such that  $C$  is cogredient to a lower block triangular matrix, if and only if  $\mathfrak{G}(C)$  is strongly connected (see Theorem 2.7 of [84]).

**Assumption 25.7** The connectivity matrix  $C \in \mathbb{R}^{q \times q}$  associated with the multiagent stochastic dynamical system  $\mathcal{G}$  is defined by

$$C_{(i,j)} \triangleq \begin{cases} 0, & \text{if } \sigma_{ij}(x_j) - \sigma_{ji}(x_i) \equiv 0, \\ 1, & \text{otherwise,} \end{cases} \quad i \neq j, \quad i, j = 1, \dots, q, \quad (25.40)$$

and

$$C_{(i,i)} \triangleq - \sum_{k=1, k \neq i}^q C_{(k,i)}, \quad i = j, \quad i = 1, \dots, q, \quad (25.41)$$

with rank  $C = q - 1$ , and for  $C_{(i,j)} = 1$ ,  $i \neq j$ ,  $\sigma_{ij}(x_j) - \sigma_{ji}(x_i) = 0$  if and only if  $x_i = x_j$ .

**Assumption 25.8** For  $i, j = 1, \dots, q$ ,

$$\sum_{j=1, j \neq i}^q C_{(i,j)}(x_i - x_j)[\sigma_{ij}(x_j) - \sigma_{ji}(x_i)] \leq -\text{row}_i(D(x))\text{row}_i^T(D(x)).$$

The information connectivity between the agents can be represented by the network communication graph topology  $\mathcal{G}(C)$  having  $q$  nodes such that  $\mathcal{G}(C)$  has an undirected edge from node  $i$  to node  $j$  if and only if agent  $j$  can receive charge from agent  $i$ . Since the coefficients scaling  $\sigma_{ij}(\cdot)$ ,  $i, j \in \{1, \dots, q\}$ ,  $i \neq j$ , are constants, the communication graph topology of the network  $\mathcal{G}(C)$  is fixed. Furthermore, note that the graph  $\mathcal{G}$  is *weakly connected* since the underlying undirected graph is connected; that is, every agent receives charge from, or delivers charge to, at least one other agent.

The fact that  $\sigma_{ij}(x_j) - \sigma_{ji}(x_i) = 0$  if and only if  $x_i = x_j$ ,  $i \neq j$ , implies that agent  $i$  and  $j$  are *connected*, and hence, can share information; alternatively,  $\sigma_{ij}(x_j) - \sigma_{ji}(x_i) \equiv 0$  implies that agent  $i$  and  $j$  are *disconnected*, and hence, cannot share information. Assumption 25.7 thus implies that if the charge (or generalized energies) in the connected agents  $i$  and  $j$  are equal, then charge exchange between the agents is not possible. This statement is reminiscent of the *zeroth law of thermodynamics*, which postulates that temperature equality is a necessary and sufficient condition for thermal equilibrium. Furthermore, if  $C = C^T$  and rank  $C = q - 1$ , then it follows that the connectivity matrix  $C$  is irreducible, which implies that for any pair of  $i$  and  $j$ ,  $i \neq j$ , of  $\mathcal{G}$  there exists a sequence information connectors (information arcs) of  $\mathcal{G}$  that connect agents  $i$  and  $j$ .

Assumption 25.8 implies that charge flows from charge rich agents to charge poor agents and is reminiscent of the *second law of thermodynamics*, which states that heat (i.e., energy in transition) must flow in the direction of lower temperatures. It is important to note here that due to the stochastic term  $D(x)dw$  capturing probabilistic variations in the charge transfer (i.e., generalized current) between the agents, the second assumption requires that the scaled net charge flow  $C_{(i,j)}(x_i - x_j)[\sigma_{ij}(x_j) - \sigma_{ji}(x_i)]$  is bounded by the negative intensity of the diffusion coefficient given by  $\frac{1}{2}\text{tr } D(x)D^T(x)$ . For further details on Assumptions 25.7 and 25.8, see [56, 69].

The intensity  $D(x)$  of the general probabilistic variations  $D(x)dw$  in the agent communication can take different forms to capture communication measurement noise or errors in the information transfer rates between agents. For example, we can consider  $D(x) = M\hat{D}(x)$ , where

$$\begin{aligned} M &\triangleq [m_{(1,2)}, \dots, m_{(1,q)}, m_{(2,3)}, \dots, m_{(2,q)}, \dots, m_{(q-1,q)}] \in \mathbb{R}^{q \times \frac{1}{2}q(q-1)} \\ \hat{D}(x) &\triangleq \text{diag}[d_{(1,2)}(x), \dots, d_{(1,q)}(x), d_{(2,3)}(x), \dots, d_{(2,q)}(x), \dots, d_{(q-1,q)}(x)] \\ &\in \mathbb{R}^{\frac{1}{2}q(q-1) \times \frac{1}{2}q(q-1)} \end{aligned}$$

and  $m_{(i,j)}d_{(i,j)}(x_i, x_j)dw_i$  represents stochastic variations in the information flow between the  $i$ th and  $j$ th agent. Furthermore, considering

$$d_{(i,j)}(x_i, x_j) = C_{(i,j)}(x_j - x_i)^p, \quad (25.42)$$

where  $p > 0$  and  $m_{(i,j)} \in \mathbb{R}^q$  satisfies  $m_{(i,j)i} \geq 0$ ,  $m_{(i,j)j} \leq 0$ ,  $m_{(i,j)i} = -m_{(i,j)j}$ ,  $m_{(i,j)k} = 0$ ,  $k \neq i, k \neq j$ , where  $m_{(i,j)i}$  denotes the  $i$ th component of  $m_{(i,j)}$ , it follows that  $\mathbf{e}_q^T m_{(i,j)} = 0$ , and hence, it can be shown that (25.39) holds. Note that (25.42) captures nonlinear relative uncertainty between interagent communication. Of course, more general nonlinear uncertainties can also be considered.

For simplicity of exposition, in the remainder of this section we let  $d = 1$  and  $p = 1$ , and consider  $q$  continuous-time agents with dynamics

$$\begin{aligned} dx_i(t) &= u_i(t)dt + \sum_{j=1, j \neq i}^q \gamma C_{(i,j)}[x_j(t) - x_i(t)]dw(t), \\ i &= 1, \dots, q, \quad x_i(0) \stackrel{\text{a.s.}}{=} x_{i0}, \quad t \geq 0, \end{aligned} \quad (25.43)$$

where  $\gamma \in \mathbb{R}$ , so that the closed-loop system (25.43) and (25.37) is given by

$$\begin{aligned} dx_i(t) &= \sum_{j=1, j \neq i}^q C_{(i,j)}[\sigma_{ij}(x_j(t)) - \sigma_{ji}(x_i(t))]dt \\ &\quad + \sum_{j=1, j \neq i}^q \gamma C_{(i,j)}[x_j(t) - x_i(t)]dw(t), \quad i = 1, \dots, q, \\ x_i(0) &\stackrel{\text{a.s.}}{=} x_{i0}, \quad t \geq 0. \end{aligned} \quad (25.44)$$

In this case, (25.38) can be cast in the form of (25.35) with

$$f(x) = \begin{bmatrix} \sum_{j=1, j \neq 1}^q C_{(1,j)}[\sigma_{1j}(x_j) - \sigma_{j1}(x_1)] \\ \vdots \\ \sum_{j=1, j \neq q}^q C_{(q,j)}[\sigma_{qj}(x_j) - \sigma_{jq}(x_q)] \end{bmatrix}, \quad (25.45)$$

$$D(x) = \begin{bmatrix} \sum_{j=1, j \neq 1}^q \gamma C_{(1,j)}[x_j(t) - x_1(t)] \\ \vdots \\ \sum_{j=1, j \neq q}^q \gamma C_{(q,j)}[x_j(t) - x_q(t)] \end{bmatrix}, \quad (25.46)$$

where the stochastic term  $D(x)dw$  represents probabilistic variations in the charge transfer rate (i.e., generalized currents) between the agents. Furthermore, Assumption 25.8 now takes the following form.

**Assumption 25.9** For  $i, j = 1, \dots, q$ ,  $C_{(i,j)}(x_i - x_j)[\sigma_{ij}(x_j) - \sigma_{ji}(x_i)] \leq -(q-1)\gamma^2 C_{(i,j)}^2(x_i - x_j)^2$ .

**Theorem 25.4** ([58]) Consider the nonlinear stochastic multiagent system given by (25.44) and assume that Assumptions 25.7 and 25.9 hold. Then, for every  $\alpha \in \mathbb{R}$ ,  $\alpha \mathbf{e}_q$  is a stochastically semistable equilibrium state of (25.44). Furthermore,  $x(t) \xrightarrow{\text{a.s.}} \frac{1}{q} \mathbf{e}_q \mathbf{e}_q^T x(0)$  as  $t \rightarrow \infty$  and  $\frac{1}{q} \mathbf{e}_q \mathbf{e}_q^T x(0)$  is a stochastically semistable equilibrium state.

Since in many consensus control protocol applications it is desirable for the closed-loop dynamical system that exhibits semistability to also possess the property that the system trajectories that almost surely converge to a Lyapunov stable in probability system state do so in finite time rather than merely asymptotically, next we develop a thermodynamically motivated finite time consensus framework for multiagent nonlinear stochastic systems that achieve finite time stochastic semistability and almost sure state equipartition.

Specifically, in place of (25.37) we can consider nonlinear consensus protocols of the form

$$\begin{aligned} u_i(t) &= \sum_{j=1, j \neq i}^q C_{(i,j)}[\sigma_{ij}(x_j(t)) - \sigma_{ji}(x_i(t))] \\ &\quad + c \sum_{j=1, j \neq i}^q C_{(i,j)} \text{sign}(x_j(t) - x_i(t)) |x_j(t) - x_i(t)|^\theta, \end{aligned} \quad (25.47)$$

where  $c > 0$  is a design constant,  $0 < \theta < 1$ ,  $\text{sign}(y) \triangleq y/|y|$ ,  $y \neq 0$ , with  $\text{sign}(0) \triangleq 0$ , and  $\sigma_{ij}(\cdot)$ ,  $i, j \in \{1, \dots, q\}$ ,  $i \neq j$ , are as in (25.37). Note that in this case the closed-loop system (25.36) and (25.47) is given by

$$\begin{aligned} dx_i(t) &= \sum_{j=1, j \neq i}^q C_{(i,j)}[\sigma_{ij}(x_j(t)) - \sigma_{ji}(x_i(t))] dt \\ &\quad + c \sum_{j=1, j \neq i}^q C_{(i,j)} \text{sign}(x_j(t) - x_i(t)) |x_j(t) - x_i(t)|^\theta \\ &\quad + \sum_{j=1, j \neq i}^q \gamma C_{(i,j)}[x_j(t) - x_i(t)] dw_i(t), \quad i = 1, \dots, q, \\ &\quad x_i(0) \stackrel{\text{a.s.}}{=} x_{i0}, \quad t \geq 0. \end{aligned} \quad (25.48)$$

Note that with  $n = q$  and  $d = 1$ , (25.48) can be cast in the form of (25.35) with

$$f(x) = \begin{bmatrix} \sum_{j=1, j \neq 1}^q C_{(1,j)} [\sigma_{1j}(x_j) - \sigma_{j1}(x_1)] \\ + c \sum_{j=1, j \neq 1}^q C_{(1,j)} \text{sign}(x_j - x_1) |x_j - x_1|^\theta \\ \vdots \\ \sum_{j=1, j \neq q}^q C_{(q,j)} [\sigma_{qj}(x_j) - \sigma_{jq}(x_q)] \\ + c \sum_{j=1, j \neq q}^q C_{(q,j)} \text{sign}(x_j - x_q) |x_j - x_q|^\theta \end{bmatrix} \quad (25.49)$$

and  $D(x)$  defined as in (25.46). Furthermore, note that since

$$\mathbf{e}_q^T dx(t) = \mathbf{e}_q^T f(x(t)) dt + \mathbf{e}_q^T D(x(t)) dw(t) = 0, \quad x(0) \stackrel{\text{a.s.}}{=} x_0, \quad t \geq 0,$$

it follows that  $\sum_{i=1}^q dx_i(t) \stackrel{\text{a.s.}}{=} 0$ ,  $t \geq 0$ , which implies that the total system charge is conserved, and hence, the controlled network satisfies an underlying conservation law.

**Theorem 25.5** ([58]) Consider the nonlinear stochastic multiagent system given by (25.48) with  $c > 0$  and  $\theta \in (0, 1)$ , and assume that Assumptions 25.7 and 25.9 hold. Then, for every  $\alpha \in \mathbb{R}$ ,  $\alpha \mathbf{e}_q$  is a stochastically finite time semistable equilibrium state of (25.48). Moreover,  $x(t) = \frac{1}{q} \mathbf{e}_q \mathbf{e}_q^T x(0)$  for all  $t \geq T(x(0))$ , where

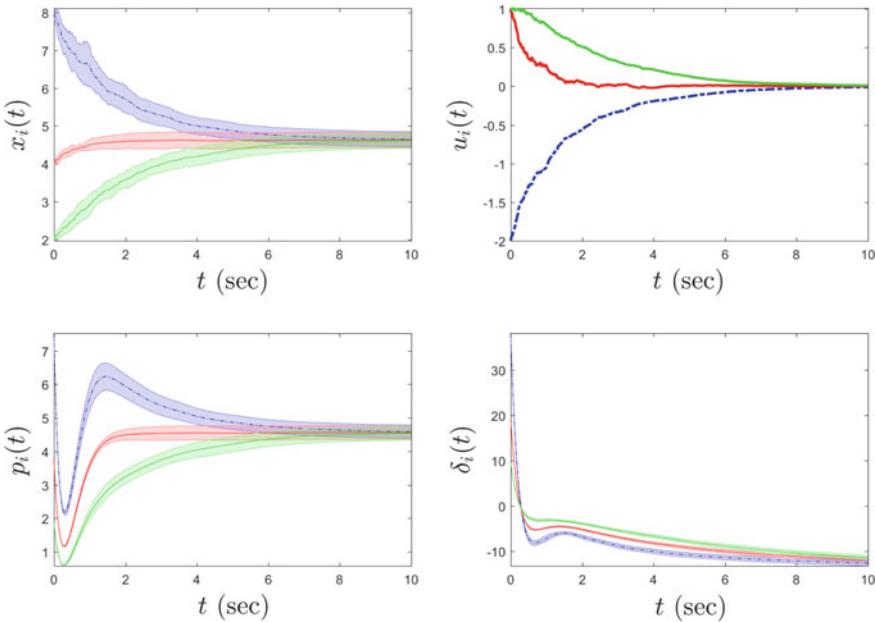
$$\mathbb{E}^{x_0}[T(x(0))] \leq \frac{4V(x_0)^{\frac{1-\theta}{2}}}{c(1-\theta)(4\lambda_2(L(C)))^{\frac{1+\theta}{2}}},$$

$\lambda_2(L(C))$  is the Fiedler eigenvalue of the graph Laplacian of  $\mathfrak{G}(C)$ , and

$$V(x_0) = \frac{1}{2} \left( x_0 - \frac{1}{q} \mathbf{e}_q \mathbf{e}_q^T x_0 \right)^T \left( x_0 - \frac{1}{q} \mathbf{e}_q \mathbf{e}_q^T x_0 \right).$$

Here, we once again use the set of aircrafts considered in Sect. 25.2 to demonstrate the proposed stochastic thermodynamic consensus framework for networks with communication uncertainty. However, instead of five aircrafts we consider three aircrafts with an uncertain triangular communication topology. For the higher level communication consensus controller design, we use (25.37) with  $\sigma_{ij}(x_j) = x_j$  and  $\sigma_{ji}(x_i) = x_i$  to generate  $x_i(t)$ ,  $t \geq 0$ , that has a direct effect on the lower level controller design to achieve pitch rate consensus. Figure 25.4 presents the sample trajectories along with the standard deviation of the states of each agent versus time for 10 sample paths for all initial conditions set to zero and  $x_1(0) \stackrel{\text{a.s.}}{=} 8$ ,  $x_2(0) \stackrel{\text{a.s.}}{=} 4$ , and  $x_3(0) \stackrel{\text{a.s.}}{=} 2$ . The mean control profile is also plotted in Fig. 25.4.

More general probabilistic variations for interagent communication as in (25.42) as well as unidirectional communication topologies can also be explored.



**Fig. 25.4** Sample average along with the sample standard deviation versus time for agent guidance state ( $x_i(t)$ ,  $t \geq 0$ ), guidance input ( $u_i(t)$ ,  $t \geq 0$ ), pitch rate ( $p_i(t)$ ,  $t \geq 0$ ), and elevator control ( $\delta_i(t)$ ,  $t \geq 0$ ) responses for the standard consensus protocol given by (25.37) with  $k_1 = 1$ . The control profile is plotted as the mean of the 10 sample runs

## 25.4 Systems Biology, Neurophysiology, Thermodynamics, and Dynamic Switching Communication Topologies for Large-Scale Multilayered Networks

To develop consensus control design protocols for nonlinear networks with dynamic switching topologies, we can look to the field of neuroscience [85–88] for inspiration. And in particular the study of the central nervous system that exhibits nearly discontinuous transitions between macroscopic states. One such example exhibiting this phenomenon is the induction of general anesthesia [89–92] resulting in a transition from consciousness to unconsciousness (i.e., a consensus state). In current clinical practice of general anesthesia, potent drugs are administered which profoundly influence levels of consciousness and vital respiratory (ventilation and oxygenation) and cardiovascular (heart rate, blood pressure, and cardiac output) functions. These variation patterns of the physiologic parameters (i.e., ventilation, oxygenation, heart rate variability, blood pressure, and cardiac output) and their alteration with levels of consciousness can provide scale-invariant fractal temporal structures to characterize the degree of consciousness in sedated patients [89].

The degree of consciousness reflects the adaptability of the central nervous system and is proportional to the maximum work output under a fully conscious state divided by the work output of a given anesthetized state in analogy to Carnot's theorem [56]. A reduction in maximum work output (and cerebral oxygen consumption) or elevation in the anesthetized work output (or cerebral oxygen consumption) will thus reduce the degree of consciousness. Hence, the fractal nature (i.e., complexity) of conscious variability is a self-organizing emergent property of the large-scale interconnected biological neuronal network since it enables the central nervous system to maximize entropy production and dissipate energy gradients.

Complex dynamical systems involving self-organizing components forming spatiotemporal evolving structures that exhibit a hierarchy of emergent system properties form the underpinning of the central nervous system. These complex dynamical systems are ubiquitous in nature and engineering systems and are not limited to the central nervous system. Such systems include, for example, biological systems, immune systems, ecological systems, quantum particle systems, chemical reaction systems, economic systems, cellular systems, modern network systems, and battle space management systems involving distributed command and control between heterogeneous air and ground assets, to cite but a few examples. The connection between the local subsystem interactions and the globally complex system behavior is often elusive. These systems are known as dissipative systems [56, 93, 94] and consume energy and matter while maintaining their stable structure by dissipating entropy to the environment.

In the central nervous system billions of neurons interact to form self-organizing dissipative nonequilibrium structures [56, 61, 93, 94]. As noted in introduction, the fundamental common phenomenon among these systems is that they evolve in accordance with the laws of (nonequilibrium) thermodynamics, which are among the most firmly established laws of nature. Dynamical thermodynamics, in the sense of [56], involves open interconnected dynamical systems that exchange matter and energy with their environment in accordance with the first law (conservation of energy) and the second law (nonconservation of entropy) of thermodynamics. Self-organization can spontaneously occur in such systems by invoking the two fundamental axioms of the science of heat.

Namely, (*i*) if the energies in the connected subsystems of an interconnected system are equal, then energy exchange between these subsystems is not possible, and (*ii*) energy flows from more energetic subsystems to less energetic subsystems. These axioms establish the existence of a system entropy function as well as lead to equipartition of energy [56, 61, 95, 96] in system thermodynamics, synchronization [97] in biological neuronal networks, and information consensus [60] in cooperative networks; an emergent behavior in thermodynamic systems as well as neuroscience and swarm dynamics. Hence, in complex interconnected dynamical systems, self-organization is not a property of the systems' parts but rather emerges as a result of the nonlinear subsystem interactions.

In recent research [56, 61, 95, 96, 98–100], we combined the two universalisms of thermodynamics and dynamical systems theory under a single umbrella to develop a dynamical system formalism for classical thermodynamics so as to harmonize it

with classical mechanics. In particular, our dynamical system formalism captures all of the key aspects of thermodynamics, including its fundamental laws, while providing a mathematically rigorous formulation for thermodynamical systems out of equilibrium by unifying the theory of heat transfer with that of classical thermodynamics. In addition, the concept of entropy for a nonequilibrium state of a dynamical process is defined, and its global existence and uniqueness is established. This state space formalism of thermodynamics shows that the behavior of heat, as described by the conservation equations of thermal transport and as described by classical thermodynamics, can be derived from the same basic principles and is part of the same scientific discipline. Connections between irreversibility, the second law of thermodynamics, and the entropic arrow of time are also established in [56, 61, 96, 98].

Building on the results in [56, 61, 95, 96, 98, 99], dynamical thermodynamics can be merged with neuroscience to develop a neuroinspired control design framework for network consensus using control architectures predicated on the network control properties of the brain by rigorously emulating the large-scale interconnected biological neuronal network control model given in [89–92]. Specifically, as in thermodynamics, neuroscience is a theory of large-scale systems wherein graph theory [101] can be used in capturing the (possibly dynamic) connectivity properties of network interconnections, with neurons represented by nodes, synapses represented by edges or arcs, and synaptic efficacy captured by edge weighting giving rise to a weighted adjacency matrix governing the underlying directed dynamic graph network topology. However, unlike thermodynamics, wherein heat (i.e., energy in transition) spontaneously flows from a state of higher temperature to a state of lower temperature, neuron membrane potential variations occur due to ion species exchanges, which evolve from regions of higher chemical potentials to regions of lower chemical potentials (i.e., Gibbs' chemical potential [56, 99]). And this evolution does not occur spontaneously but rather requires a hierarchical continuous-discrete (i.e., hybrid) control architecture for the opening and closing of specific gates within specific ion channels.

Merging the dynamical neuroscience framework developed in [89–92] with dynamical thermodynamics [56, 61, 96, 99] by embedding thermodynamic state notions (i.e., entropy, energy, free energy, chemical potential, etc.) within our dynamical systems framework can allow us to directly exploit the otherwise mathematically complex and computationally prohibitive large-scale dynamical system model given in [89–92] for designing consensus and synchronization control protocols for nonlinear multilayered network systems. In particular, a thermodynamically consistent neuroinspired controller architecture would emulate the clinically observed self-organizing, spatiotemporal fractal structures that optimally dissipate energy and optimize entropy production in thalamocortical circuits of fully conscious patients. This thermodynamically consistent neuroinspired control framework can provide the necessary tools involving multistability, synaptic drive equipartitioning (i.e., synchronization or consensus across time scales), energy dispersal, and entropy production for connecting network consensus for large-scale network dynamical systems to synchronization (i.e., agreement over time instants) in thalamocortical circuits in the necortex. This is a subject of current research.

## 25.5 Nonlinear Stochastic Optimal Control and Learning

Since multiagent systems can involve information laws governed by nodal dynamics and rerouting strategies that can be modified to minimize waiting times and optimize system throughput, optimality considerations in network systems is of paramount importance. In view of the advantages of nonlinear optimal controllers over linear controllers [29, 102], it is not surprising that a significant effort has been devoted to developing a theory of optimal nonlinear regulation [103–137]. For problems involving nonquadratic cost functionals on the infinite interval, asymptotic stability is guaranteed by means of a Lyapunov function for the closed-loop system. This Lyapunov function is given as the solution to a steady-state form of the Hamilton–Jacobi–Bellman equation [29, 102].

We propose to reverse the situation somewhat by fixing the structure of the Lyapunov function, cost functional, and feedback law prior to optimization. In this case, the structure of the Lyapunov function can be viewed as providing the *framework* for controller synthesis by guaranteeing local or global semistability or asymptotic stability for a class of feedback controllers. The *actual* controller chosen for implementation can thus be the member of this candidate class that minimizes the given performance functional. In LQG theory, for example, the Lyapunov function is the familiar quadratic functional  $V(x) = x^T P x$ , while the gains for the linear feedback control are chosen to minimize a quadratic performance functional. Thus, Lyapunov function theory provides the framework, whereas optimization fixes the gains.

To elucidate the basis for such an approach for nonlinear stochastic systems, consider the problem of evaluating a nonlinear-nonquadratic cost functional depending upon a controlled nonlinear stochastic differential equation. It can be shown that the cost functional can be evaluated in closed form so long as the cost functional is related in a specific way to an underlying Lyapunov function [138]. In accordance with practical motivations, we restrict our attention to time-invariant systems on the infinite horizon. Furthermore, for simplicity of exposition, we shall define all functions globally and assume that existence and uniqueness properties of the given stochastic differential equations are satisfied in forward time. For details, see [138].

Let  $F : \mathbb{R}^n \times \mathbb{R}^m \rightarrow \mathbb{R}^n$ ,  $D : \mathbb{R}^n \times \mathbb{R}^m \rightarrow \mathbb{R}^{n \times d}$ , assume that  $F(0, 0) = 0$ ,  $D(0, 0) = 0$ , and consider the nonlinear controlled stochastic dynamical system

$$dx(t) = F(x(t), u(t))dt + D(x(t), u(t))dw(t), \quad x(0) \stackrel{\text{a.s.}}{=} x_0, \quad t \geq 0, \quad (25.50)$$

where  $w(t)$  is a  $d$ -dimensional independent standard Wiener process. The control  $u(\cdot)$  in (25.50) is nonanticipative and is restricted to the class of admissible controls consisting of measurable functions  $u(\cdot)$  such that  $u(t) \in \mathcal{H}_m^U$ ,  $t \geq 0$ , where  $U \subseteq \mathbb{R}^m$  is given. Next, let  $L : \mathbb{R}^n \times \mathbb{R}^m \rightarrow \mathbb{R}$ , let  $V'(x)$  and  $V''(x)$  denote the Fréchet derivative and the Hessian of  $V$  at  $x$ , respectively, and, for  $p \in \mathbb{R}^n$  and  $Q \in \mathbb{R}^{n \times n}$ , define

$$H(x, p, Q, u) \triangleq L(x, u) + p^T F(x, u) + \frac{1}{2} \text{tr } D^T(x, u) Q D(x, u). \quad (25.51)$$

Furthermore, define the set of stochastic regulation controllers given by

$$\begin{aligned} \mathcal{S}(x_0) \triangleq \{u(\cdot) : u(\cdot) \text{ is admissible and } x(\cdot) \text{ given by (49)} \\ \text{satisfies } x(t) \xrightarrow{\text{a.s.}} 0 \text{ as } t \rightarrow \infty\}. \end{aligned}$$

**Theorem 25.6** ([138]) Consider the nonlinear controlled stochastic dynamical system (25.50) with performance functional

$$J(x_0, u(\cdot)) \triangleq \mathbb{E}^{x_0} \left[ \int_0^\infty L(x(t), u(t)) dt \right], \quad (25.52)$$

where  $u(\cdot)$  is an admissible control. Assume that there exists a two-times continuously differentiable radially unbounded function  $V : \mathbb{R}^n \rightarrow \mathbb{R}$  and a control law  $\phi : \mathbb{R}^n \rightarrow U$  such that

$$V(0) = 0, \quad V(x) > 0, \quad \phi(0) = 0, \quad x \in \mathbb{R}^n, \quad x \neq 0, \quad (25.53)$$

$$V'(x)F(x, \phi(x)) + \frac{1}{2}\text{tr } D^T(x, \phi(x))V''(x)D(x, \phi(x)) < 0, \quad x \in \mathbb{R}^n, \quad x \neq 0, \quad (25.54)$$

$$H(x, V'^T(x), V''(x), \phi(x)) = 0, \quad x \in \mathbb{R}^n, \quad (25.55)$$

$$H(x, V'^T(x), V''(x), u) \geq 0, \quad x \in \mathbb{R}^n, \quad u \in U. \quad (25.56)$$

Then, with the feedback control  $u(\cdot) = \phi(x(\cdot))$ , the zero solution  $x(t) \xrightarrow{\text{a.s.}} 0$  of the closed-loop system (25.50) is globally asymptotically stable in probability and  $J(x_0, \phi(x(\cdot))) = V(x_0)$ ,  $x_0 \in \mathbb{R}^n$ . Furthermore, the feedback control  $u(\cdot) = \phi(x(\cdot))$  minimizes  $J(x_0, u(\cdot))$  in the sense that

$$J(x_0, \phi(x(\cdot))) = \min_{u(\cdot) \in \mathcal{S}(x_0)} J(x_0, u(\cdot)). \quad (25.57)$$

In spite of the appealing nature of the stochastic Hamilton–Jacobi–Bellman theory, its current state of development entails several limitations in addressing real-world problems. In particular, these include (i) the ability to design static and dynamic output-feedback compensators and (ii) the development of robust controllers for uncertain plants. Hence, a principal goal is to extend the current state of the theory by removing these limitations.

For the class of network control problems that we are addressing there are typically only a small number of measurements available. An indirect solution to this problem is to implement an estimator to reconstruct the state from the available measurements. However, besides being a difficult problem itself, there is no reason to

expect that certainty equivalence (separation) will hold in the presence of nonlinearities. Consequently, we must consider the problem of designing output-feedback laws that are pre-constrained to operate solely upon the available measurements. Such controllers may be either static (proportional) or dynamic.

To achieve this goal it appears natural to develop a *fixed-structure* Hamilton–Jacobi–Bellman theory in which one can prespecify the structure of the feedback law with respect to, for example, the order of nonlinearities appearing in the dynamic compensator. The actual gain maps can then be determined by solving algebraic relations in much the same way full-state feedback controllers can be obtained.

By assuming polynomial forms for the controlled drift and diffusion terms  $F(x, u)$  and  $D(x, u)$ , and considering polynomial or multilinear parameterizations of the Lyapunov function  $V(x)$  and the control policy  $\phi(x)$ , the optimal stochastic stability criterion can be casted as in (25.54). In addition to the development of the fixed-structure synthesis for optimal stabilizing control under full-state feedback, this framework can be used to address the problem of static and dynamic output-feedback control [139].

The notions of asymptotic and exponential stability in dynamical systems theory imply convergence of the system trajectories to an equilibrium state over the infinite horizon. In many applications, however, it is desirable that a dynamical system possesses the property that trajectories that converge to a Lyapunov stable equilibrium state must do so in finite time rather than merely asymptotically. Most of the existing control techniques in the literature ensure that the closed-loop system dynamics of a controlled system are Lipschitz continuous, which implies uniqueness of system solutions in forward and backward times. Hence, convergence to an equilibrium state is achieved over an infinite time interval.

In order to achieve convergence in finite time for deterministic dynamical systems, the closed-loop system dynamics need to be non-Lipschitzian giving rise to non-uniqueness of solutions in backward time. Uniqueness of solutions in forward time, however, can be preserved in the case of finite time convergence. Sufficient conditions that ensure uniqueness of solutions in forward time in the absence of Lipschitz continuity for deterministic dynamical systems are given in [140–143], whereas [144, 145] give sufficient conditions that ensure uniqueness of solutions for stochastic dynamical systems in forward time in the absence of a uniform Lipschitz continuity and a growth restriction condition on the system drift and diffusion functions.

For deterministic dynamical system, finite time convergence to a Lyapunov stable equilibrium, that is, *finite time stability*, was rigorously studied in [146, 147] using Hölder continuous Lyapunov functions. Finite time stabilization of second-order systems was considered in [148, 149]. More recently, researchers have considered finite time stabilization of higher order systems [150] as well as finite time stabilization using output feedback [151]. Design of globally strongly stabilizing continuous controllers for nonlinear systems using the theory of homogeneous systems was studied in [152]. Alternatively, discontinuous finite time stabilizing feedback controllers have also been developed in the literature [153–155]. However, for practical implementations, discontinuous feedback controllers can lead to chattering due to system

uncertainty or measurement noise, and hence, may excite unmodeled high-frequency system dynamics.

The problem of finite time *optimal* control has received very little attention in the literature. When the performance functional (25.52) involves terms of order  $x^p$ , where  $p < 2$ , then we call the cost criterion *subquadratic*. Subquadratic cost criteria pay close attention to the behavior of the state near the origin since, for example  $x^{\frac{1}{2}} >> x^2$  for  $x$  in a neighborhood of zero. Our interest in subquadratic cost criteria stems from the fact that optimal controllers for such criteria are sublinear, and thus, exhibit finite settling-time behavior. This phenomenon was studied in [128, 148] and applied to spacecraft control in [156].

The aforementioned results yield finite interval controllers even though the original cost criterion is defined on the infinite horizon. Hence, one advantage of this approach for certain applications is to obtain finite interval controllers without the computational complexities of two-point boundary value problems. We also note that if the order of the subquadratic state terms appearing in the cost functional is sufficiently small, then the controllers actually optimize a minimum-time cost criterion. Currently, such results are only obtainable using the maximum principle, which generally does not yield feedback controllers.

The nonlinear control framework presented in this section can be readily extendable to address *optimal stochastic finite time stabilization* using a feedback control architecture by requiring (25.54) to satisfy a fractional Lyapunov differential inequality [157, 158]. To state this extension the following definition is required.

**Definition 25.6** ([158]) The zero solution  $x(t) \stackrel{\text{a.s.}}{\equiv} 0$  to (25.35) is (*globally*) *stochastically finite time stable* if there exists an operator  $T : \mathcal{H}_n \rightarrow \mathcal{H}_1^{[0, \infty)}$ , called the *stochastic settling-time operator*, such that the following statements hold:

- (i) *Finite time convergence in probability.* For every  $x(0) \in \mathcal{H}_n$ ,  $s^{x(0)}(t)$  is defined on  $[0, T(x(0)))$ ,  $s^{x(0)}(t) \in \mathcal{H}_n$  for all  $t \in [0, T(x(0)))$ , and

$$\mathbb{P}^{x_0} \left( \lim_{t \rightarrow T(x(0))} \|s^{x(0)}(t)\| = 0 \right) = 1.$$

- (ii) *Lyapunov stability in probability.* For every  $\varepsilon > 0$ ,

$$\mathbb{P}^{x_0} \left( \sup_{t \in [0, T(x(0)))} \|s^{x(0)}(t)\| > \varepsilon \right) = 0.$$

Equivalently, for every  $\varepsilon > 0$  and  $\rho \in (0, 1)$ , there exist  $\delta = \delta(\varepsilon, \rho) > 0$  such that, for all  $x_0 \in \mathcal{B}_\delta(0)$ ,  $\mathbb{P}^{x_0} \left( \sup_{t \in [0, T(x(0)))} \|s^{x(0)}(t)\| > \varepsilon \right) \leq \rho$ .

- (iii) *Finiteness of the stochastic settling-time operator.* For every  $x \in \mathcal{H}_n$  the stochastic settling-time operator  $T(x)$  exists and is finite with probability one, that is,  $\mathbb{E}^x [T(x)] < \infty$ .

**Theorem 25.7** ([158]) Consider the nonlinear stochastic dynamical system  $\mathcal{G}$  given by (25.35) with  $\mathcal{D} = \mathbb{R}^n$ . If there exist a radially unbounded positive definite function  $V : \mathbb{R}^n \rightarrow \mathbb{R}_+$  and a function  $\eta : \overline{\mathbb{R}}_+ \rightarrow \overline{\mathbb{R}}_+$  such that  $V(0) = 0$ ,  $V(x)$  is two-times continuously differentiable for all  $x \in \mathbb{R}^n$ ,  $\eta(\cdot)$  is continuously differentiable, and, for all  $x \in \mathbb{R}^n$ ,

$$V'(x)f(x) + \frac{1}{2}\text{tr } D^T(x)V''(x)D(x) \leq -\eta(V(x)), \quad (25.58)$$

$$\int_0^\varepsilon \frac{dv}{\eta(v)} < \infty, \quad \varepsilon \in [0, \infty), \quad (25.59)$$

$$\eta'(v) > 0, \quad v \geq 0, \quad (25.60)$$

then  $\mathcal{G}$  is globally stochastically finite time stable. Moreover, there exists a settling-time operator  $T : \mathcal{H}_n \rightarrow \mathcal{H}_1^{[0, \infty)}$  such that

$$\mathbb{E}^{x_0}[T(x_0)] \leq \int_0^{V(x_0)} \frac{dv}{\eta(v)}, \quad x_0 \in \mathbb{R}^n. \quad (25.61)$$

Note that if  $\eta(V) = cV^\theta$ , where  $c > 0$  and  $\theta \in (0, 1)$ , then  $\eta(\cdot)$  satisfies (25.59) and (25.60). In this case, (25.61) becomes

$$\mathbb{E}^{x_0}[T(x(0))] \leq \frac{V(x_0)^{1-\theta}}{c(1-\theta)}.$$

For deterministic dynamical systems, this specialization recovers the finite time stability result given in [146].

Using the control synthesis version (rather than the analysis version) of Theorem 25.7, a unified framework for finite time stochastic stabilization was developed in [158]. Furthermore, building on the results of [159–161] as well as on the recent stochastic finite time stabilization framework of [158] we developed a constructive universal feedback control law for stochastic finite time stabilization of stochastic dynamical systems [162]. In addition, we presented necessary and sufficient conditions for continuity of such controllers. Furthermore, we showed that for every nonlinear stochastic dynamical system for which a stochastic control Lyapunov function can be constructed there exists an inverse optimal feedback control law in the sense of [138] with guaranteed sector and gain margins of  $(\frac{1}{2}, \infty)$ .

Moreover, using the newly developed notion of stochastic dissipativity [59], in [163] we derive a return difference inequality to provide connections between stochastic dissipativity and optimality of nonlinear controllers for stochastic dynamical systems. In particular, analogous to deterministic controlled dynamical systems,

using extended Kalman–Yakubovich–Popov conditions characterizing stochastic dissipativity [59] we show that our optimal feedback control law satisfies a return difference inequality predicated on the infinitesimal generator of a controlled Markov diffusion process if and only if the controller is stochastically dissipative with respect to a specific quadratic supply rate. This allows us to develop connections between stochastic dissipativity and stochastic optimal control to address robust stability and robust stabilization problems involving both stochastic and deterministic uncertainty as well as both averaged and worst-case performance criteria.

A key extension to optimal stochastic Hamilton–Jacobi–Bellman theory is the design of *semistabilizing* optimal controllers involving controlled stochastic dynamical systems with a continuum of equilibria. In [164], we developed  $\mathcal{H}_2$  optimal semistable control theory for *linear* dynamical systems necessary for addressing *optimal consensus control* problems for multiagent systems. Unlike the standard  $\mathcal{H}_2$  optimal control problem, it is shown in [164, 165] that a complicating factor of the  $\mathcal{H}_2$  optimal semistable stabilization problem is that the closed-loop Lyapunov equation guaranteeing semistability can admit multiple solutions. In addition, we showed that the  $\mathcal{H}_2$  optimal solution is given by a least squares solution to the closed-loop Lyapunov equation over all possible semistabilizing solutions. Moreover, it is shown that this least squares solution can be characterized by a linear matrix inequality minimization problem. Building on the results of this section we can address the problem of finding a state feedback *nonlinear* control law  $u = \phi(x)$  that minimizes the performance measure (25.52) and guarantees stochastic semistability of *nonlinear* dynamical systems of the form (25.50).

A key limitation of the solution procedures for most optimal control problems is that they require full knowledge of the system dynamics. To this end, adaptive control along with robust control theory has been developed to address the problem of system uncertainty in control system design. In contrast to fixed-gain robust controllers, which are predicated on a mathematical model of the system uncertainty, and which maintain specified constants within the feedback control law to *sustain* robust stability and performance over the range of system uncertainty, adaptive controllers directly or indirectly adjust feedback gains to maintain closed-loop system stability and *improve* system performance in the face of system uncertainties. However, unlike most robust controllers, adaptive controllers are not usually optimal with respect to a prespecified performance criterion.

Online adaptive learning control algorithms have recently been developed for optimal control problems using measurements along the dynamical system trajectories. These control algorithms build on adaptive dynamic programming and neurodynamic programming methods [166, 167] and are known as *reinforcement learning* [168]. Reinforcement learning controllers are based on adaptive critic or actor–critic structures that involve simultaneous tuning of the actor–critic parameters that can learn optimal control solutions by (approximately) solving the Hamilton–Jacobi–Bellman equation in real time without requiring the full knowledge of the system dynamics. Thus, reinforcement learning-based approaches allow for the development of control algorithms that can be used in real time to affect optimal and adaptive control in the presence of system uncertainty [169, 170].

Reinforcement learning methods have been developed by observing learning mechanisms of living organisms in nature [171]. Most living things in nature are chronognostic and adopt their behavior to a dynamic (i.e., changing) environment. Thus, the behavior has a temporal component, wherein all living organisms perceive their future relative to the present. Since behavioral anticipation is vital for survival, every decision-making organism interacts with its environment and adapts its actions based on the learned responses that it receives from its actions.

In controlled dynamical systems, reinforcement learning refers to a method in which the controller can learn the unknown and (possibly) time-varying dynamics of the system by interacting with the system and its environment [169, 171]. The actor-critic structure of the algorithm evaluates the current control policy and iteratively updates it to meet a given performance measure. The control policy update is carried out by observing the response of the system predicated on the current control policy. Therefore, the reinforcement learning-based controller can learn optimal actions in the presence of system parameter uncertainty and in the absence of the complete knowledge of the system dynamics.

A direct extension of the optimal control framework outlined in this section would involve a stochastic optimal control framework based on the development of algorithms for general models of stochasticity and multi-trajectory optimization and reinforcement learning. In particular, many large-scale multilayered network systems with communication uncertainty, as well as biological, financial, and engineered processes, can be represented stochastically involving distributions such as Levy distributions and jump diffusion processes. The optimal control method presented in this section can be readily extended to address these stochastic models by generalizing work on risk sensitivity and logarithmic transformations for the case of Markov-jump diffusion processes. Whereas standard reinforcement learning methods utilize trajectory optimizers based on single trajectory optimization, the proposed extension would allow for the development of algorithms for multi-trajectory optimization and control of the full probability density of the dynamical system's states by exploring generalizations of the Hamilton–Jacobi–Bellman equation to optimal control of infinite-dimensional systems. In addition, using the framework outlined in Sects. 25.2 and 25.3, system thermodynamic notions can be used to derive symmetric Fourier type distributed control law protocols predicated on online policy iteration algorithms that achieve information consensus in the face of uncertain network communication topologies.

## 25.6 Complexity, Thermodynamics, Information Theory, and Swarm Dynamics

To enable the autonomous operation for large-scale multiagent network systems, the development of functional algorithms for agent coordination, control, and learning is needed. In particular, control algorithms need to address agent interactions, coop-

erative and non-cooperative control, task assignments, and resource allocations. To realize these tasks, appropriate sensory and cognitive capabilities such as adaptation, learning, decision-making, and agreement (or consensus) on the agent and multiagent levels are required. The common approach for addressing the autonomous operation of multiagent systems is using distributed control algorithms involving neighbor-to-neighbor interaction between agents wherein agents update their information state based on the information states of the neighboring agents.

Since most multiagent network systems are highly interconnected and mutually interdependent, both physically and through a multitude of information and communication networks, these systems are characterized by high-dimensional, large-scale interconnected dynamical systems. To develop distributed methods for control and coordination of autonomous multiagent systems, many researchers have looked to autonomous *swarm* systems appearing in nature for inspiration [1, 2, 6, 52–54].

Biology has shown that many species of animals such as insect swarms, ungulate flocks, fish schools, ant colonies, and bacterial colonies *self-organize* in nature [172–175]. These biological aggregations give rise to remarkably complex global behaviors from simple local interactions between large numbers of relatively unintelligent agents without the need for centralized control. The spontaneous development (i.e., self-organization) of these autonomous biological systems and their spatiotemporal evolution to more complex states often appears without any external system interaction. In other words, structure morphing into coherent groups is internal to the system and results from local interactions among subsystem components that are independent of the physical nature of the individual components. These local interactions often comprise a simple set of rules that lead to remarkably complex global behaviors. *Complexity* here refers to the quality of a system wherein interacting subsystems self-organize to form hierarchical evolving structures exhibiting *emergent* system properties. In addition, the spatially distributed sensing and actuation control architecture prevalent in such systems is inherently *robust* to individual subsystem (or agent) failures and unplanned behavior at the individual subsystem (or agent) level.

The connection between the local subsystem interactions and the globally complex system behavior is often elusive. Complex dynamical systems involving self-organizing components forming spatiotemporally evolving structures that exhibit a hierarchy of emergent system properties are not limited to biological aggregation systems. Such systems include, for example, nervous systems, immune systems, ecological systems, quantum particle systems, chemical reaction systems, economic systems, cellular systems, and galaxies, to cite but a few examples. As discussed in Sect. 25.4, these systems are known as *dissipative systems* [61, 176] and consume energy and matter while maintaining their stable structure by dissipating entropy to the environment. For example, as in neurophysiology involving billions of interacting neurons, in the physical universe billions of stars and galaxies interact to form self-organizing dissipative nonequilibrium structures [176]. The fundamental common phenomenon among these systems are that they evolve in accordance with the laws of (nonequilibrium) thermodynamics which are among the most firmly established laws of nature.

As discussed in Sect. 25.3, dynamical thermodynamics, in the sense of [56], involves open interconnected dynamical systems that exchange matter and energy with their environment in accordance with the first law (conservation of energy) and the second law (nonconservation of entropy) of thermodynamics. Self-organization can spontaneously occur in such systems by invoking the two fundamental axioms of the science of heat. Namely, (i) if the energies in the connected subsystems of an interconnected system are equal, then energy exchange between these subsystems is not possible, and (ii) energy flows from more energetic subsystems to less energetic subsystems. These axioms establish the existence of a system entropy function as well as *equipartition of energy* [56] in system thermodynamics and *information consensus* [61] in cooperative networks; an *emergent* behavior in thermodynamic systems as well as swarm dynamics. Hence, in complex interconnected dynamical systems, self-organization is not a property of the system's parts but rather emerges as a result of the nonlinear subsystem interactions.

In light of the above discussion, engineering swarm systems necessitates the development of relatively simple autonomous agents that are inherently distributed, self-organized, and truly scalable. Scalability follows from the fact that such systems do not involve centralized control and communication architectures. In addition, engineered swarming systems should be inherently robust to individual agent failures, unplanned task assignment changes, and environmental changes. Mathematical models for large-scale swarms can involve Lagrangian and Eulerian models. In a Lagrangian model, each agent is modeled as a particle governed by a difference or differential equation, whereas an Eulerian model describes the local energy or information flux for a distribution of swarms with an advection–diffusion (conservation) equation. The two formulations can be connected by a Fokker–Plank approximation relating jump distance distributions of individual agents to terms in the advection–diffusion equation [56, 175].

In many applications involving multiagent systems, groups of agents are required to agree on certain quantities of interest. In particular, it is important to develop information consensus protocols for networks of dynamic agents wherein, as discussed in Sect. 25.2, a unique feature of the closed-loop dynamics under any control algorithm that achieves consensus is the existence of a continuum of equilibria representing a state of equipartitioning or *consensus*. Under such dynamics, the limiting consensus state achieved is not determined completely by the dynamics, but depends on the initial system state as well. For such systems possessing a continuum of equilibria, it was argued in Sects. 25.2 and 25.3 that semistability [27, 177], and not asymptotic stability, is the relevant notion of stability.

Dynamical thermodynamics [56] can be used to develop distributed boundary control algorithms for addressing the consensus problem for an Eulerian swarm model [95]. The distributed boundary controller architectures are predicated on the recently developed notion of continuum dynamical thermodynamics [56] resulting in controller architectures involving the exchange of information between uniformly distributed swarms over an  $n$ -dimensional (not necessarily Euclidian) space that guarantees that the closed-loop system is consistent with basic thermodynamic principles. For such a thermodynamically consistent model the existence of a unique continu-

ously differentiable entropy functional for all equilibrium and nonequilibrium states of our system can also be established. Information consensus and semistability follow using the well-known Sobolev embedding theorems and the notion of generalized (or weak) solutions. Since the closed-loop system can be designed in such a way to satisfy basic thermodynamic principles, robustness to individual agent failures and unplanned individual agent behavior is automatically guaranteed. For details, see [56, 95].

## 25.7 Thermodynamic Entropy, Shannon Entropy, Bode Integrals, and Performance Limitations in Nonlinear Systems

Thermodynamics grew out of steam tables and the desire to design and build efficient heat engines, with its central problem involving hard limits on the efficiency of heat engines. Using the laws of thermodynamics, which are among the most firmly established laws of nature, Carnot's principle states that it is impossible to perform a repeatable cycle in which the only result is the performance of positive work [56]. In particular, Carnot showed that the *efficiency* of a reversible cycle—that is, the ratio of the total work produced during the cycle and the amount of heat transferred from a boiler to a cooler—is bounded by a universal maximum, and this maximum is only a function of the temperatures of the boiler and the cooler. In other words, Carnot's principle shows that for any cyclic process that is shielded from heat exchange with its environment, it is impossible to extract work from heat without at the same time discarding some heat, giving rise to an increasing quantity known as (thermodynamic) *entropy*. From a system-theoretic point of view, entropy production places hard limits on system (heat engine) performance.

Whereas thermodynamic entropy captures a measure of the amount of wasted energy in a dynamical (energy) transformation from one state (form) to another, *Shannon entropy* captures a measure of information contained in a random signal with prescribed distribution. In particular, Shannon's source-coding theorem [178] states that for a randomly generated data signal there exists a simple coding procedure, wherein the average code length for the data of length  $l$  is approximately  $l$  times the entropy of the signal source. In other words, Shannon entropy places hard limits of information performance that can be reliably retrieved from a compressed memoryless signal source.

Fundamental limits of achievable performance in linear feedback control systems were first investigated by Bode [179]. Specifically, Bode's integral theorem states that for a single-input, single-output stable system transfer function with a stable loop-gain and relative degree greater than or equal to two, the integral over all frequencies of the natural logarithm of the magnitude of the sensitivity transfer function  $S(s)$  vanishes, that is,

$$\int_0^\infty \log_e |S(j\omega)| d\omega = 0. \quad (25.62)$$

This result shows that it is not possible to decrease  $|S(j\omega)|$  below the value of 1 over all frequencies imposing fundamental limitations on achievable tracking, disturbance rejection, and robust performance for the closed-loop system. Bode's integral limitation theorem has been extended to multi-input, multi-output unstable systems [180]. In particular, the authors in [180] show that the integral over all frequencies of the natural logarithm of the magnitude of the determinant of the sensitivity transfer function is proportional to the sum of the unstable loop-gain poles, that is,

$$\int_0^\infty \log_e |\det S(j\omega)| d\omega = \pi \sum_{i=1}^{n_u} \operatorname{Re} p_i > 0, \quad (25.63)$$

where  $p_i$ ,  $i = 1, \dots, n_u$ , denotes the  $i$ th unstable loop-gain pole. The unstable poles in the right-hand side of (25.63) worsens the achievable tracking, disturbance rejection, and robust performance for the closed-loop system. Nonlinear extensions of Bode's integral theorem based on an information-theoretic interpretation, singular control, and Markov chains appear in [181–183].

Merging dynamical thermodynamics [56], information theory [184], and nonlinear dynamical systems theory, a unified nonlinear stabilization framework with a priori achievable system performance guarantees can be developed. The fact that classical thermodynamics is a physical theory concerning systems in equilibrium, information theory resorts to statistical (subjective or informational) probabilities, and control theory is based on a dynamical systems theory made it all but impossible to unify these theories, leaving these disciplines to stand in sharp contrast to one another in the seven decades of their coexistence. Yet all three theories involve fundamental limitations of performance giving rise to system entropy notions.

Using the dynamical systems framework for nonequilibrium thermodynamics recently developed in [56], we can harmoniously amalgamate thermodynamics, information theory, and control theory under a single umbrella for quantifying limits of performance for nonlinear system stabilization. The starting point of such a unified framework is to place information theory on a state space footing using graph-theoretic notions [56]. As in the case of thermodynamic entropy for systems out of equilibrium [56], this will allow us to develop an analytical description of an objective property of information entropy that can potentially offer a conceptual advantage over the subjective or informational expressions for information entropy proposed in the literature (e.g., Shannon entropy, von Neumann entropy, Kolmogorov–Sinai entropy). This can potentially allow us to quantify fundamental limitations for robustness and disturbance rejection of nonlinear feedback systems with finite capacity input–output signal communication rates.

To highlight some of the features of the aforementioned ideas and elucidate how thermodynamic entropy concepts can be merged with dynamical systems theory,

consider the nonlinear dynamical system  $\mathcal{G}$  given by

$$\dot{x}(t) = f(x(t)) + G(x(t))u(t), \quad x(0) = x_0, \quad t \geq 0, \quad (25.64)$$

$$y(t) = h(x(t)) + J(x(t))u(t), \quad (25.65)$$

where, for every  $t \geq 0$ ,  $x(t) \in \mathcal{D} \subseteq \mathbb{R}^n$  denotes the state vector,  $u(t) \in U \subseteq \mathbb{R}^m$  denotes the control input,  $y(t) \in Y \subseteq \mathbb{R}^l$  denotes the system output, and  $f : \mathcal{D} \rightarrow \mathbb{R}^n$ ,  $G : \mathcal{D} \rightarrow \mathbb{R}^{n \times m}$ ,  $h : \mathcal{D} \rightarrow \mathbb{R}^l$ , and  $J : \mathcal{D} \rightarrow \mathbb{R}^{l \times m}$ . For the dynamical system  $\mathcal{G}$  given by (25.64) and (25.65) defined on the state space  $\mathcal{D} \subseteq \mathbb{R}^n$ ,  $\mathcal{U}$  and  $\mathcal{Y}$  define input and output spaces, respectively, consisting of continuous bounded  $U$ -valued and  $Y$ -valued functions on the semi-infinite interval  $[0, \infty)$ . The spaces  $\mathcal{U}$  and  $\mathcal{Y}$  are assumed to be closed under the shift operator. Here, for simplicity of exposition, the mappings  $f(\cdot)$ ,  $G(\cdot)$ ,  $h(\cdot)$ , and  $J(\cdot)$  are assumed to be continuously differentiable and  $f(\cdot)$  has at least one equilibrium point  $x_e \in \mathcal{D}$  so that  $f(x_e) + G(x_e)u_e = 0$  and  $y_e = h(x_e) + J(x_e)u_e$  for some  $u_e \in U$ . Finally, we assume that  $\mathcal{G}$  is completely reachable [29].

A thermodynamic feedback control framework can be developed based on the notion of the thermodynamic entropy developed in [56]. To elucidate such a framework, the following definition of *feedback dissipativity* is needed. Feedback dissipative systems define a class of dynamical systems for which a continuously differentiable feedback transformation exists that renders the system  $\mathcal{G}$  dissipative and is a *generalization* of the *feedback passivation* notion introduced in [185].

**Definition 25.7** The nonlinear dynamical system  $\mathcal{G}$  is *state feedback dissipative* if there exists a state feedback transformation  $u = \phi(x) + \beta(x)v$ , where  $\phi : \mathcal{D} \rightarrow \mathbb{R}^m$  and  $\beta : \mathcal{D} \rightarrow \mathbb{R}^{m \times m}$  are continuously differentiable, with  $\det \beta(x) \neq 0$ ,  $x \in \mathcal{D}$ , such that the nonlinear dynamical system  $\mathcal{G}_s$  given by

$$\dot{x}(t) = f(x(t)) + G(x(t))\phi(x(t)) + G(x(t))\beta(x(t))v(t), \quad x(0) = x_0, \quad t \geq 0, \quad (25.66)$$

$$y(t) = h(x(t)) + J(x(t))\phi(x(t)) + J(x(t))\beta(x(t))v(t), \quad (25.67)$$

is dissipative with respect to the supply rate  $r(v, y)$ , where  $r : U \times Y \rightarrow \mathbb{R}$  is locally integrable for all input–output pairs satisfying (25.66) and (25.67), and  $r(0, 0) = 0$ . If  $r(v, y) = v^T y$ , then  $\mathcal{G}$  is *state feedback passive*.

For simplicity of exposition, here we assume that  $\beta(x) = I_m$ .

The nonlinear dynamical system  $\mathcal{G}$  given by (25.64) and (25.65) is feedback equivalent to a passive system with a two-times continuously differentiable storage function if and only if  $\mathcal{G}$  has (vector) relative degree  $\{1, \dots, 1\}$  at  $x = 0$  and is weakly minimum phase [29]. Alternatively, the Kalman–Yakubovich–Popov lemma [29] can be used to construct smooth state feedback controllers that guarantee feedback passivation as well as feedback dissipativity [185].

The following result is a direct consequence of dissipativity theory [29]. Here we assume that all storage functions  $V_s(\cdot)$  of the nonlinear dynamical system  $\mathcal{G}_s$  are continuously differentiable.

**Proposition 25.1** ([29]) *Consider the nonlinear dynamical system  $\mathcal{G}$  given by (25.64) and (25.65), and assume that  $\mathcal{G}$  is state feedback dissipative. Then there exist functions  $V_s : \mathbb{R}^n \rightarrow \mathbb{R}$ ,  $\ell : \mathbb{R}^n \rightarrow \mathbb{R}^p$ , and  $\mathcal{W} : \mathbb{R}^n \rightarrow \mathbb{R}^{p \times m}$  such that  $V_s(\cdot)$  is continuously differentiable and non-negative definite,  $V_s(x_e) = V_{se}$ , and  $\dot{V}_s(x) = r(v, y) - [\ell(x) + \mathcal{W}(x)v]^T[\ell(x) + \mathcal{W}(x)v]$ .*

Defining  $d(x, v) \triangleq [\ell(x) + \mathcal{W}(x)v]^T[\ell(x) + \mathcal{W}(x)v]$ , where  $d : \mathcal{D} \times U \rightarrow \overline{\mathbb{R}}_+$  is a continuous, non-negative-definite dissipation rate function, and  $dQ(t) \triangleq [r(v(t), y(t)) - d(x(t), v(t))]dt$ , where  $dQ(t)$  is the amount of energy (heat) received or dissipated by the state feedback dissipative system over the infinitesimal time interval  $dt$ , we arrive at a *Clausius-type equality* for  $\mathcal{G}_s$ . To see this, let  $\oint$  denote a cyclic integral evaluated along an arbitrary closed path of  $\mathcal{G}_s$ , that is,  $\oint \triangleq \int_{t_0}^{t_f}$  with  $t_f \geq t_0$  and  $v(\cdot) \in \mathcal{U}$  such that  $x(t_f) = x(t_0) = x_0 \in \mathcal{D}$ .

**Proposition 25.2** *Consider the nonlinear dynamical system  $\mathcal{G}$  given by (25.64) and (25.65), and assume that  $\mathcal{G}$  is state feedback dissipative. Then, for all  $t_f \geq t_0 \geq 0$  and  $v(\cdot) \in \mathcal{U}$  such that  $V_s(x(t_f)) = V_s(x(t_0))$ ,*

$$\int_{t_0}^{t_f} \frac{r(v(t), y(t)) - d(x(t), v(t))}{c + V_s(x(t))} dt = \oint \frac{dQ(t)}{c + V_s(x(t))} = 0, \quad (25.68)$$

where  $c > 0$ .

**Proof** It follows from Proposition 25.1 that

$$\begin{aligned} \oint \frac{dQ(t)}{c + V_s(x(t))} &= \int_{t_0}^{t_f} \frac{r(v(t), y(t)) - d(x(t), v(t))}{c + V_s(x(t))} dt \\ &= \log_e \frac{c + V_s(x(t_f))}{c + V_s(x(t_0))} \\ &= \log_e 1 \\ &= 0, \end{aligned} \quad (25.69)$$

which proves the assertion.  $\square$

In light of Proposition 25.2, we can give a definition of entropy for a feedback dissipative system.

**Definition 25.8** For the nonlinear dynamical system  $\mathcal{G}_s$  given by (25.66) and (25.67) a function  $S : \mathcal{D} \rightarrow \mathbb{R}$  satisfying

$$S(x(t_2)) \geq S(x(t_1)) + \int_{t_1}^{t_2} \frac{dQ(t)}{c + V_s(x(t))} \quad (25.70)$$

for every  $t_2 \geq t_1 \geq 0$  and  $v(\cdot) \in \mathcal{U}$  is called the *entropy* function of  $\mathcal{G}_s$ .

Recalling that  $dQ(t) = [r(v(t), y(t)) - d(x(t), v(t))]dt$  is the infinitesimal amount of the net energy received or dissipated by  $\mathcal{G}_s$  over the infinitesimal time interval  $dt$ , it follows from (25.70) that

$$dS(x(t)) \geq \frac{dQ(t)}{c + V_s(x(t))}, \quad t \geq t_0. \quad (25.71)$$

Inequality (25.71) is analogous to the classical thermodynamic inequality for the variation of entropy during an infinitesimal irreversible transformation with the shifted system energy  $c + V_s(x)$  playing the role of the thermodynamic temperature. Specifically, note that since  $\frac{dS}{dQ} = \frac{1}{c+V_s}$ , it follows that  $\frac{dS}{dQ}$  defines the reciprocal of the system thermodynamic temperature  $T_e$ . That is,

$$\frac{1}{T_e} \triangleq \frac{dS}{dQ}$$

and  $T_e > 0$ .

Next, we show that if  $\mathcal{G}_s$  given by (25.66) and (25.67) is *locally controllable* at  $\hat{x}$ , that is, the set of points that can be reached from and to  $\hat{x}$  in finite time  $T$  using admissible inputs  $v : [0, T] \rightarrow U$ , satisfying  $\|v(t) - \hat{v}\| < \varepsilon$ , contains a neighborhood of  $\hat{x}$  [29, p. 333], then all entropy functions for  $\mathcal{G}_s$  are continuous on  $\mathcal{D}$ .

**Theorem 25.8** Consider the dissipative nonlinear dynamical system  $\mathcal{G}_s$  given by (25.66) and (25.67). Assume that  $\mathcal{G}_s$  is completely reachable and assume that for every  $x_e \in \mathcal{D}$ , there exists  $v_e \in \mathbb{R}^m$  such that  $x(t) \equiv x_e$  and  $v(t) \equiv v_e$ ,  $t \geq 0$ , satisfy (25.66), and  $\mathcal{G}_s$  is locally controllable at every  $x_e \in \mathcal{D}$ . Then every entropy function  $S(x)$ ,  $x \in \mathcal{D}$ , of  $\mathcal{G}_s$  is continuous on  $\mathcal{D}$ .

**Proof** Let  $x_e \in \mathcal{D}$  and  $v_e \in \mathbb{R}^m$  be such that  $f(x_e) + G(x_e)\phi(x_e) + G(x_e)v_e = 0$ , that is,  $x_e$  is an equilibrium point of  $\mathcal{G}_s$  with  $v(t) \equiv v_e$ . Now, let  $\delta > 0$  and note that it follows from the continuity of  $f(\cdot)$ ,  $G(\cdot)$ , and  $\phi(\cdot)$  that there exist  $T > 0$  and  $\varepsilon > 0$  such that for every  $v : [0, T] \rightarrow \mathbb{R}^n$  and  $\|v(t) - v_e\| < \varepsilon$ ,  $\|x(t) - x_e\| < \delta$ ,  $t \in [0, T]$ , where  $v(\cdot) \in \mathcal{U}$  and  $x(t)$ ,  $t \in [0, T]$ , denotes the solution to (25.64) with the initial condition  $x_e$ . Furthermore, it follows from the local controllability of  $\mathcal{G}_s$  that for every  $\hat{T} \in (0, T]$ , there exists a strictly increasing, continuous function  $\gamma : \mathbb{R} \rightarrow \mathbb{R}$  such that  $\gamma(0) = 0$ , and for every  $x_0 \in \mathcal{D}$  such that  $\|x_0 - x_e\| \leq \gamma(\hat{T})$ , there exist  $\hat{t} \in [0, \hat{T}]$  and an input  $v : [0, \hat{T}] \rightarrow \mathbb{R}^m$  such that  $\|v(t) - v_e\| < \varepsilon$ ,  $t \in [0, \hat{t}]$ , and  $x(\hat{t}) = x_0$ . Hence, there exists  $\rho > 0$  such that for every  $x_0 \in \mathcal{D}$  such that  $\|x_0 - x_e\| \leq \rho$ , there exists  $\hat{t} \in [0, \gamma^{-1}(\|x_0 - x_e\|)]$  and an input  $v : [0, \hat{t}] \rightarrow \mathbb{R}^n$  such that  $\|v(t) - v_e\| < \varepsilon$ ,  $t \in [0, \hat{t}]$ , and  $x(\hat{t}) = x_0$ .

Next, since  $r(\cdot, \cdot)$  is locally integrable for all input–output pairs satisfying (25.66) and (25.67), it follows that there exists  $M \in (0, \infty)$  such that

$$\begin{aligned}
\left| \int_0^{\hat{t}} \frac{r(v(\sigma), y(\sigma)) - d(x(\sigma), v(\sigma))}{c + V_s(x(\sigma))} d\sigma \right| &\leq \int_0^{\hat{t}} \left| \frac{dQ(\sigma)}{c + V_s(x(\sigma))} \right| \\
&\leq M\hat{t} \\
&\leq M\gamma^{-1}(\|x_0 - x_e\|).
\end{aligned} \tag{25.72}$$

Now, if  $S(\cdot)$  is an entropy function of  $\mathcal{G}_s$ , then

$$S(x(\hat{t})) \geq S(x_e) + \int_0^{\hat{t}} \frac{dQ(\sigma)}{c + V_s(x(\sigma))}, \tag{25.73}$$

or, equivalently,

$$-\int_0^{\hat{t}} \frac{dQ(\sigma)}{c + V_s(x(\sigma))} \geq S(x_e) - S(x(\hat{t})). \tag{25.74}$$

If  $S(x_e) \geq S(x(\hat{t}))$ , then combining (25.72) and (25.74) yields

$$|S(x_e) - S(x(\hat{t}))| \leq M\gamma^{-1}(\|x_0 - x_e\|). \tag{25.75}$$

Alternatively, if  $S(x(\hat{t})) \geq S(x_e)$ , then (25.75) can be derived by reversing the roles of  $x_e$  and  $x(\hat{t})$ . In particular, using the assumption that  $\mathcal{G}_s$  is locally controllable from and to  $x_e$ , similar arguments can be used to show that the set of points that can be steered in small time to  $x_e$  contains a neighborhood of  $x(\hat{t})$ . Hence, since  $\gamma(\cdot)$  is continuous and  $x(\hat{t})$  is arbitrary, it follows that  $S(\cdot)$  is continuous on  $\mathcal{D}$ .  $\square$

Next, we characterize a continuously differentiable entropy function for state feedback dissipative systems.

**Proposition 25.3** *Consider the dissipative nonlinear dynamical system  $\mathcal{G}_s$  given by (25.66) and (25.67). Then the continuously differentiable function  $S : \mathcal{D} \rightarrow \mathbb{R}$  given by*

$$S(x) \triangleq \log_e [c + V_s(x)] - \log_e c, \tag{25.76}$$

where  $c > 0$ , is an entropy function of  $\mathcal{G}_s$ .

**Proof** Using Proposition 25.1 it follows that

$$\begin{aligned}
\dot{S}(x(t)) &= \frac{\dot{V}_s(x(t))}{c + V_s(x(t))} \\
&= \frac{r(v(t), y(t)) - d(x(t), v(t))}{c + V_s(x(t))} \\
&= \frac{\dot{Q}(t)}{c + V_s(x(t))}, \quad t \geq 0.
\end{aligned} \tag{25.77}$$

Now, integrating (25.77) over  $[t_1, t_2]$  yields (25.70).  $\square$

In [56], we showed that the entropy function for an energy balance equation involving a large-scale, compartmental thermodynamic model is unique. However, whether or not there exists a unique continuously differentiable entropy function for the general nonlinear dynamical system  $\mathcal{G}_s$  given by (25.66) and (25.67) is an open problem.

Finally, the following result presenting an upper and lower bound of the entropy function for a state feedback dissipative system is needed.

**Proposition 25.4** *Consider the nonlinear dynamical system  $\mathcal{G}_s$  given by (25.66) and (25.67), and let  $S : \mathcal{D} \rightarrow \mathbb{R}$  given by (25.76) be an entropy function of  $\mathcal{G}_s$ . Then,*

$$\frac{V_s(x)}{c + V_s(x)} \leq S(x) \leq \frac{1}{c} V_s(x), \quad x \in \mathcal{D}. \quad (25.78)$$

**Proof** Note that (25.76) can be rewritten as  $S(x) = \log_e[1 + V_s(x)/c]$ . The assertion now follows as a direct consequence of the inequality  $z/(1+z) \leq \log_e(1+z) \leq z$ ,  $z > -1$ .  $\square$

Using the notion of feedback dissipativity, we can develop a framework for optimal *semistabilization* of nonlinear systems; a notion briefly discussed at the end of Sect. 25.4. To address the state feedback, thermodynamic-based semistabilization problem, consider the nonlinear dynamical system  $\mathcal{G}_s$  given by (25.66) and (25.67) with performance criterion

$$J(x_0, \phi(\cdot)) = \lim_{t \rightarrow \infty} \left[ \frac{1}{t} \int_0^t S(x(\sigma)) d\sigma \right]. \quad (25.79)$$

The performance criterion  $J(x_0, \phi(\cdot))$  can be interpreted as the time-average of the entropy function for the dissipative nonlinear dynamical system  $\mathcal{G}_s$ . The key feature of this optimal control problem is that it addresses semistability instead of asymptotic stability. In the absence of energy exchange with the environment, a thermodynamically consistent nonlinear dynamical system model possesses a continuum of equilibria, and hence, is semistable; that is, the system states converge to Lyapunov stable energy equilibria determined by the system initial conditions [56]. A key question that arises is whether or not this optimal control problem is well defined; that is, whether  $J(x_0, \phi(\cdot))$  is finite, and if there exists a state feedback controller such that  $J(x_0, \phi(\cdot))$  is minimized.

The first question is addressed by the following proposition.

**Proposition 25.5** *Consider the nonlinear dissipative dynamical system  $\mathcal{G}_s$  given by (25.66) and (25.67). If there exists  $\phi : \mathcal{D} \rightarrow \mathbb{R}^m$  such that (25.66), with  $v(t) \equiv 0$ , is semistable, then  $|J(x_0, \phi(\cdot))| < \infty$ .*

**Proof** Since (25.66) with  $v(t) \equiv 0$  is semistable, it follows that  $x(t)$  is bounded for all  $t \geq 0$ . Furthermore, it follows from Theorem 25.8 that  $S(\cdot)$  is a continuous entropy function on  $\mathcal{D}$  for  $\mathcal{G}_s$ . Hence,  $S(x(t))$  is bounded for all  $t \geq 0$ . Now, let

$|S(x(t))| \leq c$  for all  $t \geq 0$ . Then, it follows that  $-c \leq (1/t) \int_0^t S(x(\sigma))d\sigma \leq c$  for all  $t \geq 0$ , which proves the result.  $\square$

To address the question of existence of a semistabilizing controller such that  $J(x_0, \phi(\cdot))$  given by (25.79) is minimized, we consider an *auxiliary minimization problem* involving the performance criterion

$$\mathcal{J}(x_0, \phi(\cdot)) = \lim_{t \rightarrow \infty} \left[ \frac{1}{t} \int_0^t V_s(x(\sigma))d\sigma \right]. \quad (25.80)$$

It follows from the auxiliary minimization problem that we seek feedback controllers that minimize the stored energy in the system in order to attain a stable minimum energy level determined by the system initial conditions and the control system effort.

The following lemma is necessary for proving the main result of this section.

**Lemma 25.1** Consider the dissipative nonlinear dynamical system  $\mathcal{G}_s$  given by (25.66) and (25.67) with continuously differentiable storage function  $V_s : \mathcal{D} \rightarrow \mathbb{R}_+$ . Suppose there exists  $\phi^* : \mathcal{D} \rightarrow \mathbb{R}^m$  such that (25.66) with  $v(t) \equiv 0$  is semistable,  $\dot{V}_s(x(t)) \leq 0$ ,  $t \geq 0$ , and  $\mathcal{J}(x_0, \phi(\cdot))$  is minimized. If  $\mathcal{J}(x_0, \phi^*(\cdot)) = 0$ , then  $\arg \min_{\phi(\cdot) \in \mathbb{R}^m} \mathcal{J}(x_0, \phi(\cdot)) = \arg \min_{\phi(\cdot) \in \mathbb{R}^m} J(x_0, \phi(\cdot))$  and  $J(x_0, \phi^*(\cdot)) = 0$ . Alternatively, if  $\mathcal{J}(x_0, \phi^*(\cdot)) \neq 0$ , then  $J(x_0, \phi^*(\cdot)) = S(x_e)$ , where  $x_e = \lim_{t \rightarrow \infty} x(t)$ .

**Proof** It follows from Proposition 25.4 and  $\dot{V}_s(x(t)) \leq 0$ ,  $t \geq 0$ , that

$$\frac{V_s(x(t))}{c + V_s(x(0))} \leq S(x(t)) \leq \frac{V_s(x(t))}{c}, \quad t \geq 0. \quad (25.81)$$

Hence,

$$\frac{\mathcal{J}(x_0, \phi(\cdot))}{(c + V_s(x(0)))} \leq J(x_0, \phi(\cdot)) \leq \frac{\mathcal{J}(x_0, \phi(\cdot))}{c}.$$

Now, if  $\mathcal{J}(x_0, \phi^*(\cdot)) = 0$ , then  $J(x_0, \phi(\cdot))$  is minimized and  $J(x_0, \phi^*(t)) = 0$ ,  $t \geq 0$ .

Alternatively, if  $\mathcal{J}(x_0, \phi^*(\cdot)) \neq 0$ , then it follows from the definition of  $\mathcal{J}(x_0, \phi^*(\cdot))$  that there exists  $c^* > 0$  such that  $\mathcal{J}(x_0, \phi^*(t)) \geq c^*$  for all  $t \geq 0$ , and hence,

$$\lim_{t \rightarrow \infty} \mathcal{J}(x_0, \phi^*(t))t/(c + V_s(x(0))) = \infty.$$

Thus,

$$\lim_{t \rightarrow \infty} \int_0^t S(x(\sigma))d\sigma = \infty.$$

Now, using l'Hôpital's rule, it follows that  $J(x_0, \phi^*(t)) = S(x_e)$ , where  $x_e = \lim_{t \rightarrow \infty} x(t)$ .  $\square$

**Theorem 25.9** Consider the dissipative nonlinear dynamical system  $\mathcal{G}_s$  given by (25.66) and (25.67) with continuously differentiable storage function  $V_s : \mathcal{D} \rightarrow \mathbb{R}_+$ . Assume that there exists  $\phi^* : \mathcal{D} \rightarrow \mathbb{R}^m$  such that (25.80) is minimized, (25.66), with  $v(t) \equiv 0$ , is semistable, and  $\dot{V}_s(x(t)) \leq 0$ ,  $t \geq 0$ . Then, for  $S : \mathcal{D} \rightarrow \mathbb{R}$  given by (25.76),  $\arg \min_{\phi(\cdot) \in \mathbb{R}^m} \mathcal{J}(x_0, \phi(\cdot)) = \arg \min_{\phi(\cdot) \in \mathbb{R}^m} J(x_0, \phi(\cdot))$ .

**Proof** If  $\mathcal{J}(x_0, \phi^*(\cdot)) = 0$ , then it follows from Lemma 25.1 that  $\phi^*(\cdot) = \arg \min_{\phi(\cdot) \in \mathbb{R}^m} J(x_0, \phi(\cdot))$ . Alternatively, if  $\mathcal{J}(x_0, \phi^*(\cdot)) \neq 0$ , then, using similar arguments as in the proof of Lemma 25.1,  $\lim_{t \rightarrow \infty} t \mathcal{J}(x_0, \phi^*(t)) = \infty$ , and hence,  $\int_0^t V_s(x(\sigma)) d\sigma = \infty$  as  $t \rightarrow \infty$ . Hence, using l'Hôpital's rule, it follows that  $\mathcal{J}(x_0, \phi^*(\cdot)) = V_s(x_e)$ .

Next, since for all  $\phi : \mathcal{D} \rightarrow \mathbb{R}^m$  such that  $\mathcal{J}(x_0, \phi(\cdot))$  is finite,

$$\lim_{t \rightarrow \infty} \mathcal{J}(x_0, \phi(t)) t / (c + V_s(x(0))) = \infty,$$

it follows from (25.81) that

$$\lim_{t \rightarrow \infty} \int_0^t S(x(\sigma)) d\sigma = \infty.$$

Consequently, for all  $\phi : \mathcal{D} \rightarrow \mathbb{R}^m$  such that  $\mathcal{J}(x_0, \phi(\cdot))$  is finite and  $\mathcal{G}_s$  is semistable, it follows from l'Hôpital's rule that  $J(x_0, \phi(\cdot)) = S(x_e) = \log_e(1 + V_s(x_e))$ , where  $x_e = \lim_{t \rightarrow \infty} x(t)$ . Next, assume that  $\phi_j^* : \mathcal{D} \rightarrow \mathbb{R}^m$  is such that  $\mathcal{G}_s$  is semistable,  $\dot{V}_s(x(t)) \leq 0$ ,  $t \geq 0$ , and  $J(x_0, \phi(\cdot))$  is minimized. Then, it follows that  $J(x_0, \phi_j^*(\cdot)) = S(x_e) = \log_e(1 + V_s(x_e))$ .

Finally, by uniqueness of solutions of  $x(\cdot)$  it follows that  $\phi^*(\cdot)$  uniquely determines  $x_e$  and  $\dot{V}_s(x(t))$ ,  $t \geq 0$ . Choosing  $V_s(x_e) = V_{se}$ , where  $V_{se} \in \mathbb{R}$ , it follows that  $\phi^*(\cdot)$  uniquely determines  $V_s(x_e)$ , and hence,  $J(x_0, \phi^*(\cdot)) = \log_e(1 + V_s(x_e))$ , which proves the result.  $\square$

It follows from Theorem 25.9 that an optimal semistable controller minimizing  $\mathcal{J}(x_0, v(\cdot))$  given by (25.80) also minimizes the entropy functional  $J(x_0, v(\cdot))$  given by (25.79). Since quadratic cost functions arise naturally in dissipativity theory [29, 186, 187], addressing the auxiliary cost (25.80) can be simpler than addressing the entropy (logarithmic) cost functional (25.79). The afore highlighted framework can be explored to quantify fundamental limits of performance for nonlinear dynamical systems using thermodynamic entropy notions. This is a subject of ongoing research.

## 25.8 Conclusion

In this paper, we highlighted the salient features for developing a general framework for the analysis and control design of complex, large-scale network systems. This framework is predicated on fundamental principles from systems biology, neurophysiology, and dynamical thermodynamics to address issues of robustness, semistability, hybrid stabilization, stochasticity, optimality, learning, system mode switching,

controller complexity, system complexity, scalability, and limits of performance in nonlinear system stabilization and spatially invariant multiagent systems. Such a framework would significantly advance the state of the art in embedded hybrid control system design as well as cooperative control and coordination of multiagent systems. In addition to network systems, the framework can be applied to queuing systems, ecological systems, economic systems, telecommunications systems, transportation systems, power systems, mean field neural systems, and large-scale aerospace systems, to cite but a few examples. Payoffs would arise from improvements in system performance, robustness, reliability, controller complexity, and maintainability, as well as reduced development cost, design time, and hardware/software implementation requirements for hierarchical and hybrid control systems.

**Acknowledgements** This research was supported in part by the Air Force of Scientific Research under Grant FA9550-16-1-0100.

## References

1. Reynolds, C.W.: Flocks, herds, and schools: a distributed behavioral model. *Comput. Graph.* **21**, 25–34 (1987)
2. Vicsek, T., Czirók, A., Ben-Jacob, E., Cohen, I., Shochet, O.: Novel type of phase transition in a system of self-deriven particles **75**(6), 1226–1229 (1995)
3. Ando, H., Oasa, Y., Suzuki, I., Yamashita, M.: Distributed memoryless point convergence algorithm for mobile robots with limited visibility. *IEEE Trans. Robot. Automat.* **15**(5), 818–828 (1999)
4. Desai, J.P., Ostrowski, J.P., Kumar, V.: Modeling and control of formations of nonholonomic mobile robots. *IEEE Trans. Robot. Automat.* **17**, 905–908 (2001)
5. Paganini, F., Doyle, J.C., Low, S.H.: Scalable laws for stable network congestion control. In: Proceedings of the IEEE Conference on Decision and Control, Orlando, FL, pp. 185–190 (2001)
6. Leonard, N.E., Fiorelli, E.: Virtual leaders, artificial potentials, and coordinated control of groups. In: Proceedings of the IEEE Conference on Decision and Control, Orlando, FL, pp. 2968–2973 (2001)
7. Liu, Y., Passino, K.M., Polycarpou, M.M.: Stability analysis of m-dimensional asynchronous swarms with a fixed communication topology. *IEEE Trans. Autom. Control* **48**, 76–95 (2003)
8. Gazi, V., Passino, K.M.: Stability analysis of swarms. *IEEE Trans. Autom. Control* **48**, 692–697 (2003)
9. Jadbabaie, A., Lin, J., Morse, A.S.: Coordination of groups of mobile autonomous agents using nearest neighbor rules. *IEEE Trans. Autom. Control* **48**, 988–1001 (2003)
10. Vidal, R., Shakernia, O., Sastry, S.: Formation control of nonholonomic mobile robots with omnidirectional visual serving and motion segmentation. In: Proceedings of the IEEE International Conference on Robotics and Automation, pp. 584–589 (2003)
11. Tanner, H.G., Jadbabaie, A., Pappas, G.J.: Stable flocking of mobile agents, Part I: fixed topology. In: Proceedings of the IEEE Conference on Decision and Control, Maui, HI, pp. 2010–2015 (2003)
12. Lin, Z., Broucke, M.E., Francis, B.A.: Local control strategies for groups of mobile autonomous agents. *IEEE Trans. Autom. Control* **49**(4), 622–629 (2004)
13. Olfati-Saber, R., Murray, R.M.: Consensus problems in networks of agents with switching topology and time-delays. *IEEE Trans. Autom. Control* **49**, 1520–1533 (2004)

14. Fax, J.A., Murray, R.M.: Information flow and cooperative control of vehicle formations. *IEEE Trans. Autom. Control* **49**, 1465–1476 (2004)
15. Marshall, J.A., Broucke, M.E., Francis, B.A.: Formations of vehicles in cyclic pursuit. *IEEE Trans. Autom. Control* **49**, 1963–1974 (2004)
16. Justh, E.W., Krishnaprasad, P.S.: Equilibria and steering laws for planar formations. *Syst. Control Lett.* **52**, 25–38 (2004)
17. Ren, W., Beard, R.W.: Consensus seeking in multiagent systems under dynamically changing interaction topologies. *IEEE Trans. Autom. Control* **50**(5), 655–661 (2005)
18. Olfati-Saber, R.: Flocking for multi-agent dynamic systems: algorithms and theory. *IEEE Trans. Autom. Control* **51**, 401–420 (2006)
19. Tanner, H.G., Jadbabaie, A., Pappas, G.J.: Flocking in fixed and switching networks. *IEEE Trans. Autom. Control* **52**, 863–868 (2007)
20. Wolfe, J.D., Chichka, D.F., Speyer, J.L.: Decentralized controllers for unmanned aerial vehicle formation flight. In: Proceedings of the AIAA Conference on Guidance, Navigation, and Control, AIAA-1996-3833, San Diego, CA (1996)
21. Cortés, J., Bullo, F.: Coordination and geometric optimization via distributed dynamical systems. *SIAM J. Control Opt.* **44**, 1543–1574 (2005)
22. Swaroop, D., Hedrick, J.K.: Constant spacing strategies for platooning in automated highway systems. In: ASME Journal of Dynamic Systems, Measurement, and Control, vol. 121, pp. 462–470 (1999)
23. Suzuki, I., Yamashita, M.: Distributed anonymous mobile robots: formation of geometric patterns. *SIAM J. Comput.* **28**, 1347–1363 (1999)
24. Lin, J., Morse, A.S., Anderson, B.D.O.: The multi-agent rendezvous problem. In: Proceedings of the IEEE Conference on Decision and Control, Maui, HI, pp. 1508–1513 (2003)
25. Hui, Q., Haddad, W.M., Bhat, S.P.: Finite-time semistability and consensus for nonlinear dynamical networks. *IEEE Trans. Autom. Control* **53**, 1887–1900 (2008)
26. Hui, Q., Haddad, W.M.: Continuous and hybrid distributed control for multiagent coordination: Consensus, flocking, and cyclic pursuit. In: Proceedings of the American Control Conference, New York, NY, pp. 2576–2581 (2007)
27. Bhat, S.P., Bernstein, D.S.: Nontangency-based Lyapunov tests for convergence and stability in systems having a continuum of equilibria. *SIAM J. Control Optim.* **42**, 1745–1775 (2003)
28. Bhat, S.P., Bernstein, D.S.: Arc-length-based Lyapunov tests for convergence and stability with applications to systems having a continuum of equilibria. *Math. Control Signals Syst.* **22**(2), 155–184 (2010)
29. Haddad, W.M., Chellaboina, V.: Nonlinear Dynamical Systems and Control: A Lyapunov-Based Approach. Princeton University Press, Princeton (2008)
30. Angeli, D., Ferrell, J., Sontag, E.D.: Detection of multistability, bifurcations, and hysteresis in a large class of biological positive-feedback systems. *Proc. Natl. Acad. Sci.* **101**, 1822–1827 (2004)
31. Angeli, D.: New analysis technique for multistability detection. *IEE Proc. Syst. Biol.* **153**, 61–69 (2006)
32. Angeli, D.: Multistability in systems with counter-clockwise input-output dynamics. *IEEE Trans. Autom. Control* **52**, 596–609 (2007)
33. Leonessa, A., Haddad, W.M., Chellaboina, V.: Hierarchical Nonlinear Switching Control Design with Applications to Propulsion Systems. Springer, London (2000)
34. Leonessa, A., Haddad, W.M., Chellaboina, V.: Nonlinear system stabilization via hierarchical switching control. *IEEE Trans. Autom. Control* **46**, 17–28 (2001)
35. Haddad, W.M., Chellaboina, V., Nersesov, S.G.: Impulsive and Hybrid Dynamical Systems: Stability, Dissipativity, and Control. Princeton University Press, Princeton (2006)
36. Peleties, P., DeCarlo, R.: Asymptotic stability of m-switched systems using Lyapunov-like functions. In: Proceedings of the American Control Conference, Boston, MA, pp. 1679–1684 (1991)
37. Branicky, M.S.: Multiple-Lyapunov functions and other analysis tools for switched and hybrid systems. *IEEE Trans. Autom. Control* **43**, 475–482 (1998)

38. Brach, R.M.: *Mechanical Impact Dynamics*. Wiley, New York (1991)
39. Brogliato, B.: *Nonsmooth Impact Mechanics: Models, Dynamics, and Control*. Springer, London (1996)
40. Lakshmikantham, V., Bainov, D.D., Simeonov, P.S.: *Theory of Impulsive Differential Equations*. World Scientific, Singapore (1989)
41. Hagiwara, T., Araki, M.: Design of a stable feedback controller based on the multirate sampling of the plant output. *IEEE Trans. Autom. Control* **33**, 812–819 (1988)
42. Passino, K.M., Michel, A.N., Antsaklis, P.J.: Lyapunov stability of a class of discrete event systems. *IEEE Trans. Autom. Control* **39**, 269–279 (1994)
43. Lygeros, J., Godbole, D.N., Sastry, S.: Verified hybrid controllers for automated vehicles. *IEEE Trans. Autom. Control* **43**, 522–539 (1998)
44. Tomlin, C., Pappas, G.J., Sastry, S.: Conflict resolution for air traffic management: a study in multiagent hybrid systems. *IEEE Trans. Autom. Control* **43**, 509–521 (1998)
45. Bainov, D.D., Simeonov, P.S.: *Systems with Impulse Effect: Stability, Theory and Applications*. Ellis Horwood Limited, England (1989)
46. Hu, S., Lakshmikantham, V., Leela, S.: Impulsive differential systems and the pulse phenomena. *J. Math. Anal. Appl.* **137**, 605–612 (1989)
47. Bainov, D.D., Simeonov, P.S.: *Impulsive Differential Equations: Asymptotic Properties of the Solutions*. World Scientific, Singapore (1995)
48. Samoilenko, A.M., Perestyuk, N.A.: *Impulsive Differential Equations*. World Scientific, Singapore (1995)
49. Haddad, W.M.: Nonlinear differential equations with discontinuous right-hand sides: Filippov solutions, nonsmooth stability and dissipativity theory, and optimal discontinuous feedback control. *Commun. Appl. Anal.* **18**, 455–522 (2014)
50. Goebel, R., Sanfelice, R.G., Teel, A.R.: Hybrid dynamical systems. *IEEE Control Syst. Mag.* **29**(2), 28–93 (2009)
51. Goebel, R., Sanfelice, R.G., Teel, A.R.: *Hybrid Dynamical Systems: Modeling Stability, and Robustness*. Princeton University Press, Princeton (2012)
52. Passino, K.M.: Biomimicry of bacterial foraging for distributed optimization and control. *IEEE Control Syst. Mag.* **22**, 52–67 (2002)
53. Levine, H., Rappel, W., Cohen, I.: Self-organization in systems of self-propelled particles. *Phys. Rev. E* **63**, 017101 (2000)
54. Mogilner, A., Edelstein-Keshet, L.: A non-local model for a swarm. *J. Math. Biology* **38**, 534–570 (1999)
55. Haddad, W.M., Chellaboina, V.S., Nersesov, S.G.: *Thermodynamics: A Dynamical Systems Approach*. Princeton University Press, Princeton (2005)
56. Haddad, W.M.: *A Dynamical Systems Theory of Thermodynamics*. Princeton University Press, Princeton (2019)
57. Haddad, W.M.: Condensed matter physics, hybrid energy and entropy principles, and the hybrid first and second laws of thermodynamics. *Comm. Nonlin. Sci. Numer. Simul.* **83**, 105096 (2020)
58. Haddad, W.M., Rajpurohit, T., Jin, X.: Stochastic semistability for nonlinear dynamical systems with application to consensus on networks with communication uncertainty. *IEEE Trans. Autom. Control* **65**(7), 2826–2841 (2020)
59. Rajpurohit, T., Haddad, W.M.: Dissipativity theory for nonlinear stochastic dynamical systems. *IEEE Trans. Autom. Control* **62**(4), 1684–1699 (2017)
60. Hui, Q., Haddad, W.M.: Distributed nonlinear control algorithms for network consensus. *Automatica* **44**(9), 2375–2381 (2008)
61. Haddad, W.M., Chellaboina, V., Nersesov, S.G.: *Thermodynamics: A Dynamical Systems Approach*. Princeton University Press, Princeton (2005)
62. Haddad, W.M., Chellaboina, V., Kablar, N.A.: Nonlinear impulsive dynamical systems. Part I: stability and dissipativity. *Int. J. Control* **74**, 1631–1658 (2001)
63. Haddad, W.M., Chellaboina, V., Kablar, N.A.: Nonlinear impulsive dynamical systems. Part II: stability of feedback interconnections and optimality. *Int. J. Control* **74**, 1659–1677 (2001)

64. Chellaboina, V., Bhat, S.P., Haddad, W.M.: An invariance principle for nonlinear hybrid and impulsive dynamical systems. *Nonlinear Anal.* **53**, 527–550 (2003)
65. Michel, A.N., Wang, K., Hu, B.: Qualitative Theory of Dynamical Systems: The Role of Stability Preserving Mappings. Marcel Dekker Inc, New York (2001)
66. Haddad, W.M., Chahine, M.: A thermodynamically-based hybrid communication control architecture for semistability and consensus of multiagent dynamical systems. *IEEE Trans. Cybern.* (to appear)
67. Bryson, A.E.: Control of Aircraft and Spacecraft. Princeton University Press, Princeton (1993)
68. Haddad, W.M., Nersesov, S.G., Hui, Q., Ghasemi, M.: Formation control protocols for general nonlinear dynamical systems via hybrid stabilization of sets. *ASME J. Dyn. Syst. Meas. Control* **136**(051020), 1–13 (2014)
69. Rajpurohit, T., Haddad, W.M.: Stochastic thermodynamics: a stochastic dynamical system approach. *Entropy* **17**(693), 1–48 (2017)
70. Justh, E.W., Krishnaprasad, P.: Equilibria and steering laws for planar formations. *Syst. & Control Lett.* **52**(1), 25–38 (2004)
71. Tanner, H.G., Jadbabaie, A., Pappas, G.J.: Flocking in fixed and switching networks. *IEEE Trans. Autom. Control* **52**(5), 863–868 (2007)
72. Cao, Y., Ren, W., Meng, Z.: Decentralized finite-time sliding mode estimators and their applications in decentralized finite-time formation tracking. *Syst. & Control Lett.* **59**(9), 522–529 (2010)
73. Yu, W., Chen, G., Cao, M., Kurths, J.: Second-order consensus for multiagent systems with directed topologies and nonlinear dynamics. *IEEE Trans. Syst. Man Cybern. Part B (Cybernetics)* **40**(3), 881–891 (2010)
74. Li, S., Du, H., Lin, X.: Finite-time consensus algorithm for multi-agent systems with double-integrator dynamics. *Automatica* **47**(8), 1706–1712 (2011)
75. Song, Q., Liu, F., Cao, J., Yu, W.:  $M$ -matrix strategies for pinning-controlled leader-following consensus in multiagent systems with nonlinear dynamics. *IEEE Trans. Cybern.* **43**(6), 1688–1697 (2013)
76. Cao, Y., Ren, W., Casbeer, D.W., Schumacher, C.: Finite-time connectivity-preserving consensus of networked nonlinear agents with unknown Lipschitz terms. *IEEE Trans. Autom. Control* **61**(6), 1700–1705 (2016)
77. Li, T., Zhang, J.-F.: Asymptotically optimal decentralized control for large population stochastic multiagent systems. *IEEE Trans. Autom. Control* **53**(7), 1643–1660 (2008)
78. Li, Z., Duan, Z., Chen, G., Huang, L.: Consensus of multiagent systems and synchronization of complex networks: a unified viewpoint. *IEEE Trans. Circuits Syst. I: Regular Papers* **57**(1), 213–224 (2010)
79. Zheng, Y., Chen, W., Wang, L.: Finite-time consensus for stochastic multi-agent systems. *Int. J. Control* **84**(10), 1644–1652 (2011)
80. Wu, X., Tang, Y., Cao, J., Zhang, W.: Distributed consensus of stochastic delayed multi-agent systems under asynchronous switching. *IEEE Trans. Cybern.* **46**(8), 1817–1827 (2016)
81. Wu, X., Tang, Y., Zhang, W.: Stability analysis of stochastic delayed systems with an application to multi-agent systems. *IEEE Trans. Autom. Control* **61**(12), 4143–4149 (2016)
82. Arapostathis, A., Borkar, V.S., Ghosh, M.K.: Ergodic Control of Diffusion Processes. Cambridge University Press, Cambridge (2012)
83. Goldstein, H.: Classical Mechanics. Addison-Wesley, Reading (1980)
84. Berman, A., Plemmons, R.J.: Nonnegative Matrices in the Mathematical Sciences. Academic Press Inc, New York (1979)
85. Dayan, P., Abbott, L.F.: Theoretical Neuroscience: Computational and Mathematical Modeling of Neural Systems. MIT Press, Cambridge (2005)
86. Ermentrout, B., Terman, D.H.: Mathematical Foundations of Neuroscience. Springer, New York (2010)
87. Shirani, F., Haddad, W.M., de la Llave, R.: On the global dynamics of an electroencephalographic mean field model of the neocortex. *SIAM J. Appl. Dyn. Syst.* **16**(4), 1969–2029 (2017)

88. Haddad, W.M.: A computational study of a spatiotemporal mean field model capturing the emergence of alpha and gamma rhythmic activity in the neocortex. *Symmetry* **10**(568), 1–16 (2018)
89. Haddad, W.M., Hui, Q., Bailey, J.M.: Human brain networks: Spiking neuron models, multistability, synchronization, thermodynamics, maximum entropy production, and anesthetic cascade mechanisms. *Entropy* **16**, 3939–4003 (2014)
90. Hui, Q., Haddad, W.M., Bailey, J.M., Hayakawa, T.: A stochastic mean field model for an excitatory and inhibitory synaptic drive cortical neuronal network. *IEEE Trans. Neural Netw. Learn. Syst.* **25**(4), 751–763 (2014)
91. Hou, S.P., Haddad, W.M., Meskin, N., Bailey, J.M.: A mechanistic neural field theory of how anesthesia suppresses consciousness: synaptic drive dynamics, bifurcations, attractors, and partial state equipartitioning. *J. Math. Neurosci.* **5**(1), 1–50 (2015)
92. Haddad, W., Hou, S., Bailey, J., Meskin, N.: A neural field theory for loss of consciousness: Synaptic drive dynamics, system stability, attractors, partial synchronization, and Hopf bifurcations characterizing the anesthetic cascade. In: Jagannathan, S., Vamvoudakis, K.G. (eds.) *Control of Complex Systems*, pp. 93–162. Elsevier, Amsterdam (2016)
93. Kondepudi, D., Prigogine, I.: *Modern Thermodynamics: From Heat Engines to Dissipative Structures*. Wiley, New York (1998)
94. Prigogine, I.: *From Being to Becoming*. Freeman, New York (1980)
95. Haddad, W.M., Hui, Q.: Complexity, robustness, self-organization, swarms, and system thermodynamics. *Nonlinear Anal.: Real World Appl.* **10**, 531–543 (2009)
96. Haddad, W.M.: Temporal asymmetry, entropic irreversibility, and finite-time thermodynamics: from Parmenides-Einstein time-reversal symmetry to the Heraclitan entropic arrow of time. *Entropy* **14**, 407–455 (2012)
97. Brown, E., Moehlis, J., Holmes, P.: On the phase reduction and response dynamics of neural oscillator populations. *Neural Comput.* **16**, 673–715 (2004)
98. Haddad, W.M., Chellaboina, V., Nersesov, S.G.: Time-reversal symmetry, Poincaré recurrence, irreversibility, and the entropic arrow of time: from mechanics to system thermodynamics. *Nonlinear Anal.: Real World Appl.* **9**, 250–271 (2008)
99. Haddad, W.M.: A unification between dynamical system theory and thermodynamics involving an energy, mass, and entropy state space formalism. *Entropy* **15**, 1821–1846 (2013)
100. Haddad, W.M.: Thermodynamics: the unique universal science. *Entropy* **19**(621), 1–70 (2017)
101. Godsil, C., Royle, G.: *Algebraic Graph Theory*. Springer, New York (2001)
102. Bernstein, D.S.: Nonquadratic cost and nonlinear feedback control. *Int. J. Rob. Nonlinear Control* **3**(3), 211–229 (1993)
103. Lewis, J.B.: The use of nonlinear feedback to improve the transient response of a servomechanism. *Trans. AIAA (Part II. Applications and Industry)* **71**, 449–453 (1953)
104. Al'brekht, E.G.: On the optimal stabilization of nonlinear systems. *J. Appl. Math. Mech.* **25**(5), 1254–1266 (1961)
105. Rekasius, Z.: Suboptimal design of intentionally nonlinear controllers. *IEEE Trans. Autom. Control* **9**(4), 380–386 (1964)
106. Bass, R., Webber, R.: Optimal nonlinear feedback control derived from quartic and higher-order performance criteria. *IEEE Trans. Autom. Control* **11**(3), 448–454 (1966)
107. Thau, F.: On the inverse optimum control problem for a class of nonlinear autonomous systems. *IEEE Trans. Autom. Control* **12**(6), 674–681 (1967)
108. Lukes, D.: Optimal regulation of nonlinear dynamical systems. *SIAM J. Control* **7**(1), 75–100 (1969)
109. Asseo, S.: Optimal control of a servo derived from nonquadratic performance criteria. *IEEE Trans. Autom. Control* **14**(4), 404–407 (1969)
110. Rugh, W.: On an inverse optimal control problem. *IEEE Trans. Autom. Control* **16**(1), 87–88 (1971)
111. Thau, F.E.: Optimum nonlinear control of a class of randomly excited systems. *J. Dyn. Syst. Meas. Control* 41–44 (1971)

112. Leeper, J., Mulholland, R.J.: Optimal control of nonlinear single-input systems. *IEEE Trans. Autom. Control* **17**(3), 401–402 (1972)
113. Mekel, R., Perujo, P.: Design of controllers for a class of nonlinear control systems. *IEEE Trans. Autom. Control* **17**(2), 206–213 (1972)
114. Moylan, P., Anderson, B.D.O.: Nonlinear regulator theory and an inverse optimal control problem. *IEEE Trans. Autom. Control* **18**(5), 460–465 (1973)
115. Halme, A., Hamalainen, R.P., Heikkila, O., Laaksonen, O.: On synthesizing a state regulator for analytic non-linear discrete-time systems. *Int. J. Control* **20**(3), 497–515 (1974)
116. Garrard, W.: Suboptimal feedback control for nonlinear systems. *Automatica* **8**(2), 219–221 (1972)
117. Halme, A., Hämäläinen, R.: On the nonlinear regulator problem. *J. Opt. Theory Appl.* **16**(3–4), 255–275 (1975)
118. Speyer, J.L.: A nonlinear control law for a stochastic infinite time problem. *IEEE Trans. Autom. Control* **21**(4), 560–564 (1976)
119. Sandor, J., Williamson, D.: Design of nonlinear regulators for linear plants. *IEEE Trans. Autom. Control* **22**(1), 47–50 (1977)
120. Garrard, W.L., Jordan, J.M.: Design of nonlinear automatic flight control systems. *Automatica* **13**(5), 497–505 (1977)
121. Willemstein, A.: Optimal regulation of nonlinear dynamical systems on a finite interval. *SIAM J. Control Opt.* **15**(6), 1050–1069 (1977)
122. Chow, J., Kokotovic, P.: Near-optimal feedback stabilization of a class of nonlinear singularly perturbed systems. *SIAM J. Control Opt.* **16**(5), 756–770 (1978)
123. Shamaly, A., Christensen, G., El-Hawary, M.: A transformation for necessary optimality conditions for systems with polynomial nonlinearities. *IEEE Trans. Autom. Control* **24**(6), 983–985 (1979)
124. Shaw, L.: Nonlinear control of linear multivariable systems via state-dependent feedback gains. *IEEE Trans. Autom. Control* **24**(1), 108–112 (1979)
125. Kosut, R.L.: Nonlinear optimal cue-shaping filters for motion bade simulators. *AIAA J. Guid. Control* **24**, 108–112 (1979)
126. Jacobson, D.: Extensions of Linear-Quadratic Control Theory. Springer, Berlin (1980)
127. Chow, J., Kokotovic, P.: A two-stage Lyapunov-Bellman feedback design of a class of nonlinear systems. *IEEE Trans. Autom. Control* **26**(3), 656–663 (1981)
128. Salehi, S.V., Ryan, E.: On optimal nonlinear feedback regulation of linear plants. *IEEE Trans. Autom. Control* **27**(6), 1260–1264 (1982)
129. Beaman, J.J.: Non-linear quadratic Gaussian control. *Int. J. Control* **39**(2), 343–361 (1984)
130. Ozgoren, M., Longman, R.: Automated derivation of optimal regulators for nonlinear systems by symbolic manipulation of poisson series. *J. Opt. Theory Appl.* **45**(3), 443–476 (1985)
131. O'Sullivan, J.A., Sain, M.K.: Nonlinear optimal control with tensors: some computational issues. In: American Control Conference, pp. 1600–1605 (1985)
132. Rouff, M., Lamnabhi-Lagarrigue, F.: A new approach to nonlinear optimal feedback law. *Syst. Control Lett.* **7**(5), 411–417 (1986)
133. Hopkins, W.J.: Optimal linear control of single-input nonlinear systems. In: American Control Conference, pp. 1981–1983 (1987)
134. Bourdache-Siguérididjane, H., Fliess, M.: Optimal feedback control of non-linear systems. *Automatica* **23**(3), 365–372 (1987)
135. Rotella, F., Dauphin-Tanguy, G.: Non-linear systems: identification and optimal control. *Int. J. Control* **48**(2), 525–544 (1988)
136. Faibusovich, L.E.: Explicitly solvable non-linear optimal control problems. *Int. J. Control* **48**(6), 2507–2526 (1988)
137. Yoshida, T., Loparo, K.A.: Quadratic regulatory theory for analytic non-linear systems with additive controls. *Automatica* **25**(4), 531–544 (1989)
138. Rajpurohit, T., Haddad, W.M.: Nonlinear-nonquadratic optimal and inverse optimal control for stochastic dynamical systems. *Int. J. Rob. Nonlinear Control* **27**(18), 4723–4751 (2017)

139. Evans, E.N., Haddad, W.M., Theodorou, E.A.: A fixed-architecture framework for stochastic nonlinear controller synthesis. In: Proceedings of the American Control Conference, Milwaukee, WI, pp. 4694–4699 (2018)
140. Agarwal, R., Lakshmikantham, V.: Uniqueness and Nonuniqueness Criteria for Ordinary Differential Equations. World Scientific, Singapore (1993)
141. Filippov, A.: Differential Equations with Discontinuous Right-Hand Sides. Mathematics and its Applications (Soviet Series). Kluwer Academic Publishers, Dordrecht (1988)
142. Kawski, M.: Stabilization of nonlinear systems in the plane. *Syst. Control Lett.* **12**, 169–175 (1989)
143. Yoshizawa, T.: Stability Theory by Liapunov's Second Methods. Publications of the Mathematical society of Japan. Mathematical Society of Japan, Tokyo, Japan (1966)
144. Yamada, T., Watanabe, S.: On the uniqueness of solutions of stochastic differential equations. *J. Math. Kyoto Univ.* **11**(1), 155–167 (1971)
145. Watanabe, S., Yamada, T.: On the uniqueness of solutions of stochastic differential equations II. *J. Math. Kyoto Univ.* **11**(3), 553–563 (1971)
146. Bhat, S.P., Bernstein, D.S.: Finite-time stability of continuous autonomous systems. *SIAM J. Control Opt.* **38**(3), 751–766 (2000)
147. Bhat, S.P., Bernstein, D.S.: Geometric homogeneity with applications to finite-time stability. *Math. Control Signals Syst.* **17**(2), 101–127 (2005)
148. Haimo, V.: Finite time controllers. *SIAM J. Control Opt.* **24**(4), 760–770 (1986)
149. Bhat, S., Bernstein, D.: Continuous finite-time stabilization of the translational and rotational double integrators. *IEEE Trans. Autom. Control* **43**(5), 678–682 (1998)
150. Hong, Y.: Finite-time stabilization and stabilizability of a class of controllable systems. *Syst. Control Lett.* **46**(4), 231–236 (2002)
151. Hong, Y., Huang, J., Xu, Y.: On an output feedback finite-time stabilization problem. *IEEE Trans. Autom. Control* **46**(2), 305–309 (2001)
152. Qian, C., Lin, W.: A continuous feedback approach to global strong stabilization of nonlinear systems. *IEEE Trans. Autom. Control* **46**(7), 1061–1079 (2001)
153. Fuller, A.T.: Optimization of some non-linear control systems by means of Bellman's equation and dimensional analysis. *Int. J. Control* **3**(4), 359–394 (1966)
154. Ryan, E.P.: Singular optimal controls for second-order saturating systems. *Int. J. Control* **30**(4), 549–564 (1979)
155. Ryan, E.: Finite-time stabilization of uncertain nonlinear planar systems. In: Skowronski, J., Flashner, H., Guttalu, R. (eds.) Mechanics and Control, vol. 151, pp. 406–414. Springer, Berlin (1991)
156. Salehi, S.V., Ryan, E.P.: Optimal non-linear feedback regulation of spacecraft angular momentum. *Opt. Control Appl. Methods* **5**(2), 101–110 (1984)
157. Haddad, W.M., L'Afliitto, A.: Finite-time stabilization and optimal feedback control. *IEEE Trans. Autom. Control* **61**, 1069–1074 (2016)
158. Rajpurohit, T., Haddad, W.M.: Stochastic finite-time partial stability, partial-state stabilization, and finite-time optimal feedback control. *Math. Control Signals Syst.* **29**(2), 1–39 (2017)
159. Florchinger, P.: Lyapunov-like techniques for stochastic stability. *SIAM J. Control Opt.* **33**(4), 1151–1169 (1995)
160. Florchinger, P.: Feedback stabilization of affine in the control stochastic differential systems by the control Lyapunov function method. *SIAM J. Control Opt.* **35**(2), 500–511 (1997)
161. Chabour, R., Oumoun, M.: On a universal formula for the stabilization of control stochastic nonlinear systems. *Stoc. Anal. Appl.* **17**(3), 359–368 (1999)
162. Haddad, W.M., Jin, X.: Universal feedback controllers and inverse optimality for nonlinear stochastic systems. *ASME J. Dyn. Syst. Meas. Control* **142**(021003), 1–10 (2020)
163. Haddad, W.M., Jin, X.: Implications of dissipativity, inverse optimal control, and stability margins for nonlinear stochastic regulators. *Int. J. Robust Nonlinear Control.* **29**, 5499–5519 (2019)
164. Haddad, W.M., Hui, Q., Chellaboina, V.:  $\mathcal{H}_2$  optimal semistable control for linear dynamical systems: an LMI approach. *J. Franklin Inst.* **348**, 2898–2910 (2011)

165. Hui, Q.: Optimal semistable control for continuous-time linear systems. *Syst. Control Lett.* **60**(4), 278–284 (2011)
166. Werbos, P.J.: Approximate dynamic programming for real-time control and neural modeling. In: White, D.A., Sofge, D.A. (eds.) *Handbook of Intelligent Control*, pp. 423–525. Van Nostrand Reinhold, New York (1992)
167. Bertsekas, D.P., Tsitsiklis, J.N.: *Neuro-Dynamic Programming*. Athena Scientific, Boston (1996)
168. Sutton, R.S., Barto, A.G.: *Reinforcement Learning-An Introduction*. MIT Press, Cambridge (1998)
169. Lewis, F.L., Vrabie, D., Vamvoudakis, K.G.: Reinforcement learning and feedback control: using natural decision methods to design optimal adaptive controllers. *IEEE Control Syst. Mag.* **6**, 76–105 (2012)
170. Kiumarsi, B., Vamvoudakis, K.G., Modares, H., Lewis, F.L.: Optimal and autonomous control using reinforcement learning: a survey. *IEEE Trans. Neural Netw. Learn. Syst.* **6**, 2042–2062 (2018)
171. Lewis, F.L., Vrabie, D.: Reinforcement learning and adaptive dynamic programming for feedback control. *IEEE Circuits Syst. Mag.* **3**, 32–50 (2009)
172. Parrish, J., Hamner, W.: *Animal Groups in Three Dimensions*. Cambridge University Press, Cambridge (1997)
173. Ben-Jacob, E., Cohen, I., Levine, H.: Cooperative self-organization of microorganisms. *Adv. Phys.* **49**, 395–554 (2000)
174. Camazine, S., Deneubourg, J., Franks, N., Sneyd, J., Theraulaz, G., Bonabeau, E.: *Self-Organization in Biological Systems*. Princeton University Press, Princeton (2001)
175. Okubo, A., Levin, S.: *Diffusion and Ecological Problems*, 2nd edn. Springer, New York (2001)
176. Kondepudi, D., Prigogine, I.: *Modern Thermodynamics: From Heat Engines to Dissipative Structures*. Wiley, Chichester (1998)
177. Bhat, S.P., Bernstein, D.S.: Arc-length-based Lyapunov tests for convergence and stability in systems having a continuum of equilibria. In: *Proceedings of the American Control Conference*, Denver, CO, pp. 2961–2966 (2003)
178. Shannon, C.E.: A mathematical theory of communication. *Bell Syst. Technol. J.* **27**, 379–423, 623–657 (1948)
179. Bode, H.W.: *Network Analysis and Feedback Amplifier Design*. D. Van Norstrand, Princeton (1945)
180. Freudenberg, J.S., Looze, D.P.: Right half plane poles and zeros and design tradeoffs in feedback systems. *IEEE Trans. Autom. Control* **30**, 555–565 (1985)
181. Seron, M.M., Braslavsky, J.H., Kokotovic, P.V., Mayne, D.Q.: Feedback limitations in nonlinear systems: from Bode integral to cheap control. *IEEE Trans. Autom. Control* **44**, 829–833 (1999)
182. Zang, G., Iglesias, P.A.: Nonlinear extension of Bode's integral based on an information-theoretic interpretation. *Syst. Control Lett.* **50**, 11–19 (2003)
183. Mehta, P.G., Vaidya, U., Banaszuk, A.: Markov chains, entropy, and fundamental limitations in nonlinear stabilization. *IEEE Trans. Autom. Control* **53**, 784–791 (2008)
184. Shannon, C.E., Weaver, W.: *The Mathematical Theory of Communication*. University of Illinois Press, Urbana (1949)
185. Byrnes, C.I., Isidori, A., Willems, J.C.: Passivity, feedback equivalence, and global stabilization of minimum phase nonlinear systems. *IEEE Trans. Autom. Control* **36**, 1228–1240 (1991)
186. Willems, J.C.: Dissipative dynamical systems part II: linear systems with quadratic supply rates. *Arch. Ration. Mech. Anal.* **45**, 352–393 (1972)
187. Hill, D.J., Moylan, P.J.: Stability results of nonlinear feedback systems. *Automatica* **13**, 377–382 (1977)

# Chapter 26

## Quantum Amplitude Amplification for Reinforcement Learning



K. Rajagopal, Q. Zhang, S. N. Balakrishnan, P. Fakhari, and J. R. Busemeyer

**Abstract** Reinforcement learning models require a choice rule for assigning probabilities to actions during learning. This chapter reviews work on a new choice rule based on an amplitude amplification algorithm originally developed in quantum computing. The basic theoretical ideas for amplitude amplification are reviewed, as well as four different simulation applications to a task with a predator learning to catch a prey in a grid world, and one application to human learning in a 4 arm bandit choice task. The applications reviewed in this chapter demonstrate that QRL can improve speed and robustness of learning compared to standard choice rules such as epsilon-greedy and softmax choice rules.

### 26.1 Exploration and Exploitation in Reinforcement Learning

Researchers across the world are racing to build quantum computers because of their great potential to speed up computational processing. For example, the Shor algorithm [18] can give an exponential speedup for factoring large integers into prime numbers, and the Grover algorithm [7] can achieve quadratic speedup in search on a quantum computer as compared to classical algorithms on a classical computer. Quantum walks perform a diffuse search at a rate of time squared as opposed to the classical rate of diffusion that is proportional to time [11]. These improvements in computational speed depend on the construction of quantum computers of sufficient size to make a practical advantage over classical computers. However, some researchers are beginning to examine the use of quantum algorithms on classical computers to improve computational speed and performance over classical algo-

---

K. Rajagopal · Q. Zhang · S. N. Balakrishnan (✉)  
Missouri University of Science and Technology, Rolla, MO, United States  
e-mail: [bala@mst.edu](mailto:bala@mst.edu)

P. Fakhari · J. R. Busemeyer  
Indiana University, Bloomington, IN, United States  
e-mail: [jbusemey@indiana.edu](mailto:jbusemey@indiana.edu)

rithms [17]. The purpose of this chapter is to explore the application of an algorithm related to the Grover algorithm to reinforcement learning.

Generally speaking, reinforcement learning (RL) algorithms have two separable parts: one is the learning algorithm that updates the future rewards expected to be produced by taking an action from a current state that leads to a new state (e.g., the Q-value for a state and action); and the second is the probabilistic choice rule for selecting an action given a state [21]. The latter moderates what is often called the exploration/exploitation properties of a RL algorithm. It is critical for efficient learning of optimal solutions to have the appropriate balance of exploration and exploitation. This chapter is primarily concerned with the type of probabilistic choice rule used in reinforcement learning. Two of the most commonly used types are the epsilon-greedy choice rule and the logistic (soft max) choice rule [21].

This chapter reviews recent work examining a new kind of choice rule based on a well-known quantum probability search algorithm called amplitude amplification [10], which is a generalization of the Grover algorithm. This algorithm was originally adapted for reinforcement learning [3, 4] and we [2, 5] developed an improved version of the quantum algorithm for reinforcement learning. Simulation studies have demonstrated faster learning using the amplitude amplification algorithm as compared to the epsilon-greedy [3, 4] and the soft max [5] choice rules. Very recently, empirical evidence has found that a learning model based on amplitude amplification predicts human learning on a 4 arm bandit task better than traditional models using a soft max type of rule [14]. The purpose of this chapter is to (a) briefly introduce quantum probability theory, (b) review the quantum reinforcement learning algorithm, and (c) review simulation work that has been used to establish its faster and more efficient learning properties.

## 26.2 Quantum Probability Theory

Before presenting the amplitude amplification algorithm, it is useful to first describe some of the basic principles of quantum probability theory [8]. Although quantum probability was originally designed for physics, recently the mathematical principles have been applied outside of physics to human judgment and decision-making [2, 12], information retrieval [16, 22], artificial intelligence and machine learning [15], social and economic science [9, 24] and finance [1].

Classical probability theory evolved over several centuries, however, an axiomatic foundation was first developed by Kolmogorov [13]. Kolmogorov theory is founded on the premise that events are represented as subsets of a larger set called the sample space. Quantum mechanics was invented by a brilliant group of physicists in the 1920, which revolutionized our world. However, not until von Neumann [23] provided an axiomatic foundation did physicists realize that they actually invented an entirely new theory of probability. Quantum probability is founded on the premise that events are represented as subspaces of a vector space (called a Hilbert space, which is a complete inner product vector space defined on a complex field). Each subspace of a vector

space corresponds to a projector that projects vectors onto the subspace. The basic axioms of quantum probability can be listed as follows:

1. An event  $A$  is represented by a subspace in the Hilbert space, which corresponds to a projector  $\mathcal{P}_A$  for event  $A$ , where  $\mathcal{P}_A = \mathcal{P}_A^\dagger = \mathcal{P}_A \cdot \mathcal{P}_A$ . For example, an event is the choice of an action, and a projector will later be used to represent the choice of an action.
2. The state of a system is represented as unit length vector, symbolized as  $|\psi\rangle$ , in the Hilbert space. For example, this state vector will later represent the potentials for an agent to take each of several actions.
3. The probability of an event  $A$  equals the squared projection:  $p(A) = \|\mathcal{P}_A \cdot |\psi\rangle\|^2$ . For example, this is how we will later compute the probability of choosing an action.
4. If two events,  $A, B$ , are mutually exclusive, then  $\mathcal{P}_A \mathcal{P}_B = 0$ , and the projector for the event that either  $A$  or  $B$  occurs is  $\mathcal{P}_A + \mathcal{P}_B$ , and probability of either event is the sum:  $p(A \vee B) = \|(\mathcal{P}_A + \mathcal{P}_B) \cdot |\psi\rangle\|^2 = \|\mathcal{P}_A \cdot |\psi\rangle\|^2 + \|\mathcal{P}_B \cdot |\psi\rangle\|^2$ .
5. If event  $A$  is known to occur, then the conditional probability of event  $B$  equals  $p(B|A) = \frac{\|\mathcal{P}_B \cdot \mathcal{P}_A \cdot |\psi\rangle\|^2}{\|\mathcal{P}_A \cdot |\psi\rangle\|^2}$ .

As noted above in item 4, the quantum algorithm is an additive measure: that is, it is a probability measure  $p$  such that if  $A, B$  are mutually exclusive events, then  $p(A \text{ or } B) = p(A) + p(B)$ . More important, any algorithm for assigning probabilities to subspaces greater than two dimensions in a vector space that satisfy an additive measure will turn out to be equivalent to the quantum algorithm [6].

The beauty of using a vector space is that there are an infinite number of bases that can be used to represent the state and the subspaces. Once a basis is selected, the state can be expressed in terms of that basis. Suppose the Hilbert space has dimension  $N$ , with a basis  $\{|V_j\rangle, j = 1, \dots, N\}$ . Then we can expand  $|\psi\rangle$  in this basis as follows  $|\psi\rangle = \sum \psi_j |V_j\rangle$ . In other words, we can represent the state vector  $|\psi\rangle$  as the  $N \times 1$  matrix  $\psi$  with coordinate  $\psi_j$  corresponding to basis vector  $|V_j\rangle$ . We can also represent the projector  $\mathcal{P}_A$  in this same basis by an  $N \times N$  matrix  $P_A$ . For example, the event  $A$  could be a one-dimensional subspace (a ray) spanned by  $V_3$  and the projector for this event is the outer product  $V_3 \cdot V_3^\dagger$ .

We might, however, find it necessary to represent the event  $B$  in a different basis  $\{|W_j\rangle = \mathcal{U} \cdot |V_j\rangle, j = 1, \dots, N\}$ , where  $\mathcal{U}$  is a unitary operator  $\mathcal{U}^\dagger \cdot \mathcal{U} = \mathcal{I}$ . In this case, the events may not commute,  $P_A \cdot P_B - P_B \cdot P_A \neq 0$ , in which case we say they are incompatible. For example, the event  $B$  could be a one-dimensional subspace (a ray) spanned by  $W_3$  and the projector for this event is the outer product  $W_3 \cdot W_3^\dagger$ . Note that  $V_3 \cdot V_3^\dagger$  does not commute with  $W_3 \cdot W_3^\dagger$ . If the events  $A, B$  are incompatible, then  $P_A \cdot P_B$  is not a projector, and  $p(A) \cdot p(B|A) \neq p(B) \cdot p(A|B)$ , and there is no joint probability for events  $A, B$ .

We can represent the unitary operator,  $\mathcal{U}$ , in the  $V$  basis by another  $N \times N$  matrix  $U$ . Note that the unitary transformation preserves lengths and inner products. Later, the Grover algorithm is expressed by a unitary transformation.

So far we have only described the structural part of quantum probability theory. The dynamic part of quantum theory forms another important part. For example, quantum walks use the dynamic part of the theory. Essentially, the dynamics describe how the unitary transformation changes with time. However, for the application considered here, we only need the structural part of the theory, and so we will not present the dynamic part.

### 26.3 The Original Quantum Reinforcement Learning (QRL) Algorithm

Consider the popular Q-learning algorithm. Define the estimate of the expected discounted future rewards for each state and action at time  $t$  as  $Q(s_i, a_k, t)$ . Suppose the last action taken at time  $t$  changed the state from  $s_i$  to state  $s_{i'}$  and the immediate reward  $r(t)$  was obtained. Then the new estimate for each state is updated according to

$$Q(s_i, a_k, t + 1) = (1 - \eta) \cdot Q(s_i, a_k, t) + \eta \cdot \left[ r(t) + \gamma \cdot \max_l Q(s_{i'}, a_l, t) \right], \quad (26.1)$$

where  $\eta$  is a learning rate parameter and  $\gamma$  is a discount rate parameter. The term in square brackets is the Q-learning “reward” signal. Then the next action is probabilistically selected based on its expected future reward value. Two commonly used methods to probabilistically choose an action are the epsilon-greedy rule and the softmax rule. The epsilon-greedy rule selects the best action with probability  $(1 - \epsilon)$ ,  $0 < \epsilon < 1$ , and otherwise randomly selects an action. The softmax rule selects action  $k$  with probability  $p(A_k) = e^{Q_k/\tau} / \sum e^{Q_j/\tau}$ , where  $\tau$  is called the temperature.

Quantum reinforcement works as follows. At any given state of the environment, suppose the agent has  $m$  possible actions. Then the dimension of the Hilbert space is set equal to  $m$ , and the basis is chosen such that each orthonormal basis vector represents an action. The state can then be represented by a  $m \times 1$  unit length state vector denoted  $\psi$ , with a coordinate (called an amplitude)  $\psi_k$  corresponding to the basis vector for action  $a_k$ . The projector for an action can be represented by a  $m \times m$  diagonal indicator matrix that simply picks out the coordinate corresponding to the action. Finally, the probability of taking action  $a_k$  simply equals  $|\psi_k|^2$ . The key new idea is the rule for amplifying the amplitudes  $\psi_k$  based on the expected future value of an action  $a_k$  from the current state  $s_i$  after some number of training trials  $t$  denoted  $Q(s_i, a_k, t)$ .

The amplitude amplification algorithm is an extension by Brassard and Hoyer [10] of Grover’s [7] search algorithm. The algorithm begins with an initial amplitude distribution represented by the  $m \times 1$  matrix  $\psi_0$ , with  $\psi_k = \frac{1}{\sqrt{m}}$ . Define  $\psi_1$  as the  $m \times 1$  matrix of amplitudes after experiencing amplification on a trial.

Suppose action  $a_j$  was chosen on the last trial. In the original QRL algorithm [3, 4], the amplitude for action  $a_j$  is amplified or attenuated in proportion to the reward signal  $[r(t) + \gamma \cdot \max_l Q(s_{i'}, a_l, t)]$  experienced by taking that action. This is done as follows.

Define  $A_k$  as an  $m \times 1$  matrix with zeros in every row except the row  $k$  corresponding to action  $a_k$ , which is set equal to one. Next define the following two unitary matrices:

$$\begin{aligned} U_1 &= I - (1 - e^{i\phi_1}) \cdot (A_k \cdot A_k^\dagger), \\ U_2 &= (1 - e^{i\phi_2}) \cdot (\psi_0 \cdot \psi_0^\dagger) - I, \end{aligned} \quad (26.2)$$

where  $\phi_1, \phi_2$  are two learning parameters that control the amount of amplification or attenuation. The matrix  $Q_1$  flips the sign of the target action, and the matrix  $Q_2$  inverts all the amplitudes around the average amplitude, and together these to act to amplify the target while having no effect (except normalization) on the non targets. Then the new amplitude distribution is formed by

$$\psi_1 = (U_2 \cdot U_1)^L \cdot \psi_0, \quad (26.3)$$

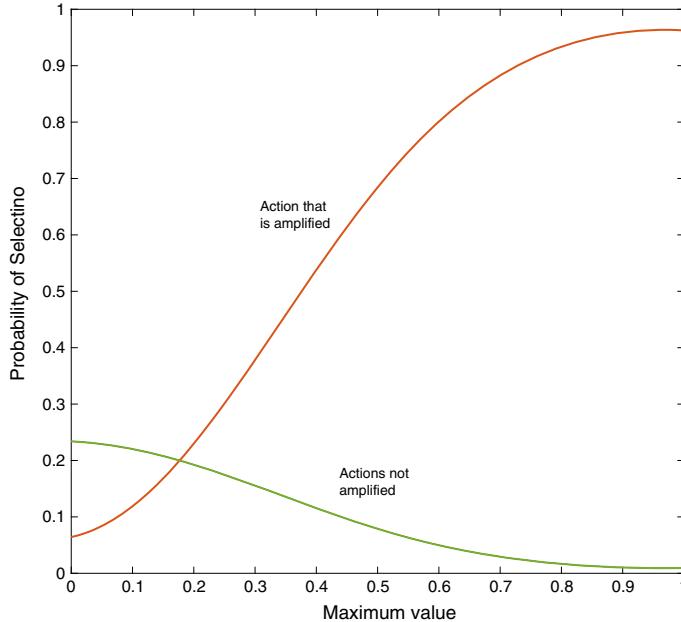
where the matrix power  $L$  indicates the integer number of applications of the update used on a single trial.

The original QRL algorithm fixed  $\phi_1 = \phi_2 = \pi \approx 3.1416$ , and related the reward  $[r(t) + \gamma \cdot \max_l Q(s_{i'}, a_l, t)]$  to  $L$  applied after each trial. The parameter  $L$  was set equal to the integer value of  $c \cdot [r(t) + \gamma \cdot \max_l Q(e, a_l, t)]$ , where  $c > 0$  is a free parameter [4]. This essentially produces the original Grover updating algorithm.

## 26.4 The Revised Quantum Reinforcement Learning Algorithm

The original QRL, based on relating the parameter  $L$  to rewards, restricts the algorithm to discrete jumps in amplitudes, and it is too restrictive for small numbers of actions, and it does not provide a good method for treating punishments. Another option first proposed by [2] and then improved in [5] was to set  $L = 1$  and vary  $\phi_1$  and  $\phi_2$  as described below

1. Find the action  $a_{max}$  corresponding to the maximum Q-value in the current state  $i$  and select it for amplitude amplification.
2. Find the set of next possible states, denoted  $S$ , from the current state  $i$ , and the next state  $i'$  produced by  $a_{max}$ , and then compute the ratio



**Fig. 26.1** Relation between choice probability for each action and normalized Q-value using parameters described in text

$$\phi = \frac{\max_l Q(s_{i'}, a_l, t)}{\max_{s'} \max_l Q(s_{i'}, a_l, t)} \quad (26.4)$$

If  $\max_{s'} \max_l Q(s_{i'}, a_l, t) = 0$  or  $\phi < 0$  then  $\phi = 0$ .

3. Set  $\phi_1 = \pi \cdot (a \cdot \phi + b)$  and  $\phi_2 = c \cdot \phi_1$ . We found  $a = 1.3$ ,  $b = 15.8$ , and  $c = 0.65$  to work well based on the relation between choice probability and normalized Q-value shown in Fig. 26.1.
4. Then apply Eqs. 26.1, 26.2 using  $\phi_1$ ,  $\phi_2$ , and  $L = 1$ .

## 26.5 Learning Rate and Performance Comparisons

The first investigation of quantum reinforcement learning (QRL) [4] conducted simulations of a predator-prey task using a 20 by 20 square grid world. Each cell represented a state with a start state in the upper left corner and a goal state in the lower right corner, and some cells designed as blocks that prevented movement. From any cell, the agent could move up, down, left, right. The reward for finding the goal was 100 points, and each step cost 1 point. This investigation used a TD(0) algorithm with

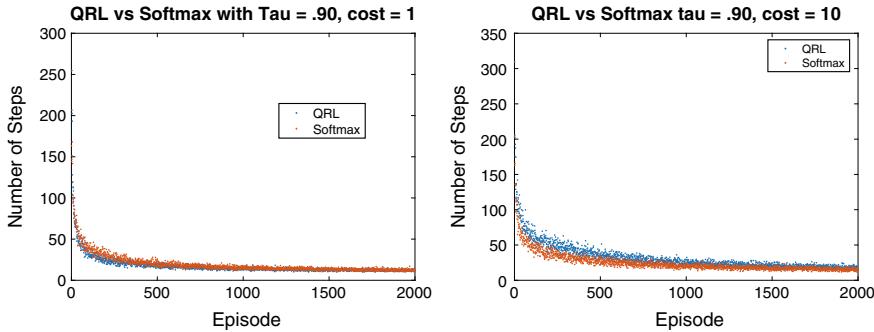
an epsilon-greedy choice rule  $\epsilon = 0.01$  and compared the QRL choice rule, using the original amplitude amplification rule. They varied the learning rate parameters of the TD(0) algorithm. Their simulation results revealed that generally the QRL rule explored more initially but learned to reach the goal faster than the epsilon-greedy algorithm based on the number of episodes required to reach the goal.

The second investigation of QRL [3] used collection of 58 states with an arrangement of permissible transitions to new states from a given state. Once again the agent started in some starting state and transitioned at a cost of 1 point each step to a goal state worth 100 points. In this investigation, the agent had to learn to find the goal with different starting positions. (Although it is not explicitly stated, we assume that an epsilon-greedy algorithm was again used for the traditional RL model, and the QRL was based on setting  $\phi_1 = \phi_2 = \pi$  and selecting  $L$  based on the reward signal). The simulation results showed that the QRL algorithm learned faster and was more robust than the traditional RL model when changing starting positions, and the results were robust across a range of learning rate parameters. The QRL and traditional RL models were also compared using a simulated robot navigation task where the goal was to avoid obstacles. The robot was endowed with 16 sensors to provide sensory data on the obstacles in the environment. Each step cost one point, reaching the goal produced 100 points, but hitting an obstacle stopped the episode and produced a loss of 100 points. Once again, the agent using the QRL algorithm learned faster to avoid obstacles than the traditional RL agent.

The third investigation of QRL [5] used simulations to investigate three different types of predator-prey tasks. The first task used a 20 by 20 square grid world with obstacles in which the goal was fixed throughout an episode (defined from start until goal capture or stop search in 4000 steps). The state was defined by a cell, and the predator had five actions (left, right, up, down, stay in place). The agent started in different states, and the reward for catching the goal was 100 with a cost of one point per step were examined. Both an epsilon-greedy ( $\epsilon = 0.01$ ) and a softmax rule ( $\tau = 0.9$ ) were examined. The QRL model was based on the revised amplitude amplification algorithm. They also simulated the same paradigm with softmax and quantum algorithms for different grid world sizes for 5 actions and 9 actions. The QRL model again produced faster learning than both the epsilon-greedy and softmax standard RL models.

The second task used a 10 by 10 grid world without obstacles. However, in this case both the predator and the prey started in arbitrary states and moved from step to step with a 100 point reward for catching the goal and a cost of one point per step. The prey moved randomly on each step. For this task, a state was defined by the vertical and horizontal steps needed to reach the prey. Again, the QRL model again produced faster learning than both the epsilon-greedy and softmax standard RL models.

Figure 26.2 (left panel) illustrates the learning results for the QRL and the softmax rule ( $\tau = 0.9$ ). A total 100 simulations were run, and each simulation contained 5000 episodes. The predator and the prey started in arbitrary states and both moved from step to step. For this task, a state was defined by the vertical and horizontal steps needed to reach the prey. The panel in the left of the figure shows the average over



**Fig. 26.2** Number of episodes required to catch prey for QRL and Softmax ( $\tau = 0.90$ )

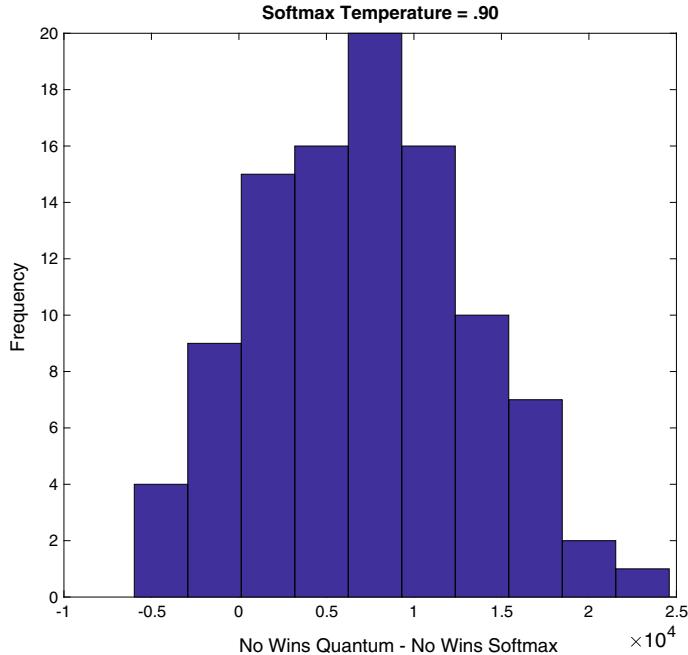
100 simulations of learning rates for the QRL and softmax ( $\tau = 0.9$ ) rule across 2000 episodes. Similar results are obtained with  $\tau = 0.5$  and 1.5.

The faster learning by the QRL compared to the softmax rule is not, however, universal. As shown in Fig. 26.2 right panel, if we increase the cost of each step from 1 to 10, then the softmax rule learns faster than the QRL rule. Nevertheless, as we next, the QRL still performs better in a competition with softmax after 20000 episodes of training.

Finally, a third task was investigated that directly pits a QRL agent against a softmax agent for chasing and reaching the prey. In this case, a 10 by 10 grid world was used, and both the predator and prey moved on each step (the prey moved randomly). For this task, a state was again defined by the vertical and horizontal steps needed to reach the prey. Catching the prey produced a reward equal to 100 points, and each step cost one point. Both agents were initially trained with just the prey for 20,000 episodes using either the softmax rule ( $\tau = 0.9$ ) or the revised amplitude amplification rule. After this training, the learning algorithm was turned off and the choice at each state was based on the maximum Q-value. Both agents started at the same position at the upper left corner, and both competed within the same episode to catch the prey. Each episode ended with either the QRL agent catching the prey first, the softmax agent catching the prey first, or both agents catching the prey at the same time. The results were that the QRL agent substantially beat the softmax agent to catch the moving prey.

We re-ran these simulations with different values for the temperature ( $\tau = 0.15, 0.2, 0.3, 0.4, 0.5, 0.6, 0.7, 0.8, 0.9, 1.5, 2.0$ ), which all produced similar results. Figure 26.3 shows the results produced by simulating 100 pairs of QRL and softmax agents ( $\tau = 0.9$ ). Each pair was tested with 20,000 preys (episodes with a different randomly generated prey for each episode). The performance measure was the number of times out of 20,000 that the QRL agent won minus the number of times the softmax agent won. As can be seen in the figure, the QRL agent most frequently beat the softmax agent by a substantial amount.

We also re-ran these simulations with different costs for each step (1, 5, and 10). Table 26.1 shows the quartiles of the distributions for a cost of 10 points per step. As



**Fig. 26.3** Frequency out of 100 pairs for each category representing the number of times QRL agent won minus number of times softmax agent won

**Table 26.1** Quartiles from 100 samples of the number of times QRL won, softmax ( $\tau = 0.90$ ) won, and ties, out of 20000 episodes for a cost of 10 points per step by predator

	QRL wins	Softmax wins	Choice rule tie
25%	1.7295	1.6252	0.9030
50%	1.9547	1.7742	1.1171
75%	2.2495	1.9612	1.5700

can be seen, the QRL distribution dominates the softmax distribution for number of wins. Similar results were obtained with costs equal to 1 and 5. It is interesting that even though the softmax learned faster with a cost of 10 points, the QRL performed better after training.

## 26.6 Other Applications of QRL

The amplitude amplification algorithm was also applied to fault detection and parameter identification for stochastic dynamic systems [25]. Detection of faults and adaptive estimation of parameters are crucial to successful control of vehicles. When a

fault occurs, the parameters of a system change and can change drastically; in order to cope with such changes, a quantum-boost scheme to work with a multiple-model Kalman filter was developed [25]. The basic estimation method is based on a multiple-model Kalman filter. At each time step, the a priori estimation and covariance of the Kalman filters are weighted with the conditional probability of the parameter set, given the new measurement. The conditional probability is obtained by Bayesian inference. The conditional probability of the parameter set closest to the correct parameter set typically rises with new information and finally converges to one. In fault detection, it is of vital significance that such probability rise and convergence happen quickly. This can be achieved with a quantum amplitude amplification boost scheme. The conditional probability is updated twice at each time step, once with Bayesian inference and once with extended amplitude amplification algorithm. The probabilities of different parameter sets can be considered as weights for the parameter sets. The weighted estimation can be considered as a superposition state and the estimations of the sub-filters with different parameter sets as basis states. Then the probabilities are the square of the amplitude of the basis states. The amplitude amplification algorithm is suitable for probability update because the operator is unitary and hence the updated probabilities will sum up to 1. A proof to assure that the boost under certain conditions can accelerate the convergence of probabilities can be found in [26].

Consider the following linear system:

$$x_k = F_{k-1}x_{k-1} + G_{k-1}u_{k-1} + w_{k-1}, \quad (26.5)$$

$$y_k = H_k x_k + v_k, \quad (26.6)$$

$$w_k \sim N(0, Q_k), \quad (26.7)$$

$$v_k \sim N(0, R_k), \quad (26.8)$$

where the subscript  $k$  indicates the  $k$ 'th time step,  $x_k$  is an  $n$ -dimensional state vector,  $y_k$  is a  $q$ -dimensional measurement vector,  $u_k$  is an  $m$ -dimensional control input,  $w_k$  is an  $n$ -dimensional system noise vector with a covariance given by  $Q_k$ ;  $v_k$  is  $q$ -dimensional measurement noise vector with a covariance of  $R_k$ . All noise sequences are assumed normal.  $G$  and  $H$  are appropriately dimensioned matrices. Parameter vector  $p$  is defined as the set  $(F, G)$ .

For derivations of a typical multiple-model Kalman filter, the reader is referred to [19, 20].

The quantum-boost multiple-model Kalman filter algorithm is given as follows:

Input: Initialize probabilities for parameter  $p_j$  given measurement  $\Pr(p_j | y_0)$  ( $j = 1, \dots, N$ ), the initial a posteriori state estimate,  $\hat{x}_{0j}^-$  and the corresponding a posteriori covariance,  $P_{0j}^-$ .

At each time step  $k > 0$ , perform the following steps:

1. Bayesian inference for multiple-model Kalman filter

- (a) Run  $N$  Kalman filters for each  $p_j$ . The a priori state estimate and covariance (indicated with minus sign in superscript) of  $j$ th filter at step  $k$  are computed from a posteriori state estimate and covariance (indicated with plus sign in superscript) at step  $k - 1$ :

$$\hat{x}_{kj}^- = F_{k-1,j} \hat{x}_{k-1}^+ \quad (26.9)$$

$$P_{kj}^- = F_{k-1,j} P_{k-1}^+ F_{k-1,j}^T + Q_{k-1}. \quad (26.10)$$

- (b) After the  $k$ th measurement, approximate  $\text{pdf}(y_k | p_j)$  is given by

$$\text{pdf}(y_k | p_j) \approx \frac{\exp(-r_k^T S_k^{-1} r_k / 2)}{(2\pi)^{q/2} |S_k|^{1/2}}, \quad (26.11)$$

where  $r_k = y_k - H_k \hat{x}_{kj}^-$  is the residual,  $S_k = H_k P_{kj}^- H_k^T + R_k$  is the covariance,  $q$  is number of measure and  $\text{pdf}(\cdot)$  refers to the probability density function of  $(\cdot)$ .

- (c) Estimate the probability that  $p = p_j$  as follows:

$$\Pr(p_j | y_k) = \frac{\text{pdf}(y_k | p_j) \Pr(p_j | y_{k-1})}{\sum_{i=1}^N \text{pdf}(y_k | p_i) \Pr(p_i | y_{k-1})}, \quad (26.12)$$

where  $\Pr(\cdot)$  represents the probability of  $(\cdot)$ .

2. Quantum boost for multiple-model Kalman filter

- (a) Prepare the following superposition state

$$|\psi\rangle = \sum_{j=1}^N \sqrt{\Pr(p_j | y_k)} |j\rangle, \quad (26.13)$$

where the canonical basis  $|j\rangle$  is simply an  $N \times 1$  vector where all elements are zero except the  $j$ th element is one, and  $|\psi\rangle$  is a superposition state.

- (b) Find  $p_j$  with greatest probability rise

$$J = \text{argmax}_j (\Pr(p_j | y_k) - \Pr(p_j | y_{k-1})). \quad (26.14)$$

- (c) Select  $|J\rangle$  as the qubit to be boosted.

- (d) Calculate the unitary extended Grover operator  $G = U_2 U_1$  according to Eq. 26.2.

(e)  $|\psi\rangle \leftarrow G |\psi\rangle$ .

(f)  $\Pr(p_j | y_k) \leftarrow |A_J|^2$  where  $A_J$  is the complex coefficient of  $|J\rangle$ .

3. Combine the sub-filters into the multiple-model Kalman filter

- (a) Weight each  $\hat{x}_{kj}^-$  and  $P_{kj}^-$  accordingly to obtain

$$\hat{x}_k^- = \sum_{j=1}^N \Pr(p_j | y_k) \hat{x}_{kj}^- \quad (26.15)$$

$$P_k^- = \sum_{j=1}^N \Pr(p_j | y_k) P_{kj}^- \quad (26.16)$$

(b) Estimate the a posteriori state estimate and covariance as

$$\hat{x}_k^+ = \hat{x}_k^- + K_k(y_k - H_k \hat{x}_k^-) \quad (26.17)$$

$$P_k^+ = (I - K_k H_k) P_k^- \quad (26.18)$$

The Grover update is implemented only once at each time step. We use  $\phi_1 = \pi\phi$  and  $\phi_2 = 0.15\phi_1$  so that the probability increases monotonically as  $\phi$  increases [5].  $\phi$  is carefully chosen so that the probability is increased while not causing divergence.

### 26.6.1 Example

Efficacy of the quantum-boosted multiple-model Kalman filter was tested with two examples. In both examples, the quantum-boost scheme was seen to accelerate the rise and convergence of the unknown initial parameter and also the changed parameter. The system equation and measurement equation of the first example (harmonic oscillator) are as follows [19, 20].

$$\dot{x} = Ax + Bw_1 = \begin{bmatrix} 0 & 1 \\ -\omega_n^2 & -2\xi\omega_n \end{bmatrix}x + \begin{bmatrix} 0 \\ \omega_n^2 \end{bmatrix}w_1 \quad (26.19)$$

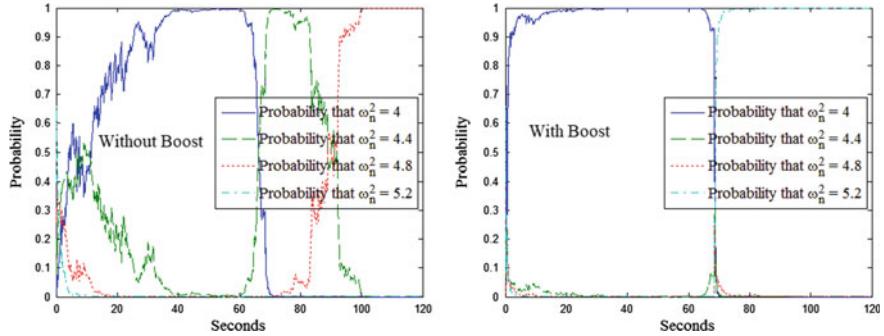
$$y_k = \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix}x_k + v_k \quad (26.20)$$

$$w_1 \sim N(0, Q_c) \quad (26.21)$$

$$v_k \sim N(0, R). \quad (26.22)$$

The damping ratio  $\zeta = 0.1$ . Process and measurement noise covariance are set at  $Q_c = 1000$  and  $R = 10I$ , where  $I$  is a 2–2 identity matrix.

Value of the natural frequency  $\omega_n$  is assumed unknown to the multiple-model filter. The objective is to find the true natural frequency  $\omega_n$  of the system in Eq. 26.19. In this example, four Kalman filters are carried with different values of  $\omega_n^2$ : 4.0, 4.4, 4.8, and 5.2 with a priori probabilities  $\Pr(\omega_n^2 = 4.0) = 0.1$ ,  $\Pr(\omega_n^2 = 4.4) = 0.2$ ,  $\Pr(\omega_n^2 = 4.8) = 0.3$ , and  $\Pr(\omega_n^2 = 5.2) = 0.4$ .



**Fig. 26.4** Histories of the probabilities of different  $\omega_n^2$

In the system equation,  $\omega_n$  retains its true normal value of 2.0 ( $\omega_n^2 = 4.0$ ) when  $t = 0 \sim 60$  s and a fault is assumed to happen changing  $\omega_n$  to an abnormal value ( $\omega_n^2 = 5.6$ ) at  $t = 60$  s.

Results are presented where the simulations were carried out with a multiple-model filter without a quantum boost and with a quantum boost and their relative performance in parameter estimation and fault detection area compared.

Histories of the probabilities of different  $\omega_n^2$  in the first example of harmonic oscillator ( $\omega_n$  is natural frequency) are shown in Fig. 26.4. Note that without the quantum boost,  $\text{Pr}(\omega_n^2 = 4)$  (the probability of the correct  $\omega_n^2$ ) converges to 1 in 40 s. With the quantum boost,  $\text{Pr}(\omega_n^2 = 4)$  converges to 1 in 20 s. In the case of fault detection, there is a fault at  $t = 60$  s and  $\omega_n^2$  jumps to 5.6; since the right value is not among the ones assumed, the best that can happen is to reach the value closest to the right value, which in this  $\omega_n^2 = 5.2$ . In the simulations without the quantum boost,  $\text{Pr}(\omega_n^2 = 5.2)$  does not show any increase in the next 60 s of the simulation; With the quantum boost,  $\text{Pr}(\omega_n^2 = 5.2)$ , however, converges to 1 in just 15 s after the fault has occurred. Note that by switching the parameter values around the newly converged value and carrying the same number of filters, we can converge to the true value. This is shown in an upcoming paper. With quantum-boosted multiple-model Kalman filter, fast fault detection can be achieved by monitoring the probabilities of the assumed values of the parameters.

## 26.7 Application to Human Learning

Recently, a QRL model was applied to human decision-making with 100 participants trained for 180 trials on a 4 arm bandit type of reinforcement learning task [14]. Two different QRL model were compared with twelve traditional RL models based on the trial by trial behavioral choices. In addition, the relation between estimated model parameters and functional magnetic resonance imaging (fMRI) was examined. The QRL models performed well compared with the best traditional RL models. The

internal state representation from the QRL model was also related to fMRI brain activity measures in the medial frontal gyrus. This was the first study to relate quantum decision processes to neural substrates of human decision-making.

## 26.8 Concluding Comments

The trade-off between exploration and exploitation is critical for reinforcement learning theory, which is an issue concerning the assignment of probabilities to actions during learning. This chapter reviews work on a new method based on an amplitude amplification algorithm originally developed in quantum computing. The new method is called quantum reinforcement learning, however, it still uses traditional learning algorithms, such as Q-learning, and only modifies the choice rule for selecting actions based on the learned expectations. The applications reviewed in this chapter demonstrate that QRL can improve speed and robustness of learning compared to standard choice rules such as epsilon-greedy and softmax choice rules. However, more work is needed to investigate the range of learning situations where this advantage is found, and also more work is needed to compare other types of choice rules (such as, for example, simulated annealing). Finally, amplitude amplification employs only a very small part of quantum computing tools, and other quantum applications, such as, for example, quantum walks for modeling flow of information in networks [17], may provide additional engineering contributions.

## References

1. Shor, P.W.: Algorithms for quantum computation: discrete logarithms and factoring. In: Proceedings 35th Annual Symposium on Foundations of Computer Science, pp. 124–134. IEEE, Santa Fe, NM (1994)
2. Grover, L.K.: Quantum mechanics helps in searching for a needle in a haystack. Phys. Rev. Lett. **79**(2), 325–327 (1997)
3. Kempe, Julia: Quantum random walks: an introductory overview. Contemp. Phys. **44**(4), 307–327 (2003)
4. Sánchez-Burillo, E., Duch, J., Gómez-Gardenes, J., Zueco, D.: Quantum navigation and ranking in complex networks. Sci. Rep. **2**, 605 (2012)
5. Sutton, R., Barto, A.G.: Reinforcement learning: an introduction. MIT Press, Cambridge (1998)
6. Hoyer, P.: Arbitrary phases in quantum amplitude amplification. Phys. Rev. A **62**, 052304–1–052304–5 (2000)
7. Dong, D., Chen, C., Chu, J., Tarn, T.J.: Robust quantum inspired reinforcement learning for robot navigation. IEEE/ASME Trans. Mechatron. **17**, 1–12 (2010)
8. Dong, D., Chen, C., Li, H., Tarn, T.J.: Quantum reinforcement learning. IEEE Trans. Syst. Man Cybernet. B: Cybernet. **38**(5), 1207–1220 (2008)
9. Busemeyer, J.R., Bruza, P.D.: Quantum models of cognition and decision. Cambridge University Press, Cambidge (2012)
10. Rajagopal, K., Balakrishnan, S.N., Busemeyer, J.R., Fakhari, P.: Quantum inspired reinforcement learning in changing environments. New Math. Nat. Comput.: Spec. Issue Eng. Mind Cogn. Sci. Robot. **9**(3), 273–294 (2013)

11. Li, J., Dong, D., Wei, Z., Liu, Y., Yu, P., Nori, F., Zhang, X.: Quantum reinforcement learning during human decision making. *Nat. Human Behav.* **4**(3), 294–307 (in press)
12. Gudder, S.P.: *Quantum Probability*. Academic Press, Cambridge (1988)
13. Khrennikov, A.Y.: *Ubiquitous Quantum Structure: From Psychology to Finance*. Springer, Berlin (2010)
14. Melucci, M.: *Introduction to information retrieval and quantum mechanics*. Springer, Berlin (2015)
15. Van Rijsbergen, K.: *The geometry of infomation retrieval*. Cambridge University Press, Cambridge (2004)
16. Wichert, A.M.: *Principles of quantum artificial intelligence*. World scientific, Singapore (2013)
17. Haven, E., Khrennikov, A.: *Quantum Social Science*. Cambirdge University Press, Cambirdge (2013)
18. Wendt, A.: *Quantum mind and social science*. Cambridge University Press, Cambridge (2015)
19. Baaquie, B.E.: Quantum mechanics and option pricing. In: *Proceedings of the Second Quantum Interaction Symposium (QI-2008)*, pp. 54–59 (2008)
20. Kolmogorov, A.N.: *Foundations of the theory of probability*. Chelsea Publishing Co., New York (1933/1950)
21. Von Neumann, J.: *Mathematical Foundations of Quantum Theory*. Princeton University Press, Princeton (1932/1955)
22. Gleason, A.M.: Measures on the closed subspaces of a hilbert space. *J. Math. Mech.* **6**, 885–893 (1957)
23. Zhang, Q., Balakrishnan, Sivasubramanya N., Busemeyer, J.: Fault detection and adaptive parameter estimation with quantum inspired techniques and multiple-model filters. In: *AIAA Guidance, Navigation, and Control Conference*, pp. 1124 (2018)
24. Qizi Z., Balakrishnan, S.N., Busemeyer, J.: Parameter estimation with quantum inspired techniques and adaptive multiple-model filters. In: *2018 Annual American Control Conference (ACC)*, pp. 925–930. IEEE, Milwaukee, WI (2018)
25. Simon, D.: *Optimal state estimation: Kalman, H infinity, and nonlinear approaches*. Wiley, Berlin (2006)
26. Stengel, R.F.: *Optimal control and estimation*. Courier Corporation, Chelmsford (1994)