

Efficient Bayesian Nonparametric Methods for
Model-Free Reinforcement Learning in Centralized
and Decentralized Sequential Environments

by

Miao Liu

Department of Electrical and Computer Engineering
Duke University

Date: _____

Approved:

Lawrence Carin, Supervisor

David Dunson

Michael Zavlanos

Galen Reeves

Ronald Parr

Dissertation submitted in partial fulfillment of the requirements for the degree of
Doctor of Philosophy in the Department of Electrical and Computer Engineering
in the Graduate School of Duke University

2014

ABSTRACT

Efficient Bayesian Nonparametric Methods for Model-Free
Reinforcement Learning in Centralized and Decentralized
Sequential Environments

by

Miao Liu

Department of Electrical and Computer Engineering
Duke University

Date: _____

Approved:

Lawrence Carin, Supervisor

David Dunson

Michael Zavlanos

Galen Reeves

Ronald Parr

An abstract of a dissertation submitted in partial fulfillment of the requirements for
the degree of Doctor of Philosophy in the Department of Electrical and Computer
Engineering
in the Graduate School of Duke University
2014

Copyright © 2014 by Miao Liu
All rights reserved except the rights granted by the
Creative Commons Attribution-Noncommercial Licence

Abstract

As a growing number of agents are deployed in complex environments for scientific research and human well-being, there are increasing demands for designing efficient learning algorithms for these agents to improve their control policies. Such policies must account for uncertainties, including those caused by environmental stochasticity, sensor noise and communication restrictions. These challenges exist in missions such as planetary navigation, forest firefighting, and underwater exploration. Ideally, good control policies should allow the agents to deal with all the situations in an environment and enable them to accomplish their mission within the budgeted time and resources. However, a correct model of the environment is not typically available in advance, requiring the policy to be learned from data. Model-free reinforcement learning (RL) is a promising candidate for agents to learn control policies while engaged in complex tasks, because it allows the control policies to be learned directly from a subset of experiences and with time efficiency. Moreover, to ensure persistent performance improvement for RL, it is important that the control policies be concisely represented based on existing knowledge, and have the flexibility to accommodate new experience. Bayesian nonparametric methods (BNPMs) both allow the complexity of models to be adaptive to data, and provide a principled way for discovering and representing new knowledge.

In this thesis, we investigate approaches for RL in centralized and decentralized sequential decision-making problems using BNPMs. We show how the control policies

can be learned efficiently under model-free RL schemes with BNPMs. Specifically, for centralized sequential decision-making, we study Q learning with Gaussian processes to solve Markov decision processes, and we also employ hierarchical Dirichlet processes as the prior for the control policy parameters to solve partially observable Markov decision processes. For decentralized partially observable Markov decision processes, we use stick-breaking processes as the prior for the controller of each agent. We develop efficient inference algorithms for learning the corresponding control policies. We demonstrate that by combining model-free RL and BNPMs with efficient algorithm design, we are able to scale up RL methods for complex problems that cannot be solved due to the lack of model knowledge. We adaptively learn control policies with concise structure and high value, from a relatively small amount of data.

Contents

Abstract	iv
List of Tables	x
List of Figures	xi
Acknowledgements	xiv
1 Introduction	1
1.1 Overview	1
1.2 Sequential Decision-making Models and Reinforcement Learning . . .	4
1.3 Overview of Relevant Work and Issues	8
1.4 Summary of Contributions	10
1.5 Dissertation Organization	12
2 Background	14
2.1 Centralized and Decentralized Sequential Decision Making Models . .	14
2.1.1 Markov Decision Processes	14
2.1.2 Partially Observable Markov Decision Processes	17
2.1.3 Decentralized Partially Observable Markov Decision Processes	29
2.2 Reinforcement Learning	32
2.2.1 Reinforcement Learning in MDPs	34
2.2.2 Reinforcement Learning in POMDPs	37
2.2.3 Reinforcement Learning in DEC-(PO)MDPs	44

2.3	Bayesian Nonparametric Methods: Models and Inference	46
2.3.1	Bayesian Nonparametric Models	47
2.3.2	Inference Methods	50
3	Q Learning with Gaussian Processes	51
3.1	Overview	51
3.2	Gaussian Processes	53
3.2.1	Sequential GP Updates with Sparsification	54
3.3	Off-Policy RL with a GP	55
3.4	GPQ Algorithm and Convergence Analysis	55
3.4.1	Batch GP-Fitted Q-Iteration	56
3.4.2	Online learning with GPQ	59
3.4.3	Optimistic Exploration for GPQ	60
3.5	Experiments	61
3.6	Discussion	64
3.7	Appendix	64
3.7.1	Details of Empirical Results	64
3.7.2	Sequential GP Updates	68
3.7.3	Proof of Theorem 7	70
4	Online Expectation Maximization for Reinforcement Learning in POMDPs	72
4.1	Overview	72
4.2	Batch-mode Expectation Maximization	74
4.3	Online Nested Expectation Maximization	75
4.3.1	Time and Memory Complexity	79
4.3.2	Convergence Analysis	80
4.4	Experiments	82

4.5	Discussion	90
4.6	Appendix	91
5	The Infinite Regionalized Policy Representation for POMDPs	95
5.1	Bayesian Policy Learning	97
5.2	The Infinite RPR	99
5.3	Exploration vs Exploitation in iRPR	102
5.4	Theoretical Analysis of the Relation Between Exploration and Optimality	105
5.5	Experiments	106
5.5.1	Effects of History Information and u_1	107
5.5.2	Performance Comparisons	109
5.6	Discussion	109
5.7	Appendix	110
6	Policy Learning for Decentralized POMDPs with Bayesian Non-parametric Priors	112
6.1	Overview	112
6.2	Policy Representations for infinite horizon DEC-POMDPs	114
6.2.1	Policy Learning by Direct Value Maximization	115
6.3	Bayesian Learning of Fixed-size FSCs	116
6.3.1	Policy Posterior under Dirichlet Priors	118
6.4	Bayesian Learning of Variable-sized FSCs	119
6.4.1	Computational complexity	122
6.4.2	Tradeoff between Exploration and Exploitation	123
6.4.3	Theoretical Analysis of Exploration and Optimality	125
6.5	Prior parameter selection and initialization	126
6.6	Experiments	127

6.7	Discussion	132
6.8	Appendix	133
6.8.1	Proof of Theorem 16	133
6.8.2	(Mean-field) variational Bayesian Inference for DEC-SBPR	134
6.8.3	Some Basics of Stick-breaking Priors	139
7	Conclusions	142
7.1	Summary of Contributions	143
7.2	Future work	145
7.2.1	Optimal exploration strategies for GPQ	145
7.2.2	Temporal and spatial(state) abstraction and BNPMs	146
7.2.3	Speed up BNPMs	147
7.2.4	Unified view of Model-free RL for (DEC)-POMDPs	148
7.2.5	Domains with continuous states, observations and actions	149
7.2.6	Transfer Learning	149
	Bibliography	151
	Biography	163

List of Tables

4.1	Five benchmark POMDP problems	84
4.2	A comparison, in terms of averaged reward, to U-trees on small problems. The parenthesized integers indicate the numbers of nodes used in U-trees, which correspond to the numbers of decision states $ Z $ for the RPR. . . .	88
4.3	A comparison of online EM-RPR to iPOMDP and Policy Prior (PP), in terms of cumulative reward.	88
6.1	Computational Complexity of Algorithm 5.	123
6.2	Results for DEC-POMDP benchmark problems comparing DEC-SBPR and other state-of-art algorithms. The numbers in the table are values and running times for each algorithm, with higher value indicating better performance. $ Z $ is the averaged controller size inferred by DEC-SBPR.	129

List of Figures

1.1	Relationships among learning, planning, and acting, adapted from [132].	6
1.2	The relationship among different sequential decision-making models, adapted from [14].	7
2.1	A simple MDP model: Graphical representation (left) and Domain illustration (right).	15
2.2	The graphical representation of time series generated from a POMDP.	19
2.3	A 3-step policy tree for POMDP policy representation ($ \mathcal{O} = 3$). Each node is labeled with the action that should be taken if it is reached. . .	23
2.4	The DBN of a POMDP with a standard FSC.	27
2.5	A simple DEC-POMDP model: Graphical representation(left) and Domain illustration (right).	30
2.6	The planning and learning framework for POMDPs	38
3.1	The maximum error $\ \hat{Q} - Q^*\ $ is plotted for GP-FQI with insufficient regularization $\omega_n^2 = 0.1$ and sufficient regularization $\omega_n^2 = 1$	57
3.2	Average sum of discounted rewards for the experimental domains. The GQ and the QL variants are given more information because their bases are specified a priori, yet GPQ is able to reach comparable performance (often faster) while choosing its own basis functions. . .	62
3.3	The performance of GPQ with ϵ -greedy exploration on Baird’s counterexample (the “Star” problem) which can cause divergence with fixed-basis linear function approximation.	65

3.4	The performance for Inverted Pendulum under different budget (quantization levels), 36 (left) and 100 (right). GPQ is not sensitive to the quantization level because it selects its own bases, while the quantization level impacts the other algorithms significantly. For example, with higher a quantization level GQ converges slowly, and with a lower quantization level QL-tab performances poorly.	66
3.5	The performance of the algorithms in Puddleworld. The bandwidth is set to be 0.1 for the basis function in all methods. The number of basis (budget) is set to be 400. FA-FB and GQ diverge when the bases are placed uniformly over the state space.	67
3.6	The performance for Puddleworld under different budget (quantization levels).	68
4.1	Performance Comparison between online and batch EM-RPR learning algorithms. Left column and right column correspond to averaged accumulative CPU time and reward respectively. The rows from top to bottom correspond to 5 benchmark problems; for all of the latter, the horizontal axis corresponds to the number of episodes seen by the agent when developing the policy.	83
4.2	The results on Hallway (top two rows) and Tag (bottom two rows) produced by the online EM-RPR with various settings of $ \mathcal{Z} $	86
4.3	A comparison of online EM-RPR to EM-MDP on Hallway. $ h $ is the number of observations. The performance of the EM-MDP is reported only for $n = 200 \times 10^3$	89
5.1	The iRPR's performance on the noisy 1D maze: (top right) relative reward (bottom left) exploration rate (bottom right) $ \mathcal{Z} $	106
5.2	Performance comparison between iRPR, RPR, iPOMDP on Hallway .	108
5.3	The probability density function(PDF) and cumulative distribution function (CDF) of Beta distribution with different setting of parameters.	111
6.1	An illustration of DSBPR and its initialization for two agents. Left: sample trajectories; Right: the initial policy graphs.	127
6.2	A demonstration of convergence and optimality of Algorithm 5 on the Marsrover problem.	128
6.3	Performance on the traffic control domain with 10^{20} states and 100 agents. Top: Domain illustration; Bottom left: test reward; Bottom right: Inferred decision state number.	132

6.4 Left: The ratio between two gamma functions $\frac{\Gamma(x+a)}{\Gamma(x)}$ and its lower and upper bounds. Right: The absolute difference between $\frac{\Gamma(x+a)}{\Gamma(x)}$ and its lower and upper bound. 141

Acknowledgements

There are a number of people to whom I have my uttermost appreciation in my pursuit of Ph.D. at Duke. First and foremost, it is my advisor Larry. Your supervision has been so instrumental in shaping my interest and pushing forward my research. Your vision, help and support have been an invaluable asset for my study. To Xuejun, thanks for your generous guidance and many insightful discussions during the last four years which directly contribute to my research. To Ron, thank you for teaching me MDPs and opening the door of reinforcement learning for me, as well as providing constructive criticisms on my coursework. To other members in my committee, including David, Michael and Galen, thanks for your time and advice.

I also need to thank Jon who not only personally provided insightful suggestions for my research, but by welcoming me into his research group (ACL) at MIT, where I spent the past two summers exploring Bayesian nonparametric methods and reinforcement learning. Since then, I have been collaborating dynamically with those awesome people at MIT including Trevor, Girish, Tom, Rob, Alborz and Chris. These experiences broadened my research horizon and enriched my knowledge in a way that would not have been possible otherwise.

In no uncertain terms, I am in debt to my friends and family. Mom and Dad thank you for giving me a sound body and mind, taking care of my life well when I am too busy with coursework and research, and celebrating with me when making breakthroughs. To friends from Duke - Minhua, John, Eric, Mingyuan, Lingbo, Zheng-

ming, Xianxing, Yingjian, Ray, Shaobo, Qisong, Mingtao, Bo, Xuefei, Nathaniel, Esther, David, Kyle, Wenzhao, Changwei, Jose, Isabella and countless more - thanks for your help and sharing with me your knowledge and many funny moments. To my friends at MIT - Beipeng, Kemal, Luke, Buddy, Dan, Mark, Bobby, Jack, Sam, Susie and everyone else-thanks for making me feel at home when I was in Boston.

Last, but definitely not least, I thank my wonderful girlfriend Leslie. Your love and encouragement are the foundations where I stand.

Introduction

1.1 Overview

Developing efficient methods for learning and planning under uncertainty has been a long-standing research focus in artificial intelligence and machine learning. Such research aims to help a decision maker to better explore a domain and understand both its dynamics and utilities, so that a good strategy can be planned for him or her to accomplish a mission. In a strategic planning or sequential decision-making problem, the decision maker is called an agent, which can be a human or machine. Consider a recycling robot domain, for example. The agent is a robot who works in an office building. The robot has to move back and forth from a charger station to different rooms. The planning problem in this example is to prescribe an optimal course of moving directions for the agent so that it can finish the trash-collecting task within a given amount time and without depleting batteries.

Solving a planning problem would be easy if the world were simple and deterministic. However, real life is complex and stochastic, so an agent has to envision various uncertain factors in order to come up with a high quality strategic plan. One type of uncertainty an agent has to cope with is the stochasticity of world dynamics. In

the recycling robot domain, a motor may fail or the floor may be slippery. Without considering these uncertain factors, the robot might not be able to reach the intended location within a given amount of time. Another source of uncertainty which can confound an agent's decision-making is the noisy observation of the state information, including the agent's battery level and location. For example, the battery level might be incorrectly read because of the malfunction of a voltmeter, causing the agent to fail at recharging the battery on time. Meanwhile, an agent's physical location might be inaccurately estimated because of imperfect sensors, limited computation power, as well as the complexity of the environment (for example, occlusions can hide some areas from sensing). Furthermore, when there are multiple agents working together in a decentralized fashion, there is uncertainty about the choices of the other agents and their local observations. Having to reason about each other's plans and observations, this situation further complicates the process of obtaining an optimal plan for each individual.

Besides robotic control, there are many other applications for single agent sequential decision-making under uncertainty, such as spoken dialogue management [157], inventory management [50] and homecare [108]. In a spoken dialogue management system, human users are allowed to interact with a machine using voice, and the task of an agent (machine) is to infer the goal of the intended user from both the user's input and the record of the dialogue history. The dialogue cycles continuously until either the user's goal is satisfied or the dialogue fails. In inventory management, a wholesale company needs to determine the ordering plan for every selling period, so that the customers' demand can be satisfied and the company's profit can be maximized in the long run. To achieve the goal, the company has to consider the dynamic demands of customers and contemporary inventory levels, and it also needs to track the seasonal variation of customer demands. Many other applications are surveyed by White [151] and Cassandra [29].

Besides handling the uncertainty inherited from single agent cases, decision makers in multiagent problems must cope with the uncertainty over the action choices and resultant observations of others. In cooperative and decentralized settings, the agents share the same objective function; however, the plan of each agent has to be developed based only on local observations¹. Due to bandwidth limits, budgets and other physical constraints, many real-world applications require decentralized learning and planning. The potential applications include domains such as Marsrover [2] and RoboCup rescue [94]. In the Marsrover domain, the distributed rovers conduct scientific experiments at different locations on Mars. Due to high cost and slow speed of communication with a central station located on Earth, these rovers must operate autonomously based on their own locations and partial view of the Mars surface, without constant online communication. This process has to be optimized so that the greatest scientific value of a mission on Mars can be realized within the given budget and time constraints. Similarly, in the decentralized rescue domain, the agents, including teams of firefighters, police officers and ambulances, have to make decisions based on local information. This information includes estimation of how fast the fire will spread and the location of other agents, so that the casualties and building damage can be minimized.

In order to obtain a high-quality plan which is capable of handling various uncertain factors, probabilistic models are often employed for describing the dynamics and utilities of a domain. If these models are available, a planner might be able to simulate all possible events and find an optimal strategy before executing a mission. However for many real-world problems, accurate models might not be available in advance. In this case, the agents have to collect samples through their interaction

¹ Here we emphasize the difference between decentralized decision-making and distributed computation. In distributed computation, the agents are allowed to communicate their information, whereas in decentralized decision-making, each agent executes its own policy based only on local data without explicit communicating to each other or a central authority.

with the domain, and learn the domain’s dynamics and utilities from trial and error, so that gradually, the value of their strategy can be improved. Such a learning process is data-driven and called reinforcement learning (RL), which is a stepping stone towards the goal of the agents learning by themselves to discover new knowledge and acquire higher levels of intelligence and autonomy.

1.2 Sequential Decision-making Models and Reinforcement Learning

In this thesis, we aim to design efficient RL algorithms for both centralized and decentralized sequential decision-making in stochastic environments. We will focus on three major models, namely Markov decision processes (MDPs) [112], partially observable Markov decision processes (POMDPs) [62], and decentralized partially observable Markov decision processes (DEC-POMDPs) [14]. We briefly introduce these models here, providing their formal definitions in the next section.

For single agent decision-making, when state information is fully observable, an MDP is a widely used framework. Essentially an MDP is a controlled Markov model, in which the state transitions satisfy the Markov property, i.e., at each time step, after the agent takes an action, the system transits to a new state and the transition is only dependent on the system’s current state, and independent of previous states and actions. The goal in an MDP is to maximize the cumulative reward or minimize the cumulative cost for a given number of decision steps, also called the horizon of a problem, which can be finite or infinite.

When the state information, such as the location of a robot, is partially observable, a POMDP serves as an expressive model for single agent decision-making. Essentially a POMDP is a controlled hidden Markov model. At each step, after taking an action, the agent receives a noisy observation of the state of the process. The observations no longer satisfy the Markov property, which makes POMDPs more complicated and more challenging to solve than MDPs. However, an observation

provides some information about the underlying states and can be used to update the belief state, which satisfies the Markov property. Therefore one can convert a POMDP into a belief-state MDP and use the special structure of value function to obtain a solution.

When multiple distributed agents are working collaboratively in partially observable stochastic domains with limited communication, a DEC-POMDP can be employed as a control model. In a DEC-POMDP, each agent receives its own local observation after executing its own action according to a local plan (decision rule). A global reward signal is generated to measure the immediate effect of the joint actions. Because of the decentralized nature, the global state cannot be calculated directly from local observations during policy execution. The goal of a DEC-POMDP is that the agents maximize the shared objective function while choosing actions based on local information.

Given the accurate mathematical model of domain dynamics, observation functions and utility functions, the process of searching an optimal strategy is called solving a planning problem [132]. When any one of these functions is unknown, an agent has to resort to RL techniques to learn an optimal decision rule. There are two categories of RL methods. One of them is first to obtain an estimate of the model from the agent's experiences, and then compute an (approximate) optimal policy for the estimated model, with this used as an approximation to the optimal policy for the true model. An alternative is to learn the policy directly from experiences, without the intermediate step of estimating the model. We refer to the former as a model-based approach or indirect RL and the latter as a model-free approach or direct RL. The interplay between experience, model and policy/value are depicted in Figure 1.1, where arrows indicate the direct impact and expected improvement. More details on RL will be reviewed in Chapter 2.

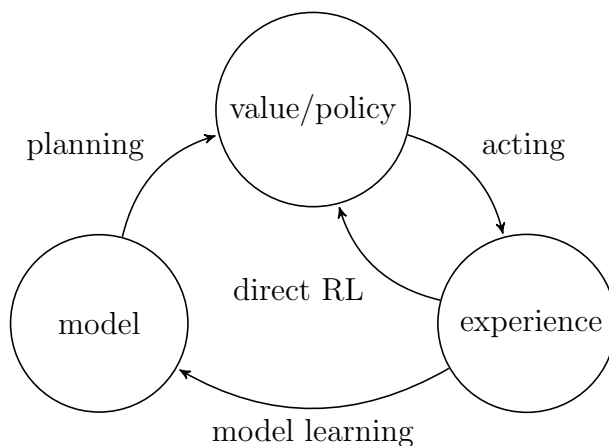


FIGURE 1.1: Relationships among learning, planning, and acting, adapted from [132].

Related disciplines The sequential decision-making models in this thesis are strongly related to classic control theory, operations research and game theory.

- Control theory

MDPs and (DEC-)POMDPs are closed-loop control models. The typical solution to these models is stochastic dynamic programming (value/policy iteration) which is essentially stochastic optimal control [120]. The main distinction between control theory and these sequential decision models is that control theory concerns more about tracking the underlying state space with controllability and stability analysis, whereas these sequential decision-making frameworks care more about making plans that can achieve long term goals by using tools, such as probabilistic inference and programming. In control theory, the problem of estimating the models of dynamical system is called system identification, whereas in (PO)MDPs, estimating the system dynamics is a task in model-based RL.

- Operations research

MDPs and (DEC-)POMDPs are extensively studied in operational research [112, 124, 14]. Operations research focus more on studying the methods for searching

the optimal control policy, usually assuming the model of a domain is given, whereas for RL, the agents have to solve the optimal policy based on the experiences (samples) from the agent-environment interactions.

- Game theory

Sequential decision-making and RL are also studied in the context of game theory [52]. A stochastic game can be viewed as a MDP when multiple agents can fully observe the world state. These agents can be cooperative or non-cooperative. Their joint actions determine the state transitions and rewards. The objective for each agent is to maximize the cumulative reward received during the game. When the underlying state is partially observable, the game is called partially observable stochastic game (POSG). When there is only one agent in the game, a POSG boils down to a POMDP. A DEC-POMDP is a special case of POSGs where all the players share the same payoff function. The relationship among the models of MDPs, POMDPs, DEC-POMDPs, and POSGs are depicted by the Venn diagram shown in Figure 1.2.

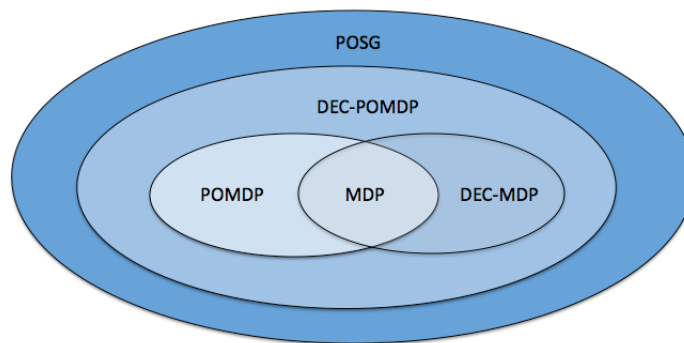


FIGURE 1.2: The relationship among different sequential decision-making models, adapted from [14].

1.3 Overview of Relevant Work and Issues

Exact and approximate solutions for MDPs can be obtained by dynamic programming or linear programming, when the MDP models are known [132, 112]. If the models are unknown or incomplete, we have to resort to RL techniques to obtain optimal control policies. The classical RL methods have been mainly developed for discrete space problems. However, many real problems have continuous state spaces where the methods developed for discrete space problems are not directly applicable. To address this issue, function approximation methods are often necessary in practice, and recent research has focused on both developing efficient function approximation methods for RL in continuous domains [42, 133, 87, 69, 79, 61, 32] and designing principled and efficient exploration strategies [130, 111, 77, 68, 48, 104].

Developing efficient algorithms for POMDPs is also a thriving research area where significant progress has been made in the last three decades. Amato [2] provided a survey of the exact and approximate solution methods before 2010. Because of the difficulty of obtaining the exact solution of POMDPs, recent research has focused on designing efficient approximate methods and scaling up existing solvers for problems with large spaces [39, 38, 76, 25, 81, 159, 118, 123, 127, 46, 83]. For instance, controller-based algorithms [76] combine the EM algorithm and the Monte Carlo simulation to achieve policy iteration; tree-based methods [123, 127] use the Monte Carlo tree search and heuristics to produce high-value solutions for large scale POMDP problems; Bayesian methods impose priors over policy parameters [38, 154] and are able to achieve efficient exploration [25]; moreover, methods based on Bayesian non-parametric methods allow both model learning [39] and policy learning [81] with the additional benefit of controlling the complexities of models and policies.

Compared to the maturity of the theoretical and technical developments of RL in MDPs and POMDPs, the research into RL in DEC-POMDPs is preliminary. Even

though a DEC-POMDP can be viewed as a POMDP controlled by multiple distributed agents, each acting based on local observations, many POMDP algorithms (especially value-iteration based) cannot be easily extended for DEC-POMDPs, because the decentralized nature of the problem prevents accurate calculation of the shared belief state during policy execution for each individual. Even though many open issues still remain, as surveyed by Amato [2], there were breakthroughs in the past decade for developing exact and approximate algorithms for DEC-POMDPs. The recent progress has mainly focused on developing efficient algorithms that can adapt to variable temporal scales and scale up to more realistic domains with large state space sizes and increased number of agents [71, 98, 10, 9, 156, 85, 6]. For example, finite state controller (FSC) based methods [71, 98, 156] are able to recast the planning problem as inference in a mixture of dynamic Bayesian networks, and use the EM algorithm to solve large problems and improve solution quality. In addition, macro-action based methods [6] retain both temporal extended actions and the coordination ability with memory bounded policy tree, while allowing near-optimal solutions to be generated for realistic problems [5]. Moreover, model-free RL methods [10, 9, 156] have also debuted for use in finite and infinite horizon problems, achieving comparable results to planning methods that use the complete model information.

Despite the significant progress for developing efficient RL algorithms for MDPs and (DEC)-POMDPs, current solvers, especially model-free methods for (DEC)-POMDPs, are still not sufficiently scalable to offer high-quality approximations to the optimal solution for real problems. Because MDPs are applicable for real physical systems with state information that is fully observable, it is desirable to consider safe exploration strategies, efficient feature selections and convergence guarantees to develop high-quality MDP-RL algorithms. These algorithms are important for increasing the agents' robustness and efficiency, while engaged in more complex mis-

sion scenarios. POMDP models are capable of expressing many real-world planning problems that involve partial observability, yet their solution methods are still far from mature. Being able to scale up current RL algorithms will transform POMDPs from theoretical models to practical tools for missions in complex and stochastic environments. Because the DEC-POMDP model is newer and more complex, and it is a challenge to solve the optimal policy for problems of desired size, a very limited number of studies have been considered DEC-POMDP RL. Adding learning ingredients to DEC-POMDPs will enable agents to acquire higher levels of autonomy for learning and planning. However, many open issues still exist, including the dilemma between exploration and exploitation, and the challenge of decentralized learning and coordination. Given the decentralized nature of the problem, no universal strategies can address these issues thoroughly.

1.4 Summary of Contributions

In order to address the existing issues in RL for sequential decision-making problems, we consider the flexibility of policy representations as a foundation. A policy is essentially a decision rule that maps the information state to actions. Depending on the types of problems, policies may have different representations. For example, in a discrete MDP, the policy is determined by a value function which is represented by a table, with each entry corresponding to the value of a state-action pair; in a continuous MDP, the value function is approximated by linear functions. In a discrete (DEC-)POMDP, the policy can be represented by a tree or graph. Under a particular form of policy representation, one needs to decide an appropriate number of parameters in such a representation, since the complexity of the representation affects both the policy value and the convergence rate. When the number of parameters is set too small, the parameters may be unable to represent the optimal policy and, therefore, will quickly converge to a sub-optimal policy. In contrast, when the number is set too

large, the policy representation is overly complex, often yielding slow convergence to a sub-optimal policy that overfits the agent’s trajectories. Bayesian nonparametric (BNP) models allow hierarchical modeling and data-driven inference, and provide powerful frameworks for reasoning about objects and relations in settings where these objects and relations are not predefined. This feature is particularly attractive for RL, because it is often difficult to correctly specify the complexity of a policy representation or the number of parameters *a priori*.

In this dissertation, we take a Bayesian nonparametric (BNP) point-of-view for policy representation and learning. We improve the state-of-the-art of model-free RL for MDPs, POMDPs and DEC-POMDPs in several ways. The contribution includes increasing the scalability of RL algorithms for these three decision-making models, as well as increasing their solution quality.

- **Efficient off-policy RL for MDPs based on Gaussian processes** We propose an off-policy RL method for learning the control policy for MDPs with Gaussian processes (GPs). We approximate the Q function with a GP regression and develop an efficient online learning algorithm, which allows sparse representation of Q function (with an upper limit to the number of parameters). Convergence is proved for both batch and online settings. This approach was first discussed by Girish *et al.* [31, 32].
- **Online learning for policy graph** We develop an online nested EM algorithm for model-free RL in a POMDP. The algorithm evaluates the policy only in the current learning episode, discarding the episode after the evaluation and memorizing the sufficient statistic, from which the policy is computed in closed form. As a result, the online algorithm has a time complexity $O(n)$ and a memory complexity $O(1)$, compared to $O(n^2)$ and $O(n)$ for the corresponding batch-mode algorithm, where n is the number of learning episodes. The online

algorithm has a provable convergence to a local optimal policy. This approach was first presented by Liu *et al.* [83]

- **Learning infinite policy graph: iRPR** We introduce the infinite regionalized policy representation (iRPR), as a nonparametric policy for reinforcement learning in POMDPs. The iRPR assumes an unbounded set of decision states *a priori*, and infers the number of states to represent the policy given the experiences. We propose algorithms for learning the number of decision states while maintaining a proper balance between exploration and exploitation. The analysis of the relation between the exploration rate and optimality is provided. This approach was first introduced by Liu *et al.* [81]
- **Policy learning for infinite horizon DEC-POMDPs** Recent work has shown that expectation maximization (EM) is an efficient algorithm for learning finite-state controllers (FSCs) in large decentralized domains. However, the solution quality of current methods is limited by their inability to define an appropriate controller size and the methods often converge to maxima that are far from optimal. We address this problem by using a variable-sized FSC to represent the local policy of each agent. The variable-sized FSC is constructed using the stick-breaking prior, leading to a new framework termed a decentralized stick-breaking policy representation (DEC-SBPR). This approach allows one to study the RL setting, and develop a variational Bayesian algorithm to infer the variable-size FSC.

1.5 Dissertation Organization

This dissertation is organized as follows. Chapter 2 reviews the basics of sequential decision making, reinforcement learning, and Bayesian nonparametric methods. Collectively, this information provides the background for developing our policy learning

methods and makes the thesis self-contained. We begin discussing the contribution of this dissertation in Chapter 3, which summarizes the convergence results of GP Q learning and presents its online implementation. Chapter 4 introduces an online learning algorithm for POMDPs. Chapter 5 presents a Bayesian nonparametric policy representation and learning method in POMDPs. Chapter 6 describes a stick-breaking policy learning approach for DEC-POMDPs. Chapter 7 concludes this dissertation by summarizing the main contributions and discussing possible future work.

2

Background

If I have seen further it is by standing on the shoulders of giants.

—Isaac Newton

The work in this dissertation is a marriage between two fields: reinforcement learning and Bayesian nonparametric statistics. In this chapter, we provide a brief overview of these two fields, as well as the essentials for solving the problems in the next four chapters.

2.1 Centralized and Decentralized Sequential Decision Making Models

2.1.1 Markov Decision Processes

An MDP is a sequential decision-making framework with states being fully perceivable and state dynamics satisfying the Markov property. Figure 2.1 pictorializes a simple navigation problem that can be modeled as an MDP. Mathematically, an

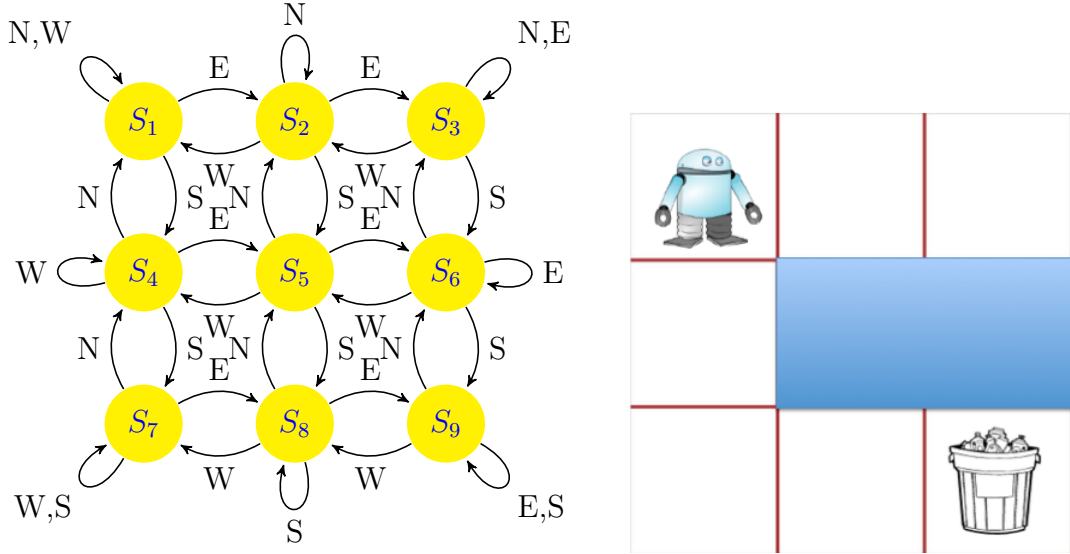


FIGURE 2.1: A simple MDP model: Graphical representation (left) and Domain illustration (right).

MDP is a quadruple $\mathcal{M}_{MDP} = \langle \mathcal{S}, \mathcal{A}, T, R \rangle$, with

- \mathcal{S} , the state space of the process. It manifests all the current information of a system, and is also called information state. \mathcal{S} can be finite and continuous. In the example of figure 2.1, the state space $\mathcal{S} = \{s_1, s_2, \dots, s_9\}$ represents the set of discretized locations on the gridworld, where an agent navigates.
- \mathcal{A} , the action space. The action space can be finite, countably infinite or continuous. For the example in figure 2.1, the agent can choose to move in any of the four directions or stay, so $\mathcal{A} = \{E, W, N, S, stay\}$. A sequence of actions are prescribed by a control policy for an agent to interact with an environment.
- T , the state transition function: $T_{s,a}^{s'} = Pr(s'|s, a)$ is the probability of transitioning to state s' after taking action a in state s , and it captures the uncertainty of the world dynamics under the influence of an agent's action. In our example, because of the slippery floor or the malfunction of actuator, the agent might not be able to reach the intended state.
- R , the reward function: $R(s, a, s')$ is the immediate reward for taking action a

in state s and transitioning to state s' . The reward is the feedback from the environment, which provides an immediate evaluation of the performance of agents at given states. The goal of an agent is to maximize the accumulated (long term) reward. In our example, a positive reward +100 is assigned for reaching the goal state (trash can), a large negative reward -100 for falling into a puddle, a -2 for any movement (energy consumption) and a -1 for stay (wasting time).

Given the MDP model, the objective is to find the control policy that maximizes the long term reward. A deterministic policy $\pi : \mathcal{S} \rightarrow \mathcal{A}$ is a mapping from states to the actions, *i.e.*, $a_t = \pi(s_t)$. Depending on the planning horizon (finite vs infinite) and performance criteria (averaged reward vs total reward), there are several ways to construct the value functions [112]. Here we consider infinite horizon planning problems, where the state-action value function or Q-function of each state-action pair under policy π is defined as

$$\begin{aligned}
 Q^\pi(s, a) &= \mathbb{E}_\pi \left[\sum_{t=0}^{\infty} \gamma^t r(s_t, s_{t+1}, a_t) \mid s_0 = s, a_0 = a \right] \\
 &= R(s, a) + \gamma \sum_{s'} Pr(s'|s, a) \sum_{a'} \pi(s', a') Q^\pi(a', s')
 \end{aligned} \tag{2.1}$$

which is the expected sum of discounted reward obtained by starting with taking action a in the initial state s and following policy π thereafter. In equation (2.1), $\gamma \in [0, 1)$, is a discount factor, which determines the importance of future rewards. The optimal value function Q^* satisfies $Q^* = \max_\pi Q^\pi(s, a)$ and is captured by the Bellman equation:

$$Q^*(s_t, a_t) = \mathbb{E}_{s_{t+1}} \left[r(s_t, a_t, s_{t+1}) + \max_{a'} \gamma Q^*(s_{t+1}, a') \right]. \tag{2.2}$$

The optimal policy for a given state s is $\pi^*(s) = \arg \max_a Q^*(s, a)$. When \mathcal{S} and \mathcal{A} are discrete, T and R are known, the optimal policy of an MDP can be solved by dynamic programming (value/policy iteration with fixed γ) or linear programming (turn the nonlinear max into a collection of linear constraints) within polynomial time [112]. When the state space is continuous, one has to use function approximations to value functions and apply approximate dynamic programming [24].

2.1.2 Partially Observable Markov Decision Processes

POMDPs are single-agent sequential decision-making frameworks with the state information partially perceivable [124]. Formally, a POMDP is a sextic-tuple: $\mathcal{M}_{POMDP} = \langle \mathcal{S}, \mathcal{A}, \mathcal{O}, T, \Omega, R \rangle$, where

- $\mathcal{S}, \mathcal{A}, T, R$ are the same as for MDPs
- \mathcal{O} denote a finite set of observations. It provides partial information about the underlying state. In our example (without using SLAM), the agent can observe whether there is a wall in front of it based on the input images from sensors, so $\mathcal{O} = \{\text{observe wall}, \text{observe no wall}\}$. In this case, the observation is aliased which means two different states can result the same observation.
- Ω are a finite set observation functions with $\Omega_{s',a}^o = P(o|a, s')$ denoting the probability of observing o after taking action a and transiting to state s' . Ω characterize the sensor noise. That is, different observations can be perceived in the same state.

Due to state aliasing and sensor noise, an observation only provides partial information about the world state, hence it cannot be directly used as the information state for planning. Instead, the observation provides some evidence of the current state and allows the uncertainty of the world state to be updated. The uncertainty of

the world state is characterized by the belief state, which is a probability distribution of the state s given the history of actions, observations, and the initial belief b_0 , *i.e.*,

$$b_t \stackrel{def.}{=} P(s_t | b_0, a_0, o_1, \dots, o_{t-1}, a_{t-1}, o_t). \quad (2.3)$$

It has been shown that the belief state is the sufficient statistic of observation history [124]. Assume at time t , the belief state $b_t = b$ and an action a is taken, resulting an observation o at time $t + 1$, then the agents can reason the belief state at time $t + 1$ by Bayes rule

$$b_{t+1}(s') = \frac{\Omega_{a,s'}^o \sum_{s \in \mathcal{S}} T_{s,a}^{s'} b_t(s)}{p(o|b, a)} \quad (2.4)$$

where

$$p(o|b, a) = \sum_{s' \in \mathcal{S}} \sum_{s \in \mathcal{S}} b_t(s) T_{s,a}^{s'} \Omega_{a,s'}^o \quad (2.5)$$

is the probability of transiting from b to b' when taking action a . The belief update (2.4) is similar to the forward algorithm in HMMs [114] and the Kalman filter used in continuous state space models [64], which is computationally efficient.

Figure 2.2 illustrates the time series from a POMDP, from which we can see the transitions of belief states satisfy the Markov property; that is, the belief state at time step $t + 1$, only depends on its previous belief states b_t , action a_t , and current observation o_t . This property implies that the belief state and the history of actions and observations provide the same information about the current world state.

Since the transition of belief states satisfies the Markov property, the POMDP can be formulated as a continuous space “belief MDP” by tracking the belief states without memorizing the whole action and observation history. Formally, the belief MDP can be defined as a tuple $\langle \mathcal{B}, \mathcal{A}, \tau, \rho \rangle$ [62], with

- \mathcal{B} , the set of belief states
- \mathcal{A} , the set of actions

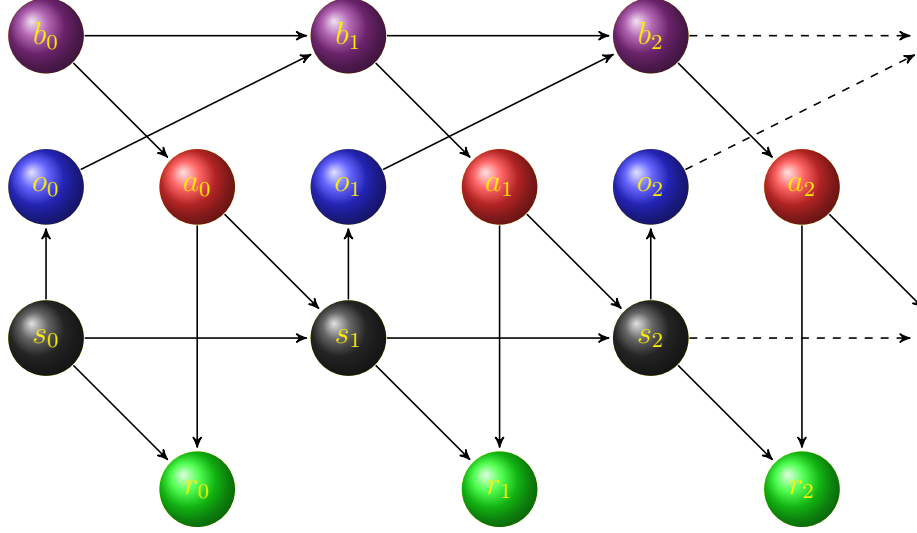


FIGURE 2.2: The graphical representation of time series generated from a POMDP.

- τ , the state-transition function: $\tau(b, a, b')$ is the probability of transitioning to state b' after taking action a in state b , defined as

$$\tau(b, a, b') = Pr(b'|b, a) = \sum_{o \in \mathcal{O}} Pr(b'|b, a, o)Pr(o|a, b) \quad (2.6)$$

where

$$Pr(b'|b, a, o) = \begin{cases} 1 & \text{if } b' = f(b, a, o) \\ 0, & \text{otherwise} \end{cases} \quad (2.7)$$

- ρ , the reward function on belief states given by

$$\rho(b, a) = \sum_{s \in \mathcal{S}} b(s)R(s, a) \quad (2.8)$$

The policy π in this case, is characterized by a value function, $V(b) : \Delta \rightarrow \mathbb{R}$, which is defined as the expected future discounted reward that the agent can receive starting at initial belief state b and with actions selected according to π thereafter.

$$V^\pi(b) = \mathbb{E}_\pi \left[\sum_{t=0}^{\infty} \gamma^t r(b_t, \pi(b_t)) | b_0 = b \right] \quad (2.9)$$

The value of an optimal policy π^* is defined by the optimal value function V^* , that satisfies the Bellman equation [124]

$$\begin{aligned} V^*(b) &= \max_{a \in \mathcal{A}} \left[R(b, a) + \gamma \sum_{b' \in \mathcal{B}} p(b'|b, a) V(b') \right] \\ &= \max_{a \in \mathcal{A}} \left[R(b, a) + \gamma \sum_{o \in \mathcal{O}} p(o|b, a) V(\tau(b, a, o)) \right]. \end{aligned} \tag{2.10}$$

The optimal policy can be found by solving the following problem

$$\pi^*(b) = \arg \max_{a \in \mathcal{A}} \left[R(b, a) + \gamma \sum_{o \in \mathcal{O}} p(o|b, a) V(\tau(b, a, o)) \right]. \tag{2.11}$$

Depending the number of horizons, the value function and policy may have different representations. For finite-horizon problems, the value function (2.10) is a piece-wise linear convex (PWLC) function [124], and π can be represented by policy trees with each hyper-plane corresponding to one subtree[62]. For infinite horizon problems, the value function (2.10) can be approximated by a PWLC function with arbitrary accuracy [128], and the policy is often represented by finite state controllers [62].

There are two categories of POMDP solvers, including value iteration (VI) and policy iteration (PI). They can be further classified into optimal solution methods and approximate methods. We summarize the basic algorithms in each category in the following subsections.

Value Iteration

First we summarize an exact value iteration (VI) method for solving finite horizon POMDPs. The main idea of this algorithm is that for a given optimal value function V_t and a belief point b , the optimal vector α_{t+1}^b representing V_{t+1} can be computed by back-projecting all vectors α_t in horizon t one step from the future and returning

the vector that maximizes the value of b , denoted as follows

$$\alpha_{t+1}^b = \arg \max_{\alpha \in G_{t+1}} \alpha^T b \quad (2.12)$$

which can be performed by taking a sequence of operations based on G_t , the set of vectors representing V_t . The main steps of these operations are summarized below.

First, calculate the initial value function:

$$\begin{aligned} V_1(b) &= \max_a \sum_{s \in \mathcal{S}} R(s, a) b(s) \\ &= \max_{\alpha \in G_1} \alpha^T b \end{aligned} \quad (2.13)$$

which is a PWLC function represented by a set of vectors G_1 . Then, use G_1 to construct the second step value function V_2 , which represents the expected future reward (since we have only two steps, the best value for the future is just what we obtained from the first step); hence using (2.4), we have

$$\begin{aligned} V_2(b) &= \max_{a \in \mathcal{A}} \left[R(b, a) + \gamma \sum_{o \in \mathcal{O}} P(o|a, b) V_1(\tau(b, a, o)) \right] \\ &= \max_{a \in \mathcal{A}} \left[R(b, a) + \gamma \sum_{o \in \mathcal{O}} p(o|a, b) \max_{\alpha \in G_1} \sum_{s'} \alpha(s') b(s') \right] \\ &= \max_{a \in \mathcal{A}} \left[R(b, a) + \gamma \sum_{o \in \mathcal{O}} \max_{\alpha \in G_1} \sum_{s'} p(o|a, b) \alpha(s') b(s') \right] \\ &= \max_{a \in \mathcal{A}} \left[R(b, a) + \gamma \sum_{o \in \mathcal{O}} \max_{\alpha \in G_1} \sum_{s'} \frac{\sum_s T_{s,a}^{s'} \Omega_{s',a}^o b(s)}{b(s')} \alpha(s') b(s') \right] \\ &= \max_{a \in \mathcal{A}} \left[R(b, a) + \gamma \sum_{o \in \mathcal{O}} \max_{\alpha \in G_1} \sum_{s'} \sum_s T_{s,a}^{s'} \Omega_{s',a}^o b(s) \alpha(s') \right] \end{aligned} \quad (2.14)$$

By induction, given that at the time step t , the value function is represented by a set of vectors $G_t = \{\alpha_0, \dots, \alpha_{|V_t|}\}$, the $t + 1$ horizon value function for the belief point b

is computed as

$$V_{t+1}(b) = \max_{a \in \mathcal{A}} \left[R(b, a) + \gamma \sum_{o \in \mathcal{O}} \max_{\alpha \in G_t} \sum_{s' \in \mathcal{S}} \sum_{s \in \mathcal{S}} T_{s,a}^{s'} \Omega_{a,s'}^o b(s) \alpha(s') \right], \quad (2.15)$$

and the corresponding set G_{t+1} can be determined through the following procedures

- Step 1: generate intermediate sets $G_{t+1}^{a,\star}$ and $G_{t+1}^{a,o}, \forall a \in \mathcal{A}, o \in \mathcal{O}$,

$$G_{t+1}^{a,\star} \leftarrow \alpha^{a,+}(s) = R(s, a) \quad (2.16)$$

$$G_{t+1}^{a,o} \leftarrow \alpha_i^{\alpha,o} = \gamma \sum_{s' \in \mathcal{S}} T_{s,a}^{s'} \Omega_{s',a}^o \alpha_i^t(s'), \forall \alpha_i \in G_t \quad (2.17)$$

- Step 2: combine expected future reward and immediate reward by summing the sets¹

$$G_{t+1}^a = G_{t+1}^{\star} + G_{t+1}^{a,o_1} \oplus G_{t+1}^{a,o_2} \oplus \dots \oplus G_{t+1}^{a,|\mathcal{O}|} \quad (2.18)$$

- Step 3: take the union of G_{t+1}^a

$$G_{t+1} = \cup_{a \in \mathcal{A}} G_{t+1}^a \quad (2.19)$$

The above procedure is called backup operation at horizon $t + 1$. The optimal value function V_{t+1}^* is extracted from

$$V_{t+1}(b) = \max_{\alpha \in G_{t+1}} \alpha \cdot b. \quad (2.20)$$

In this procedure, step 1 generates $|\mathcal{A}||\mathcal{O}||G_t|$ projections and step 2 produces $|\mathcal{A}||G_t|^{|\mathcal{O}|}$ cross-sums; hence without pruning dominated α vectors (the worst case), the $t + 1$ horizon value function requires

$$|G_{t+1}| = \mathcal{O}(|\mathcal{A}||G_t|^{|\mathcal{O}|}) \quad (2.21)$$

¹ the summation between two sets is denoted by cross sum \oplus , which is element-wise summation between two sets, e.g. let $A = \{a_1, \dots, a_n\}$ and $B = \{b_1, \dots, b_m\}$, then $A \oplus B = \{a_i + b_j | a_i \in A, b_j \in B\}$.

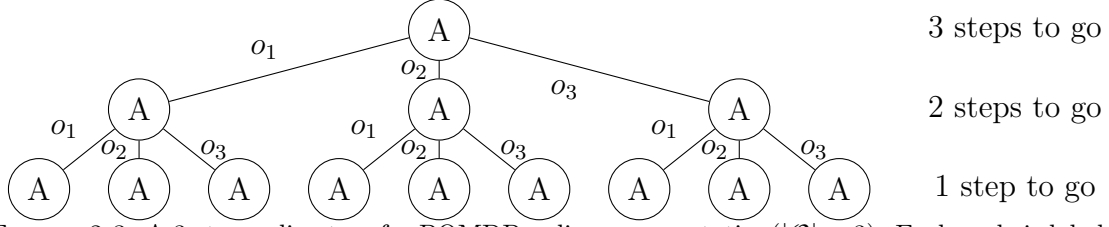


FIGURE 2.3: A 3-step policy tree for POMDP policy representation ($|\mathcal{O}| = 3$). Each node is labeled with the action that should be taken if it is reached.

memory and $O(|\mathcal{S}|^2|\mathcal{A}||G_t|^{|\mathcal{O}|})$ computation time.

To make value iteration more tractable, it is useful to remove the dominated α vectors by solving a linear program with $|\mathcal{S}|$ variables and $|G_t|$ constraints. This step is the main bottleneck of the application of POMDPs for realistic problems.

Another way to understand the VI algorithm is based on the process of building a policy tree. For a finite horizon problem (assume T steps), the policy can be represented by policy trees with depth T [62]. In a policy tree, nodes represent actions and subtrees correspond to different observations. This definition recurses until leaf nodes. Each tree corresponds to a hyper-plane in the belief state space. An example of a policy tree is shown in figure 2.3; there are $|\mathcal{A}|$ action choices at each node and $|\mathcal{O}|$ edges (observations) right below each action node. There are in total $|\mathcal{A}|^{\sum_{t=0}^{T-1} |\mathcal{O}|^t} = |\mathcal{A}|^{\frac{1-|\mathcal{O}|^T}{1-|\mathcal{O}|}}$ policy trees. Each branch of the tree represents an action-observation history. The size of policy trees grows double exponentially with the planning horizons. To bound the memory, we prune the policy trees by evaluating the value of each branch and eliminating those dominated or impossible police trees. It can be shown that the set of α -vectors G_T correspond to the policy trees with depth T . Pruning the dominated policy trees is equivalent to removing the dominated α -vectors in the VI algorithm.

Policy Iteration

For a POMDP, a value iteration method solves the optimal policy by searching in value function space and extracting the policy (represented by α vectors) directly from the value function, whereas a policy iteration method searches the policy from the policy space by alternating between policy evaluation and policy improvement. Policy iteration was first used by Sondik [128], and was improved by Hansen [51] where the policy is represented by an FSC. We briefly review these basic PI algorithm below.

It is shown by Sondik [128] that the optimal infinite horizon value function can be approximated arbitrarily closely by successive finite horizon value functions V_t , as $t \rightarrow \infty$. When the optimal policy can be represented by a PWLC function², the gradients (α vectors) of value function form a Markov partition of the belief space, *i.e.*, $\mathcal{B} = \cup_{i=1}^N \mathcal{B}_i$. Under the optimal action and resultant observation, all belief states in a region B_i are completely mapped into another set or itself. The partition and belief transitions constitutes a policy graph or finite state controller (FSC), where the nodes correspond to belief state partition with the associated optimal action, and transitions are guided by observations. Another way to understand the construction of FSCs is by considering the case that the finite horizon value functions V_t and V_{t+1} are equal, and at every step beyond t , the corresponding plans have the same value, one can redraw the edge from level t to itself, so the policy-tree branches can be arranged to form a stationary cyclic graph [62].

A finite-state controller (policy graph) Θ is a sex-tuple $\langle \mathcal{A}, \mathcal{O}, \mathcal{Z}, W, \mu, \pi \rangle$, where

- \mathcal{A} and \mathcal{O} are the same as for POMDPs;
- \mathcal{Z} is a finite set of nodes/decision-states;

² also called finitely transient deterministic policy

- W is a set of node-transition matrices, with $W_{z'}^{z,a,o}$ denoting the probability of transiting from z to z' when taking action a in z results in observation o :

$$W_{zao}^{z'} = Pr(Z^{t+1} = z' | Z^t = z, A^t = a, O^{t+1} = o) \quad (2.22)$$

- μ is the initial node distribution, with μ_z the probability of initially being in z ;
- π is a set of stochastic local policy, with π_z^a the probability of taking action a in z :

$$\pi_z^a = Pr(A^t = a | Z^t = z) \quad (2.23)$$

A policy iteration algorithm constitutes two steps: policy evaluation and policy improvement, which are summarized below

- Policy evaluation involves computing the value of every state-node pair $V(s, z)$ under current policy parameterized by Θ , *i.e.*,

$$V_{\Theta}(s, z) = R(s, a) + \gamma \sum_{s', o, z'} \Omega_{s', a}^o T_{s, a}^{s'} W_{z'}^{z, a, o} V_{\Theta}(s', z'), \forall s \in \mathcal{S}, z \in \mathcal{Z} \quad (2.24)$$

where $a \sim \pi^z$. The value of state-node pair (s, z) corresponds to a α -vector, *i.e.*, $V_{\Theta}(s, z) = \alpha(s)$. The time complexity of policy iteration is $O(|\mathcal{Z}|^2 |\mathcal{S}|^2)$.

- Policy improvement involves performing dynamic programming backup, in which the value function V^{π} represented by a finite set of α vectors G^{π} is transformed into an improved value function V' represented by another finite set of α -vectors G' . In the policy evaluation step, G^{π} is calculated from the FSC Θ using $V_{\Theta}(s, z)$ obtained in the policy evaluation step. Then the improved α -vector set G' is constructed from Θ based on the following rules

- For each vector $\alpha' \in G'$:

- * If α' has the same action and successor links as that is already in G , keep the same nodes in G' .
 - * if α' point-wisely dominates some nodes in G , then replace the nodes in G with a node corresponding to α' .
 - * else, add a node to G' that has the action and observation associated with α' .
- Prune any node in G that has no corresponding α -vector in G' if that cannot be reached from a node with an associated vector in G' .

Policy iteration usually converges more quickly than value iteration. However the dynamic programming is still a bottleneck for scaling up the algorithm to solve problems with large sizes. In the worst case, up to $|\mathcal{A}||\mathcal{Z}|^{|\mathcal{O}|}$ nodes may be added after each DP backup, hence the size of the controller quickly grows making many POMDPs intractable. Hansen’s policy is guaranteed to converge to the optimal policy, if the infinite horizon value function does have a finite representation (though it is still NP-hard to find the optimal FSC, even if the number of states is given [91]), otherwise, an ϵ -optimal FSC can be found in finite number of iterations.

Approximate methods

Searching the optimal plan for a POMDP is generally a very hard problem. It has been shown that finding the optimal solution for a finite horizon POMDP is PSPACE-hard [100], and verifying the existence of a policy whose value is greater than ϵ for an infinite horizon POMDP is undecidable [86]. The main reasons for the difficulty of obtaining the optimal policy for POMDPs are from two independent curses, namely the curse of dimension and the curse of history. To step down these curses, various efficient approximate planning methods have been proposed in the last decade. There are two categories of methods. One category of methods are point/region

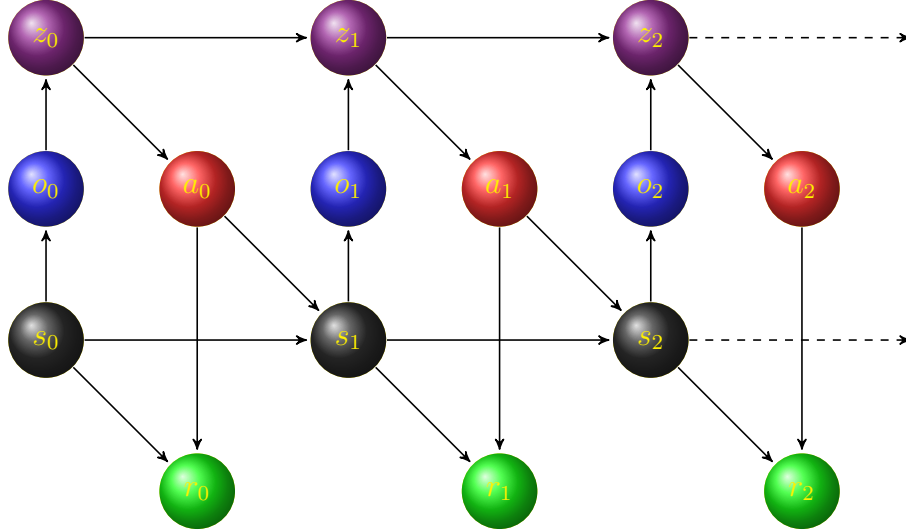


FIGURE 2.4: The DBN of a POMDP with a standard FSC.

based, including PBVI [106], HSVI [125], Persus [129], RBVI [75], and many others as surveyed by Pineau *et al.* [107]. These methods sample the belief states and then perform value iteration or policy iteration over the belief states/regions. They are differentiated by both the ways of selecting belief points and updating values. Approximate policy iteration methods include PBPI (point-based policy iteration) [58], BPI [109], gradient search [91] and EM-based methods [144]. PBPI (point-based policy iteration) [58] replaces the policy evaluation with PBVI and achieves monotonic increase of policy value. BPI(bounded policy iteration) [109], in which the policy improvement step comprises improving each node by solving linear programs with local escape by adding new nodes. The gradient search method [91] reformulates the task of finding optimal policy as a nonlinear optimization problem. The EM algorithm [144] realize policy iteration with the E-step corresponds to policy evaluation and the M-step corresponds to policy improvement.

Here, we describe the details of an EM algorithm introduced by Toussaint *et al.* [144] for solving the planning problem in POMDPs. This algorithm can be considered as a counter-part of our model-free RL methods to be introduced in the next section. The main idea of the method by Toussaint *et al.* [144] is to convert

a POMDP model to a dynamic Bayesian network (DBN) and reformulate a policy optimization problem as a maximum likelihood estimation problem.

Denote \hat{R} , a binary random variable, such that $Pr(\hat{R} = 1|A_t = a, S_t = s, T = t) = \frac{r(s,a)-r_{min}}{r_{max}-r_{min}}$, and T , a random variable representing the length of horizon, which is set to be geometrically distributed $Pr(T = t) = \gamma^t(1 - \gamma)$. The dynamic network in figure 2.4 can be considered as a mixture of finite time process with 0 – 1 reward R earned at the end of the process. Given the policy (FSC) represented by Θ , the value of the policy can be evaluated by computing the likelihood of $\hat{R} = 1$, *i.e.*,

$$\begin{aligned} L(\Theta) &= P(\hat{R} = 1|\Theta) = \sum_{T=0}^{\infty} p(T)p(\hat{R} = 1|T; \Theta) = (1 - \gamma) \sum_{T=0}^{\infty} \gamma^T \mathbb{E}[\hat{R}|T; \Theta] \\ &\propto \mathbb{E} \left[\sum_{T=0}^{\infty} \gamma^T r_t | \Theta \right] = V(\Theta) \end{aligned} \quad (2.25)$$

where $V(\Theta)$ is given by equation (2.24).

Given a finite process with length T , the sequence of state-node pairs constitute a Markov chain (Parr and Russell’s HAM theorem [102], given a POMDP and a finite policy graph). Therefore, the state-node transition can be computed by marginalizing the action and observation in the concatenation of state (node) transition and action (observation) emission probabilities, *i.e.*,

$$P(s', z'|s, z) = \sum_{a,o} \Omega_{a,s'}^o \pi_z^a T_{sa}^{s'} W_{z,a,o}^{z'} \quad (2.26)$$

In the E step, the distributions of hidden variables under the current estimation of Θ are computed by the recursive forward and backward computation

$$\alpha_t(s', z') = Pr(s_t = s', z_t = z|T; \Theta) = \sum_{s,z} P(s', z'|s, z) \alpha_{t-1}(s, z) \quad (2.27)$$

$$\beta_t(s, z) = Pr(\hat{R} = 1|s_{T-t} = s, z_{T-t} = z, T; \Theta) = \sum_{s',z'} P(s', z'|s, z) \beta_{t-1}(s', z') \quad (2.28)$$

with the initializations

$$\alpha_0(s, z) = \mu_z P_s, \quad \beta_0(s, z) = \sum_{a,o} \hat{R}(s, a) \pi_z^a \Omega_{s,a}^o \quad (2.29)$$

The results of E-step are

$$\hat{\alpha}(s, z) = \sum_{t=0}^{\infty} P(T = t) \alpha_t(s, z), \quad \hat{\beta} = \sum_{\tau=0}^{\infty} P(T = \tau) \beta_\tau(s, z). \quad (2.30)$$

We can understand $\hat{\alpha}(s, z)$ as the joint probability of visiting state s and node z , and $\hat{\beta}(s, z)$ as the joint value of state s and node z . The results of M-step are derived from maximization of the expected log-likelihood, *i.e.*,

$$\Theta^* = \arg \max_{\Theta} \sum_{T=0}^{\infty} \sum_z p(\hat{R} = 1, z, T; \Theta) \log p(\hat{R}, z, T; \Theta^*) \quad (2.31)$$

with the results listed below

$$\pi_z^a = \hat{\pi}_z^a \sum_s \left[\frac{\gamma}{1 - \gamma} \sum_{z', s'} \hat{\beta}(z', s') \hat{W}_{zao}^{z'} T_{sz}^{s'} + R(s, a) \right] \Omega_s^o \hat{\alpha}(z, s) \quad (2.32)$$

$$W_{z,o}^{z'} = \hat{W}_{z,o}^z \sum_{s', a, s} \hat{\beta}(s', z') T_{sz}^{s'} \hat{\pi}_z^a \Omega_s^o \hat{\alpha}(s, z) \quad (2.33)$$

$$\mu_z = \hat{\mu}_z \sum_s \hat{\beta}(s, z) P(s_0 = s). \quad (2.34)$$

2.1.3 Decentralized Partially Observable Markov Decision Processes

A decentralized POMDP [14] can be represented as $\mathcal{M} = \langle \mathcal{N}, \mathcal{A}, \mathcal{S}, \mathcal{O}, \mathcal{T}, \Omega, \mathcal{R}, \gamma \rangle$, where

- $\mathcal{N} = \{1, \dots, N\}$ denote a finite set of agents;
- $\mathcal{A} = \otimes_n \mathcal{A}_n$ are a set of joint actions, with \mathcal{A}_n available to agent n ;
- $\mathcal{O} = \otimes_n \mathcal{O}_n$ are a set of joint observations, with \mathcal{O}_n available to agent n ;

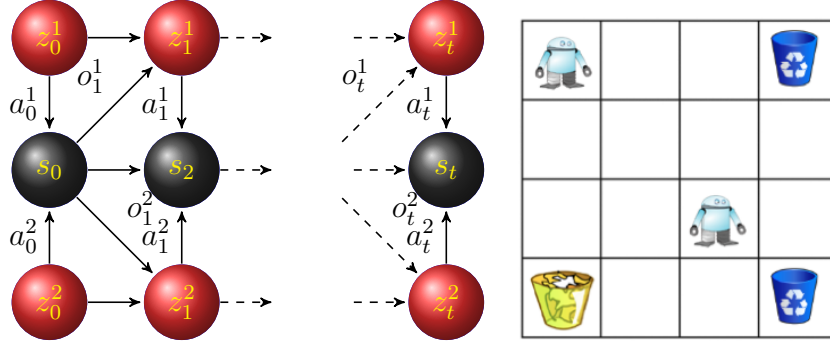


FIGURE 2.5: A simple DEC-POMDP model: Graphical representation(left) and Domain illustration (right).

- \mathcal{S} is a set of finite world states;
- $\mathcal{T} : \mathcal{S} \times \mathcal{A} \rightarrow \mathcal{S}$ is the state transition function with $\mathcal{T}(s'|s, \vec{a})$ denoting the probability of transitioning to s' after taking joint action \vec{a} in s ;
- $\Omega : \mathcal{S} \times \mathcal{A} \rightarrow \mathcal{O}$ is the observation function with $\Omega(\vec{o}|s', \vec{a})$ the probability of observing \vec{o} after taking joint action \vec{a} and arriving in state s' ;
- $\mathcal{R} : \mathcal{S} \times \mathcal{A} \rightarrow \mathbb{R}$ is the reward function with $r(s, \vec{a})$ the immediate reward received after taking joint action \vec{a} in s ;
- $\gamma \in [0, 1)$ is a discount factor.

A DEC-POMDP can be viewed as a POMDP controlled by multiple distributed agents, each acting based on local observations. The joint actions of all agents control the state dynamics and expected global rewards (the global rewards are only used during policy learning; they are not needed when executing the policy). In a DEC-POMDP, each agent has access to only its own action/observation trajectories, but cannot access the trajectories of other agents. Since the state transitions depend jointly on the actions of all agents, this implies that no agent (in general) has enough information to compute the global belief state, a sufficient statistic for decision making in POMDPs. Moreover the joint spaces of state and action grow

exponentially with the number of agents. For these reasons, a DEC-POMDP is more difficult to solve than a POMDP (as seen by the NEXP complexity [14]). Nevertheless, DEC-POMDPs provide a general and expressive framework for multiagent sequential decision-making problems arising in diverse applications, including robotic soccer [90], cooperative transportation [121], extra-planetary exploration [15], traffic control [156]. Other applications are discussed by Oliehoek *et al.* [95] and Amato *et al.* [4].

In a DEC-POMDP, due to the lack of access to other agents' observations, each agent has a policy π_n , which is a mapping from local observation histories to actions. A joint policy consists of the local policies of all agents. For an infinite-horizon DEC-POMDP with initial state s_0 , the objective is to find a joint policy $\pi = \otimes_n \pi_n$, such that the value function of π , $V^\pi(s_0) = \mathbb{E}[\sum_{t=0}^{\infty} \gamma^t r(s_t, \vec{a}_t) | s_0, \pi]$, is maximized.

The methods for solving a DEC-POMDP can be categorized into optimal methods and approximate methods [95]. Except in certain finite horizon cases for which optimal algorithms exist [54, 135], recent research focuses on approximate methods, for both finite and infinite horizon problems. Due to its expressiveness and scalability, the finite state controller (FSC) is the most widely studied approximation method, which achieves state-of-the-art performance. Formally, the stochastic FSC for agent n is defined as $\Theta_n = \langle \mathcal{A}_n, \mathcal{O}_n, \mathcal{Z}_n, W_n, \mu_n, \pi_n \rangle$, where, \mathcal{A}_n and \mathcal{O}_n are the same as defined in the DEC-POMDP; \mathcal{Z}_n is a finite set of nodes, W_n is a set of Markov transition matrices with $W_{zz'}^{n,a,o}$ denoting the probability of agent n transiting from z to z' when taking action a in z results in observation o ; μ_n is the initial node distribution with μ_z^n the probability of agent n initially being in z ; π_n is a set of stochastic policies with $\pi_{z,a}^n$ the probability of agent n taking action a in z . The value function starting the controllers in nodes $\vec{z} = \langle z_1, \dots, z_N \rangle$ at a state s is given

by

$$\begin{aligned}
V(\vec{z}, s) = & \sum_{a_1, \dots, a_N} \prod_{i=1}^N \pi_{z_n, a_n}^n \left[R(s, \vec{a}) + \right. \\
& \left. \gamma \sum_{s'} P(s'|s, \vec{a}) \sum_{o_1, \dots, o_N} P(\vec{o}|s', \vec{a}) \sum_{z'_1, \dots, z'_N} \prod_{n=1}^N W_{z_n, z'_n}^{n, a_n, o_n} V(\vec{z}', s') \right].
\end{aligned} \tag{2.35}$$

For a planning problem, the goal is to find the parameters $\Theta = \cup_n \Theta_n$ that maximize the expected discounted reward for the initial belief b_0 :

$$V(b_0) = \sum_{s, z_1, \dots, z_N} \prod_{n=1}^N \mu_{z_n}^n b_0(s) V(\vec{z}, s). \tag{2.36}$$

Note that (2.36) serves as a global objective function for generating decentralized policies that are parameterized by Θ . Based on FSCs, Amato *et al.* [3] formulate the DEC-POMDP as a nonlinear programming (NLP) problem. Recently, several researchers [71, 98, 96] use the expectation-maximization (EM) algorithm to generate FSCs. These EM-based methods are similar to the method that is developed for POMDPs [144], which is described in the last section, except that they generate decentralized policies. The details are similar, hence they are omitted here.

2.2 Reinforcement Learning

In the previous section, we have reviewed three important sequential decision-making models and the associated basic solution methods. These solution methods require knowing the domain models in advance. In this section, we review the techniques for reinforcement learning (RL), where an agent must learn a policy via using the experiences collected through interacting with an environment, without assuming the knowledge of underlying models. In this case, the policy can be obtained by using either of the following approaches: (i) a model-based approach, in which the

agent first learns the underlying model using its experience with the environment and then solves the learned model to obtain an approximate policy; (ii) a model-free approach, in which the agent learns the policy directly from the experience, skipping the model-learning step.

The two approaches each have their advantages and disadvantages. Since the underlying model is a complete description of an environment, a learned model can be used not only to find the optimal policy, but also for other purposes; thus, a model-based approach is versatile and sample efficient. However, the agent needs a comprehensive set of experiences with the environment to make the learned model complete, implying the model-learning step is not efficient in time. In addition, the agent needs a second step to solve the learned model to determine the best action to take after each update, which could be computationally costly.

A model-free approach, by contrast, does not learn the domain model directly. Though one can find the optimal policy given the model, inverting the learned policy into the model is generally not possible. The fact that the policy is not as rich in information as the model implies: (i) learning a policy may need only a subset of the experience required in learning a complete underlying model of an environment; (ii) the policy can only be used for a subset of the purposes which the model can be used for. For these reasons, a model-free approach is typically more efficient in time, but less versatile and sample efficient, than its model-based counterpart; moreover, the avoidance of a second step of model-to-policy conversion might enhance the time efficiency.

An indispensable ingredient, for both of the aforementioned approaches, is a mechanism for maintaining a proper balance between looking for new information (exploration) and using current models (or policies learned from past experience) to maximize reward (exploitation). Until the current policy is optimal, insofar the agent should always explore the consequences of actions that are not encouraged by the

current policy, to see whether the new actions will lead to higher expected long-term rewards. Exploration is the only way to ensure continual improvement of the policy. However, excessive exploration makes the policy converge unnecessarily slowly. To keep a balance, the agent needs to switch appropriately between exploration and exploitation.

We start this section by first introducing the RL techniques in MDPs, for which the theoretical and technical developments are more advanced. We will give more elaboration on Q-learning and a linear function approximation method, both of which are the foundations for our work introduced in the next chapter. Then we review the RL methods for POMDPs, which have witnessed increased development in the last two decades. We will focus on regionalized policy representation (RPR), an efficient framework for learning FSC-type of policy representations. Finally, we review RL methods for DEC-POMDPs, which is still a nascent field.

2.2.1 Reinforcement Learning in MDPs

For RL in MDPs, model-based methods learn domain models by approximating the reward function $R(s, a)$ and state transition function $P(s'|s, a)$ for each action and state, and then use dynamic programming or linear programming to obtain the optimal policy [132]. Model-free approaches only keep value functions (or parameters in continuous case) which are directly estimated from samples, skipping the step of learning the MDP models. According to Strehl *et al.* [130], a learning algorithm is said to be model-free if its space complexity is $o(|\mathcal{S}|^2|\mathcal{A}|)$, which means asymptotically less than the space for storing an MDP.³

To balance exploration and exploitation (EE), a number of heuristics have been proposed for both model-based and model-free approaches. The most simple exploration strategy is ϵ -greedy [132], in which with probability $1 - \epsilon$, the agent

³ This definition is only applicable for problems with finite state and action spaces.

chooses action according to the greedy policy and with probability ϵ chooses random actions. Other heuristics including the Boltzmann strategies [143], which select actions according to the Boltzmann distribution parameterized by the current value function. These methods are able to guarantee exploring the entire state-action space eventually, however they may suffer from sample efficiency, that is, they may need a huge or possibly unbounded number of samples for learning a policy which is near-optimal. There are a class of more sample efficient algorithms called PAC-MDPs, which enjoy bounded sample numbers with high probabilities [130, 77].

- Model-based methods

Classical model-based methods include Dyna, polarized-sweeping, and others discussed by Sutton and Barto[132]. The PAC-MDP model-based methods include E^3 [65], R-max [21], KWIK [147], and they use the heuristic of “optimism under uncertainty” bias to guide exploration. These PAC-MDP methods are guaranteed to learn near optimal policies in polynomial time. Another class of model-based algorithms use PAC-Bayes exploration, including BEB [68], BEE-TLE [111], BOSS [7], and BAMCP [48] (as well as other references therein). These PAC-Bayes methods maintain distributions over possible models and simply act to maximize the expected future rewards; hence the EE dilemma can be elegantly solved.

- Model-free methods

Classical model-free methods, including Q-learning [150], SARSA, and Actor-critic [132], use ϵ -greedy heuristic for exploration and do not have bounded sample complexity. Later, Delayed Q-learning [130], is designed to wait for m (s, a) samples before updating $Q(s, a)$ and use “optimism under uncertainty” bias for guiding exploration, and is shown to be a PAC-MDP method. However, all these methods are only applicable for problems with discrete action-state

spaces. Recent research have focused on designing PAC-MDP methods, such as Metric E^3 [63], C-PACE [104], which are able to solve problems with large or even continuous state-action spaces.

Here we elaborate more on Q learning which serves the foundation for the method to be introduced in chapter 3. Specifically, Q learning [150] is a classical model-free RL algorithm, which updates Q-value (table) on every time step according to the following equation

$$Q_{t+1}(s_t, a_t) = Q_t(s_t, a_t) + \alpha_t \Delta_t \quad (2.37)$$

with

- Δ_t , temporal difference (TD) error: $\Delta_t = r(s_t, a_t) + \gamma \max_{a'} Q_t(s_{t+1}, a') - Q_t(s_t, a_t)$, where a' is the action maximizes $Q_t(s_{t+1}, a')$
- α_t , the learning rate, which need to satisfy $\sum_{t=1}^{\infty} \alpha_t = \infty, \sum_{t=1}^{\infty} \alpha_t^2 < \infty$: the necessary conditions for guaranteeing convergence of the iteration in (2.37).

The on-policy counter-part to Q-learning is SARSA, which uses a different TD error based on the current policy

$$\Delta_t = r(s_t, a_t) + \gamma Q_t(s_{t+1}, a') - Q_t(s_t, a_t). \quad (2.38)$$

While on-policy algorithms are generally easy to use and has convergence guarantees, off-policy algorithms are more preferable in situations where batch data is available or safe exploration is desired. Although classical Q learning has a number of nice properties, its tabular representation does not generalize to continuous RL domains. In this case, linear function approximation is often used to approximate Q function as a weighted combination of basis function $\phi(s, a)$, *i.e.*, $Q(s_t, a_t) = w^T \phi(s_t, a_t)$. Then the off-policy gradient descend is applied to the Q function for updating the weights

w as follows

$$w_{t+1} = w_t - \frac{1}{2} \alpha_t \nabla_{w_t} [Q^*(s_t, a_t) - Q_t(s_t, a_t)]^2 \quad (2.39)$$

$$= w_t + \alpha_t [Q^*(s_t, a_t) - Q_t(s_t, a_t)] \nabla_{w_t} Q_t(s_t, a_t) \quad (2.40)$$

$$= w_t + \alpha_t \Delta_t \phi(s_t, a_t) \quad (2.41)$$

with the following TD error

$$\Delta_t = r(s_t, a_t) + \gamma \max_{a'} w_t^T \phi(s_{t+1}, a') - w_t^T \phi_t(s_t, a_t). \quad (2.42)$$

However, function approximations can cause divergence for off-policy RL methods, including Q-learning and linear least square approach. Several recent approaches ensure convergence (in the online case) by adding some form of regularization to TD algorithms for policy evaluation, as done in TDC [133], GQ [87], LARS-TD [69], and RO-TD [79]. These algorithms use different versions of the SARSA-style TD error, but the regularization ensures their convergence even when the samples are obtained off-policy.

2.2.2 Reinforcement Learning in POMDPs

Because the state information is noisy, RL in POMDPs is notoriously more difficult than RL in MDPs. Depending on whether we learn the model of POMDPs, the methods of RL in POMDPs can also be categorized into model-based and model-free. For model-based methods, the policy is represented by the value function of a particular action and belief state pair with the belief state computed with the estimated POMDP model. In contrast, for the model-free methods, the policy is usually a mapping from observation histories to actions. The mechanism for learning to interact with a POMDP can be illustrated by figure 2.6, where the solid arrow indicates model-free RL and the dashed arrow indicates model-based RL.

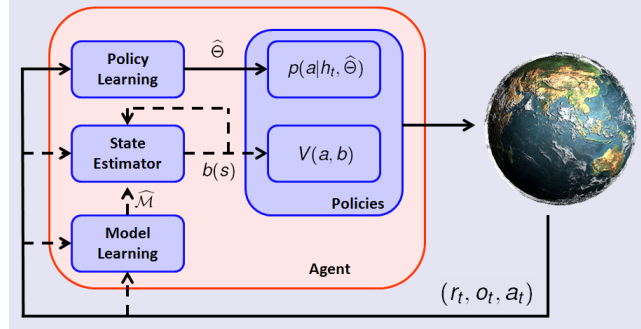


FIGURE 2.6: The planning and learning framework for POMDPs

Typically, model-based approaches first learn the POMDP models using algorithms such as EM, and then use the planning techniques discussed in the previous section to obtain control policies. Because the POMDP models are essentially controlled hidden Markov models (HMMs) and an HMM learning with performance guarantees is known to be intractable [140], model-based methods in general are computationally expensive. However, such difficulty does not prevent developing efficient approximate model-based RL methods, especially if some good prior knowledge is available. Therefore, model-based approaches usually employ Bayesian RL for an implicit exploration and exploitation trade-off, treating the uncertain POMDP parameters as additional states, and attempting to solve an augmented POMDP with the model uncertainty incorporated into the augmented belief states [110, 118]. Specifically, these methods impose Dirichlet priors on the transition function T and observation function Ω , and define the belief state as the distribution of current state s_t and the counts associated with T and Ω , and finally solve an ϵ -optimal policy by using standard POMDP solvers. However, the augmented POMDP is intractable and approximations are used. Moreover, the approximations are usually based on policy-solving of model samples, ignoring the model uncertainty in future steps. To handle the model uncertainty issue, Bayesian nonparametric priors have been applied for solving RL problems in POMDPs [39, 38]. These methods use Monte Carlo Markov chain (MCMC) algorithms for inference. MCMC methods are powerful for

capturing complex structures in data without requiring explicit model selection, yet they suffer some practical shortcomings. Especially they are not ideal for use in contexts where performing inference quickly and reliably on large volumes of streaming data is crucial for timely decision making, such as autonomous robotic systems.

Because of the difficulty of estimating the POMDP model, many RL methods avoid to explicitly model the environment, but directly learn the policy based on the experience of agent-environment interactions. For policy-based RL, there are three main policy representations, including tabular presentation, policy tree and policy graph. Early work for model-free RL include U-tree [88, 159] and Monte Carlo based reactive policy(MCESP) [105]. U-tree uses internal states to memorize the history and build a decision tree online. The issue with this algorithm is that the tree can grow rapidly with the episode length. MCESP uses tabular policy representation and learns the value function over action observation pairs, similar to that of Monte Carlo Exploring-Starts [132]. In addition, MCESP is able to offer a PAC-style guarantee for convergence to a local optimum. In addition, policy graphs have been learned by gradient descent by Meuleau *et al.* [92]. More recently, Monte Carlo tree search based methods [123, 127] use UCT algorithm to control exploration and exploitation, and are able to solve problems with large state space.

The choice between a model-based approach and a model-free approach is, therefore, a tradeoff between versatility and time efficiency. When the policy is of the primary interest, the right choice would be an approach that is just versatile enough for finding the optimal policy, but not more. This idea has been pursued in model-based methods. For example, the work in [152] approximates a POMDP as an utile distinction hidden Markov model which creates memory to preserve perceptual and utility distinction only when necessary. Whereas the work by Shani *et al.* [122] uses U-tree [88] to learn a POMDP model and then use existing POMDP solver to obtain the control policy. In addition, the recent work by Grady *et al.* [46] solves a less

complex model, with reduced state and action spaces, to find approximate policies for the original POMDP. All these methods solve a compressed model of the original POMDP to find approximate policies, enhancing time efficiency while reducing the model’s versatility. A model-free approach can be considered as an extreme case, in which the POMDP is compressed into the policy itself, *i.e.*, the most compressed “model” that preserves the policy.

Because of their conciseness and representation power, finite state controllers (FSCs) are adopted by most model-free RL methods for representing the policies of infinite horizon POMDPs. Early work learns FSCs using stochastic gradient descent [92, 1]. Recently, it has been shown by Li *et al.* [76] that expectation maximization (EM), a popular algorithm in statistics [37], provides an effective tool for learning a family of policies that include FSCs as a special case. The policies considered by Li *et al.* [76], collectively referred to as *regionalized policy representation* (RPR), treat belief-state as a latent random variable and integrate it out to yield a marginalized policy that is expressed as a probability distribution of the current action conditional on the history of past actions and observations. The RPR is a general form of FSC, where each internal memory unit (called a decision state in the RPR and a machine node in the FSC) is associated with a distribution of actions, instead of a single action; as a result, the transition from unit z to unit z' depends jointly on the action at z and the observation at z' , instead of on the observation only. The action-dependency reflects the inherent uncertainty of the action choice in each belief region, which, as discussed in depth by Sondik [128], cannot be resolved unless the policy is finitely transient, in which case the RPR specializes to a FSC.

Here we briefly introduce the RPR that determines the control policy of POMDPs and the auxiliary RPR that controls the balance between exploration and exploitation. This introduction summarizes the necessary terminologies and formulae that will be used in both chapter 4 and chapter 5.

Regionalized Policy Representation

Definition 1. [76] A regionalized policy representation is a tuple $(\mathcal{A}, \mathcal{O}, \mathcal{Z}, W, \mu, \pi)$, where \mathcal{A} , \mathcal{O} , and \mathcal{Z} are respectively a finite set of actions, observations, and decision states; W is a set of Markov transition matrices, with $W_{zao}^{z'}$ denoting the probability of transiting from z to z' when taking action a in z results in observation o ; μ is the initial distribution of decision states, with μ_z the probability of initially being in z ; π is a set of stochastic policies, with π_z^a the probability of taking action a in z .

For simplicity, \mathcal{Z} is denoted as $\{1, 2, \dots, |\mathcal{Z}|\}$, where $|\mathcal{Z}|$ is the cardinality; \mathcal{A} and \mathcal{O} are denoted in similar ways. The set of RPR parameters is denoted as $\Theta = \{\pi, \mu, W\}$. A consecutively indexed variable is abbreviated as the variable with its index range, $W_{zao}^{1:|\mathcal{Z}|} = (W_{zao}^1, W_{zao}^2, \dots, W_{zao}^{|\mathcal{Z}|})$, etc.

Given $h_t = \{a_{0:t-1}, o_{1:t}\}$, the history of actions and observations up to t , the RPR chooses action a_t according to

$$p(a_t|h_t, \Theta) = \frac{p(a_{0:t}|o_{1:t}, \Theta)}{p(a_{0:t-1}|o_{1:t}, \Theta)} = \frac{p(a_{0:t}|o_{1:t}, \Theta)}{p(a_{0:t-1}|o_{1:t-1}, \Theta)}, \quad (2.43)$$

where the second equality arises because o_t has no influence on the actions before t , and $p(a_{0:t}|o_{1:t}, \Theta)$ results from

$$p(a_{0:t}, z_{0:t}|o_{1:t}, \Theta) = \mu_{z_0} \pi_{z_0}^{a_0} \prod_{\tau=1}^t W_{z_{\tau-1} a_{\tau-1} o_{\tau}}^{z_{\tau}} \pi_{z_{\tau}}^{a_{\tau}}, \quad (2.44)$$

by marginalizing out decision states $z_{0:t}$. It follows from (2.43) that $p(a_{0:t}|o_{1:t}, \Theta) = \prod_{\tau=0}^t p(a_{\tau}|h_{\tau}, \Theta)$.

The RPR parameters are learned from the agent experiences by using an empirical value function defined below. Assuming the interaction between the POMDP and the agent is episodic [132], the experiences are represented as a set of episodes. An episode of length T_k is denoted by $(a_0^k r_0^k o_1^k a_1^k r_1^k \dots o_{T_k}^k a_{T_k}^k r_{T_k}^k)$, where r is a nonnegative

immediate reward, k indexes the episodes, and the subscripts index discrete time steps.

Definition 2. [76] Let $\mathcal{D}^{(K)} = \{(a_0^k r_0^k o_1^k a_1^k r_1^k \dots o_{T_k}^k a_{T_k}^k r_{T_k}^k)\}_{k=1}^K$ be a set of episodes resulting from the interaction between the POMDP and an agent who chooses actions according to Π , an arbitrary stochastic policy with action-selecting distributions $p^\Pi(a|h) > 0, \forall$ action a, \forall history h . The empirical value function is defined as

$$\hat{V}(\mathcal{D}^{(K)}; \Theta) \stackrel{\text{def.}}{=} \frac{1}{K} \sum_{k=1}^K \sum_{t=0}^{T_k} \gamma^t r_t^k \frac{\prod_{\tau=0}^t p(a_\tau^k | h_\tau^k, \Theta)}{\prod_{\tau=0}^t p^\Pi(a_\tau^k | h_\tau^k)} \quad (2.45)$$

where $h_t^k = (a_{0:t-1}^k, o_{1:t}^k)$, $0 < \gamma < 1$ is the discount as defined in the POMDP.

It has been shown in [76] that $\lim_{K \rightarrow \infty} \hat{V}(\mathcal{D}^{(K)}; \Theta)$ is the expected sum of discounted rewards by following the RPR parameterized by Θ for an infinite number of steps. Therefore, the RPR resulting from maximization of the empirical value function is an approximation of the optimal policy, assuming the number of decision states, *i.e.*, $|\mathcal{Z}|$, is large enough to accommodate the optimal policy. The optimal policy of a POMDP can be represented by a RPR because the RPR subsumes as a special case the finite state controller (FSC) [76], which is known to approximate the optimal policy of any POMDP to a arbitrary precision [128]. An appropriate $|\mathcal{Z}|$ can be inferred by the method which will be discussed in chapter 5.

Given batch data $\mathcal{D}^{(K)}$, a local optimal RPR parameter Θ can be learned by nested expectation maximization [76], where the outer-loop EM evaluates the policy in the E-step and improves the policy in the M-step, and the inner-loop EM realizes the outer-loop M-step. The nested EM is better interpreted as a consequence of the policy improvement theorem of a POMDP [17]. As detailed in [76] (pages 1140-1143), the outer-loop E-step re-computes the immediate rewards such that a discounted sum of the new rewards, averaged over the episodes, represents the value of the current

policy; in addition, the inner-loop EM updates the policy by maximizing the weighted log-likelihood function of a cost-sensitive hidden Markov model, with the weights constituted by the recomputed new rewards. The weighted log-likelihood function corresponds to the logarithm of the action-value function in [17], parameterized by the RPR and averaged over all belief points, and the average value is maximized by the inner-loop EM to yield an improved policy.

Exploration-Exploitation Trade-off Based on the Auxiliary RPR

The method in [25] employs an auxiliary policy to decide between exploration and exploitation at any time during an episode, and the decision is conditional on the history of past actions and observations. The auxiliary policy, which also is an RPR, is affiliated with the primary RPR that controls the regular actions. The auxiliary RPR has parameters (σ, μ, W) , where (μ, W) are shared with the primary RPR, and σ is distinct from π , with σ_y^z denoting the probability of choosing exploration ($y = 1$) or exploitation ($y = 0$) in decision state z . The primary RPR can be learned by the methods discussed in the previous section. The auxiliary RPR only updates σ , using (μ, W) as they are learned for the primary RPR. The σ is governed by a set of beta distributions,

$$\sigma_0^z \sim \text{Beta}(u_0^z, u_1), \text{ with } \sigma_1^z = 1 - \sigma_0^z, \forall z \in \mathcal{Z}, \quad (2.46)$$

where $u_1 > 0$ is a given constant and $\{u_0^z\}_{z=1}^{|\mathcal{Z}|}$ are updated using the rule,

$$u_0^i = \sum_{k=1}^K \sum_{t=0}^{T_k} \hat{\nu}_t^k \sum_{\tau=0}^t \phi_{t,\tau}^k(i), \forall i \in \mathcal{Z}, \quad (2.47)$$

where $\hat{\nu}_t^k$ and $\phi_{t,\tau}^k(i)$ are as given in (6.5) and (6.9), respectively. At time t during an episode, the probability $p(y_t|h_t)$ is computed using (2.43) and (2.44), replacing a_t with y_t and π with σ .

Intuitively, u_0^z represents the total amount of immediate and future rewards⁴ (over

⁴ Recall that $\hat{\nu}_t^k$ approximates a rescaled reward received by following the RPR.

all time steps in previous episodes) that the agent receives when executing $\pi_{1:|\mathcal{A}|}^z$, the local policy in decision state z . Since $\pi_{1:|\mathcal{A}|}^z$ is executed only when z is occupied, the reward at t is allocated to u_0^z in proportion to $\sum_{\tau=0}^t \phi_{t,\tau}^k(z)$, where $\phi_{t,\tau}^k(z)$ is the probability that z is occupied at τ in episode k , given the actions and observations that have led to the reward at t , as one recalls from (6.9).

When $u_0^z \gg u_1$, one has $\sigma_0^z \gg \sigma_1^z$, which implies the agent almost never performs exploration. Therefore, u_1 defines, up to a constant multiplier, the total reward required in z for the agent to stop exploration in z .

As rewards accumulate in each decision state, the probability of exploration gradually decreases. It is shown in [25] that, with a sufficiently large u_1 , the RPR is guaranteed to converge to the optimal policy (assuming $|\mathcal{Z}|$ is appropriate).

2.2.3 Reinforcement Learning in DEC-(PO)MDPs

Despite a significant research progress in single agent RL, multiagent RL, especially in decentralized settings, is still a nascent field. The distributed RL algorithms with global performance guarantee only exist for some special class of problems, such as network distributed MDPs (ND-MDPs) [47] and network distributed POMDPs (ND-POMDPs) [93]. For example, a decentralized Bayesian RL in a cooperative multiagent system is developed by Teacy *et al.* [138]. This method assumes the value function and the global state are factorable, and the factorized state-transitions are independent. In addition, the distributed RL method developed by Zhang and Lesser [158], assumes that the state transition and observation are independent and interaction among agents are local. Both two methods use a message passing (communication between neighboring agents) and the max-sum algorithm to coordinate distributed Learning.

Except for those special class of multiagent RL problems, there are a very limited amount of research into RL for general DEC-POMDPs. Because of the decentralized

nature of policy execution and the difficulty of solving DEC-POMDP planning with a given model, coordinating RL for DEC-POMDPs are extremely difficult. Among a limited amount of published work on RL for DEC-POMDPs, we discuss two recent methods, which are model-free.

- Banerjee *et al.* [10] have developed a model-free RL framework called Monte Carlo Q alternation (MCQ), in which agents take turns to learn the best response (their own Q-function) to each other's policies. When it is the turn for one agent to learn its policy, this agent updates its Q function over the action-history pairs and uses R-max [21] type of heuristic for exploration. This algorithm is proved to be convergent; however, its sample complexity could grow exponentially with the problem horizon. Later, they develop a pruning technique [10] to reduce the sample complexity. However, this method is still only applicable for finite horizon problems, and it requires the agent to communicate with each other after the learning process of one agent is finished.
- Wu *et al.* [156] have studied RL in infinite horizon DEC-POMDPs and designed a method called MCEM. In this method, a Monte Carlo EM algorithm is adopted to learn the policies represented by FSCs. Specifically, in the E-step, trajectories are generated from the DEC-POMDP model and FSCs; and in the M-step, the policies represented by FSCs are improved. Moreover, MCEM applies UCB type of heuristic for exploration and is able to solve problems with large state-action spaces and increased number of agents. Despite its generality and scalability, the MCEM algorithm is sensitive to initialization and prone to local optima. Moreover, by fixing the number of nodes in controllers, the learned policies from EM algorithms might be over/under represented.

The existing issues in the above methods motivate us to apply Bayesian nonparametric methods to further improve the research of RL in DEC-POMDPs.

2.3 Bayesian Nonparametric Methods: Models and Inference

In the previous section, we introduced RL methods for three important sequential decision models. We are interested in learning control policies directly from experiences of agent-environment interactions. This problem can be formulated as a statistical modeling problem, the essence of which is to infer the model that best explain data (experiences). When solving a statistical inference problem. One need to consider several issues, including how to encode useful prior information and how to select models. The simplest approach for inference is maximum likelihood (ML) estimation which assumes data are drawn independently from $p(x|\theta)$ and uncovers the model that generates x by solving $\arg \max_{\theta} p(x|\theta)$ to provide a point estimate of θ . ML methods can provide consistent and unbiased estimations, however, they are difficult to encode prior information. To address this issue, one can express the prior knowledge of θ by a prior distribution $p(\theta)$, and applies Bayes rule to obtain the posterior distribution

$$p(\theta|x) = \frac{p(x|\theta)p(\theta)}{p(x)} \quad (2.48)$$

, where $p(x) = \int p(x|\theta)p(\theta)d\theta$ is called marginal likelihood or model evidence. One can solve $\arg \max p(\theta|x)$ to obtain a MAP solution. Here we are interested in Bayesian inference which centers on computing both the posterior distribution (instead of a point estimate) and the prediction distribution of the value y_{\star} at a new input x_{\star} location, *i.e.*, $p(y_{\star}|x_{\star}, x, y) = \int P(y_{\star}|x_{\star}, \theta)p(\theta|x, y)d\theta$.

Besides encoding useful prior information, another critical issue in statistics inference is model selection, *i.e.*, to determine the appropriate number of parameters of a model, which has direct impact on the quality of a learned model. When the number of parameters is set too small (the model is overly simple), the inferred model can suffer from under-fitting, which means the model cannot capture the underlying

trend of data. When the number of parameters is set too large (the model is overly complex), the inferred model can suffer from overfitting, which is also undesirable. To address the model selection issue, parametric methods use cross-validation or statistical test to determine an approximate number of parameters, whereas nonparametric methods provide a principled framework to simultaneously infer the model complexity and parameter value. Specifically, by allowing the prior to be nonparametric, we can use posterior inference to determine the appropriate number of model parameters, avoiding the cross-validation which has to be performed by parametric methods when this number is unknown.

This section summarizes two Bayesian nonparametric models, including Gaussian processes (GPs) and Dirichlet processes (DPs), which underpin our value function modeling and policy learning.

2.3.1 Bayesian Nonparametric Models

- Gaussian Processes

A Gaussian Process (GP) is a stochastic process defined over an input space $X \subset \mathbb{R}^n$, with the output f characterized by the mean function μ and covariance function K , which are defined as follows:

$$\mathbb{E}[f(x)] = \mu_x \tag{2.49}$$

$$K(x, x') = \sigma_x^2 \exp \left\{ - \sum_{i=1}^n \frac{(x[i] - x'[i])^2}{2w_i} \right\} + \sigma_n \delta(x, x') \tag{2.50}$$

where $x, x' \in X$, σ_n represents within-point variation caused by measurement noise, and we consider radial basis function as the kernel function. The ratio between σ_n and w_i weights the relative effect of measurement noise and the influences from nearby points.

A GP can be applied as a prior for representing smooth functions, and can be used for solving nonparametric regression/classification problems. For a GP

(over a trajectory) trained with N measurements $\{x_k, f_{x_k}\}_{k=1}^N$, by solving a least square regression problem, given a new input x^* , we can obtain the predictive distribution over f_{x^*} as a Gaussian with predictive mean and variance given by

$$\mu_{f_{x^*}} = K(x^*, X_{tr})K(X_{tr}, X_{tr})^{-1}f_x \quad (2.51)$$

$$\sigma_{f_{x^*}}^2 = K(x^*, x^*) - K(x^*, X_{tr})K(X_{tr}, X_{tr})^{-1}K(X_{tr}, x^*) \quad (2.52)$$

where X_{tr} is a $p \times N$ matrix with each column represents a input sample vector, and f_x is a p dimensional vector concatenating all the output samples.

Gaussian process is nonparametric model, because the parameters is a function of data. More detailed treatment of GPs can be found in [116].

Here we are mainly interested in the application of GPs for RL. A series of papers [41, 42] have showed that GPs could be used to capture the value function using on-policy updates, specifically SARSA and approximate policy iteration, which may not be able to learn the value of a new policy from batch data. Others have used GPs in model-based RL to learn the MDP parameters (\mathcal{P} and \mathcal{R}) [115, 35], including a variant of the R_{\max} algorithm that performs exploration through an “optimism in the face of uncertainty” heuristic [61]. However, model-based approaches require a planner to determine the best action to take after each update, which could be computationally costly. The relationship of GPs and other nonparametric methods for RL are discussed in [136].

- Dirichlet Processes

A Dirichlet Process (DP) [44] is defined as a discrete probability measure over a non-overlapping partition of a probability space Ω . It is characterized by a concentration parameter α and base measure G . DPs are often used as a prior for mixture modeling, in which a single draw of DPs can be represented

as $\{\theta_k, \pi_k\}_{k=1}^{\infty} \sim \text{DP}(\alpha, G)$, where θ_k is a mixture component k and π_k is a mixture weight.

Dirichlet process is widely used for mixture modeling. It can be used for clustering and without knowing the number of clusters a priori. There are two ways to realize DPs. One way is called Chinese restaurant process, where a new data point (customer) is clustered into an existing cluster (table) with the probability proportional to the number of data points (customers) that are already assigned to that cluster (table), and assigned to be a new cluster (table) with probability proportional to α . The other way to present a DP is by using stick-breaking presentation, in which a stick with unit length is broken into K parts (mixture components), with the length of k -th ($k < K$) component $V_k \prod_{j=1}^{k-1} (1 - V_j)$, and the length of remain component $\prod_{k=1}^K (1 - V_k)$, where $\{V_k\}_{k=1}^K \sim \text{Beta}(1, \alpha)$.

- Stick Breaking Processes

Here we summarize the basics of stick-breaking prior and its connection to a Dirichlet process. For more detailed mathematical treatment, readers are referred to [57, 155].

Definition 3. *The stick-breaking priors [57] are almost surely discrete random probability measures \mathcal{P} over the measurable space (Ω, \mathcal{B}) which are partitioned into d disjoint regions with $\Omega = \cup \mathcal{B}_k$ for $1, \dots, d$. It is expressed as*

$$\mathcal{P}_{a,b}^d = \sum_{k=1}^d p_k \delta_{\Theta_k} \tag{2.53}$$

and

$$p_1 = V_1 \quad \text{and} \quad p_i = (1 - V_1)(1 - V_2) \cdots (1 - V_{i-1})V_i, i > 2 \tag{2.54}$$

are the weights with V_i are independent $Beta(a_i, b_i)$ random variables for $a_i, b_i > 0$.

SBP allows Beta-distributed RVs $V_i, \forall i$ and the atoms $\Theta_i, \forall i$ associated with the resulting weights to be drawn simultaneously. Moreover, SBP boils down to DP, when $a_i = 1, \forall i$.

2.3.2 Inference Methods

The aim of Bayesian Inference is to obtain the posterior approximation of the desired models. There are two main categories of algorithms for Bayesian Inference, namely Markov Chain Monte Carlo (MCMC) inference [117] and variational Bayes (VB) optimization [16]. MCMC inference utilizes the Markov chain Monte Carlo simulation to obtain the samples that approximate the posterior distribution. MCMC methods are guaranteed to approximate the posterior distribution accurately, as long as the sampler runs long enough. Although MCMC methods can obtain accurate posterior approximation, the samplers have to run long enough which can be very time consuming. To accelerate convergence, VB is often adopted. VB inference uses a proposal distribution q to approximate the posterior distribution p by minimizing the K-L divergence between p and q .

Q Learning with Gaussian Processes

3.1 Overview

As mentioned in the previous chapter, the core of obtaining the control policy of an MDP lies in computing the value function. For an MDP with continuous state space, the value function is represented by a parameterized functional form. Here we consider linear function approximation. When computing the value function in RL contexts, we have to consider several issues, including feature selection for concise representation, regularization for avoiding overfitting and convergence guarantees. Gaussian Processes (GPs) [116] are Bayesian Nonparametric (BNP) models that are capable of automatically adjusting features based on the observed data. GPs have been successfully employed in high-dimensional approximate RL domains, such as a simulated octopus arm [42], but several aspects of their use, particularly convergence guarantees and off-policy learning have not been fully addressed.

More specifically, unlike RL algorithms employing a tabular representation, and even some function approximation techniques [43, 89], no convergence results for RL algorithms with GPs exist. Also, existing RL methods with GPs have either required

burdensome computation in the form of a planner [115, 35, 61] or require that the policy being learned is the same as the one being executed (on-policy learning) [42]. The latter approach is less general than off-policy RL, which enables learning the optimal value function using samples collected with a safe or exploratory policy.

To address these issues, we present a method for approximate RL using GPs that has provable convergence guarantees in the off-policy setting. More specifically, we present a model-free off-policy approximate RL technique, termed as GPQ, that uses a GP model to approximate the value function and does not require a planner. Because GPQ is off-policy, it can be used in both online or batch settings no matter whether the data is obtained using a safe or exploratory policy. In addition, we present an extension of GPQ that uses a heuristic exploration strategy based on the GP’s inherent measures on predictive confidence of the Q-function.

In addition to presenting the GPQ framework, sufficient conditions for convergence of GPQ to the best achievable optimal Q-function given the data (Q^*) are presented in the batch and online setting, and it is shown that these properties hold even as new features are added or less-important features removed to maintain computational feasibility. Our work also contributes to off-policy approximate RL in general, because unlike other recent papers on off-policy RL with fixed-parameter linear function approximation [133, 87, 69, 79], our approach allows the basis functions to be automatically identified from data. Furthermore, our condition for convergence reveals why in the batch case GPQ or kernel base Fitted Q-Iteration [43] could lead to divergence in the worst case, and how the divergence can be prevented by tuning a regularization-like parameter of the GP. Finally, we designed a practical online implementation of this framework that makes use of a recent budgeted online sparse GP inference algorithm [34]. Our theoretical and empirical results show off-policy RL using a GP provides provable convergence guarantees and competitive learning speeds.

3.2 Gaussian Processes

Gaussian Processes (GPs) [116] are Bayesian nonparametric (BNP) function approximation models: they do not specify a model structure a priori and explicitly model noise and uncertainty. A GP is defined as a collection of random variables, any finite subset of which has a joint Gaussian distribution with mean (prediction) function $m(z)$ and covariance kernel, such as a Radial Basis Function (RBF), $k(z', z)$, for input points z and z' . A common choice of covariance kernel, and the one used in our empirical results, is the Gaussian radial RBF:

$$k(z, z') = \exp\left(-\sum_{i=1}^m \frac{(z_i - z'_i)^2}{2\sigma_i^2}\right) \quad (3.1)$$

where m is the dimensionality of the input domain. In the case of modeling the Q-function, the input domain is the space of all state action pairs and the model captures a distribution over possible Q-functions.

For ease of exposition, denote $Z = [z_1, \dots, z_\tau] = [s_0, a_0, \dots, s_\tau, a_\tau]$ be a set of state action pairs observed at discrete sample times, where each state action pair is concatenated as z . In our analysis, we assume a finite set of actions, but all analysis extends to the continuous action space as well. Let $\vec{y} = [y_1, \dots, y_\tau]^T$ denote the vector of observation values at the respective state action pairs. Given some set of data points \vec{y} at corresponding locations in the input domain Z , we would like to predict the expected value of the Q-function $y_{\tau+1}$ at some possibly new location $z_{\tau+1}$.

Define $K(Z, Z)$ as the kernel matrix with entries $K_{i,j} = k(z_i, z_j)$, $\mathbf{k}(Z, z_{\tau+1}) \in \mathbb{R}^\tau$ as the kernel vector corresponding to the $\tau + 1^{th}$ measurement, and ω_n^2 as the variance of the uncertainty in our measurement. Using Bayes theorem and properties of Gaussian distributions, the conditional probability $p(z_{\tau+1}|)$ can be calculated as a normal variable [116] with mean

$$m(z_{\tau+1}) = \alpha^T \mathbf{k}(Z, z_{\tau+1}), \quad (3.2)$$

where $\alpha = [K(Z, Z) + \omega_n^2 I]^{-1} \vec{y}$ are the kernel weights, and covariance

$$\Sigma(z_{\tau+1}) = k(z_{\tau+1}, z_{\tau+1}) + \omega_n^2 - \mathbf{k}^T(Z, z_{\tau+1}) [K(Z, Z) + \omega_n^2 I]^{-1} \mathbf{k}(Z, z_{\tau+1}). \quad (3.3)$$

Due to the addition of the positive definite matrix $\omega_n^2 I$, the matrix inversions above are well defined. An important insight obtained through the representer theorem [116] is that even if the true function is described by an infinite number of bases, the predictive mean function of a GP will be the best possible description of the function using a finite number of bases given the data available.

3.2.1 Sequential GP Updates with Sparsification

Performing batch prediction using GPs requires an expensive inversion of a matrix that scales as $O(\tau^3)$ with the size of the data. Many sparsification schemes have been proposed to reduce this computational complexity to $O(\tau m^2)$ [34, 74], where m is a number of reduced parameters. Here we use the sparsification method of [34], which allows for sequential updates and is suitable many applications with computational resource constraints. This sparsification algorithm works by building a dictionary of basis vector points that adequately describe the feature of an input domain without including a basis point at the location of every observed data point. A full description of this algorithm is available in Appendix 3.7.2.

In order to determine when a new point should be added to the dictionary, a linear independence test ¹ is performed:

$$\beta_{\tau+1} = k(z_{\tau+1}, z_{\tau+1}) - \mathbf{k}(Z_d, z_{\tau+1})^T K(Z_d, Z_d)^{-1} \mathbf{k}(Z_d, z_{\tau+1}). \quad (3.4)$$

When $\beta_{\tau+1}$ is larger than a specified threshold β_{tol} , then a new data point is added to the dictionary. Otherwise, the weights α_τ are updated, but the dimensionality of α_τ remains the same.

¹ The linear independence test measures the length of the basis vector $\phi(z_{\tau+1})$ that is perpendicular to the linear subspace spanned by the current bases.

3.3 Off-Policy RL with a GP

As a slight abuse of notation, we let Q^* represent the best possible representation of the true Q-function given the data available. In GPQ, we model Q^* as a GP with mean function $m^*(s, a)$ and positive semi-definite covariance kernel $k([s, a], [s', a'])$. To do so, we place a zero mean Gaussian prior on Q , so $Q(s, a) \sim \mathcal{N}(0, k(\cdot, \cdot))$. Our goal is to perform posterior inference using available information so that the current estimate of the mean \hat{m} approaches the mean of Q^* . Let the current estimate of the mean of the Q-function be $\hat{Q}(s, a) = \hat{m}(s, a)$. Since samples of Q^* are not available, posterior inference needs to be performed using the best estimate of Q^* at the current time as:

$$\hat{Q}(s_t, a_t) = r(s_t, a_t) + \gamma \max_{a'}(\hat{Q}(s_{t+1}, a')). \quad (3.5)$$

Essentially equation (3.5) provides a measurement model for generating GP samples. That is every time when $\hat{Q}(s_t, a_t)$ is computed, $\{\hat{Q}(s_t, a_t), s_t, a_t\}$ becomes a new sample for updating the parameters of GPQ. It is worth mentioning that GPQ has a close relation to Bayesian regression. In GPQ, the parameter ω_n^2 is viewed as the variance of Gaussian measurement noise, which act as an regularization term. It accounts for the fact that current measurements are not necessarily drawn from the true model and therefore prevents our model from converging too quickly to an incorrect estimate of Q^* . We will show that ω_n^2 plays a pivotal role in preventing divergence.

3.4 GPQ Algorithm and Convergence Analysis

The update rule in (3.5) with a GP model can be applied both in batch setting and online setting. batch setting means all sample are stored and reused for updating the parameter of Q functions, whereas online setting refers to the case in which each sample is used only once without storing. We briefly summarize the batch algorithm

and its convergence property before presenting an practical online algorithm and its convergence.

3.4.1 Batch GP-Fitted Q-Iteration

Using GPs and the update rule in Equation (3.5) in the batch setting gives us Algorithm 1, which we call GP-FQI because it is a member of the Fitted Q-Iteration family of algorithms. At each iteration, the values of the stored points are updated based on the stored rewards and transitions as well as the previous iteration’s approximation of the Q-values, which is the form of Fitted Q-iteration.

Algorithm 1: Batch GPQ (GP-FQI)

- 1: Input: Experience tuples $\langle s, a, r, s' \rangle_{1 \dots N}$
 - 2: Output: A GP representing \hat{Q}^*
 - 3: $\hat{Q} \leftarrow$ Initialized GP.
 - 4: **repeat**
 - 5: $\hat{Q}' \leftarrow$ Initialized GP.
 - 6: **for** each experience tuple $\langle s, a, r, s' \rangle_i$ **do**
 - 7: $y_i = r_i + \gamma \max_b \hat{Q}(s', b)$
 - 8: **end for**
 - 9: Train \hat{Q} on all $(\langle s_i, a_i \rangle, y_i)$
 - 10: $\hat{Q} = \hat{Q}'$
 - 11: **until** The convergence condition is satisfied
-

Algorithm 2: Online GPQ

- 1: **for** for each time step τ **do**
 - 2: Choose a_τ from s_τ , using ϵ -greedy exploration
 - 3: Take action a_τ , observe $r_\tau, s_{\tau+1}$
 - 4: Let $z_\tau = \langle s, a \rangle$ and $y_\tau = r + \gamma \max_b \hat{Q}(s', b)$
 - 5: **if** $\beta_{\tau+1} > \beta_{tol}$ **then**
 - 6: Add z_τ to the \mathcal{BV} set.
 - 7: **end if**
 - 8: Compute $\mathbf{k}_{z_{\tau+1}}$ and $\alpha_{\tau+1}$ according to [34]
 - 9: **if** $|\mathcal{BV}| > \text{Budget}$ **then**
 - 10: Delete $z_i \in \mathcal{BV}$ with lowest score according to [34]
 - 11: **end if**
 - 12: update $\hat{Q}(z_{\tau+1}) = \sum_{i=1}^{\infty} \alpha_i k(z_i, \cdot)$
 - 13: **end for**
-

It has been shown that GP-FQI with no restrictions on its parameters may actually diverge (a result that has not been reported previously for GP RL), but we later

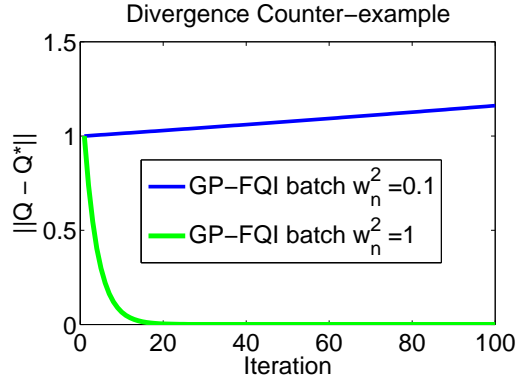


FIGURE 3.1: The maximum error $\|\hat{Q} - Q^*\|$ is plotted for GP-FQI with insufficient regularization $\omega_n^2 = 0.1$ and sufficient regularization $\omega_n^2 = 1$.

show how setting a single regularization parameter for the GP based on the other input parameters will guarantee the convergence of GP-FQI.

Below, we prove that GP-FQI can diverge if the regularization parameter is not properly set. However, we also prove that for any set of hyperparameters and desired density of data, a proper regularization constant can be determined to ensure convergence. We begin with a counter-example showing divergence in the batch setting if ω_n^2 is insufficient, but show convergence when ω_n^2 is large enough.

Consider a system with three nodes located along the real line at locations $-1, 0,$ and 1 . At each time step, the agent can move deterministically to any node or remain at its current node. The reward associated with all actions is zero. All algorithms are initialized with $\hat{Q}(z) = 1\forall z$, $\gamma = 0.9999$, and we use a RBF kernel with bandwidth $\sigma = 1$ in all cases. We consider two settings of the regularization parameter, $\omega_n^2 = 0.1$ and $\omega_n^2 = 1$. Figure 3.1 shows that when ω_n^2 is set too low, the Bellman operation can produce divergence in the batch setting. If the regularization is set to the higher value, GP-FQI converges. In the following sections, we show that determining the sufficient regularization parameter ω_n^2 depends only on the density of the data and the hyperparameters, not the initialization value of \hat{Q} or γ .

Let T denote the approximate Bellman operator that updates the mean of the current estimate of the Q-function using the measurement model of (3.5), that is

$\hat{m}_{k+1} = T\hat{m}_k$, we argue that T is a contraction, so a fixed point exists. For a GP model, we define the approximate Bellman operator in the batch case as training a new GP with the observations $y_i = r(s_i, a_i) + \gamma \max_b \hat{Q}(s'_i, b)$ at the input locations $z_i = (s_i, a_i)$.

The properties of Algorithm 1 can be summarized by the following theorems, the proofs are discussed in [31].

Theorem 4. *Given a GP with data Z of finite size N , and Mercer kernel that is bounded above by k_{\max} , there exists a finite regularization parameter ω_n^2 such that the T is a contraction in the batch setting. In particular, $\omega_n^2 = 2(\|K(Z, Z)\|_\infty - k_{\max}) \leq 2N$*

Theorem 4 shows that in the case of finite data, a finite regularization term always exists which guarantees convergence.

Theorem 5. *Given a GP with infinite data generated using a sparse approximation with acceptance tolerance β_{tol} , and given a Mercer kernel function that decays exponentially, there exists a finite regularization parameter ω_n^2 such that T is a contraction in the batch setting.*

Theorem 5 shows that for a GP with infinite data, a finite regularization term also exists if the added data points (to the GP) exceed the linear independence test β_{tol} .

Theorem 5 provides a powerful insight into the convergence properties of GPs. As the density of basis vectors increases or as the bandwidth of the kernel function grows, corresponding to decreasing β_{tol} , the basis vector weights α_i becomes increasingly correlated. As the weights become correlated, changing the weight at one basis vector also changes the weights of nearby basis vectors. It is this sharing of weights that can result in divergence, as seen in [8]. Theorem 5 shows that for a given

β_{tol} and kernel function, there exists a finite regularization parameter ω_n^2 that will prevent divergence, however. This regularization technique can also be applied to provide convergence guarantees for FQI using a linear function approximator. In related work, [84] provides convergence guarantees for linear FQI using a similar regularization technique solved using an optimization-based framework. In practice, ω_n^2 does not have to be set very large to prevent divergence. Both Theorem 5 and [84] consider worst case analysis, which generally is not encountered in practice. For most applications, reasonable values of $\omega_n^2 \in [0.01, 1]$ will prevent divergence.

Theorem 6. *If the sparse GP algorithm is used, the error $\|\mathbb{E}[\hat{Q} - Q^*]\|$ is uniformly, ultimately bounded for the approximate Bellman operator.*

Theorem 6 indicates the approximation error from using a sparse representation of a GP versus a full GP is bounded. But it does not compute the bound exactly as this depends on the topology of the space and kernel function. The expectation in Theorem 6 is with respect to the difference between two GPs: \hat{Q} and Q^* .

3.4.2 Online learning with GPQ

In theory, GP-FQI provides a method with provable convergence, however the computational requirements can be intense. In our empirical results, we employ several approximations to GP-FQI to reduce the computational burden. We call this modified algorithm *Online GPQ* and display it in Algorithm 2. At each step of Online GPQ, we take an action according to some policy π that ensures ergodicity of the induced Markov chain, and observe the value $y_\tau = r + \gamma \max_b \hat{Q}_\tau(s', b)$ at location z_τ . The sparse online GP algorithm of [34] is used to determine whether or not to add a new basis vector to the active bases set \mathcal{BV} and then update the kernel weights. We provide a set of sufficient conditions for convergence in the online case.

Theorem 7. *For an ergodic sample obtaining policy π , and for each active basis set, a sufficient condition for convergence of $\hat{m}(z_t) \rightarrow m^*(z_t)$ as $t \rightarrow \infty$ on-line GPQ is $\mathbb{E}_\pi [C_t \mathbf{k}_t \mathbf{k}_t^T + K_t^{-1} \mathbf{k}_t \mathbf{k}_t^T] \geq \gamma \mathbb{E}_\pi [C_t \mathbf{k}_t \mathbf{k}_t^\alpha + K_t^{-1} \mathbf{k}_t \mathbf{k}_t^\alpha]$, where $\mathbf{k}_t^\alpha \alpha_t = \max_{a'} (\mathbf{k}^T(x_{t+1}, a')) \alpha_t$.*

Here, C_t is a negative definite and K_t^{-1} is a positive definite matrix related to the posterior and the prior covariance explained in [34]. These sufficient conditions for convergence are less restrictive than [89] for Q-learning. The proof style follows closely to that of [89]. The details are discussed in [32] and attached in Appendix 3.7.3.

It should be noted that while the convergence results presented here are significant because no such results have been available before, these results only guarantee the asymptotic convergence of the Q function to the approximate Bellman operator’s fixed point, within the projection of the selected bases. Which means the algorithm will eventually converge, yet no guarantees on the rate of convergence are provided here. Such guarantees are expected to depend on the choice of the exploration scheme, the algorithm’s eventual selection of bases, and the rate at which the predictive variance decreases.

3.4.3 Optimistic Exploration for GPQ

The Online GPQ algorithm above used an ϵ -greedy exploration strategy, which may not collect samples in an efficient manner. We now consider a more targeted exploration heuristic facilitated by the GP representation. Others have considered similar heuristics based on information theory [33]. Here we use a simpler strategy based on the “optimism in the face of uncertainty” principle, which has been a cornerstone of efficient exploration algorithms (e.g. [131]).

In the discrete-state case, optimistic value functions can be maintained by initializing Q-values to $R_{\max}/(1 - \gamma)$ and performing updates that maintain optimism

until the values are nearly accurate. Pairing this over-estimate with greedy action selection causes the agent to explore areas of the state space that may yield higher long-term values, but not at the expense of “known” areas which have higher values grounded in real data. We propose using the upper confidence tails from the GP as an optimistic value function. Specifically, for any point $\langle s, a \rangle$, the GP will report an upper confidence tail of $m(s) + 2\Sigma(s_{\tau+1})$ where m and Σ are defined in Section 3.2. We modify GPQ to use these optimistic estimates in two ways: change value used in GP update to $\hat{Q}(s_i, a_i) = r(s_i) + \gamma \max_a [\hat{Q}(s_{i+1}, a) + 2\Sigma(s, a)]$, always take actions that are greedy with respect to the upper confidence tail.

The first change uses the upper tail of the next state’s Q-value in the Bellman update to maintain optimism of the value function and is reminiscent of the backups performed in Model-Based Interval Estimation [131]. The second change makes the algorithm select greedy actions with respect to an optimistic Q-function.

A degree of caution needs to be used when employing this optimistic strategy because of potentially slow convergence rates for GPQ when large amounts of data has been collected. Once a large amount of data has been collected at a certain point, the confidence interval at that location can converge before GPQ has centered the mean around the true Q-value. These temporarily incorrect Q-values will still be optimistic and encourage exploration, but it may take a large amount of time for these points to converge to their true values, meaning this technique is prone to *over*-exploration. However, in many of our empirical tests, this variance-based exploration significantly outperformed ϵ -greedy exploration.

3.5 Experiments

Our experiments cover three domains, a discrete 5×5 Gridworld, a continuous state Inverted Pendulum (see [73]), and continuous state Puddle-World [20]. Specific details of the domains are listed in Appendix 3.7.1. These domains pose increas-

ingly more difficult challenges to GPQ when choosing basis points. The algorithms compared in each domain include the two variants of Online GPQ (ϵ -greedy and optimistic), both using a budgeting scheme [34]. We also implemented a tabular Q-learner (QL-Tab) using discretization, Q-learning with fixed-basis linear function approximation (QL-FB), and the GQ algorithm [87]. We chose these algorithms because they are each off-policy and model-free online approaches for dealing with continuous state spaces. We report results for the best case parameter settings (see Appendix 3.7.1) of 1) the learning rate (QL-tab, FA-FB, GQ), 2) the exploration rate (QL-tab, FA-FB, GQ, GPQ- ϵ -greedy), 3) bandwidth of kernel (FA-FB, GQ, GPQ), 4) the position of kernels (GQ, FA-FB, GPQ) and 5) the number of kernels (quantization level for QL-tab). After cross-validation, the policy learned from all these methods for the three domains are evaluated based on discounted cumulative reward averaged over 20 independent runs and are shown in Figure 3.2.

The Gridworld consisted of 25 cells, noisy transitions, and a goal reward of 1 (step reward of 0). While all of the algorithms find the optimal policy, the GPQ based methods converge much faster by quickly identifying important areas for basis points. We also see that optimistic exploration using the GP’s variance is advantageous, as the algorithm very quickly uncovers the optimal policy.

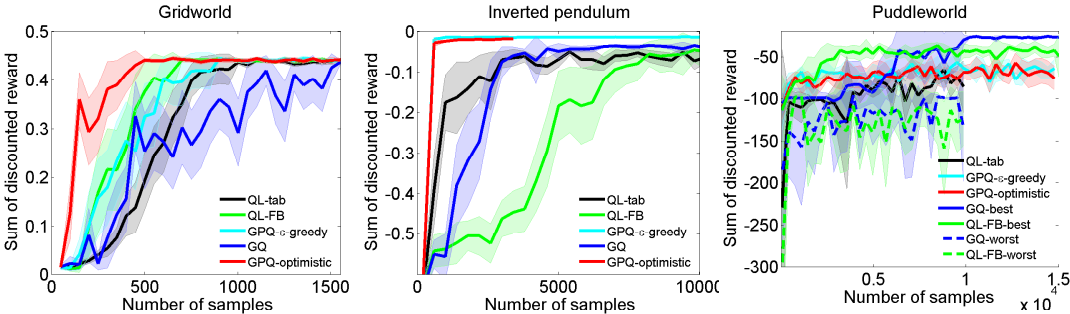


FIGURE 3.2: Average sum of discounted rewards for the experimental domains. The GQ and the QL variants are given more information because their bases are specified a priori, yet GPQ is able to reach comparable performance (often faster) while choosing its own basis functions.

The Inverted Pendulum is a continuous 2-dimensional environment with the re-

ward defined as the difference between the absolute value of the angle for two consecutive states. Again, GPQ quickly finds adequate bases and converges to a near optimal policy while GQ requires more samples. Q-learning with fixed bases and a tabular representation achieve adequate policies as well but require thousands of more samples. Optimistic exploration is not as helpful in this domain since the pendulum usually starts near the goal so targeted exploration is not required to find the goal region. Additional graphs in the appendix show that optimistic exploration is beneficial for certain parameter settings. The graphs in the appendix also demonstrate that GPQ methods are more resilient against small quantizations (budgets) because they are able to select their own bases, while GPQ and QL-FB are far more sensitive to the number and placement of bases.

Finally, we performed experiments in the Puddle-World domain, a continuous 2-dimensional environment with Gaussian transition noise and a high-cost puddle between the agent and the goal. Because of the large range of values around the puddle, basis placement is more challenging here than in the other domains and GPQ sometimes converges to a cautious (puddle adverse) policy. Multiple lines are shown for QL-FB and GQ, depicting their best and worst case in terms of parameter settings, as they were extremely sensitive to these settings in this domain. While the best case versions of GQ and QL-FB reached better policies than GPQ, in the worst case, their Q-values appear to *diverge*. While this is not immediately evident from the discounted-reward graph, an additional graph in the appendix shows the average steps to the goal, which more clearly illustrates this divergence. While GQ has convergence guarantees when data comes from a fixed policy, those conditions are violated here, hence the potential for divergence. In summary, while very careful selection of parameters for QL-FB and GQ leads to slightly better performance, GPQ performs almost as well as their best case with less information (since it does not need the bases a priori) and far outperforms their worst-case results.

3.6 Discussion

We presented a nonparametric Bayesian framework (GPQ) for model-free off-policy RL. GPQ uses GPs to approximate the value function and allows features to be automatically adjusted based on data. We presented algorithms using this framework in the batch and online case and provided sufficient conditions for their convergence. Recognizing that GP’s predictive variance provides a measure of uncertainty of the predicted value, we designed a principled exploration strategy for the online case by using GP’s prediction confidence, which is a function of the predictive variance. Moreover, A GP sparsification technique is implemented based on linearly independence test, and is adapted into the online GPQ framework. Hence, our online GPQ algorithm enjoys bounded memory and significant reduced computational complexity, and is able to balance exploration and exploitation in a principled way. Our empirical results show that GP’s representational power and our efficient algorithm design allows GPQ to perform as well or better than other off-policy RL algorithms.

3.7 Appendix

3.7.1 Details of Empirical Results

Our experiments cover three domains, a discrete 5×5 Gridworld, a continuous state Inverted Pendulum (similar to [73]), and continuous state Puddle-World [20]. The Gridworld consisted of 25 cells with the agent always starting in the lower-left corner, a 0 step cost, and a goal state in the upper-right corner with reward 1 and $\gamma = 0.9$. Transitions were noisy with a .1 probability of the agent staying in its current location.

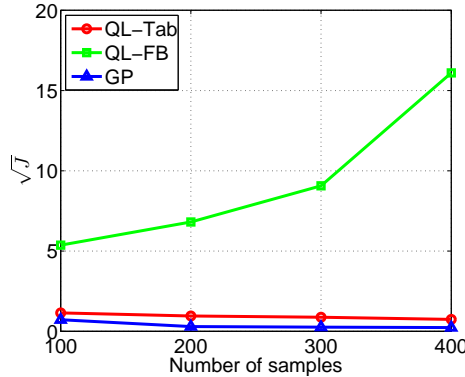


FIGURE 3.3: The performance of GPQ with ϵ -greedy exploration on Baird’s counterexample (the “Star” problem) which can cause divergence with fixed-basis linear function approximation.

We also performed experiments on the Inverted Pendulum, a continuous 2D environment with three actions: applying forces of -50 , 0 , or 50 Newtons. The goal is to balance the pendulum upright within a threshold of $(-\pi/2, \pi/2)$ and the reward is defined as the difference between the absolute value of the angle for two consecutive states.

The Puddle-World domain is a continuous 2-dimensional environment with an agent moving in the four compass directions and Gaussian noise added to its movements. The initial state is at the bottom left of the domain and the goal region is near the top-right corner. The goal region and puddle placement follows the description of [20]. The “puddle” (which is really two overlapping puddles) between the agent and the goal causes negative reward proportional to the agent’s distance to the center of each puddle. Steps outside of the puddle cause a reward of -1 except at the goal where the reward is 0 .

The results are influenced by several parameters, including 1) the learning rate (QL-tab, FA-FB, GQ), 2) the exploration rate (QL-tab, FA-FB, GQ, GPQ- ϵ -greedy), 3) bandwidth of kernel (FA-FB, GQ, GPQ), 4) the position of kernels (GQ, FA-FB, GPQ) and 5) the number of kernels (quantization level for QL-tab). The learning rate is set as $0.5/t^\alpha$ with $\alpha \in \{0.1, 0.3, 0.5\}$, the exploration rate is set according to

$1/t^\beta$ with $\beta \in \{0.1, 0.3, 0.5\}$. For Gridworld the kernel budget is 25; for Inverted Pendulum, the budget is chosen from $\{36, 100\}$ and for Puddle World we use budgets from $\{100, 400\}$. The quantization level for tabular Q-Learning is set the same as the budget for those function approximation methods. For each of the algorithms, the best combination of parameter settings was used for the figures in the main paper.

The following experiments from the Inverted Pendulum domain show the robustness of GPQ against changes in the quantization (budgeting) level and also the sensitivity of the fixed basis methods to the number of basis points. Figure 3.4 demonstrates that GPQ methods are more resilient against small budgets because they are able to select their own bases, while GPQ and QL-FB are far more volatile. These graphs also show that optimistic exploration is beneficial for certain parameter settings.

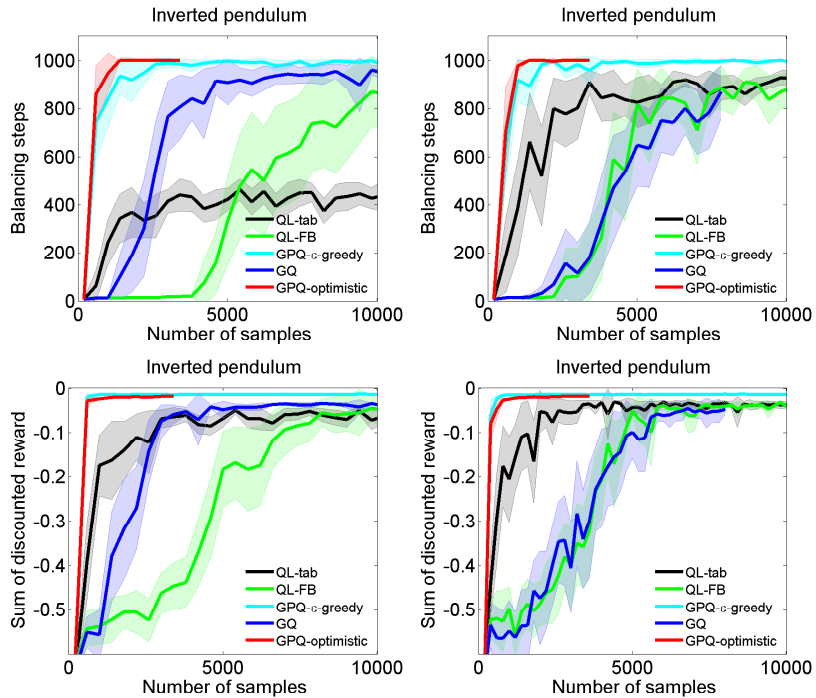


FIGURE 3.4: The performance for Inverted Pendulum under different budget (quantization levels), 36 (left) and 100 (right). GPQ is not sensitive to the quantization level because it selects its own bases, while the quantization level impacts the other algorithms significantly. For example, with higher a quantization level GQ converges slowly, and with a lower quantization level QL-tab performances poorly.

The graphs evaluating performance in Puddle World in Figure 3.2(c) include an additional graph plotted in Figure 3.5 that shows the average steps to the goal, which more clearly illustrates the divergence of GQ and QL-FB. The divergence in these worst cases is caused by having many bases (> 100) with overlapping bandwidths ($> .05$). In contrast, a smaller number of bases with the same small bandwidth actually produces the best performances for these algorithms, because weights are updated almost independently. Figure 3.6 elaborates on this sensitivity by showing the performance of all the algorithm under different budget (quantization) constraints with all other parameters fixed. We see GQ and QL-FB are very sensitive to the budget, and QL-tab is sensitive to the quantization level, while GPQ is relatively stable. In summary, while very careful selection of parameters for QL-FB and GQ leads to slightly better performance, GPQ performs almost as well as their best case with less information (since it does not need the bases a priori) and far outperforms their worst-case results.

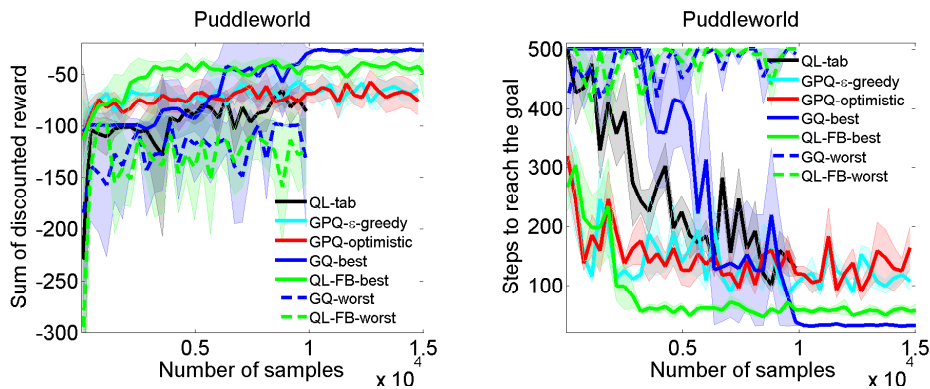


FIGURE 3.5: The performance of the algorithms in Puddleworld. The bandwidth is set to be 0.1 for the basis function in all methods. The number of basis (budget) is set to be 400. FA-FB and GQ diverge when the bases are placed uniformly over the state space.

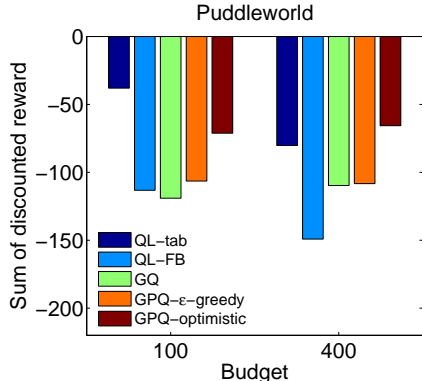


FIGURE 3.6: The performance for Puddleworld under different budget (quantization levels).

3.7.2 Sequential GP Updates

Performing batch prediction using GPs requires the inversion of a matrix that scales with the size of the data. Inverting this matrix at every iteration is computationally taxing, and many sparsification schemes have been proposed to reduce this burden [34, 126, 74]. In this paper, we use the sparsification method used in [34], which allows for sequential updates. The sparsification algorithm in [34] works by building a dictionary of basis points that adequately describe the input domain without including a basis point at the location of every observed data points.

Given a dictionary of bases, Z_d , the prediction and covariance equations are computed as,

$$m(z_{\tau+1}) = \alpha_t^T k(Z_d, z_{\tau+1}) \quad (3.6)$$

$$\Sigma(z_{\tau+1}) = k(z_{\tau+1}, z_{\tau+1}) + k^T(Z_d, z_{\tau+1})C_t k(Z_d, z_{\tau+1}) \quad (3.7)$$

where $C = -(K + \omega^2 I)^{-1}$, i.e. C is the negative of the inverse of the regularized covariance matrix which is computed recursively. A natural and simple way to determine whether to add a new point to the subspace is to check how well it is approximated by the elements in Z . This is known as the kernel linear independence test [34], and has deep connections to reproducing kernel Hilbert spaces (RKHS) [116]. The linear independence test measures the length of the basis vector $\phi(z_{\tau+1})$

that is perpendicular to the linear subspace spanned by the current bases. For GPs, the linear independence test is computed as

$$\beta_{\tau+1} = k(z_{\tau+1}, z_{\tau+1}) - k(Z_d, z_{\tau+1})^T K(Z_d, Z_d)^{-1} k(Z_d, z_{\tau+1}). \quad (3.8)$$

When $\beta_{\tau+1}$ is larger than a specified threshold ϵ_{tol} , then a new data point should be added to the dictionary. Otherwise, the associated weights α_τ are updated, but the dimensionality of α_τ remains the same. When incorporating a new data point into the GP model, the inverse kernel matrix and weights can be recomputed with a rank-1 update. To compute the updates in an online fashion, first define the scalar quantities

$$q^{(\tau+1)} = \frac{y - \alpha_\tau^T k_{x_\tau}}{\omega_n^2 + k^T(Z_d, z_{\tau+1}) C_t k(Z_d, z_{\tau+1}) + k(z_\tau, z_\tau)}, \quad (3.9)$$

$$r^{(\tau+1)} = -\frac{1}{\omega_n^2 + k^T(Z_d, z_{\tau+1}) C_t k(Z_d, z_{\tau+1}) + k(z_\tau, z_\tau)}, \quad (3.10)$$

Let $e_{\tau+1}$ be the $(\tau+1)$ coordinate vector, and let $T_{\tau+1}(\cdot)$ and $U_{\tau+1}(\cdot)$ denote operators that extend a τ -dimensional vector and matrix to a $(\tau+1)$ vector and $(\tau+1) \times (\tau+1)$ matrix by appending zeros to them, respectively. The GP parameters can be solved recursively by using the equations

$$\begin{aligned} \alpha_{\tau+1} &= T_{\tau+1}(\alpha_\tau) + q^{(\tau+1)} s_{\tau+1}, \\ C_{\tau+1} &= U_{\tau+1}(C_\tau) + r^{(\tau+1)} s_{\tau+1} s_{\tau+1}^T, \\ s_{\tau+1} &= T_{\tau+1}(C_\tau k_{x_{\tau+1}}) + e_{\tau+1}. \end{aligned} \quad (3.11)$$

The inverse of the matrix $K(Z, Z)$, denoted by P , needed to solve for $\gamma_{\tau+1}$ is updated online through the equation

$$P_{\tau+1} = U_{\tau+1}(P_\tau) + \gamma_{\tau+1}^{-1} (T_{\tau+1}(\hat{e}_{\tau+1}) - e_{\tau+1}) (T_{\tau+1}(\hat{e}_{\tau+1}) - e_{\tau+1})^T \quad (3.12)$$

where $\hat{e}_{\tau+1} := P_\tau k(z_{\tau+1}, Z_d)^T$. In order to maintain a dictionary of fixed size, many point deletion criterion can be used [34]. Alternatively, [40] shows that if data is

drawn from a Banach space and if $\epsilon_{tol} > 0$, then the number of points added to the dictionary will always be finite. If points are not deleted from the dictionary, the approximation error can be bounded analytically. In this paper, we consider a sparse representation of GPs without deletion of dictionary elements in order to facilitate analysis. Empirically, we have found that maintaining a fixed budget size does not significantly decrease performance.

3.7.3 Proof of Theorem 7

In the following we present a sufficient condition for convergence of online GPQ (see Section 4.1). The main idea is to show that the mean of the GP estimate of the Q-function, \hat{m} , converges to the mean m^* . Note that $m^*(z) = \mathbf{k}^T(Z, z)\alpha^*$, where α^* is the unique minimizer of the regularized least squares cost function $\sum_t \|Q^*(z_t) - \mathbf{k}^T(Z, z_t)\alpha^*\|^2$. For ease of exposition, we define the following notation: $\mathbf{k}(Z_t, z_t) = \mathbf{k}_t$, $K_t = K(Z_t, Z_t)$, and $\mathbf{k}_t^\alpha \alpha_t = \max_{a'}(\mathbf{k}^T(x_{t+1}, a'))\alpha_t$. Similar to proofs of other online RL algorithms, including TD learning ([145] and Q-learning [89]), an ODE approach is used to establish stability [12]. The update in (3.11) is rewritten here for the case when the active basis set is fixed, that is, when $\beta_t \leq \beta_{tol}$

$$\begin{aligned}\alpha_{\tau+1} &= \alpha_\tau + \zeta_\tau q^{(\tau+1)}(s_{\tau+1}), \\ C_{\tau+1} &= (C_\tau) + \zeta_\tau r^{(\tau+1)} s_{\tau+1} s_{\tau+1}^T, \\ s_{\tau+1} &= (C_\tau k_{x_{\tau+1}}).\end{aligned}\tag{3.13}$$

where $\hat{s}_{t+1} = C_t \mathbf{k}_{t+1} + K_t^{-1} \mathbf{k}_{t+1}$ and ζ_τ is a decreasing sequence with $\sum_\tau \zeta_\tau = \infty$.

Theorem 4. *For an ergodic sample obtaining policy π , and for each active basis set \mathcal{BV} , a sufficient condition for convergence with probability 1 of $\hat{m}(z_t) \rightarrow m^*(z_t)$ as $t \rightarrow \infty$ when using the online sparse GP algorithm of (3.13) with the measurement model \hat{Q} in (3) of the main paper is*

$$\mathbb{E}_\pi [C_t \mathbf{k}_t \mathbf{k}_t^T + K_t^{-1} \mathbf{k}_t \mathbf{k}_t^T] \geq \gamma \mathbb{E} [C_t \mathbf{k}_t \mathbf{k}_t^\alpha + K_t^{-1} \mathbf{k}_t \mathbf{k}_t^\alpha].\tag{3.14}$$

Proof. Ensuring that assumptions required for Theorem 17 of [12] hold, the following ODE representation of the α update equation of 3.13 exists:

$$\dot{\alpha}(t) = \mathbb{E}_\pi [q_t S_t]. \quad (3.15)$$

In the following it is shown that $\alpha \rightarrow \alpha^*$ for each active basis set. Let $\tilde{\alpha} = \alpha - \alpha^*$ and consider the continuously differentiable function $V(\tilde{\alpha}) = \frac{1}{2} \tilde{\alpha}_t^T \tilde{\alpha}_t$. $V(\tilde{\alpha}_t) > 0$ for all $\tilde{\alpha}_t \neq 0$, therefore, V is a Lyapunov like candidate function [49, 67]. The first derivative of V along the trajectories of $\alpha(t)$ is

$$\dot{V}(\tilde{\alpha}) = \tilde{\alpha}_t^T \mathbb{E}_\pi \left[\frac{(C_t \mathbf{k}_t + K_t^{-1} \mathbf{k}_t)}{\sigma_x^2} (r_t + \gamma \mathbf{k}_t^\alpha \alpha_t - \mathbf{k}_t^T \alpha_t) \right], \quad (3.16)$$

where we have set $\sigma_x^2 = \omega_n^2 + \mathbf{k}^T(Z, z) C_t \mathbf{k}(Z, z') + k(z, z')$ for notational convenience. Note that $m^*(z) = \mathbf{k}^T(Z, z) \alpha^*$, and add and subtract $\mathbf{k}_t^T \alpha^*$ we have

$$\dot{V}(\tilde{\alpha}) = -\tilde{\alpha}_t^T \mathbb{E}_\pi \left[\frac{(C_t \mathbf{k}_t + K_t^{-1} \mathbf{k}_t)}{\sigma_x^2} \mathbf{k}_t^T \tilde{\alpha} + \tilde{\alpha}_t^T \mathbb{E}_\pi \left(\frac{(C_t \mathbf{k}_t + K_t^{-1} \mathbf{k}_t)}{\sigma_x^2} \right) (r_t + \gamma \mathbf{k}_t^\alpha \alpha_t - \mathbf{k}_t^T \alpha_t^*) \right]. \quad (3.17)$$

Adding and subtracting $\gamma \mathbf{k}_t^{al*} \alpha^*$, noting that $\mathbf{k}_t^{al*} \alpha^* \geq \mathbf{k}^\alpha \alpha^*$, and $m_t^* = r_t + \mathbf{k}^\alpha \alpha^*$ due to the Bellman equation, we have

$$\dot{V}(\tilde{\alpha}) = -\tilde{\alpha}_t^T \mathbb{E}_\pi \left[\frac{(C_t \mathbf{k}_t + K_t^{-1} \mathbf{k}_t)}{\sigma_x^2} \mathbf{k}_t^T - \gamma \frac{(C_t \mathbf{k}_t + K_t^{-1} \mathbf{k}_t)}{\sigma_x^2} \mathbf{k}_t^\alpha \right] \tilde{\alpha}. \quad (3.18)$$

Hence, if the condition in (3.14) is satisfied, $\dot{V}(\tilde{\alpha}) < 0$ and $\alpha \rightarrow \alpha^*$ w.p. 1 for each active basis set. \square

The argument above can be extended to show that $m(z) \rightarrow m^*(z)$ even as new bases are added to the active basis set. To see this, let k be the time instant when a new basis is added and let α_{k+1}^* be the associated ideal weight vector. Note that the nature of the sparse GP algorithm ensures that $k+1 \geq k + \Delta T$ with $\Delta T > 0$. If ΔT is sufficiently large such that $\mathbb{E}_\pi (V(\alpha_{k+1}^*) - V(\alpha_k^*)) < 0$ then the convergence is uniform across all bases.

Online Expectation Maximization for Reinforcement Learning in POMDPs

4.1 Overview

Because of its conciseness and approximation power, finite-state controllers (FSCs) have been a dominant policy representation for infinite-horizon POMDPs. Under reinforcement learning settings, early work learns FSCs using stochastic gradient descent [92, 1]. Recently, it has been shown in [76] that expectation maximization (EM), a popular algorithm in statistics [37], provides an effective tool for learning a family of policies that include FSCs as a special case. The policies considered in [76], collectively referred to as *regionalized policy representation (RPR)*, treat belief-state as a latent random variable and integrate it out to yield a marginalized policy that is expressed as a probability distribution of the current action conditional on the history of past actions and observations. The RPR is a general form of FSC, where each internal memory unit (called a decision state in the RPR and a machine node in the FSC) is associated with a distribution of actions, instead of a single action; as a result, the transition from unit z to unit z' depends jointly on the action at z

and the observation at z' , instead of on the observation only. The action-dependency reflects the inherent uncertainty of the action choice in each belief region, which, as discussed in depth in [128], cannot be resolved unless the policy is finite transient, in which case the RPR specializes to a FSC.

Given batch data $\mathcal{D}^{(K)}$, a local optimal RPR can be learned by nested expectation maximization [76], where the outer-loop EM evaluates the policy in the E-step and improves the policy in the M-step, and the inner-loop EM realizes the outer-loop M-step. It is noteworthy that the nested EM maximizes a value function, which cannot be interpreted as a probability of the observed variables as in the case of a likelihood function. The nested EM is better interpreted as a consequence of the policy improvement theorem of a POMDP [17]. As detailed in [76](pages 1140-1143), the outer-loop E-step re-computes the immediate rewards such that a discounted sum of the new rewards, averaged over the episodes, represents the value of the current policy; in addition, the inner-loop EM updates the policy by maximizing the weighted log-likelihood function of a cost-sensitive hidden Markov model, with the weights constituted by the recomputed new rewards. The weighted log-likelihood function corresponds to the logarithm of the action-value function in [Blackwell, 1965], parameterized by the RPR and averaged over all belief points, and the average value is maximized by the inner-loop EM to yield an improved policy.

The RPR has been introduced in [76] as a tool for multi-task RL across multiple POMDPs. Subsequent work has focused on using the RPR for reinforcement learning in a single POMDP [Cai et al., 2009; Liu et al., 2011]. All previous work assumes batch-mode learning, updating the policy based on the entire set of experiences collected so far. Consequently, as new experiences arrive sequentially, the computational cost and memory demand increase fast. To reduce the time and memory complexity of learning an RPR, in this chapter, we proposed an online nested expectation maximization for learning the FSCs which is the parameter of RPR. The major change

from the batch-mode EM is that the policy is updated each time based on a partial policy evaluation, with only the latest episode of agent-environment interaction used in the outer-loop E-step and inner-loop Estep. Therefore each episodes contribution to the sufficient statistics is computed only when the episode is new, instead of computed repeatedly when each subsequent episode comes in; this makes the E-steps computationally more efficient. Furthermore, as each episode is used only when it is new, the algorithm need only memorize the latest episode, discarding all older ones, leading to tremendous memory savings.

4.2 Batch-mode Expectation Maximization

Before presenting the online Expectation and Maximization algorithm, we first review the batch-model EM algorithm in [76], The solution produced by the EM algorithm in [76] is re-stated in Theorem 5.

Theorem 5. *Let $\{\Theta_n\}_{n \geq 0}$ be a sequence produced by the iterative update $\Theta_{n+1} = \arg \max_{\hat{\Theta} \in \Xi} \text{LB}(\hat{\Theta}|\Theta_n)$, where Θ_0 is an arbitrary initialization and $\Xi = \left\{ \Theta = (\mu, \pi, W) : \sum_{j=1}^{|\mathcal{Z}|} \mu_j = 1, \sum_{a=1}^{|\mathcal{A}|} \pi_a^i = 1, \sum_{j=1}^{|\mathcal{Z}|} W_j^{iao} = 1, i = 1, 2, \dots, |\mathcal{Z}|, a = 1, 2, \dots, |\mathcal{A}|, o = 1, 2, \dots, |\mathcal{O}| \right\}$,*

$$\begin{aligned} \text{LB}(\hat{\Theta}|\Theta_n) &= \frac{1}{K} \sum_{k=1}^K \sum_{t=0}^{T_k} \nu_t^k \left\{ \sum_{i=1}^{|\mathcal{Z}|} \phi_{t,0}^k(i) \ln \hat{\mu}_i \right. \\ &\quad \left. + \sum_{\tau=0}^t \left[\sum_{i=1}^{|\mathcal{Z}|} \phi_{t,\tau}^k(i) \ln \hat{\pi}_{a_\tau^k}^i + \sum_{i,j=1}^{|\mathcal{Z}|} \xi_{t,\tau}^k(i,j) \ln \hat{W}_i^{j,a_{\tau-1}^k, o_\tau^k} \right] \right\}, \\ \nu_t^k &= \frac{\gamma^t r_t^k p(a_{0:t}^k | o_{1:t}^k, \Theta_n)}{\prod_{\tau=0}^t p^\Pi(a_\tau^k | h_\tau^k) \hat{V}(\mathcal{D}^{(K)}; \Theta_n)}, \forall t, k, \end{aligned} \quad (4.1)$$

$$\xi_{t,\tau}^k(i,j) = p(z_\tau^k = i, z_{\tau+1}^k = j | a_{0:t}^k, o_{1:t}^k, \Theta_n), \quad (4.2)$$

$$\phi_{t,\tau}^k(i) = p(z_\tau^k = i | a_{0:t}^k, o_{1:t}^k, \Theta_n), \quad (4.3)$$

then $\{\Theta_n\}_{n \geq 0}$ monotonically increases LB, until convergence to a maxima.

LB is derived from the lower bound of log value function(2.45). The quantity ν_t^k in (11) is the re-computed reward at time t in episode k ; it represents the expected reward of following policy Θ_n , and is computed in the outer-loop E-step. The qualities in (12)-(13) are standard soft counts resulting from the Baum-Welch formula for learning an HMM [114], and they are computed in the inner-loop E-step.

4.3 Online Nested Expectation Maximization

To prepare for the online EM algorithm, we rewrite the local objective in Theorem 5 equivalently as

$$\text{LB}(\hat{\Theta}|\Theta_n) = \frac{1}{K} \sum_{i=1}^{|\mathcal{Z}|} v_i \ln \hat{\mu}_i + \frac{1}{K} \sum_{i=1}^{|\mathcal{Z}|} \sum_{a=1}^{|\mathcal{A}|} \rho_a^i \ln \hat{\pi}_a^i + \frac{1}{K} \sum_{a=1}^{|\mathcal{A}|} \sum_{o=1}^{|\mathcal{O}|} \sum_{i=1}^{|\mathcal{Z}|} \sum_{j=1}^{|\mathcal{Z}|} \omega_j^{iao} \ln \hat{W}_j^{iao}, \quad (4.4)$$

where, for any $i, j \in \{1, \dots, |\mathcal{Z}|\}$, $a \in \{1, \dots, |\mathcal{A}|\}$, and $o \in \{1, \dots, |\mathcal{O}|\}$,

$$v_i = \frac{1}{K} \sum_{k=1}^K \sum_{t=0}^{T_k} \nu_t^k \phi_{t,0}^k(i), \quad (4.5)$$

$$\rho_a^i = \frac{1}{K} \sum_{k=1}^K \sum_{t=0}^{T_k} \nu_t^k \sum_{\tau=0}^t \phi_{t,\tau}^k(i) \delta(a_\tau^k, a), \quad (4.6)$$

$$\omega_j^{iao} = \frac{1}{K} \sum_{k=1}^K \sum_{t=0}^{T_k} \nu_t^k \sum_{\tau=1}^{t-1} \xi_{t,\tau}^k(i, j) \delta(a_\tau^k, a) \delta(o_{\tau+1}^k, o), \quad (4.7)$$

and $\delta(a, b) = \begin{cases} 1, & a = b \\ 0, & a \neq b \end{cases}$. Since ν 's are normalized and ϕ 's and ξ 's are probabilities, the quantities $\{v, \rho, \omega\}$ are all bounded.

The constrained maximization $\max_{\hat{\Theta} \in \Xi} \text{LB}(\hat{\Theta} | \Theta_n)$ has a unique solution given by

$$\hat{\mu}_i = \frac{v_i}{\sum_{i=1}^{|\mathcal{Z}|} v_i}, \quad \hat{\pi}_a^i = \frac{\rho_a^i}{\sum_{a=1}^{|\mathcal{A}|} \rho_a^i}, \quad \hat{W}_j^{iao} = \frac{\omega_j^{iao}}{\sum_{j=1}^{|\mathcal{Z}|} \omega_j^{iao}}, \quad (4.8)$$

for $i, j = 1, 2, \dots, |\mathcal{Z}|$, $a = 1, \dots, |\mathcal{A}|$, and $o = 1, \dots, |\mathcal{O}|$. Define

$$\begin{aligned} f(\hat{\Theta}; v, \rho, \omega) \stackrel{Def.}{=} & \text{LB}(\hat{\Theta} | \Theta_n) + (1 - \sum_{i=1}^{|\mathcal{Z}|} \mu_i) \sum_{i=1}^{|\mathcal{Z}|} v_i + \sum_{i=1}^{|\mathcal{Z}|} (1 - \sum_{a=1}^{|\mathcal{A}|} \hat{\pi}_a^i) \sum_{a=1}^{|\mathcal{A}|} \rho_a^i \\ & + \sum_{a=1}^{|\mathcal{A}|} \sum_{o=1}^{|\mathcal{O}|} \sum_{i=1}^{|\mathcal{Z}|} (1 - \sum_{j=1}^{|\mathcal{Z}|} \hat{W}_j^{iao}) \sum_{j=1}^{|\mathcal{Z}|} \omega_j^{iao}. \end{aligned} \quad (4.9)$$

One may verify that $f(\hat{\Theta}; v, \rho, \omega)$ has a unique stationary point that is equal to $\hat{\Theta}$ as given in (4.8).

Let $\boldsymbol{\theta} = [\boldsymbol{\mu}, \boldsymbol{\pi}, \boldsymbol{w}]^T$ and $\boldsymbol{s} = [\boldsymbol{v}, \boldsymbol{\rho}, \boldsymbol{\omega}]^T$, with the definitions $\boldsymbol{\mu} = [\mu_i]_{i=1:|\mathcal{Z}|}$, $\boldsymbol{\pi} = [\pi_a^i]_{a=1:|\mathcal{A}|, i=1:|\mathcal{Z}|}$, $\boldsymbol{w} = [W_j^{iao}]_{j=1:|\mathcal{Z}|, i=1:|\mathcal{Z}|, a=1:|\mathcal{A}|, o=1:|\mathcal{O}|}$, and similarly $\boldsymbol{v} = [v_i]_{i=1:|\mathcal{Z}|}$, $\boldsymbol{\rho} = [\rho_a^i]_{a=1:|\mathcal{A}|, i=1:|\mathcal{Z}|}$, $\boldsymbol{\omega} = [\omega_j^{iao}]_{j=1:|\mathcal{Z}|, i=1:|\mathcal{Z}|, a=1:|\mathcal{A}|, o=1:|\mathcal{O}|}$, where the elements in a vectorization are arranged by following a left-to-right order in sorting the indices. For example, $[\pi_a^i]_{a=1:|\mathcal{A}|, i=1:|\mathcal{Z}|} = [\pi_1^1, \dots, \pi_{|\mathcal{A}|}^1, \dots, \pi_1^{|\mathcal{Z}|}, \dots, \pi_{|\mathcal{A}|}^{|\mathcal{Z}|}]$.

Define $\mathbf{C} = \text{diag}(\mathbf{C}_\mu, \mathbf{C}_\pi, \mathbf{C}_W)$, where \mathbf{C}_μ is a $|\mathcal{Z}| \times |\mathcal{Z}|$ matrix of ones, $\mathbf{C}_\pi = \text{diag}\{(\mathbf{C}_\pi^i)_{i=1:|\mathcal{Z}|}\}$ with \mathbf{C}_π^i a $|\mathcal{A}| \times |\mathcal{A}|$ matrix of ones, and

$$\mathbf{C}_W = \text{diag}\{(\mathbf{C}_W^{iao})_{i=1:|\mathcal{Z}|, a=1:|\mathcal{A}|, o=1:|\mathcal{O}|}\}$$

with \mathbf{C}_W^{iao} a $|\mathcal{Z}| \times |\mathcal{Z}|$ matrix of ones. By construction, \mathbf{C} is a block-diagonal matrix with each block a matrix of all ones, and each block is associated with the corresponding sub-vector in $\boldsymbol{\theta}$ whose elements sum up to one.

Let $\ln(\boldsymbol{\theta})$ denote a vector resulting from element-wise application of the natural logarithm to $\boldsymbol{\theta}$. Define

$$\boldsymbol{\psi}(\boldsymbol{\theta}) = \ln \boldsymbol{\theta} - \mathbf{C}\boldsymbol{\theta}. \quad (4.10)$$

Then one can rewrite (4.9) as

$$f(\hat{\boldsymbol{\theta}}; \boldsymbol{s}) = \boldsymbol{s}^T \boldsymbol{\psi}(\hat{\boldsymbol{\theta}}), \quad (4.11)$$

and the solution in (4.8) is obtained by solving the equation $\nabla_{\hat{\boldsymbol{\theta}}} f(\hat{\boldsymbol{\theta}}; \mathbf{s}) = \mathbf{0}$ or equivalently $\nabla_{\hat{\boldsymbol{\theta}}} \boldsymbol{\psi}^T(\hat{\boldsymbol{\theta}}) \mathbf{s} = \mathbf{0}$, where $\nabla_{\boldsymbol{\theta}}$ denotes the gradient with respect to $\boldsymbol{\theta}$.

Using the vectorized notation, one can restate the EM algorithm in Theorem 5 as:

Outer- and/or inner-loop E-steps: Update $\mathbf{s} = \mathbf{s}(\mathcal{D}^{(K)}, \boldsymbol{\theta}_{n-1})$ using (6.5) - (6.8) and (4.5)-(4.7).

Inner-loop M-step: Solve the equation $\nabla_{\boldsymbol{\theta}} \boldsymbol{\psi}^T(\boldsymbol{\theta}) \mathbf{s}|_{\boldsymbol{\theta}=\boldsymbol{\theta}_n} = \mathbf{0}$ for $\boldsymbol{\theta}_n$.

We have written $\mathbf{s} = \mathbf{s}(\mathcal{D}^{(K)}, \boldsymbol{\theta}_{n-1})$ to emphasize that the sufficient statistic \mathbf{s} is a function of the previous parameter vector $\boldsymbol{\theta}_{n-1}$ and episodes $\mathcal{D}^{(K)}$. Note that the constraint $\hat{\Theta} \in \Xi$ in Theorem 5 has been accounted for in (4.9) and (4.11).

To see the possibility of an online version of the EM algorithm, we note from (6.5) - (6.8), the sufficient statistic is a sum over the episodes, and therefore one can write

$$\mathbf{s}(\mathcal{D}^{(n)}, \boldsymbol{\theta}_{n-1}) = \frac{1}{n} \sum_{k=1}^n \mathbf{s}(\varepsilon_k, \boldsymbol{\theta}_{n-1}), \quad (4.12)$$

where $\mathbf{s}(\varepsilon_k, \boldsymbol{\theta}_{n-1})$, a vector collecting only the terms with index k in (6.5) - (6.8), represents the contribution from ε_k and $\mathcal{D}^{(n)} = \{\varepsilon_1, \varepsilon_2, \dots, \varepsilon_n\}$ are episodes coming in sequentially.

The online EM algorithm employs a noisy (indicated by the hat above \mathbf{s}) version of true sufficient statistic in (4.12),

$$\hat{\mathbf{s}}(\mathcal{D}^{(n)}, \boldsymbol{\theta}_{n-1}) = \sum_{k=1}^n \beta_k \mathbf{s}(\varepsilon_k, \boldsymbol{\theta}_{k-1}), \quad (4.13)$$

where $\beta_k = \alpha_k \prod_{j=k+1}^n (1 - \alpha_j)$, and $\{0 \leq \alpha_k \leq 1\}_{k=1}^n$ referred to as learning rates, discount the contribution from each episode, with earlier giving greater discounts because they are computed using earlier versions of $\boldsymbol{\theta}$ and are thus less accurate.

One can equivalently write (4.13) as

$$\hat{\mathbf{s}}(\mathcal{D}^{(n)}, \boldsymbol{\theta}_{n-1}) = \hat{\mathbf{s}}_n, \quad (4.14)$$

with the recursive definition

$$\hat{\mathbf{s}}_k = (1 - \alpha_k)\hat{\mathbf{s}}_{k-1} + \alpha_k \mathbf{s}(\varepsilon_k, \boldsymbol{\theta}_{k-1}), k = 1, 2, \dots, n \quad (4.15)$$

The online nested EM algorithm processes the episodes on the fly, and removes an episode from the memory once its contribution to the sufficient statistic has been recorded using (4.15). The contribution from the k -th episode, $\mathbf{s}(\varepsilon_k, \boldsymbol{\theta}_{k-1})$, is computed using (6.5) - (6.8) and (4.5)-(4.7), considering only the terms with index k . Recalling that (6.5) constitutes the outer-loop E-step and (6.9) - (6.8) the inner-loop E-step, one need only compute (6.5) very δn iterations of the inner-loop EM, where δn may be determined by the convergence of the inner-loop EM. When n is small, however, the sufficient statistics is very noisy, and in this case it is preferable to run only a few iterations for both the outer-loop EM and inner-loop EM, to prevent premature convergence. The RPR parameter vector is updated in the inner-loop M-step, as the solution to

$$\nabla_{\boldsymbol{\theta}} \boldsymbol{\psi}^T(\boldsymbol{\theta}) \hat{\mathbf{s}}_n = \mathbf{0}. \quad (4.16)$$

The solution, given in closed-form in (4.8), is essentially a set of normalized sub-vectors of the sufficient statistic.

The complete online algorithm is given in Algorithm 3. The algorithm performs on-policy learning ([132]), where the agent learns the RPR while using it to choose actions in collecting experiences. In order to achieve a balance between exploration and exploitation, the behavior policy used in the algorithm, Π_n , is a mixture of two experts , i.e.,

$$p^{\pi_n}(a|h) = p(y = 0|h)p(a|h, \Theta_n) + p(y = 1|h)/|\mathcal{A}|, \forall h,$$

where the first expert, $p(a|h, \Theta_n)$, is the RPR policy as given by (2.43), the second is a random policy, and the gating network $p(y|h)$ is learned, along with the EM algorithm, using the dual-policy approach described in [25]. The behavior policy used here generalizes the ϵ -greedy policy [132] in such a sense that the probability of choosing random action is not a constant ϵ ; rather, it depends on the history of past actions and observations.

Algorithm 3: The online nested EM algorithm for RPR

Initialize $n = 0$, $\alpha_0 \ll 1$, $\hat{\mathbf{s}} = \mathbf{0}$, Π_0 as random policy, ;
while $P_e = \mathbb{E}[p(y = 1|h)] < \epsilon$ **do**
 Step 1: collect the n -th episode \mathcal{D}_n following Π_{n-1} ;
 Step 2: (outer-loop E-step) recompute rewards with (4.1).;
 while *inner-loop EM not convergent* **do**
 Step 3-1 (inner-loop E-step) compute (4.2)-(4.3);
 Step 3-2 update the sufficient statistic using (4.15).;
 Step 3-3 (inner-loop M-step) update the RPR with (4.8).;
 Step 3-4 (exploitation-exploration balance) update Π_n (the gating network) using the method in [25].
 end
 Step 4: adjust learning rate α_n Step 5: increase pointer $n := n + 1$
end

Algorithm 4: The online EM algorithm for RPR with a fixed behavior policy

while $\hat{V}(\mathcal{D}^{(n)}; \theta_n)$ not converging **do**
 Step 1: collect the n -th episode \mathcal{D}_n following a fixed behavior policy Π ;
 Step 2 (E-step): evaluate the value of \mathcal{D}_n and update the sufficient statistics as in (4.15);
 Step 3 (M-step): update the primary RPR parameters by solving (4.16);
 Step 4: $n = n + 1$, adjust learning rate α_n ;
end while

4.3.1 Time and Memory Complexity

The batch-mode EM algorithm has a time complexity of $O(|\mathcal{A}||\mathcal{O}||\mathcal{Z}|^2 \sum_{m=1}^n \sum_{k=1}^m T_k^2)$ when there is a nonzero reward at every step in an episode (dense reward), or

$O(|\mathcal{A}||\mathcal{O}||\mathcal{Z}|^2 \sum_{m=1}^n \sum_{k=1}^m T_k)$ when nonzero reward is received only at the terminal step in each episode.

In comparison, the proposed online EM algorithm has a time complexity of $O(|\mathcal{A}||\mathcal{O}||\mathcal{Z}|^2 \sum_{k=1}^n T_k^2)$ in the densely-rewarded case, and $O(|\mathcal{A}||\mathcal{O}||\mathcal{Z}|^2 \sum_{k=1}^n T_k)$ in the terminally-rewarded case.

When all episodes have the same number of steps, say T , the complexity of batch-mode EM becomes $O(|\mathcal{A}||\mathcal{O}||\mathcal{Z}|^2 T^2 n^2)$ in the case of dense rewards and $O(|\mathcal{A}||\mathcal{O}||\mathcal{Z}|^2 T n^2)$ in the case of terminal rewards, while the complexity of online EM is $O(|\mathcal{A}||\mathcal{O}||\mathcal{Z}|^2 T^2 n)$ and $O(|\mathcal{A}||\mathcal{O}||\mathcal{Z}|^2 T n)$, in the two respective cases.

Therefore, the accumulative learning time, as a function of the number of episodes, is approximately on the order of $O(n^2)$ for batch-mode EM, while it is approximately on the order of $O(n)$ for online EM.

The batch algorithm stores all episodes, and thus it has an memory complexity of $O(n)$. The online algorithm stores only the latest episode; its memory complexity is $O(1)$.

4.3.2 Convergence Analysis

The convergence of the online algorithm depends on the asymptotic behavior of the noisy sufficient statistic in (4.15). If $\hat{\mathbf{s}}_n$ approaches to the accurate (noise-free) sufficient statistic $\mathbf{s}(\mathcal{D}^\infty, \boldsymbol{\theta}_n)$ as n is sufficiently large, the online algorithm converges to the batch-mode solution after n iterations; thus, the online algorithm also converges to the optimal RPR, considering that the batch-mode solution gives the optimal RPR when the number of episodes is sufficiently large (see the discussion below Definition 2.44).

We have mentioned below Definition 13 that $\lim_{K \rightarrow \infty} \hat{V}(\mathcal{D}^{(K)}; \Theta)$ is the true value function, and its maximization gives the optimal RPR. Therefore $\mathbf{s}(\mathcal{D}^{(\infty)}, \boldsymbol{\theta})$ represents the accurate (noise-free) sufficient statistic computed by $\boldsymbol{\theta}$ using all available

episodes. In other word, for any $\boldsymbol{\theta}$, $\mathbf{s}(\mathcal{D}^{(\infty)}, \boldsymbol{\theta})$ is the best sufficient statistic one would get in batch-mode learning.

The goal of our convergence analysis is to show that $\hat{\mathbf{s}}_n$ approaches $\mathbf{s}(\mathcal{D}^{(\infty)}, \boldsymbol{\theta}_n)$ as n is sufficiently large. With this holding true, the online EM iteration between (4.15) and (4.16) converges to the batch-mode solution after n iterations.

We begin the analysis by rewriting the recursion in (4.15) as

$$\hat{\mathbf{s}}_{n+1} = \hat{\mathbf{s}}_n + \alpha_{n+1}(\mathbf{s}(\varepsilon_{n+1}, \boldsymbol{\theta}_n) - \hat{\mathbf{s}}_n) = \hat{\mathbf{s}}_n + \alpha_{n+1}(\mathbf{g}(\hat{\mathbf{s}}_n) + \mathbf{e}_{n+1}), \quad (4.17)$$

where

$$\mathbf{g}(\hat{\mathbf{s}}_n) \stackrel{def.}{=} \mathbf{s}(\mathcal{D}^{(\infty)}, \hat{\boldsymbol{\theta}}(\hat{\mathbf{s}}_n)) - \hat{\mathbf{s}}_n, \quad (4.18)$$

is the bias, and

$$\mathbf{e}_{n+1} \stackrel{def.}{=} \mathbf{s}(\varepsilon_{n+1}, \boldsymbol{\theta}_n) - \mathbf{s}(\mathcal{D}^{(\infty)}, \hat{\boldsymbol{\theta}}(\hat{\mathbf{s}}_n)) \quad (4.19)$$

is a noise sequence. Let

$$\Delta \stackrel{def.}{=} \{\mathbf{s} : \mathbf{g}(\mathbf{s}) = \mathbf{0}\}. \quad (4.20)$$

With $\mathbf{g}(\mathbf{s})$ defined in (4.18), $\lim_{n \rightarrow \infty} \hat{\mathbf{s}}_n = \mathbf{s}(\mathcal{D}^{(\infty)}, \boldsymbol{\theta}_n)$ if and only if

$$\lim_{n \rightarrow \infty} \text{dist}(\hat{\mathbf{s}}_n, \Delta) = 0.$$

Definition 6. $\mathcal{S} = \{\mathbf{s} = \mathbf{s}(\mathcal{D}^{(K)}, \boldsymbol{\theta}) : \boldsymbol{\theta} \in \Xi, K \geq 1\}$, where Ξ is defined in Theorem 5.

Since $\nu_t^k \leq 1$ and $T_k < \infty$, $\|\mathbf{s}\| < \infty$ holds for any $\mathbf{s} \in \mathcal{S}$, by (4.5)-(4.7), and therefore, $\|\mathbf{e}_n\| < \infty$, $\forall n$, by (4.19).

The main result of our analysis is stated in Theorem 9, the proof of which is based on the following two lemmas. The lemmas are proven in the appendix of this chapter.

Lemma 7. *Define*

$$l(\mathbf{s}) \stackrel{def.}{=} -[\boldsymbol{\psi}(\hat{\boldsymbol{\theta}}(\mathbf{s}))]^T \mathbf{s}(\mathcal{D}^{(\infty)}, \hat{\boldsymbol{\theta}}(\hat{\mathbf{s}}_n)). \quad (4.21)$$

Then $\mathbf{g}^T(\mathbf{s}) \nabla_{\mathbf{s}} l(\mathbf{s}) \leq 0$.

Lemma 8. *Assume $\{\mathbf{e}_i\}_{i \geq n}$ are serially uncorrelated, with mean zero, if $\sum_{n=1}^{\infty} \alpha_n = \infty$ and $\sum_{n=1}^{\infty} \alpha_n^2 < \infty$, then $\lim_{n \rightarrow \infty} \sup_{k \geq n} \left| \sum_{i=n}^k \alpha_i \mathbf{e}_i \right| = 0$.*

Lemma 8 provides a characterization of the noise term in (4.19). The conditions $\sum_{n=1}^{\infty} \alpha_n = \infty$ and $\sum_{n=1}^{\infty} \alpha_n^2 < \infty$ are standard in stochastic approximation. Using the above lemmas, we prove the main result.

Theorem 9. *If $\sum_{n=1}^{\infty} \alpha_n = \infty$ and $\sum_{n=1}^{\infty} \alpha_n^2 < \infty$, then $\lim_{n \rightarrow \infty} \text{dist}(\hat{\mathbf{s}}_n, \Delta) = 0$.*

Theorem 9 shows that the noise in $\hat{\mathbf{s}}_n$ indeed vanishes as $n \rightarrow \infty$, under certain conditions on the learning rate $\{\alpha_n\}$. Thus convergence of the online algorithm is guaranteed. Theorem 4 can be proven using the techniques in [36] and [28]. The details can be found in the appendix of this chapter.

4.4 Experiments

We investigate the empirical performance of the proposed online EM algorithm for the RPR on five benchmark problems described in Table 4.1. The POMDP models for these problems are available at <http://www.cs.brown.edu/research/ai/pomdp/examples>. These models are assumed unknown to the agent; they are used as black boxes to simulate the episodes of agent-environment interactions. For all problems, an episode starts from a random position and ends when the goal is achieved.

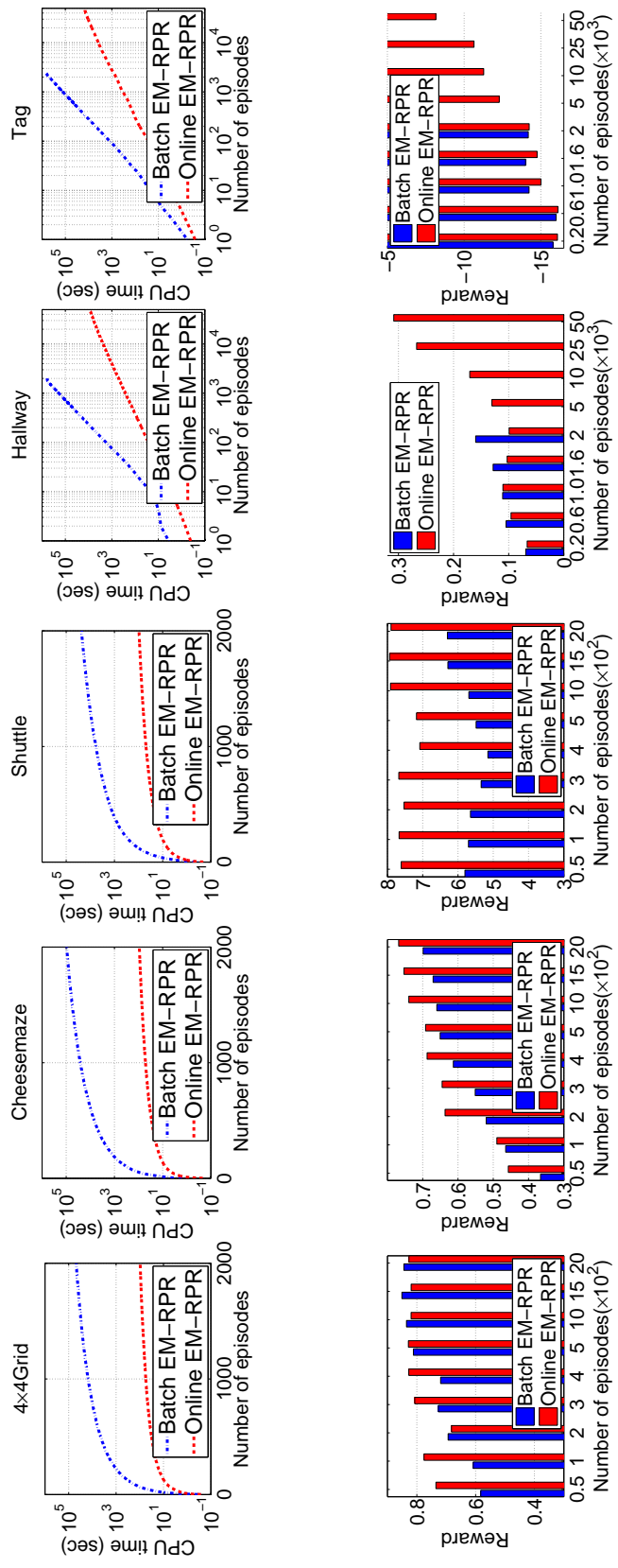


FIGURE 4.1: Performance Comparison between online and batch EM-RPR learning algorithms. Left column and right column correspond to averaged accumulative CPU time and reward respectively. The rows from top to bottom correspond to 5 benchmark problems; for all of the latter, the horizontal axis corresponds to the number of episodes seen by the agent when developing the policy.

Table 4.1: Five benchmark POMDP problems

Name	Shuttle	Cheesemaze	4×4 grid	Hallway	Tag
$ \mathcal{A} $	8	4	4	5	5
$ \mathcal{O} $	3	7	2	21	30
Number of States	3	11	16	60	870

Online RPR versus Batch RPR

We compare the performance of the proposed online EM-RPR against that of the batch EM-RPR. The policy is evaluated based on discounted accumulative reward (averaged over 5 independent runs, each using 251 (101) test episodes of at most 251 (101) steps for the first three (other) problems). The computation efficiency is compared based on the accumulative CPU time in the learning phase. The number of states is set to $|\mathcal{Z}| = 20$ for the first three small problems, $|\mathcal{Z}| = 40$ for Hallway and Tag. The comparison results are plotted in Figure 4.1.

As can be seen from the leftmost three columns of Figure 4.1, the online EM algorithm shows some advantages in performance, especially during the early stage of learning when episodes are collected by exploration. A comparison of CPU time shows that online EM is significantly more efficient than the batch method, and the time agrees with the complexity analysis in Section 4.3.1. Moreover it is noted that, in the Shuttle and Cheesemaze problems, batch EM is prone to a local optimal solution, whereas online EM is a stochastic algorithm which seems capable of jumping out of local optima and achieving a better solution.

The rightmost two columns in Figure 4.1 demonstrate that the online EM algorithm makes the RPR scale to larger problems. As can be seen, online EM achieves a better tradeoff between the number of episodes (n) and the policy quality. When n is small, batch EM is better than online EM, which can be explained by the fact that the batch method updates the sufficient statistics using all available data. However,

given the same amount of learning time, the online RPR achieves better policy performance; for example, by the time that online EM has processed 50,000 episodes and achieved a test reward of 0.3 for Hallway and -8 for Tag, batch EM has processed less than 1000 episodes, achieving only 0.1 and -14 for these two problems, respectively. These comparisons show that the RPR can scale up to larger POMDP problems with online learning.

The Performance as a Function of $|\mathcal{Z}|$

We consider six settings of $|\mathcal{Z}|$ to investigate the influence of the number of decision states on the RPR policy learned with online EM. Each policy is learned from 50000 episodes, with the intermediate policies evaluated every 500 episodes during the learning. The performance of a policy is measured by three overlapping metrics: (i) the discounted reward averaged over 251 test episodes, (ii) the percentage of test episodes in which goal is reached, (iii) the number of steps for reaching the goal (it is 251 when the goal is not reach within 251 steps) averaged over the 251 episodes. The results are reported in Figure 4.2. As shown in Figure 4.2, the RPR gives comparable convergent performances over a wide range of choices for $|\mathcal{Z}|$ (from 30 to 60), indicating robustness of the online EM-RPR to the number of decision states. However, when $|\mathcal{Z}|$ is too small, the RPR cannot accommodate a good policy, which explains why the RPR quickly converges to sub-optimality when $|\mathcal{Z}| = 10$. The degraded performance at $|\mathcal{Z}| = 10$ motivate us to infer $|\mathcal{Z}|$ online in our future work by using statistic test or online nonparametric Bayesian technique such as the hierarchical Dirichlet process [149].

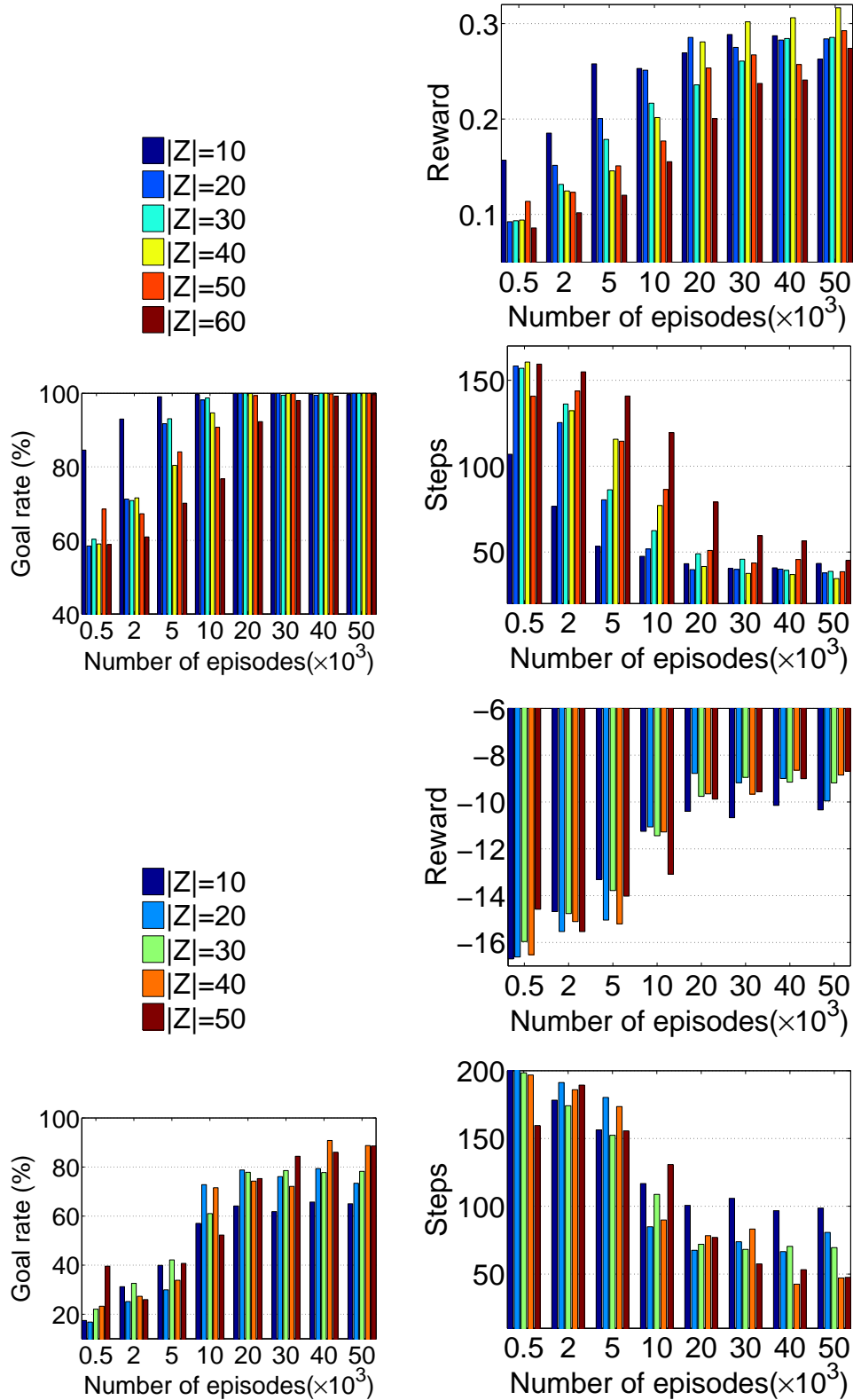


FIGURE 4.2: The results on Hallway (top two rows) and Tag (bottom two rows) produced by the online EM-RPR with various settings of $|Z|$.

Comparison with Other Methods

We compare the RPR policy learned with the proposed online EM algorithm to three existing algorithms, which all assume the POMDP models are unknown. The first one is McCallum’s U-tree [88], including its modified version [159], which, like the RPR, is a policy-based approach and does not attempt to learn the POMDP models. The second is iPOMDP[39], including its variant Policy Prior (PP) [38], which learn both POMDP models and control policies. The last competing algorithm is mixture learning [146], referred to MDP-EM, which is an online EM algorithm treating the observation in a POMDP as the state in an MDP. We do not repeat the experiments for the three competing methods, but rather cite the results from the literature. Since the cited results are based on different testing procedure, we report the comparison to each method separately, using the same test procedure as in the literature for each respective method.

We compare to the U-tree results as reported in [159]. To make the results comparable, we strictly follow the same procedures to set up our experiment for the RPR. Specifically, the experiment for each problem consists of 21 independent trials. In each trial, we consider three settings of $|\mathcal{Z}|$ for the RPR policy; for each setting, the RPR policy is learned using the proposed online EM algorithm, based on episodes collected by following a behavior policy for 75000 steps, and then the learned RPR policy is followed 25000 steps to evaluate the policy’s quality. The quality of a policy in each trial is measured by the average reward earned over the 25000 steps of evaluation. The results from the 21 trials are used to calculate the mean and standard deviation for the RPR in each setting of $|\mathcal{Z}|$, which are reported in Table 4.2 along with the results of U-trees cited from [159]. For RPR, we set $|\mathcal{Z}|$ to the values $\{10, 20, 30\}$, which are in the range around the number of the nodes inferred by the modified U-tree algorithm (the number in the braces behind the results for

Table 4.2: A comparison, in terms of averaged reward, to U-trees on small problems. The parenthesized integers indicate the numbers of nodes used in U-trees, which correspond to the numbers of decision states $|\mathcal{Z}|$ for the RPR.

METHODS	SHUTTLE	CHEESE MAZE	4 × 4 GRID
U-TREE	1.833 (62)	0.184 (53)	0.259 (47)
MODIFIED U-TREE	1.820 (25)	0.184 (19)	0.259 (20)
RPR ($ \mathcal{Z} = 10$)	1.76 ± 0.136	0.161 ± 0.018	0.259 ± 0.033
RPR ($ \mathcal{Z} = 20$)	1.824 ± 0.046	0.151 ± 0.021	0.207 ± 0.049
RPR ($ \mathcal{Z} = 30$)	1.820 ± 0.029	0.148 ± 0.022	0.219 ± 0.031

Table 4.3: A comparison of online EM-RPR to iPOMDP and Policy Prior (PP), in terms of cumulative reward.

PROBLEM	RPR (10)	RPR (40)	iPOMDP	PP-2	PP-3
HALLWAY	6.0 ± 2.8	3.0 ± 1.3	0.2	1.6	6.6
TAG	-4908 ± 95	-4987 ± 13	-16000	-7400	-3500

U-tree algorithms are the inferred nodes).

The RPR with online EM performs comparably as U-trees on Shuttle and performs better on 4 × 4 Grid. The RPR performs worse than U-trees on Cheese Maze for the given budget of 75000 steps of learning. However, we notice that the RPR’s performance can catch up after additional learning steps. Since the decision states in the RPR correspond to the nodes in U-trees, it is interesting to compare $|\mathcal{Z}|$ used by the RPR and the nodes generated by U-trees, which are shown as the parenthesized numbers in Table 4.2. As can be seen, the RPR generally needs less decision states to achieve superior performance, while U-trees generate more nodes than necessary.

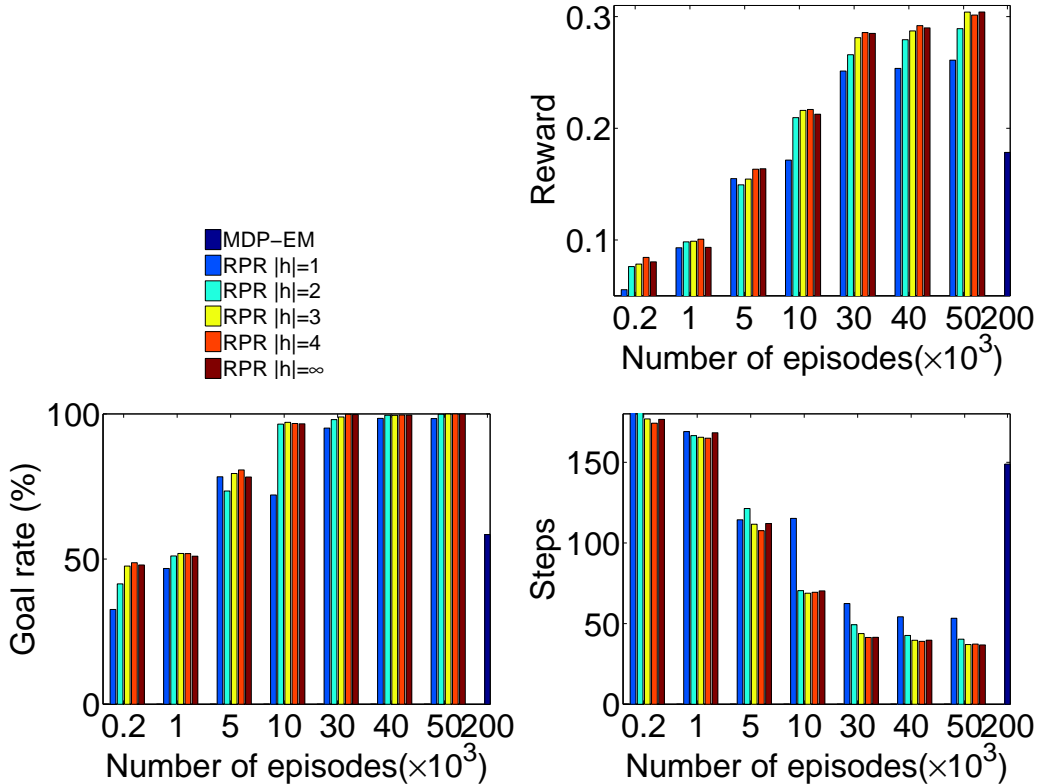


FIGURE 4.3: A comparison of online EM-RPR to EM-MDP on Hallway. $|h|$ is the number of observations. The performance of the EM-MDP is reported only for $n = 200 \times 10^3$.

To compare to iPOMDP [39] and policy prior [38], we follow their experimental setting and report the cumulative rewards as a function of the learning step (3500 steps for Hallway, 5000 steps for Tag). The results are summarized in Table 4.3, which show that online EM-RPR achieves significantly larger accumulative rewards than iPOMDP, and performs comparably to the policy prior methods which utilize expert information.

The MDP-EM method learns a reactive policy (it does not memorize past observations). To make the comparison fair, we consider the RPR policy conditioning on different numbers of observations (lengths of history). The policy evaluation procedure is the same as described in Section 4.4, with the MDP-EM policy provided by the authors of [146]. The results are plotted in Figure 4.3, where $|h|$ indicates the length

of history, e.g. $|h| = 1$ means only the current observation is used, which makes the RPR a reactive policy as the MDP-EM. As can be seen from Figure 4.3, the RPR policies perform generally better than the MDP-EM; the performance increases as the number of observations increases from one to three, and then becomes saturated when more observations are used. It is noted that the MDP-EM policy provided to us by the authors of [146] is learned from $n = 200,000$ episodes, while the online RPR policies are learned from only 50,000 episodes or less ($200 \leq n \leq 50,000$). The performance improvement with $|h| > 1$ is expected, considering that the MDP-EM policy uses only the current observation while the RPR also uses past observations. The improvement with $|h| = 1$ can be explained by the fact that the RPR uses μ as an initial guess of what local policy should be used. The reduction of learning episodes in the RPR may be attributed to the internal reward re-computation in (6.5). Since an episode with re-computed rewards functions like a new episode [76] for the improved policy, reward re-computation effectively multiplies the information for the update of sufficient statistics (for policy improvement in the next iteration).

4.5 Discussion

We then presented an online nested EM algorithm for learning the RPR, a parametric policy for RL in POMDPs. The online algorithm uses only the latest learning episode to update the policy's value in the outer-loop E-step and update the sufficient statistic in the inner-loop E-step, and computes the improved policy in closed-form given the sufficient statistic. The algorithm reduces both the time and memory complexity an order of magnitude, in comparison with the corresponding batch-mode algorithm. Under standard conditions on the learning rates, the online algorithm provably converges to the optimal batch-mode policy when the number of learning episodes becomes sufficiently large. Experimental results on five benchmark POMDP problems show that: (i) the online RPR produces policies as good as the batch-mode

RPR by using only a small fraction of the time; (ii) the stochastic nature of the online algorithm can help it to jump out of local optima; (iii) model-free methods have the potential to learn better policies than model-based methods; (vi) the RPR can produce high-quality policies by using fewer internal memory units than the U-tree; (v) the RPR has a performance increasing with the history length, and is a competitive reactive policy when the length is one. Future work includes online inference of the number of decision states and multi-agent online learning.

4.6 Appendix

Proof of Lemma 7. Recall $l(\mathbf{s}) = -f(\hat{\boldsymbol{\theta}}(\mathbf{s}), \mathbf{s}(\mathcal{D}^{(\infty)}, \hat{\boldsymbol{\theta}}(\mathbf{s})))$. Application of the chain rule of differentiation gives

$$\begin{aligned}\nabla_{\mathbf{s}} l(\mathbf{s}) &= -\nabla_{\mathbf{s}} \hat{\boldsymbol{\theta}}^T(\mathbf{s}) \nabla_{\boldsymbol{\theta}} \boldsymbol{\psi}^T(\hat{\boldsymbol{\theta}}(\mathbf{s})) \mathbf{s}(\mathcal{D}^{(\infty)}, \hat{\boldsymbol{\theta}}(\mathbf{s})), \\ &= -\nabla_{\mathbf{s}} \hat{\boldsymbol{\theta}}^T(\mathbf{s}) \nabla_{\boldsymbol{\theta}} \boldsymbol{\psi}^T(\hat{\boldsymbol{\theta}}(\mathbf{s})) [\mathbf{s} + \mathbf{g}(\mathbf{s})],\end{aligned}\tag{4.22}$$

Since $\hat{\boldsymbol{\theta}}(\mathbf{s})$ minimizes $-f(\boldsymbol{\theta}; \mathbf{s})$, one has

$$\mathbf{0} = \nabla_{\boldsymbol{\theta}} f(\boldsymbol{\theta}; \mathbf{s})|_{\boldsymbol{\theta}=\hat{\boldsymbol{\theta}}(\mathbf{s})} = \nabla_{\boldsymbol{\theta}} \boldsymbol{\psi}^T(\hat{\boldsymbol{\theta}}(\mathbf{s})) \mathbf{s},\tag{4.23}$$

which reduces (4.22) to

$$\nabla_{\mathbf{s}} l(\mathbf{s}) = -\nabla_{\mathbf{s}} \hat{\boldsymbol{\theta}}^T(\mathbf{s}) [\nabla_{\boldsymbol{\theta}} \boldsymbol{\psi}^T(\hat{\boldsymbol{\theta}}(\mathbf{s})) \mathbf{g}(\mathbf{s})].\tag{4.24}$$

Differentiating both sides of (4.23) w.r.t. \mathbf{s} , we obtain

$$\begin{aligned}\mathbf{0} &= \nabla_{\boldsymbol{\theta}} \boldsymbol{\psi}^T(\hat{\boldsymbol{\theta}}(\mathbf{s})) + \nabla_{\mathbf{s}} \hat{\boldsymbol{\theta}}^T(\mathbf{s}) \nabla_{\boldsymbol{\theta}}^2 \boldsymbol{\psi}(\hat{\boldsymbol{\theta}}(\mathbf{s})) \mathbf{s} \\ &= \nabla_{\boldsymbol{\theta}} \boldsymbol{\psi}^T(\hat{\boldsymbol{\theta}}(\mathbf{s})) + \nabla_{\mathbf{s}} \hat{\boldsymbol{\theta}}^T(\mathbf{s}) \nabla_{\boldsymbol{\theta}}^2 f(\boldsymbol{\theta}; \mathbf{s})|_{\boldsymbol{\theta}=\hat{\boldsymbol{\theta}}(\mathbf{s})},\end{aligned}$$

from which follows

$$\nabla_{\mathbf{s}} \hat{\boldsymbol{\theta}}^T(\mathbf{s}) = -\nabla_{\boldsymbol{\theta}} \boldsymbol{\psi}^T(\hat{\boldsymbol{\theta}}(\mathbf{s})) \{ \nabla_{\boldsymbol{\theta}}^2 f(\boldsymbol{\theta}; \mathbf{s})|_{\boldsymbol{\theta}=\hat{\boldsymbol{\theta}}(\mathbf{s})} \}^{-1},$$

which is plugged into (4.24) to give

$$\begin{aligned} \mathbf{g}^T(\mathbf{s})\nabla_{\mathbf{s}}l(\mathbf{s}) &= \mathbf{g}^T(\mathbf{s})\nabla_{\boldsymbol{\theta}}\boldsymbol{\psi}^T(\hat{\boldsymbol{\theta}}(\mathbf{s}))[\nabla_{\boldsymbol{\theta}}^2f(\boldsymbol{\theta}; \mathbf{s})|_{\boldsymbol{\theta}=\hat{\boldsymbol{\theta}}(\mathbf{s})}]^{-1} \\ &\times \left[\nabla_{\boldsymbol{\theta}}\boldsymbol{\psi}^T(\hat{\boldsymbol{\theta}}(\mathbf{s}))\mathbf{g}(\mathbf{s}) \right] \leq \mathbf{0}, \end{aligned} \quad (4.25)$$

where the inequality arises because $\nabla_{\boldsymbol{\theta}}^2f(\boldsymbol{\theta}; \mathbf{s})|_{\boldsymbol{\theta}=\hat{\boldsymbol{\theta}}(\mathbf{s})}$ is a negative definite matrix, and the inequality reduces to an equality if and only if $\nabla_{\boldsymbol{\theta}}\boldsymbol{\psi}^T(\hat{\boldsymbol{\theta}}(\mathbf{s}))\mathbf{g}(\mathbf{s}) = \mathbf{0}$. However, $\nabla_{\boldsymbol{\theta}}\boldsymbol{\psi}^T(\hat{\boldsymbol{\theta}}(\mathbf{s}))$ is a full-rank diagonal matrix, and thus the inequality reduces to an equality if and only if $\mathbf{g}(\mathbf{s}) = \mathbf{0}$. \square

Proof of Lemma 8. Let $M_{n,k} = \sum_{i=n}^k \alpha_i \mathbf{e}_i$. One can show that $\mathbb{E}[\mathbf{e}_n] = \mathbf{0} \forall n$ and $\mathbb{E}(\mathbf{e}_n^T \mathbf{e}_m) = 0 \forall m \neq n$ [72]. Thus

$$\mathbb{E}[M_{n,k}^2] \leq 2 \sup \|\mathbf{s}\|^2 \sum_{i=n}^k \alpha_i^2.$$

Using Chebyshev's inequality we have

$$\begin{aligned} \mathbb{P}(\sup_{k \geq n} |M_{n,k}| \geq \eta) &\leq \frac{\mathbb{E}[M_{n,k}^2]}{\eta^2}, \\ &\leq \frac{2}{\eta^2} \sup \|\mathbf{s}\|^2 \sum_{i=n}^{\infty} \alpha_i^2, \end{aligned} \quad (4.26)$$

Because $\lim_{n \rightarrow \infty} \sum_{i=n}^{\infty} \alpha_i^2 = 0$, it follows from (4.26) that

$$\lim_{n \rightarrow \infty} \mathbb{P} \left(\sup_{k \geq n} \left| \sum_{i=n}^k \alpha_i \mathbf{e}_i \right| \geq \eta \right) = 0, \quad (4.27)$$

which implies, using the fact that $\eta > 0$ is arbitrarily chosen, that

$$\lim_{n \rightarrow \infty} \sup_{k \geq n} \left| \sum_{i=n}^k \alpha_i \mathbf{e}_i \right| = \mathbf{0}. \quad (4.28)$$

The claim thus follows. \square

Proof of Theorem 9. We first prove the sequence $\{l(\hat{\mathbf{s}}_n)\}_{n \geq 0}$ converges, and then show that $\{\hat{\mathbf{s}}_n\}_{n > 0}$ converges to Δ . We begin with the definition

$$\mathbf{s}'_n = \hat{\mathbf{s}}_n + \sum_{i=n+1}^{\infty} \alpha_i \mathbf{e}_i. \quad (4.29)$$

Plugging $\hat{\mathbf{s}}_n$ given by (4.17) into (4.29), we have a recursive expression $\mathbf{s}'_n = \mathbf{s}'_{n-1} + \alpha_n \mathbf{g}(\hat{\mathbf{s}}_{n-1})$. Since \mathbf{g} is bounded on \mathcal{S} , one has

$$\|\mathbf{s}'_n - \mathbf{s}'_{n-1}\| \leq \alpha_n \sup \|\mathbf{g}(\hat{\mathbf{s}}_{n-1})\|, \quad (4.30)$$

and $\{\mathbf{s} : \mathbf{s} = \beta \mathbf{s}'_n + (1 - \beta) \mathbf{s}'_{n-1}, \beta \in [0, 1]\} \subset \mathcal{S}$.

By the mean-value theorem, there exist $t \in [0, 1]$, such that

$$\begin{aligned} l(\mathbf{s}'_n) &= l(\mathbf{s}'_{n-1}) + \langle \nabla_{\mathbf{s}} l(\mathbf{s}'_{n-1} + t\alpha_n \mathbf{g}(\hat{\mathbf{s}}_n)), \mathbf{s}'_n - \mathbf{s}'_{n-1} \rangle \\ &= l(\mathbf{s}'_{n-1}) + \alpha_n \langle \nabla_{\mathbf{s}} l(\mathbf{s}'_{n-1} + t\alpha_n \mathbf{g}(\hat{\mathbf{s}}_n)), \mathbf{g}(\hat{\mathbf{s}}_{n-1}) \rangle \\ &= l(\mathbf{s}'_{n-1}) + \alpha_n \langle \nabla_{\mathbf{s}} l(\mathbf{s}'_{n-1}), \mathbf{g}(\mathbf{s}'_{n-1}) \rangle + \alpha_n \zeta_n \end{aligned} \quad (4.31)$$

where,

$$\begin{aligned} \zeta_n &= \langle \nabla_{\mathbf{s}} l(\mathbf{s}'_{n-1}), \mathbf{g}(\hat{\mathbf{s}}_{n-1}) - \mathbf{g}(\mathbf{s}'_{n-1}) \rangle + \\ &\quad \langle \nabla_{\mathbf{s}} l(\mathbf{s}'_{n-1} + t\alpha_n \mathbf{g}(\hat{\mathbf{s}}_n)) - \nabla_{\mathbf{s}} l(\mathbf{s}'_{n-1}), \mathbf{g}(\hat{\mathbf{s}}_{n-1}) \rangle. \end{aligned} \quad (4.32)$$

By the continuity of $\nabla_{\mathbf{s}} l$ and \mathbf{g} , $\zeta_n \rightarrow 0$ as $n \rightarrow \infty$; thus $\zeta_n \in o(1)$ and

$$l(\mathbf{s}'_n) = l(\mathbf{s}'_{n-1}) + \alpha_n \langle \nabla l(\mathbf{s}'_{n-1}), \mathbf{g}(\mathbf{s}'_{n-1}) \rangle + o(\alpha_n) \quad (4.33)$$

Note that $o(\alpha_n)$ goes to 0 faster than the second term $\alpha_n \langle \nabla l(\mathbf{s}'_{n-1}), \mathbf{g}(\mathbf{s}'_{n-1}) \rangle$ on the right hand side of (4.33). Since $\langle \nabla l(\mathbf{s}'_{n-1}), \mathbf{g}(\mathbf{s}'_{n-1}) \rangle \leq 0$ is bounded, (4.33) implies that there exist $M < \infty$ such that for n large enough

$$|l(\mathbf{s}'_n) - l(\mathbf{s}'_{n-1})| \leq M\alpha_n. \quad (4.34)$$

(4.30) implies $\{\mathbf{s}'_n\}_{n>0}$ is a Cauchy sequence, thus $\{\mathbf{s}'_n\}_{n>0}$ has a limit. Since l is the lower bound of the log-value function which is finite, hence $\lim_{n \rightarrow \infty} l(\mathbf{s}'_n)$ exists. Moreover, by Lemma 8, $\hat{\mathbf{s}}_n \rightarrow \mathbf{s}'_n$, thus we have $\lim_{n \rightarrow \infty} l(\mathbf{s}'_n) = \lim_{n \rightarrow \infty} l(\hat{\mathbf{s}}_n)$.

Next we show $\{\hat{\mathbf{s}}_n\}_{n>0}$ converges to Δ . Let $\mathcal{N} \subset \mathcal{S}$ to be an arbitrary open set containing Δ . If $\mathbf{s}'_{n-1} \in \mathcal{N}$, we have from (4.34) that

$$l(\mathbf{s}'_n) \leq l(\mathbf{s}'_{n-1}) + M\alpha_n. \quad (4.35)$$

If $\mathbf{s}'_{n-1} \notin \mathcal{N}$, by Lemma 7 there exist $\epsilon_{\mathcal{N}} > 0$, such that for n large enough, $\langle \nabla l(\mathbf{s}'_{n-1}), g(\mathbf{s}'_{n-1}) \rangle \leq -2\epsilon_{\mathcal{N}} < 0$ and $\zeta_n \leq \epsilon_{\mathcal{N}}$, we have from (4.31) that

$$l(\mathbf{s}'_n) \leq l(\mathbf{s}'_{n-1}) - \alpha_n \epsilon_{\mathcal{N}}. \quad (4.36)$$

Combining (4.35) and (4.36), we obtain

$$l(\mathbf{s}'_n) \leq l(\mathbf{s}'_{n-1}) - \alpha_n \epsilon_{\mathcal{N}} + \alpha_n (M + \epsilon_{\mathcal{N}}) \mathbb{I}(\mathbf{s}'_{n-1} \in \mathcal{N}). \quad (4.37)$$

Since $l(\mathbf{s}'_n)$ converges and $\sum_{n=1}^{\infty} \alpha_n^2 < \infty$, for any $\lambda > 0$, there exist an integer $N_{\lambda} < \infty$ such that $|l(\mathbf{s}'_n) - l(\mathbf{s}'_p)| \leq \lambda$, $N_{\lambda} \leq n \leq p$ and $\alpha_k \epsilon_{\mathcal{N}} < \lambda$, for any $k \geq N_{\lambda}$. By adding and subtracting consecutive terms between $l(\mathbf{s}'_n)$ and $l(\mathbf{s}'_p)$, we have

$$\begin{aligned} \lambda &\geq |l(\mathbf{s}'_n) - l(\mathbf{s}'_p)| \geq l(\mathbf{s}'_n) - l(\mathbf{s}'_p) = l(\mathbf{s}'_n) - l(\mathbf{s}'_{n+1}) + \cdots + l(\mathbf{s}'_{p-1}) - l(\mathbf{s}'_p) \\ &\geq \epsilon_{\mathcal{N}} \sum_{k=n+1}^p \alpha_k - (M + \epsilon_{\mathcal{N}}) \sum_{k=n+1}^p \alpha_k \mathbb{I}(\mathbf{s}'_{k-1} \in \mathcal{N}). \end{aligned} \quad (4.38)$$

where the last inequality is from (4.37). Since $\alpha_k \epsilon_{\mathcal{N}} < \lambda$ and $\sum_{n=1}^{\infty} \alpha_n = \infty$, denote by p^* the first integer larger than n such that $\lambda/\epsilon_{\mathcal{N}} < \sum_{k=n+1}^{p^*} \alpha_k < 2\lambda/\epsilon_{\mathcal{N}}$. Comparing with (4.38), we can deduce that there exist $k^* \in [n, p^*]$ such that $\mathbf{s}'_{k^*} \in \mathcal{N}$. Denote $K = \sup \|g(\mathbf{s}'_n)\|$ which is a finite constant. According to (4.30), for all $n \geq 1$, $\|\mathbf{s}'_n - \mathbf{s}'_{n-1}\| \leq K\alpha_n$, therefore we have $\|\mathbf{s}'_{k^*} - \mathbf{s}'_n\| \leq K \sum_{m=n+1}^{k^*} \alpha_m \leq K \sum_{m=n+1}^{p^*} \alpha_m \leq 2K\lambda/\epsilon_{\mathcal{N}}$. Since $\mathbf{s}'_{k^*} \in \mathcal{N}$ and λ can be arbitrarily small, thus $\lim_{n \rightarrow \infty} \text{dist}(\mathbf{s}'_n, \mathcal{N}) = 0$. Moreover, since \mathcal{N} is arbitrary, we have $\lim_{n \rightarrow \infty} \text{dist}(\mathbf{s}'_n, \Delta) = 0$. Using definition (4.29) and Lemma 8, we have $\{\hat{\mathbf{s}}_n\}_{n>0}$ converges to Δ . \square

The Infinite Regionalized Policy Representation for POMDPs

An indispensable ingredient for RL in POMDPs, for both of model-based and model-free approaches, is a mechanism for maintaining a proper balance between exploration and exploitation. Until the current policy is optimal, the agent should always explore the consequences of actions that are not encouraged by the current policy, to see whether the new actions will lead to higher expected long-term rewards. Exploration is the only way to ensure continual improvement of the policy. However, excessive exploration makes the policy converge unnecessarily slowly. To keep a balance, the agent needs to switch appropriately between exploration and exploitation.

Model-based approaches usually employ Bayesian RL for an implicit exploration and exploitation trade-off, treating the uncertain POMDP parameters as additional states, and attempting to solve an augmented POMDP with the model uncertainty incorporated into the augmented belief states [110]. However, the augmented POMDP is intractable and approximations are used. The approximations are usually based on policy-solving of model samples, ignoring the model uncertainty in future steps [39, 38]; as a result, their ability to balance exploration and exploitation is limited.

There has been little work on exploration and exploitation in POMDPs using policy-based approaches. One recent study addressing this problem is reported in [25], in which an explicit exploration and exploitation algorithm is given for POMDPs, employing an idea motivated by E^3 [66] and R-MAX [21], two RL algorithms in Markov decision processes (MDPs). The method employs a primary policy for choosing the regular actions, and an auxiliary policy for switching between exploration and exploitation. The primary policy is a regionalized policy representation (RPR) [76]. The auxiliary policy is affiliated with the primary one, using the same RPR but with a different set of local policies (see Section 2.2.2).

The RL algorithm in [25] is guaranteed to converge to the optimal policy. However, the optimality is based on the assumption that the RPR has an appropriate number of decision states. In practice, this number is never known and has to be set manually. When this number is too small, the RPR cannot express the optimal policy; when it is too large, the RPR requires an unnecessarily large amount of exploration to converge. Therefore, an appropriate choice of this number is crucial to the success of the algorithm.

In this chapter, we introduce the infinite regionalized policy presentation (iRPR) as a solution to this problem. An iRPR is an extension of the RPR where the parameters are *a priori* drawn from the hierarchical Dirichlet process (HDP) [139], which allows an infinite number of decision states and yet is biased towards a small number of states. Given the agent experiences, we infer the number of decision states while maintaining a proper balance between exploration and exploitation. We provide theoretical analysis which guarantees the iRPR converges to the optimal policy as the rate of exploration decreases to zero. Experiments on benchmark problems demonstrate the performance of the iRPR.

5.1 Bayesian Policy Learning

In addition to the expectation and maximization algorithm introduced in the previous chapter, a Bayesian approach can also be used to learn the RPR. which employs $\widehat{V}(\mathcal{D}^{(K)}; \Theta)$ as the likelihood function of Θ given the episodes $\mathcal{D}^{(K)}$. This Bayesian approach is nonstandard, as $\widehat{V}(\mathcal{D}^{(K)}; \Theta)$ is not equal to the probability of $\mathcal{D}^{(K)}$ given Θ . But this does not prevent one from obtaining a legitimate posterior of Θ , because, during the posterior inference, one is only interested in $\widehat{V}(\mathcal{D}^{(K)}; \Theta)$ as a function of Θ . Since the Bayesian approach forms the basis for the technical developments in Section 2.2.2, we provide a review of it below.

With a prior distribution $G_0(\Theta)$, the posterior is defined as

$$p(\Theta|\mathcal{D}^{(K)}) \stackrel{def.}{=} \widehat{V}(\mathcal{D}^{(K)}; \Theta)G_0(\Theta) [\widehat{V}(\mathcal{D}^{(K)})]^{-1} \quad (5.1)$$

where $\widehat{V}(\mathcal{D}^{(K)}) = \int \widehat{V}(\mathcal{D}^{(K)}; \Theta)G_0(\Theta)d\Theta$ is the marginal empirical value. The posterior generally does not have an analytic form. However, by employing the variational Bayesian technique [11], one may obtain an approximation to the posterior, along with the following byproducts: approximations to $p(z_{0:t}^k|a_{0:t}^k, o_{1:t}^k)$, $\forall t, k$, and an approximation to the nonnegative sequence $\nu = \{\nu_{0:T_k}^k\}_{k=1}^K$, where $\nu_t^k = \frac{\gamma^t r_t^k p(a_{0:t}^k|o_{1:t}^k)}{\prod_{\tau=0}^t p^\Pi(a_\tau^k|h_\tau^k) \widehat{V}(\mathcal{D}^{(K)})}$ is a rescaled discounted reward¹ averaged over the RPRs drawn from G_0 [76]. The rescaling leads to $\frac{1}{K} \sum_{k=1}^K \sum_{t=0}^{T_k} \nu_t^k = 1$, which normalizes $\widehat{V}(\mathcal{D}^{(K)})$ to a unit. Denote by $g(\Theta)$ the approximation to $p(\Theta|\mathcal{D}^{(K)})$, by $\hat{\nu} = \{\hat{\nu}_{0:T_k}^k\}_{k=1}^K$ the approximation to ν , and by $q_t^k(z_{0:t}^k)$ the approximation to $p(z_{0:t}^k|a_{0:t}^k, o_{1:t}^k)$, $\forall t, k$. Letting $\text{KL}(q||p)$ denote the Kullback-Leibler (KL) distance between probability measures q and p , the approximations are found by point-wise maximization of the lower bound

¹ The ν_t^k results from the fact one is evaluating the RPR using episodes collected by a different policy Π .

of log marginal empirical value function $\widehat{V}(\mathcal{D}^{(K)})$

$$\begin{aligned} \text{LB}(g(\Theta), \hat{\nu}, \{q_t^k\}) = & \epsilon \widehat{V}(\mathcal{D}^{(K)}) - \text{KL}\left(\frac{\hat{\nu}}{K} \parallel \frac{\nu}{K}\right) \\ & - \frac{1}{K} \sum_{k=1}^K \sum_{t=0}^{T_k} \hat{\nu}_t^k \text{KL}(q_t^k(z_{0:t}^k)g(\Theta) \parallel p(z_{0:t}^k, \Theta | a_{0:t}^k, o_{1:t}^k)), \end{aligned} \quad (5.2)$$

subject to the non-negativity and normalization constraints on $g(\Theta)$, $\frac{\hat{\nu}}{K}$, $\{q_t^k\}$. The first term on the right side, log of the marginal empirical value, is a constant, thus the maximization is equivalent to minimization of the KL distance between each approximation and the associated true. The joint distribution $p(z_{0:t}^k, \Theta | a_{0:t}^k, o_{1:t}^k)$ is factorized into a product of two marginals, i.e., $q_t^k(z_{0:t}^k)g(\Theta)$, in the approximation, and their KL distance is minimized in proportion to the weight $\hat{\nu}_t^k$. Since the weight sequence $\hat{\nu}$ is an approximation to the reward sequence ν , this ensures the approximations associated with higher rewards are more accurate.

Maximization of (5.2) leads to analytic solutions when G_0 is a product of Dirichlet distributions,

$$\begin{aligned} G_0(\Theta) = & \left[\text{Dir}(\mu_{1:|\mathcal{Z}|} | v_{1:|\mathcal{Z}|}) \right] \left[\prod_{i=1}^{|\mathcal{Z}|} \text{Dir}(\pi_{1:|\mathcal{A}|}^i | \rho_{1:|\mathcal{A}|}^i) \right] \\ & \times \left[\prod_{a=1}^{|\mathcal{A}|} \prod_{o=1}^{|\mathcal{O}|} \prod_{i=1}^{|\mathcal{Z}|} \text{Dir}(W_{1:|\mathcal{Z}|}^{iao} | \omega_{1:|\mathcal{Z}|}^{iao}) \right], \end{aligned} \quad (5.3)$$

with hyper-parameters (v, ρ, ω) , where $v = v_{1:|\mathcal{Z}|}$, $\rho = \{\rho_{1:|\mathcal{A}|}^i\}_{i=1}^{|\mathcal{Z}|}$, and

$$\omega = \{\omega_{1:|\mathcal{Z}|}^{iao}\}_{i=1:|\mathcal{Z}|, a=1:|\mathcal{A}|, o=1:|\mathcal{O}|}.$$

The solutions, which are given in [76], are re-stated in Theorem 10.

Theorem 10. *Let $g(\Theta)$ initially be the form of (5.3) with hyper-parameters $(\hat{v}, \hat{\rho}, \hat{\omega})$, then iterative application of the following updates leads to monotonic increase of*

(5.2), until convergence to a maxima. The updates of \hat{v} and $\{q_t^k\}$ are given by

$$\hat{v}_t^k = \frac{\gamma^t r_t^k p(a_{0:t}^k | o_{1:t}^k, \tilde{\Theta})}{\prod_{\tau=0}^t p^\Pi(a_\tau^k | h_\tau^k) \widehat{V}(\mathcal{D}^{(K)} | \tilde{\Theta})}, \forall t, k, \quad (5.4)$$

$$q_t^k(z_{0:t}^k) = p(z_{0:t}^k | a_{0:t}^k, o_{1:t}^k, \tilde{\Theta}), \forall t, k, \quad (5.5)$$

where $\tilde{\Theta} = \{\tilde{\pi}, \tilde{\mu}, \tilde{W}\}$ is a set of under-normalized probability mass functions², with $\tilde{\pi}_m^i = e^{\psi(\hat{\rho}_m^i) - \psi(\sum_{a=1}^{|\mathcal{A}|} \hat{\rho}_a^i)}$, $\tilde{\mu}_i = e^{\psi(\hat{v}_i) - \psi(\sum_{j=1}^{|\mathcal{Z}|} \hat{v}_j)}$, and $\tilde{W}_j^{iao} = e^{\psi(\hat{\omega}_j^{iao}) - \psi(\sum_{z=1}^{|\mathcal{Z}|} \hat{\omega}_z^{iao})}$, and ψ is the digamma function. The hyper-parameters of $g(\Theta)$ are updated as

$$\begin{aligned} \hat{v}_i &= v_i + \frac{1}{K} \sum_{k=1}^K \sum_{t=0}^{T_k} \hat{v}_t^k \phi_{t,0}^k(i) \\ \hat{\rho}_a^i &= \rho_a^i + \frac{1}{K} \sum_{k=1}^K \sum_{t=0}^{T_k} \hat{v}_t^k \sum_{\tau=0}^t \phi_{t,\tau}^k(i) \delta(a_\tau^k, a) \\ \hat{\omega}_j^{iao} &= \omega_j^{iao} + \frac{1}{K} \sum_{k=1}^K \sum_{t=0}^{T_k} \hat{v}_t^k \sum_{\tau=1}^t \xi_{t,\tau-1}^k(i, j) \\ &\quad \times \delta(a_{\tau-1}^k, a) \delta(o_\tau^k, o), \end{aligned} \quad (5.6)$$

where

$$\xi_{t,\tau}^k(i, j) = p(z_\tau^k = i, z_{\tau+1}^k = j | a_{0:t}^k, o_{1:t}^k, \tilde{\Theta}) \quad (5.7)$$

$$\phi_{t,\tau}^k(i) = p(z_\tau^k = i | a_{0:t}^k, o_{1:t}^k, \tilde{\Theta}) \quad (5.8)$$

are marginals of $q_t^k(z_{0:t}^k)$.

5.2 The Infinite RPR

Based on the Bayesian policy learning framework introduced in the last section, we are ready to present the Bayesian nonparametric policy model: iRPR.

Definition 11. *The infinite regionalized policy representation (iRPR) is a tuple $(\mathcal{A}, \mathcal{O}, \mathcal{Z}, \lambda, \alpha, \rho)$, where \mathcal{A} and \mathcal{O} are as in Definition 1; \mathcal{Z} is an unbounded set of*

² Note that $q_t^k(z_{0:t}^k) = p(z_{0:t}^k | a_{0:t}^k, o_{1:t}^k, \tilde{\Theta})$ is always properly normalized by $p(a_{0:t}^k | o_{1:t}^k, \tilde{\Theta}) = \sum_{z_0, \dots, z_t=1}^{|\mathcal{Z}|} p(z_{0:t}^k, a_{0:t}^k | o_{1:t}^k, \tilde{\Theta})$.

decision states indexed by positive integers; (λ, α, ρ) determine (W, μ, π) , the RPR parameters in Definition 1, as follows (notations described under Definition 1 in section 2.2.2),

$$\begin{aligned}\eta_i &\sim \text{Beta}(1, \lambda), \quad i = 1, 2, \dots, \infty, \\ \beta_i &= \eta_i \prod_{j=1}^{i-1} (1 - \eta_j), \quad i = 1, 2, \dots, \infty, \\ W_{1:\infty}^{iao} &\sim \text{DP}(\alpha, \beta_{1:\infty}), \quad i = 1, 2, \dots, \infty, \\ \mu_{1:\infty} &\sim \text{DP}(\alpha, \beta_{1:\infty}), \\ \pi_{1:|\mathcal{A}|}^i &\sim \text{Dir}(\cdot | \rho_{1:|\mathcal{A}|}), \quad i = 1, 2, \dots, \infty,\end{aligned}$$

$\forall a \in \mathcal{A}$ and $\forall o \in \mathcal{O}$, where $\text{DP}(\alpha, \beta_{1:\infty})$ denotes a Dirichlet process [44] with concentration α and base probability measure $\beta_{1:\infty}$.

In Definition 11, \sim denotes sampling from a distribution proportional to the right side. The iRPR is defined based on the hierarchical Dirichlet process (HDP) [139], which places a nonparametric prior on $\Theta = (W, \mu, \pi)$. The stick-breaking weights $\beta_{1:\infty}$ specify a probability measure on $\mathcal{Z} = \{1, 2, \dots, \infty\}$. The initial state distribution (μ) and the next-state distributions (W) are independently drawn from $\text{DP}(\alpha, \beta_{1:\infty})$. The local policies π are drawn independently from a Dirichlet distribution with parameters $\rho_{1:|\mathcal{A}|}$.

Assuming $\rho_a = 1/|\mathcal{A}|$, $\forall a \in \mathcal{A}$ (i.e., the agent takes random actions *a priori*), we are interested in learning the RPR parameters Θ , along with the hyper-parameters (λ, α) if inference of the latter is desirable. The learning is based on a hybrid Gibbs-variational algorithm, which iteratively samples the decision-state occupancies $\{z_t^k\}$, the RPR parameters Θ , the latent parameters $\beta_{1:\infty}$, and the hyper-parameters (λ, α) (if desirable), with the samples of one group of parameters conditional on all other parameters.

During the inference, one may employ Theorem 10 as a basic ingredient. To see why this is possible, we first note that the nonparametric prior can be expressed

using the parametric prior in (5.3) by introducing a special state z^* summarizing all decision states currently unoccupied by the episodes. Given that the number of (distinct) unoccupied decision states is n , one may write a parametric prior as

$$G_0^n(\Theta) = \left[\text{Dir}(\mu_{1:n+1} | \alpha \beta_{1:n+1}) \right] \left[\prod_{i=1}^{n+1} \text{Dir}(\pi_{1:|\mathcal{A}|}^i | \rho_{1:|\mathcal{A}|}) \right] \\ \times \left[\prod_{a=1}^{|\mathcal{A}|} \prod_{o=1}^{|\mathcal{O}|} \prod_{i=1}^{n+1} \text{Dir}(W_{1:n+1}^{iao} | \alpha \beta_{1:n+1}) \right], \quad (5.9)$$

where z^* is indicated by $n + 1$, $\{\beta_i\}_{i=1}^n$ are the same as in definition 11 and $\beta_{n+1} = 1 - \sum_{i=1}^n \beta_i$.

Given $G_0^n(\Theta)$, one may employ Theorem 10 to obtain $\hat{\nu}$, $\{q_t^k(z_{0:t}^k)\}_{\forall t,k}$, and $g(\Theta)$, which are the approximations to the rescaled rewards ν , the decision-state posterior $\{p(z_{0:t}^k | a_{0:t}^k, o_{1:t}^k)\}_{\forall t,k}$, and the RPR posterior $p(\Theta | \mathcal{D}^{(K)})$, respectively. Given $\{q_t^k(z_{0:t}^k)\}_{\forall t,k}$, one can obtain decision-state occupancies $\{z_{t,0:t}^k\}_{t=0:T_k, k=1:K}$, where $z_{t,0:t}^k \sim q_t^k(z_{0:t}^k)$ with $q_t^k(z_{0:t}^k)$ given in (5.5). Let \mathbb{I}_s be an indicator function that equals one if s is true and zero otherwise. Define

$$\varphi_j^{iao} = \sum_{k=1}^K \sum_{t=1}^{T_k} \hat{\nu}_t^k \sum_{\tau=1}^t \mathbb{I}_{z_{t,\tau-1}^k=i, a_{\tau-1}^k=a, o_{\tau}^k=o, z_{t,\tau}^k=j}$$

which is the reward-weighted sum of transitions from i to j given that action a results in observing o .

If $\sum_{i,a,o} \varphi_{n+1}^{iao} = 0$, there is currently no occupancy at z^* , then β is updated as

$$\beta_{1:n+1} \sim \text{Dir}(\beta_{1:n+1} | \sum_{i,a,o} m_1^{iao}, \dots, \sum_{i,a,o} m_n^{iao}, \lambda)$$

with $\{m_j^{iao}\}$ a set of auxiliary variables sampled from

$$p(m_j^{iao}=m | z, \beta, \alpha) \propto S([\varphi_j^{iao}]_m) (\alpha \beta_j)^m, \quad m = 1, \dots, n,$$

where $S(\cdot, \cdot)$ is a Stirling number of the first kind [139]; $[x]$ is the smallest integer no less than x .

If $\sum_{i,a,o} \varphi_{n+1}^{iao} > 0$, it indicates there is at least one occupancy at z^* , then one generates a new decision state to hold the occupancy, releasing z^* as a special state. Assuming $n + 1$ and $n + 2$ respectively indicates the new decision state and z^* , one first samples $\{m_{1:n+1}^{iao}\}$ and then samples $\beta_{1:n+2}$, similarly as above.

Given the samples of $\{\varphi^{iao}\}$, $\{m^{iao}\}$, and β , one may sample the concentration parameters (α, λ) , assuming they have gamma priors. The details are similar to those as described in the appendix of [139].

So far, one has completed a single iteration of Gibbs sampling. Given the update of (α, β) , the prior in (5.9) is updated, and one begins a new iteration. The process is repeated until the Gibbs sampler converges.

5.3 Exploration vs Exploitation in iRPR

We extend the exploration-exploitation method in Section 2.2.2 to account for the unbounded set of decision states. Definition 13 requires $p^\Pi(a|h) > 0, \forall$ action a and history h , so that the empirical value function converges to the true value function as the episodes grow. We consider Π as a mixture of the iRPR and the uniformly random policy, i.e., $\forall h$, we let $p^\Pi(a|h) = p(y = 0|h)p(a|h, \Theta) + p(y = 1|h)/|\mathcal{A}|$. The mixing proportions, $p(y|h) = \sum_{z \in \mathcal{Z}} \sigma_y^z p(z|h)$, are computed using the auxiliary policy in Section 2.2.2, with the special state z^* included in \mathcal{Z} to represent any potential unseen experiences, and $\sigma_y^z \sim \text{Beta}(u_0, u_1)$.

Before the $(K+1)$ -th episode starts, the agent has learned Θ based on the previous episodes $\mathcal{D}^{(K)}$ and the prior in (5.9), where the special state $z^* = n + 1$ is currently unoccupied and reserved to hold possible occupancies in future episodes.

Moreover, the agent has used (2.47) to allocate each previous reward to u_0^z in proportion to the probabilities that z is occupied at and *before* the time of receiving the reward. A large u_0^z , therefore, implies either or both of the following events: (i) the local policy $\pi_{1:|\mathcal{A}|}^z$ has contributed to large immediate or *future* rewards; (ii) there

has been a large amount of visits to z leading to small rewards. Note that $u_0^{z^*} \equiv 0$, since z^* is not occupied in $\mathcal{D}^{(K)}$.

Since $p(y = 0|h) \gg p(y = 1|h)$ implies that $\sigma_0^z \gg \sigma_1^z$ and hence $u_0^z \gg u_1$ for any $z \in \{z : p(z|h) \gg 0\}$ ³, which in turn implies that the decision states closely associated with h have received a large amount (relative to u_1) of rewards and/or visits so far. As a result, when u_1 in (2.46) is sufficiently large, one can conclude

$$\mathcal{H}_{\text{known}} \stackrel{\text{Def.}}{=} \{h : p(y = 0|h) \gg p(y = 1|h)\} \quad (5.10)$$

represents the part of \mathcal{H} in which the agent has acquired so much information, in the form of either a few large rewards or a large number of small rewards, that the iRPR policy has learned is nearly optimal there, where \mathcal{H} is the set of all possible histories. Let u_1^{\min} denote the minimum u_1 such that this is true.

Define $\mathcal{H}_{\text{unseen}} = \{h : p(z = z^*|h) \gg 0\}$, which is the part of \mathcal{H} that can not be represented by current decision states. Note that $h \in \mathcal{H}_{\text{known}}$ implies $h \notin \mathcal{H}_{\text{unseen}}$, because $h \in \mathcal{H}_{\text{unseen}}$ contradicts $p(y = 0|h) \gg p(y = 1|h)$, using the fact that $u_0^{z^*} \equiv 0$. Thus $\mathcal{H}_{\text{unseen}} \subset \mathcal{H}_{\text{unknown}} = (\mathcal{H} \setminus \mathcal{H}_{\text{known}})$. Letting $\mathcal{H}_{\text{seen}} = \mathcal{H}_{\text{unknown}} \setminus \mathcal{H}_{\text{unseen}}$, one may write $\mathcal{H} = \mathcal{H}_{\text{known}} \cup \mathcal{H}_{\text{seen}} \cup \mathcal{H}_{\text{unseen}}$, with the subsets mutually exclusive.

During the $(K + 1)$ -th episode, the agent follows the iRPR policy in $\mathcal{H}_{\text{known}}$ to obtain high rewards (exploitation), takes random actions to increase knowledge in $\mathcal{H}_{\text{seen}}$ or start new knowledge in $\mathcal{H}_{\text{unseen}}$ (both are exploration). Upon completion of the episode, $\{u_{0,1}^z\}_{z=1}^{\mathcal{Z}}$ are updated to reflect the increased knowledge, with a new decision state introduced to hold the new knowledge if $\mathcal{H}_{\text{unseen}}$ has been visited.

The iRPR always maintains an appropriate (possibly minimum) \mathcal{Z} for any given episodes $\mathcal{D}^{(K)}$, which is a key difference from the RPR apart from inferring \mathcal{Z} . Identification of $\mathcal{H}_{\text{unseen}}$ leads to incremental augmentation of \mathcal{Z} , and all new decision

³ To validate this deduction, we can examine the PDFs of Beta distribution $Beta(a, b)$ under different parameter settings, which are plotted in Figure 5.3 in the appendix.

states (including z^*) are initially activated for constant exploration, recalling from Section 5.2 that $\rho_z = 1/|\mathcal{Z}|, \forall z \in \mathcal{Z}$.

In contrast, the RPR assumes the optimal policy is representable on a fixed \mathcal{Z} . When $|\mathcal{Z}|$ is under-specified, an apparently large $p(y = 0|h)$ does not necessarily imply h is known, because the RPR cannot identify $\mathcal{H}_{\text{unseen}}$. Thus, if \mathcal{Z} is not set appropriately, even if $u_1^z \gg u_0, \forall z \in \mathcal{Z}$, the RPR cannot claim $\mathcal{H}_{\text{known}}$ as known; instead, the RPR will converge to a suboptimal policy in this case. It is clear then, the above definition for $\mathcal{H}_{\text{known}}$ is correct for the RPR only when $|\mathcal{Z}|$ is appropriate, while it is correct for the iRPR even if $|\mathcal{Z}|$ is initially under-specified.

It is important to note that the agent does not have to see every history in \mathcal{H} , nor try all actions, to obtain a good policy. Recall an RPR policy is a mapping from \mathcal{H} to \mathcal{A} . Given that any $h \in \mathcal{H}$ corresponds to a belief state in the underlying POMDP, one may define the similarity between two histories based on the similarity between the corresponding belief states. As similar belief states are likely to have the same optimal action [128], so are similar histories. Therefore, the agent needs to see typical histories only, instead of every single history in \mathcal{H} .

When no new decision state emerges, it means $\mathcal{H}_{\text{unseen}}$ is null; when all existing decision states have their u_0 's significantly exceeding u_1 , it means $\mathcal{H}_{\text{seen}}$ is null. When both occur, one has $\mathcal{H} = \mathcal{H}_{\text{known}}$ and the learning stops.

Thus, we have provided an approach to balancing exploration and exploitation while at the same time inferring the number of decision states. A formal analysis of the relation between the exploration rate from this approach and the optimality of iRPR policy is given below.

5.4 Theoretical Analysis of the Relation Between Exploration and Optimality

The following theorem guarantees that an agent following the Π specified above will continue exploration until the iRPR has converged to the optimal policy. Moreover, the theorem quantitatively relates the exploration rate to the difference between the optimal value and the value of the current iRPR. The theorem extends the analysis in [25] to account for the unbounded \mathcal{Z} . The proof is given in the appendix.

Let \mathcal{M} denote the true model of the POMDP. Then

$$V(\mathcal{M}; \Theta) = \sum_{t=0}^{\infty} \sum_{a_{0:t}, o_{1:t}, r_t} \gamma^t r_t p(a_{0:t}, o_{1:t}, r_t | \Theta, \mathcal{M}), \quad (5.11)$$

is the true value function of Θ , where $r_t = 0, \forall t > T$, for an episode of length T .⁴ Let R_{\max} denote the maximum r . Since $r \geq 0$, as one recalls from Section 2.2.2, one must have $R_{\max} > 0$ (otherwise $r \equiv 0$).

Theorem 12. *Let Θ^* be the optimal iRPR for the underlying POMDP. Let Θ be the iRPR learned from $\mathcal{D}^{(K)}$, and σ be governed by (2.46), with $u_1 \geq u_1^{\min}$ and $\{u_0^z\}_{z=1}^{|\mathcal{Z}|}$ updated as in (2.47). For any $\epsilon \geq 0$, if $V(\mathcal{M}; \Theta) < V(\mathcal{M}; \Theta^*) - \epsilon$, then*

$$P_e = 1 - p(y_{0:\infty} = 0 | \sigma, \Theta) > (1 - \gamma)\epsilon / R_{\max}. \quad (5.12)$$

where P_e denotes the probability of exploration, and $y_{0:t} = 0$ is a shorthand for “ $y_\tau = 0, \forall \tau \in [0, t]$ ”.

Theorem 16 shows that, when the value of the current iRPR is ϵ away from the optimal value, the agent will perform exploration with probability $P_e > (1 - \gamma)\epsilon / R_{\max}$. Conversely, when $P_e \leq (1 - \gamma)\epsilon / R_{\max}$, the value of the current iRPR is guaranteed to be ϵ close to the optimal value.

⁴ After an episode terminates, the agent stays in an absorbing state with zero reward [132].

Given a history h , the agent may explore in either of the two cases: (i) $z^* \in \mathcal{Z}(h) = \{z : p(z|h) \gg 0\}$, (ii) $\mathcal{Z}(h)$ contains an occupied decision state z for which $u_0^z \gg u_1$ is false. In case (i), $h \in \mathcal{H}_{\text{unseen}}$, the agent explores to start the learning in h , while in case (ii), $h \in \mathcal{H}_{\text{seen}}$, the agent resumes the learning in h . When neither (i) nor (ii) occurs, the iRPR is optimal with at least the minimum necessary number of decision states.

5.5 Experiments

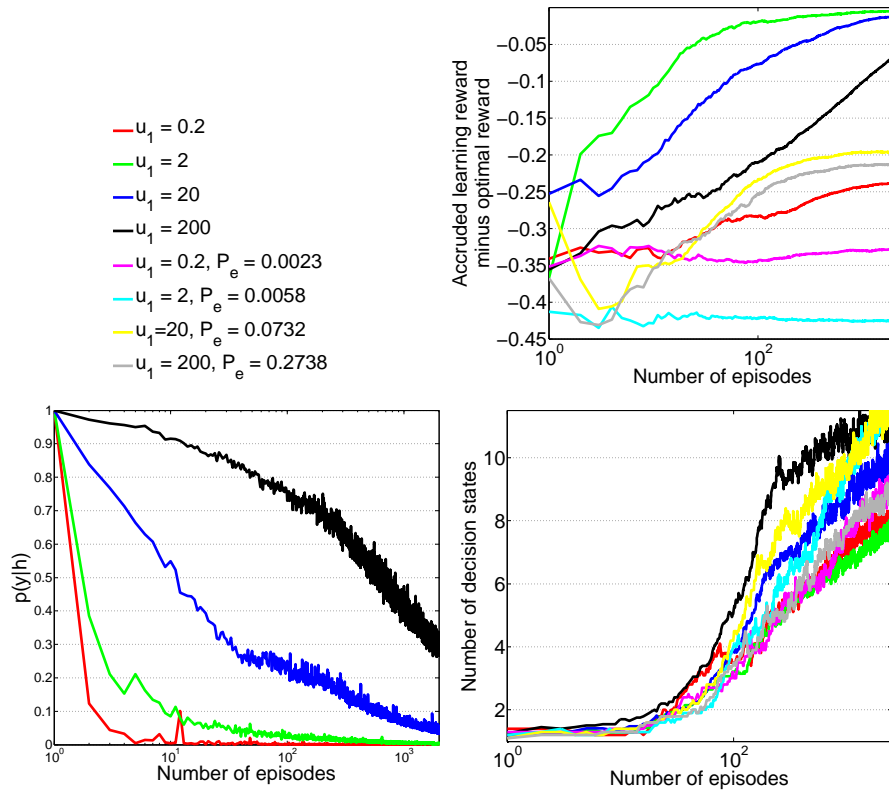


FIGURE 5.1: The iRPR’s performance on the noisy 1D maze: (top right) relative reward (bottom left) exploration rate (bottom right) $|\mathcal{Z}|$.

We study the empirical performance of the iRPR based on three benchmark POMDP models, i.e, Littman’s noisy 1D maze and Hallway, and Tag.⁵

⁵ The models are available at <http://www.cs.brown.edu/research/ai/pomdp/examples/index.html> and <http://www.science.uva.nl/~mtjspaam/pomdp>

In all the experiments, we assume gamma priors for the HDP concentration parameters, i.e., $\alpha \sim \text{Ga}(10, 10)$ and $\lambda \sim \text{Ga}(3, 10)$, where $\text{Ga}(a_g, b_g)$ is a gamma distribution with scale a_g and shape b_g . Upon completion of each episode, the iRPR parameters are updated using the inference algorithm presented in Section 5.2, based on all available episodes. All results shown result from an average over ten independent Monte Carlo runs, with error bars showing the variances.

5.5.1 Effects of History Information and u_1

We first examine the effects of u_1 and history information on the performance of balancing exploitation and exploration. Figure 5.1 plots the following experimental outputs as a function of $\log(k)$: (i) the cumulative discounted reward averaged over episodes 1 through k , with the optimal reward subtracted, (ii) the average exploration rate in the k -th episode, i.e., $\frac{1}{T_{k+1}} \sum_{t=0}^{T_k} p(y_t^k | h_t^k)$, (iii) the number of decision states $|\mathcal{Z}|$ learned from episodes 1 through k . For a curve labeled with u_1 only, the exploration or exploitation is determined by $y_t^k \sim p(y_t^k | h_t^k)$, using the u_1 shown. For a curve labeled with u_1 and P_e , the agent draws y_t^k from a Bernoulli distribution with $p(y_t^k = 1) = P_e$, where $P_e = \frac{1}{\sum_{i=1}^k \frac{1}{T_{i+1}}} \sum_{i=1}^k \sum_{t=0}^{T_i} p(y_t^k | h_t^k)$ is the average exploration rate for the curve that is labeled with the corresponding u_1 only.

It is seen from Figure 5.1 that, when $u_1 = 2$, the iRPR converges to optimality after 1000 learning episodes, and the exploration rate drops to zero accordingly. This indicates the amount of exploration allowed by u_1 is appropriate. When $u_1 = 20$, the iRPR converges to optimality, but at a lower convergence rate. This indicates that the amount of exploration as specified by $u_1 = 20$ is excessively large. With $u_1 = 200$, the convergence is too slow to be seen within the number of episodes shown here. When $u_1 = 0.2$, the exploration rate quickly drops to zero, without giving the agent enough time for exploration, and as a result, the iRPR only converges to a suboptimal policy. The results show relations between exploration rates and values

(accumulative discounted rewards) that are in agreement with Theorem 16.

The performances significantly degrade when using fixed exploration rate (not considering history information), demonstrating that the use of history information is crucial to balancing exploitation and exploration. The number of decision states inferred by the iRPR generally increases with the number of episodes, and with the increase of exploration rates.

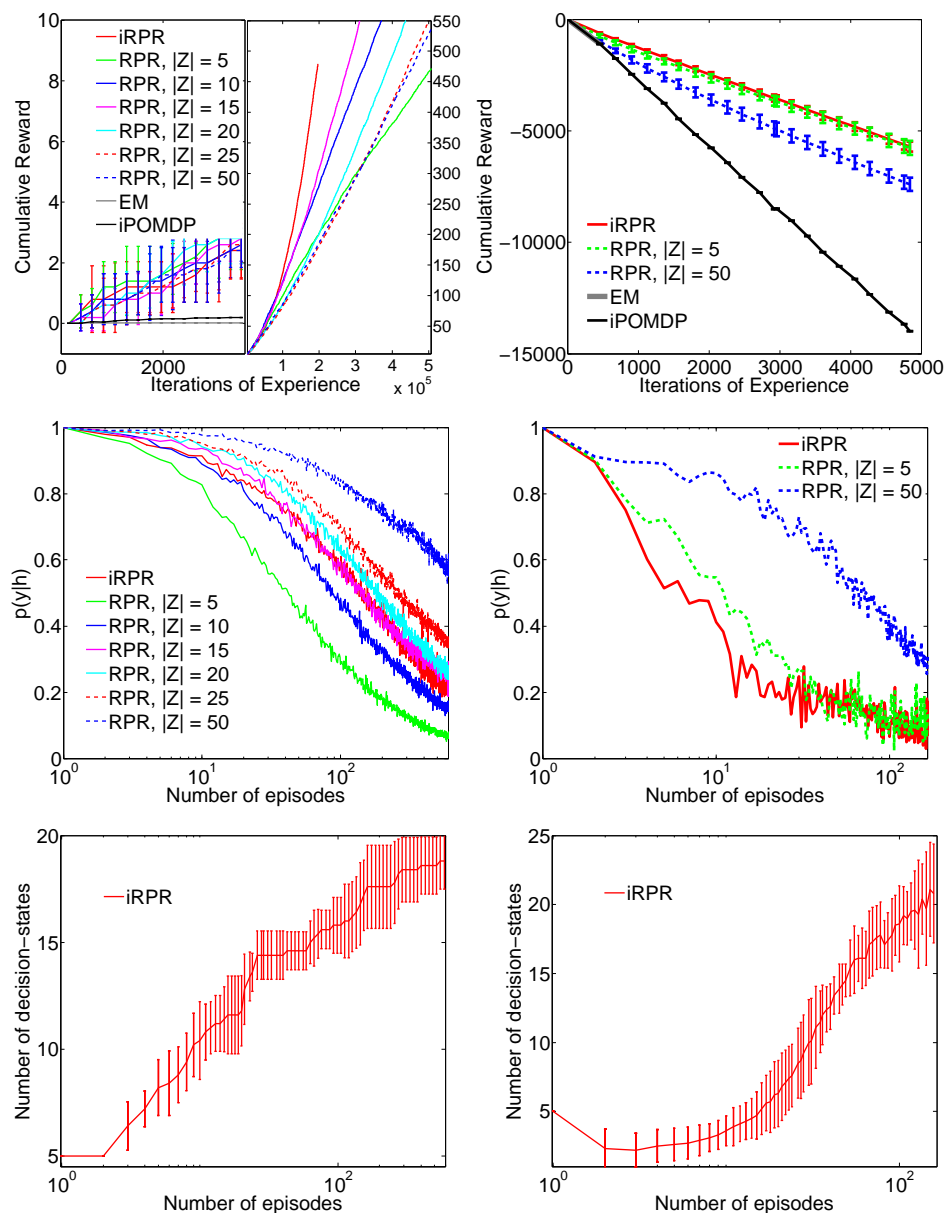


FIGURE 5.2: Performance comparison between iRPR, RPR, iPOMDP on Hallway

5.5.2 Performance Comparisons

We compare the performance of the iPRP to those of the RPR and the iPOMDP [39], the latter is a nonparametric model-based approach that infers the number of world states of a POMDP.⁶

We report the results on Hallway and Tag, in the form of the un-discounted reward summed over interactions. The results of iPOMDP and EM (which is the finite counterpart of iPOMDP), which are cited from [38], are available only within a small portion of the interactions shown here. It is seen from Figure 5.2 that the iPRR performs much better than the iPOMDP and EM.

The RPR has its performance dependent on the number of decision states. The iRPR always achieves superior performance by using appropriate numbers of decision states. The advantage of iRPR is more prominent as the agent has accumulated more experiences to make the inference of $|Z|$ more accurate.

5.6 Discussion

We presented a Bayesian nonparametric framework (iRPR) for model-free reinforcement learning in POMDPs. By using HDP-HMM type of prior over FSCs, iRPR extends the RPR to represent the POMDP policy on an *a priori* unbounded set of decision states. Such an extension is nontrivial and has multi-function. Since the empirical value function 2.45 is a large mixture of probability distributions, we have to resort to a hybrid Gibbs-variational algorithm to achieve efficient inference. Using such a inference algorithm, we are able to infer the *a posteriori* number of decision

⁶ The iPOMDP was employed in [38] to implement an infinite FSC (iFSC), which can be regarded as a special case of the iRPR (refer to the text under Definition 13 for the relations between an iRPR and a FSC). However, the iFSC was learned by fitting to expert trajectories (using standard likelihood functions), while learning of the iRPR uses the empirical value function in (2.45). As shown in [76], learning with the empirical value function is closely related to policy iteration [128]. The iFSC and the iRPR are based on totally different learning frameworks, which should not be confused.

states, to match policy complexity dynamically to the experiences. Moreover we are able to provide a principled way for exploration while inferring the decision states. This approach is based on an auxiliary policy similar to that of R-max which is based on "knowness" heuristic. Convergence analysis guarantees that the iRPR performs exploration with a rate commensurate with the difference from the optimal value. Experimental results agree with the theoretical analysis and demonstrate the iRPR's superior performance over those of the RPR and the iPOMDP. Future work involves developing more efficient data driven learning algorithm as such those introduced in chapter 4.

5.7 Appendix

Proof of Theorem 16: We note that $V(\mathcal{M}; \Theta^*)$ is upper bounded by

$$\begin{aligned}
V_f(\mathcal{M}; \sigma, \Theta) &= \sum_{t=0}^{\infty} \sum_{a_{0:t}, o_{1:t}, r_t} \gamma^t r_t p(a_{0:t}, o_{1:t}, r_t, y_{0:t}=0 | \sigma, \Theta, \mathcal{M}) \\
&+ \sum_{t=0}^{\infty} \gamma^t R_{\max} \sum_{a_{0:t}, o_{1:t}, r_t} \sum_{y_{0:t} \neq 0} p(a_{0:t}, o_{1:t}, r_t, y_{0:t} | \sigma, \Theta, \mathcal{M}), \tag{5.13}
\end{aligned}$$

where $y_{0:t} = 0$ is an abbreviation for " $y_\tau = 0, \forall \tau \in [0, t]$ " and $y_{0:t} \neq 0$ for " $y_\tau \neq 0, \exists \tau \in [0, t]$ ".

To verify the upper bound, one notes that V_f is constructed as an optimistic value function, in which the agent receives R_{\max} at any time t unless $y_{0:t} = 0$. However, observing $y_{0:t} = 0$ implies $\{h_\tau : \tau \in [0, 1]\} \subset \mathcal{H}_{\text{known}}$, in which Θ is optimal (see (5.10) and the discussions thereabout). Note that the probability of observing $y_{0:t} = 0$, i.e., $p(y_{0:t} = 0)$ can be small, which means the first term in V_f may also be small.

The premise implies $\epsilon < V_f(\mathcal{E}; \Theta) - V(\mathcal{E}; \Theta)$. Substituting (5.11), (5.13), the equation $p(a_{0:t}, o_{1:t}, r_t | \Theta, \mathcal{M}) = \sum_{y_{0:t}} p(a_{0:t}, o_{1:t}, r_t, y_{0:t} | \sigma, \Theta, \mathcal{M})$, and integrating out

a 's and σ 's, one obtains

$$\begin{aligned}
 \epsilon &< \sum_{t=0}^{\infty} \sum_{r_t} \gamma^t (R_{\max} - r_t) \sum_{y_{0:t} \neq 0} p(r_t, y_{0:t} | \Theta, \sigma), \\
 &< \sum_{t=0}^{\infty} \sum_{r_t} \gamma^t R_{\max} \sum_{y_{0:t} \neq 0} p(r_t, y_{0:t} | \Theta, \sigma), \\
 &= \sum_{t=0}^{\infty} \gamma^t R_{\max} \sum_{y_{0:t} \neq 0} p(y_{0:t} | \Theta, \sigma), \\
 &= \sum_{t=0}^{\infty} \gamma^t R_{\max} (1 - p(y_{0:t} = 0 | \Theta, \sigma)) \\
 &< \sum_{t=0}^{\infty} \gamma^t R_{\max} (1 - p(y_{0:\infty} = 0 | \Theta, \sigma)) \\
 &= \frac{R_{\max}}{1 - \gamma} (1 - p(y_{0:\infty} = 0 | \Theta, \sigma)),
 \end{aligned}$$

from which (5.12) follows. □

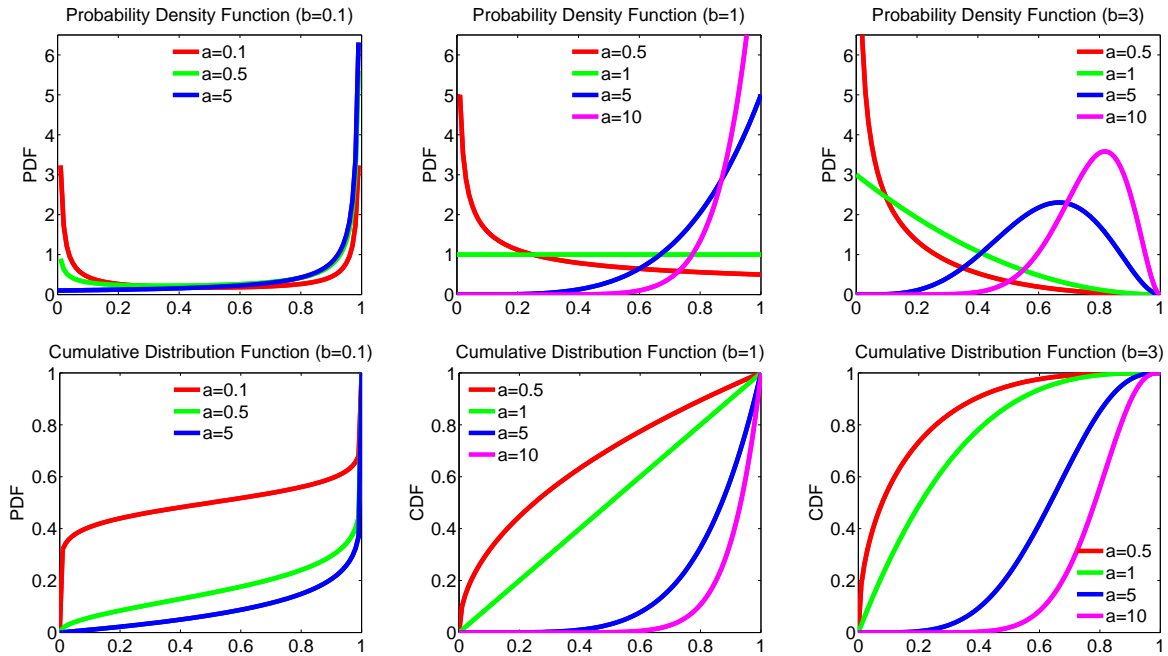


FIGURE 5.3: The probability density function(PDF) and cumulative distribution function (CDF) of Beta distribution with different setting of parameters.

Policy Learning for Decentralized POMDPs with Bayesian Nonparametric Priors

6.1 Overview

In the previous chapters, we have introduced model-free RL for centralized decision making models. In this chapter, we are going to shift our focus to multi-agent RL in DEC-POMDPs. Because of the lack of centralized belief states in DEC-POMDPs, finite-horizon algorithms have represented the agent policies as mappings from observation histories to actions [95]. History-based policy representations are difficult to generalize to the infinite-horizon case, due to their exponential complexity in the horizon length. To overcome this issue, infinite-horizon DEC-POMDP algorithms have based agent policies on finite-state controllers (FSC). While early algorithms employed exhaustive backups [13] or non-linear programming [3] to obtain FSC policies, recent work indicates that expectation maximization (EM) [37], a popular algorithm for maximum-likelihood (ML) estimation, can be used as a scalable method for generating controllers for large DEC-POMDPs [71]. In addition, [156] recently showed that EM is also an efficient algorithm for model-free reinforcement learning

(RL) in DEC-POMDPs, where each agent learns a FSC based on locally observable trajectories, without knowing the DEC-POMDP model.

An important and yet unanswered question for EM-based DEC-POMDP algorithms is how to define an appropriate number of nodes in each FSC. This is essentially a model-selection problem, and has not yet been addressed in the DEC-POMDP literature. The number of nodes affects both the quality of the resulting policies and the convergence rate. Previous work has assumed a fixed FSC size for each agent. When the number of nodes is set too small, the FSC is unable to represent the optimal policy and therefore will quickly converge to a sub-optimal policy. On the contrary, when the number is set too large, the FSC is overly complex, often yielding slow convergence to a sub-optimal policy that overfits the agent trajectories.

In this chapter, we address the model-selection problem in DEC-POMDPs based on a nonparametric Bayesian approach. In particular, we represent the local policy of each agent by a variable-size FSC, which has an unspecified number of nodes. Such a controller is constructed using the stick-breaking (SB) prior [57]. At each node of the controller, the probability distribution of successor nodes, conditional on both the observation and action, is a random draw from the SB prior. This gives rise to a distribution over an infinite number of possible nodes, indexed $\{1, 2, \dots, \infty\}$, and yet the probability mass concentrates on only a few nodes with small indices. The SB prior therefore encourages a small number of successor nodes without specifying how many *a priori*. Since the favored successors of every node all tend to have small indices, this encourages a compact set of nodes to be actively used by the controller. Those nodes that are actually used are determined by the posterior, combining the SB prior and the information from trajectory data. In light of the role played by the SB prior, we name the framework the *decentralized stick-breaking policy representation (DEC-SBPR)*.

Although the proposed DEC-SBPR may be used to find the policy for a given

DEC-POMDP model, our focus in this chapter is on reinforcement learning in DEC-POMDPs, assuming the model is not available. Toward this end, we derive a variational Bayesian (VB) algorithm for learning the DEC-SBPR based on the agents' trajectories (or episodes) of actions, observations, and rewards. The VB algorithm is linear in the number of agents and at most square in the problem size, and is therefore scalable to large application domains. To the best of our knowledge, this is the first time that nonparametric Bayesian methods have been applied to model-free reinforcement learning in DEC-POMDPs.

6.2 Policy Representations for infinite horizon DEC-POMDPs

In a (centralized) POMDP, the agent has access to the belief state, and the policy can be expressed as a mapping from belief states to actions. Sondik [128] showed that in the infinite-horizon case, if the belief-state space can be quantized into a finite set of disjoint regions which form a Markov partition, the action choice over each region is constant and the POMDP policy is a deterministic finite-state controller. When this condition is only approximately satisfied, a stochastic FSC is usually preferred as it has a better tolerance to the resulting uncertainty. A framework known as regionalized policy representation (RPR) has been developed in [76], in which the node of a stochastic FSC is marginalized out (as a latent variable) to yield a probabilistic mapping from histories to actions, and the FSC parameters are optimized by applying a nested EM algorithm to the (empirical) value function.

In a DEC-POMDP, each agent operates locally and does not have access to the full observation-action space; as a result, no agent has enough information to compute the global belief state. However, it may be possible for each agent to compute a local belief state, defined as a joint probability distribution over the world state and other agents' local actions and observations [53]. Assuming there exists an approximate Markov partition for each agent's local belief space, one can use a stochastic FSC to

represent each agent’s local policy.

Formally, the stochastic FSC for agent n is defined as $\Theta_n = \langle \mathcal{A}_n, \mathcal{O}_n, \mathcal{Z}_n, W_n, \mu_n, \pi_n \rangle$, where, \mathcal{A}_n and \mathcal{O}_n are the same as defined in the DEC-POMDP; \mathcal{Z}_n is a finite set of nodes, W_n is a set of Markov transition matrices with $W_{zz'}^{n,a,o}$ denoting the probability of agent n transiting from z to z' when taking action a in z results in observation o ; μ_n is the initial node distribution with $\mu_{n,z}$ the probability of agent n initially being in z ; π_n is a set of stochastic policies with $\pi_{z,a}^n$ the probability of agent n taking action a in z .

For simplicity, we use the following notational conventions. $\mathcal{Z}_n = \{1, 2, \dots, |\mathcal{Z}_n|\}$, where $|\mathcal{Z}_n|$ is the cardinality, and \mathcal{A}_n and \mathcal{O}_n follow similarly. $\Theta = \{\Theta_1, \dots, \Theta_N\}$ is the joint FSC of all agents. A consecutively indexed variable is abbreviated as the variable with its index range, which is indicated in the subscript of the variable; for example, $a_{n,0:T} = (a_{n,0}, a_{n,1}, \dots, a_{n,T})$, $W_{z,1:|\mathcal{Z}_n|}^{n,a,o} = (W_{z1}^{n,a,o}, W_{z2}^{n,a,o}, \dots, W_{z|\mathcal{Z}_n|}^{n,a,o})$, etc. A variable with an arrow bar is a vector, so $\vec{z}_t^k = (z_{1,t}^k, \dots, z_{N,t}^k)$

Given $h_{n,t} = \{a_{n,0:t-1}, o_{n,1:t}\}$, a local history of actions and observations up to time t , the n -th agent chooses its action $a_{n,t}$ according to the distribution $p(a_{n,t}|h_{n,t}, \Theta_n) = \frac{p(a_{n,0:t}|o_{n,1:t}, \Theta_n)}{p(a_{n,0:t-1}|o_{n,1:t-1}, \Theta_n)}$, where $p(a_{n,0:t}|o_{1:t}, \Theta_n)$ is a marginal distribution of $p(a_{n,0:t}, z_{n,0:t}|o_{n,1:t}, \Theta_n) = \mu_{n,z_{n,0}} \prod_{\tau=1}^t W_{z_{n,\tau-1} z_{n,\tau}}^{n a_{n,\tau} o_{n,\tau}} \pi_{n,a_{n,\tau}}^{z_{n,\tau}}$, obtained by integrating out the latent node variables $z_{n,0:t}$.

6.2.1 Policy Learning by Direct Value Maximization

We first discuss learning fixed-size FSCs, based on the following global empirical value function, which is extended from the single-agent case [76].

Definition 13. (*global empirical value function*) Let $\mathcal{D}^{(K)} = \{(\vec{a}_0^k r_0^k \vec{o}_1^k \vec{a}_1^k r_1^k \dots \vec{o}_{T_k}^k \vec{a}_{T_k}^k r_{T_k}^k)\}_{k=1, \dots, K}$ be a set of episodes resulting from N agents who choose actions according to $\Pi = \otimes_n \Pi_n$, a set of arbitrary stochastic policies with $p^{\Pi_n}(a|h) > 0, \forall$ action

a, \forall history h . The global empirical value function is defined as $\hat{V}(\mathcal{D}^{(K)}; \Theta) \stackrel{def.}{=} \sum_{k=1}^K \sum_{t=0}^{T_k} \frac{\gamma^t r_t^k}{K} \frac{\prod_{\tau=0}^t \prod_{n=1}^N p(a_{n,\tau}^k | h_{n,\tau}^k, \Theta_n)}{\prod_{\tau=0}^t \prod_{n=1}^N p^{\Pi_n}(a_{n,\tau}^k | h_{n,\tau}^k)}$ where $h_{n,t}^k = (a_{n,0:t-1}^k, o_{n,1:t}^k)$, $0 < \gamma < 1$ is the discount.

The global empirical value function $\hat{V}(\mathcal{D}^{(K)}; \Theta)$ is a function of the joint FSC, given only the trajectories collected from the interactions between the agents and the DEC-POMDP model, with the DEC-POMDP model parameters \mathcal{T} , Ω and \mathcal{R} assumed unknown. The empirical value function offers an objective for learning the decentralized policies, and the objective can be directly maximized by the nested EM algorithm [76], with the following modifications: (i) during the policy evaluation in the outer E-step, the rewards are recomputed based on the information from all agents; (ii) given recomputed rewards (the sum of which amounts to the improved value), the outer M-step consists of N independent applications of the inner EM, each performed for a local agent. Step (ii) can be implemented by parallel or distributed computation. Step (i) collects $\{p(a_{n,t}^k | h_{n,t}^k, \Theta_n) : \forall t, k\}$ from each agent n to update the recomputed rewards $\{\nu_t^k : \forall t, k\}$; this requires minimal communication of only $T_1 + \dots + T_K$ numbers per agent, which can be implemented efficiently.

6.3 Bayesian Learning of Fixed-size FSCs

An alternative approach is Bayesian learning, which provides the convenience of encoding expert knowledge and imposing priors for model selection. By measuring the likelihood of Θ by its empirical value, we can treat $\hat{V}(\mathcal{D}^{(K)}; \Theta)$ as a likelihood function, which is combined with the prior $p(\Theta)$ in Bayes' rule to yield the posterior

$$p(\Theta | \mathcal{D}^{(K)}) \stackrel{def.}{=} \hat{V}(\mathcal{D}^{(K)}; \Theta) p(\Theta) \left[\hat{V}(\mathcal{D}^{(K)}) \right]^{-1} \quad (6.1)$$

where $\hat{V}(\mathcal{D}^{(K)}) = \int \hat{V}(\mathcal{D}^{(K)}; \Theta) p(\Theta) d\Theta$ is the marginal empirical value of the joint FSCs.

To get rid of the sum in $\hat{V}(\mathcal{D}^{(K)}; \Theta)$, we consider a variational Bayesian (VB) approximation [16]. Denote by $q(\Theta)$ the variational approximation to $p(\Theta|\mathcal{D}^{(K)})$, and by $q_t^k(z_{0:t}^k)$ the approximation to $p(z_{0:t}^k|\sigma_{1:t}^k)$. The variational lower bound to $\ln \hat{V}(\mathcal{D}^{(K)})$ (free energy) is given by

$$\begin{aligned} \text{LB}(\{q_t^k(z_{0:t}^k)\}_{k=1}^K, q(\Theta)) &= \ln \hat{V}_p(\mathcal{D}^{(K)}) - \text{KL}(\hat{\nu}/K || \nu/K) \\ &\quad - \frac{1}{K} \sum_{k=1}^K \sum_{t=0}^{T_k} \hat{\nu}_t^k \text{KL}(q_t^k(z_{0:t}^k)q(\Theta) || p(z_{0:t}^k, \Theta)) \end{aligned} \quad (6.2)$$

where $\text{KL}(q||p)$ denotes the Kullback-Leibler (KL) divergence between probability measures q and p . The goal is to minimize the KL divergence. Since the first term on the right is a constant, this is equivalent to maximizing the lower bound. This leads to the following constrained-optimization problem

$$\begin{aligned} &\max_{\{q_t^k(z_{0:t}^k)\}_{k=1}^K, q(\Theta)} \text{LB}(\{q_t^k(z_{0:t}^k)\}_{k=1}^K, q(\Theta)) \\ \text{subject to: } &q_t^k(z_{0:t}^k, \Theta) = \prod_{n=1}^N q_t^k(z_{n,0:t}^k)q(\Theta_n), \\ &\sum_{k=1}^K \sum_{t=0}^{T_k} \sum_{z_{n,0}^k, \dots, z_{n,t}^k=1}^{|Z_n|} q_t^k(z_{n,0:t}^k) = K, \quad q_t^k(z_{n,0:t}^k) \geq 0, \forall z_{n,0:t}^k, n, t, k, \\ &\int p(\Theta)d\Theta = 1 \quad \text{and} \quad p(\Theta) \geq 0, \forall \Theta \end{aligned} \quad (6.3)$$

where the second line arises from the mean-field approximation and the consideration for generating local policies, and the last two lines are normalization constraints.

6.3.1 Policy Posterior under Dirichlet Priors

The problem in (6.3) has an analytic solution when $p(\Theta)$ is a product of Dirichlet distributions,

$$\begin{aligned} p(\Theta) &= \prod_{n=1}^N p(\Theta_n) \\ &= \prod_{n=1}^N \left[\text{Dir}(\mu_{1:|\mathcal{Z}_n}^n | \nu_{1:|\mathcal{Z}_n}^n) \right] \left[\prod_{z=1}^{|\mathcal{Z}_n|} \text{Dir}(\pi_z^n | \rho_z^n) \right] \\ &\quad \times \left[\prod_{a=1}^{|\mathcal{A}_n|} \prod_{o=1}^{|\mathcal{O}_n|} \prod_{z=1}^{|\mathcal{Z}_n|} \text{Dir}(W_{1:|\mathcal{Z}_n}^{nzo} | \omega_{1:|\mathcal{Z}_n}^{nzo}) \right], \end{aligned} \quad (6.4)$$

with hyper-parameters $\nu^n = \nu_{1:|\mathcal{Z}_n}^n$, $\rho^n = \{\rho_z^n\}_{z=1}^{|\mathcal{Z}_n|}$, and $\omega^n = \{\omega_{z,1:|\mathcal{Z}_n}^{n,a,o}\}_{z=1:|\mathcal{Z}_n|}^{o=1:|\mathcal{O}_n|, a=1:|\mathcal{A}_n|}$.

The solution to (6.3) under the Dirichlet priors is given in Theorem 14.

Theorem 14. *Let $p(\Theta)$ initially be the form of (6.4) with hyper-parameters $(\hat{\nu}, \hat{\rho}, \hat{\omega})$, then iterative application of the following updates leads to monotonic increase of (6.2), until convergence to a maxima. The updates of $\hat{\nu}$ and $\{q_t^k\}$ are given by*

$$\hat{\nu}_t^k = \frac{\gamma^t r_t^k \prod_{n=1}^N p(a_{n,0:t}^k | o_{n,1:t}^k, \tilde{\Theta}_n)}{\prod_{n=1}^N \prod_{\tau=0}^t p^{\Pi_n}(a_{n,\tau}^k | h_{n,\tau}^k) \hat{V}(\mathcal{D}^{(K)} | \tilde{\Theta})}, \quad \forall t, k, \quad (6.5)$$

$$q_t^k(z_{n,0:t}^k) = p(z_{n,0:t}^k | o_{n,1:t}^k, a_{n,0:t}^k, \tilde{\Theta}_n), \quad \forall n, t, k, \quad (6.6)$$

where $\tilde{\Theta} = \{\tilde{\eta}, \tilde{\mu}, \tilde{W}\}$ is a set of under-normalized probability (mass) functions¹, with $\tilde{\pi}_a^{n,i} = e^{\psi(\hat{\rho}_a^{n,i}) - \psi(\sum_{m=1}^{|\mathcal{A}_n|} \hat{\rho}_m^{n,i})}$, $\tilde{\mu}_i^n = e^{\psi(\hat{\nu}_i^n) - \psi(\sum_{j=1}^{|\mathcal{Z}|} \hat{\nu}_j^n)}$, and $\tilde{W}_{i,j}^{n,a,o} = e^{\psi(\hat{\omega}_{i,j}^{n,a,o}) - \psi(\sum_{z=1}^{|\mathcal{Z}|} \hat{\omega}_{iz}^{n,a,o})}$, and ψ is the digamma function. The hyper-parameters of the posterior distribution are updated as

$$\begin{aligned} \hat{\nu}_i^n &= \nu_i^n + \frac{1}{K} \sum_{k=1}^K \sum_{t=0}^{T_k} \hat{\nu}_t^k \phi_{t,0}^{n,k}(i) \\ \hat{\rho}_{i,a}^n &= \rho_{i,a}^n + \frac{1}{K} \sum_{k=1}^K \sum_{t=0}^{T_k} \hat{\nu}_t^k \sum_{\tau=0}^t \phi_{i,\tau}^{n,k}(i) \mathbb{I}(a_{i,\tau}^k, a) \end{aligned}$$

¹ Note that $q_t^k(z_{n,0:t}^k) = p(z_{n,0:t}^k | o_{n,1:t}^k, \tilde{\Theta}_n)$ is always properly normalized by $p(a_{n,0:t}^k | o_{1:t}^k, \tilde{\Theta}_n) = \sum_{z_{n,0}, \dots, z_{n,t}=1}^{|\mathcal{Z}_n|} p(z_{n,0:t}^k, a_{n,0:t}^k | o_{n,1:t}^k, \tilde{\Theta}_n)$.

$$\hat{\omega}_{i,j}^{n,a,o} = \omega_{i,j}^{n,a,o} + \frac{1}{K} \sum_{k=1}^K \sum_{t=0}^{T_k} \hat{\nu}_t^k \sum_{\tau=1}^t \xi_{t,\tau-1}^{n,k}(i,j) \mathbb{I}(a_{i,\tau}^k, a) \mathbb{I}(o_{i,\tau+1}^k, o) \quad (6.7)$$

where $\mathbb{I}(\cdot)$ is the indicator function, and

$$\xi_{t,\tau}^{n,k}(i,j) = p(z_{n,\tau}^k = i, z_{n,\tau+1}^k = j | a_{n,0:t}^k, o_{n,1:t}^k, \tilde{\Theta}_n) \quad (6.8)$$

$$\phi_{t,\tau}^{n,k}(i) = p(z_{n,\tau}^k = i | a_{n,0:t}^k, o_{n,1:t}^k, \tilde{\Theta}_n) \quad (6.9)$$

are marginals of $q_t^k(z_{n,0:t}^k)$.

The update equations in Theorem 14 constitute the VB algorithm for learning a fixed-size joint FSC under Dirichlet priors, and have a strong connection to the policy iteration algorithm. In particular, (6.5) is a policy-evaluation step where the rewards are recomputed to reflect the improved marginal value of the new policy posterior updated in the previous iteration, and (6.7) is a policy-improvement step where the recomputed rewards are used to further improve the policy posterior. Both steps require (6.8)-(6.9), which are computed based on $\alpha_{\tau}^{n,k}(i) = p(z_{n,\tau}^k = i | a_{n,0:\tau}^k, o_{n,1:\tau}^k, \tilde{\Theta}_n)$ and $\beta_{t,\tau}^{n,k}(i) = \frac{p(a_{n,\tau+1:t}^k | z_{n,\tau}^k = i, o_{n,\tau+1:t}^k, \tilde{\Theta}_n)}{\prod_{\tau'=\tau}^t p(a_{n,\tau'}^k | h_{n,\tau'}^k, \tilde{\Theta}_n)}$, $\forall n, k, t, \tau$. The (α, β) are similar to the forward-backward variables in the Baum-Welch algorithm for hidden Markov models [114]. These variables are computed recursively by each agent using (6.10)-(6.36) (shown on the top of next page).

6.4 Bayesian Learning of Variable-sized FSCs

We generalize the Bayesian learning approach described above to the case of variable-sized FSCs, using the stick-breaking prior to specify the policy's structure. The resulting DEC-SBPR generalizes the nonparametric Bayesian policy representation of POMDP [81] to the decentralized domain, employing a non-hierarchical structure for simplicity.

$$\alpha_\tau^{n,k}(i) = \begin{cases} \frac{\mu(z_{n,0}^k=i)\pi(z_{n,0}^k=i, a_{n,0}^k)}{p(a_{n,0}^k|h_{n,0}^k, \Theta_n)} & \text{if } \tau = 0 \\ \frac{\sum_{j=1}^{|\mathcal{Z}_n|} \alpha_{\tau-1}^{n,k}(j)W(z_{i,\tau-1}^k=j, a_{n,\tau-1}^k, o_{n,\tau}^k, z_{n,\tau}^k=i)\pi(z_{n,\tau}^k=i, a_{n,\tau}^k)}{p(a_{n,\tau}^k|h_{n,\tau}^k, \Theta_n)} & \text{if } \tau > 0 \end{cases} \quad (6.10)$$

$$\beta_{t,\tau}^{n,k}(j) = \begin{cases} \frac{1}{p(a_{n,0}^k|h_{n,0}^k, \Theta_n)} & \text{if } \tau = 0 \\ \frac{\sum_{n=1}^{|\mathcal{Z}_n|} W(z_{n,\tau}^k=i, a_{n,\tau-1}^k, o_{n,\tau}^k, z_{n,\tau+1}^k=j)\pi(z_{n,\tau+1}^k=j, a_{n,\tau+1}^k)\beta_{n,t,\tau+1}^k(j)}{p(a_{n,\tau}^k|h_{n,\tau}^k, \Theta_n)} & \text{if } \tau > 0 \end{cases} \quad (6.11)$$

$$p(a_{n,\tau}^k|h_{n,\tau}, \Theta_n) = \begin{cases} \sum_{j=1}^{|\mathcal{Z}_n|} \mu(z_{n,0}^k=i)\pi(z_{n,0}^k=i, a_{n,0}^k) & \text{if } \tau = 0 \\ \sum_{i,j=1}^{|\mathcal{Z}_n|} \alpha(z_{n,\tau-1}^k=i)W(z_{n,\tau-1}^k=i, a_{n,\tau-1}^k, o_{n,\tau}^k, z_{n,\tau}^k=j)\pi(z_{n,\tau}^k=j, a_{n,\tau}^k) & \text{if } \tau > 0 \end{cases} \quad (6.12)$$

Definition 15. *The decentralized stick breaking policy representation (DEC-SBPR) is a tuple $(\mathcal{N}, \mathcal{A}, \mathcal{O}, \mathcal{Z}, \Sigma, \Lambda, \rho)$, where \mathcal{N}, \mathcal{A} and \mathcal{O} are as in the definition of DEC-POMDP; \mathcal{Z} is an unbounded set of nodes indexed by positive integers; (Σ, Λ, ρ) determine (W, μ, π) , the FSC parameters defined in section 6.2, as follows*

$$\begin{aligned} W_{i,1:\infty}^{n,a,o} &\sim \text{SB}(\sigma_i^{n,a,o}, \Sigma_i^{n,a,o}) \\ \mu_{1:\infty}^n &\sim \text{SB}(\Lambda^n, \Lambda^n) \\ \pi_{1:|\mathcal{A}_n|}^n &\sim \text{Dir}(\rho_{1:|\mathcal{A}_n|}^n) \\ \Sigma_{i,j}^{n,a,o} &\sim \text{Gamma}(c, d) \\ \Lambda_i^n &\sim \text{Gamma}(e, f) \end{aligned} \quad (6.13)$$

where SB represents the stick-breaking process with $W_{i,j}^{n,a,o} = V_{i,j}^{n,a,o} \prod_{m=1}^{j-1} (1 - V_{i,m}^{n,a,o})$ and $V_{i,j}^{n,a,o} \sim \text{Beta}(\sigma_j^{n,a,o}, \Sigma_j^{n,a,o})$, and $\mu_i^n = U_i^n \prod_{m=1}^{i-1} (1 - U_m^n)$ and $U_i^n \sim \text{Beta}(\Lambda_i^n, \Lambda_i^n)$, $n = 1, \dots, N$ and $i, j = 1, \dots, \infty$.

The stick-breaking process (SBP) is a general class of priors for discrete measures. It has been used as the prior for state transition probabilities in HMMs [97] and also

for the graphical structures in topic modeling [30]. The basic properties of SBP are summarized in the supplement. The update equations for the posteriors, derived in the Supplementary Material, are partly listed below:

$$\begin{aligned}\tilde{\pi}_{i,a}^n &= \exp \left\{ \psi(\hat{\rho}_{i,a}^n) - \psi(\sum_{m=1}^{|\mathcal{A}_n|} \hat{\rho}_{i,m}^n) \right\}, \\ \tilde{\mu}_i^n &= \exp \left\{ \langle \ln \mu_i^n \rangle_{p(\mu|\lambda,\Lambda)} \right\},\end{aligned}\tag{6.14}$$

$$\tilde{W}_{i,j}^{n,a,o} = \exp \left\{ \langle \ln W_{i,j}^{n,a,o} \rangle_{p(W|\sigma,\Sigma)} \right\},$$

$$\begin{aligned}\hat{\lambda}_i^n &= \lambda_i^n + \sum_{k=1}^K \sum_{t=0}^{T_k} \frac{\hat{\nu}_t^k}{K} \phi_{t,0}^{n,k}(i) \\ \hat{\Lambda}_i^n &= \Lambda_i^n + \sum_{j=i+1}^{|\mathcal{Z}_n|} \sum_{k=1}^K \sum_{t=0}^{T_k} \frac{\hat{\nu}_t^k}{K} \phi_{t,0}^{n,k}(i)\end{aligned}\tag{6.15}$$

$$\begin{aligned}\hat{\rho}_{i,a}^n &= \rho_{i,a}^n + \sum_{k=1}^K \sum_{t=0}^{T_k} \frac{\hat{\nu}_t^k}{K} \phi_{t,\tau-1}^{n,k}(i) \mathbb{I}(a_{n,\tau}^k = a) \\ \hat{\sigma}_{i,j}^{n,a,o} &= \sigma_{i,j}^{n,a,o} + \sum_{k=1}^K \sum_{t=0}^{T_k} \frac{\hat{\nu}_t^k}{K} \xi_{t,\tau-1}^{n,k}(i,j) \mathbb{I}(a_{n,\tau}^k = a) \mathbb{I}(o_{n,\tau+1}^k = o) \\ \hat{\Sigma}_{i,j}^{n,a,o} &= \Sigma_{i,j}^{n,a,o} + \sum_{l=j+1}^{|\mathcal{Z}_n|} \sum_{k=1}^K \sum_{t=0}^{T_k} \frac{\hat{\nu}_t^k}{K} \sum_{\tau=1}^t \xi_{t,\tau-1}^{n,k}(i,j) \mathbb{I}(a_{n,\tau}^k = a) \mathbb{I}(o_{n,\tau+1}^k = o),\end{aligned}$$

where $\xi_{t,\tau}^{n,k}(i,j)$ and $\phi_{t,\tau}^{n,k}(i)$ are the same as (6.8) and (6.9) respectively.

The purpose of using SBPs is to encourage a small number of FSC nodes. Compared to a Dirichlet process (DP) [44], which has only a single concentration parameter, the SBP can represent richer patterns of sparse transition between the nodes of an FSC, because the parameters that determine the stick proportions are infinite and independent. To see this mechanism, we calculate the posterior of a SBP parameter², i.e., $p(\Sigma_{i,j}^{n,a,o} | V_{i,j}^{n,a,o}, c, d) = \text{Gamma}(c+1, d - \ln(1 - V_{i,j}^{n,a,o}))$, from which we can see the posterior mean $\mathbb{E}[\Sigma_{i,j}^{n,a,o} | V_{i,j}^{n,a,o}, c, d] = \frac{c+1}{d - \ln(1 - V_{i,j}^{n,a,o})}$. When $c, d \rightarrow 0$ and a large portion of the remaining stick is broken, i.e., $V_{i,j}^{n,a,o} \rightarrow 1$, $\Sigma_{i,j}^{n,a,o}$ will be driven down to zero, with the opposite occurring when $V_{i,j}^{n,a,o} \rightarrow 0$. Therefore, the prior over $\Sigma_{i,j}^{n,a,o}$ controls usage of a particular state j . In a similar way, Λ_n encodes the prior preference for initial node occupancy.

² The derivation is based on the assumption that $\sigma_{i,j}^{n,a,o} = 1$ and is shown in the appendix.

To accommodate an unbounded number of nodes, we apply the retrospective representation of SBPs [101] to DEC-SBPR. For agent n , the stick breaking-prior is set with a truncation level $|\mathcal{Z}_n|$, taking into account the current occupancy as well as additional nodes reserved for future new occupancies.

To determine the occupancy of the nodes in a FSC, we compute $\{|\mathcal{Z}_n|\}_{n=1}^N$ directly by checking if there is a reward assigned to a node. For example, for action a and node i , $\hat{\rho}_{i,a}^n - \rho_{i,a}^n$ is the reward being assigned. If this quantity is greater than zero, then node i is visited. By summing over all actions, we can determine whether there is a value in visiting node i . Hence we can directly compute $|\mathcal{Z}_n|$ based on the following formula

$$|\mathcal{Z}_n| = \sum_{i=1}^{\infty} \mathbb{I}(\sum_{a=1}^{|\mathcal{A}_n|} (\hat{\rho}_{i,a}^n - \rho_{i,a}^n) > 0). \quad (6.16)$$

It is shown in [57] that the random weights constructed by the stick-breaking prior are equivalently governed by a generalized Dirichlet distribution (GDD) and is therefore conjugate to the multinomial distribution; hence we can derive an efficient batch variational Bayesian algorithm for learning the decentralized policies. The complete algorithm is described in Algorithm 5. Upon the convergence of Algorithm 5, a point estimate of the decentralized policies may be obtained by calculating the expectation:

$$\mathbb{E}[\hat{\mu}_i^n] = \frac{\hat{\lambda}_i^n \prod_{l=1}^{i-1} \hat{\Lambda}_l^n}{\prod_{l=1}^i (\hat{\lambda}_l^n + \Lambda_l^n)}, \quad \mathbb{E}[\hat{\pi}_a^{n,i}] = \frac{\hat{\rho}_a^{n,i}}{\sum_{j=1}^{|\mathcal{A}_n|} \hat{\rho}_a^{n,j}}, \quad \mathbb{E}[\hat{W}_{i,j}^{n,a,o}] = \frac{\hat{\sigma}_{i,j}^{n,a,o} \prod_{l=1}^{i-1} \hat{\Sigma}_{l,j}^{n,a,o}}{\prod_{l=1}^i (\hat{\sigma}_{i,j}^{n,a,o} + \hat{\Sigma}_{i,j}^{n,a,o})}.$$

6.4.1 Computational complexity

The time complexity of Algorithm 5 in each iteration is summarized in Table 6.1, assuming the averaged episode length is T and averaged nodes number is $|\mathcal{Z}|$, which are the same for all agents. In Table 6.1, the worst case refers to the case when there is a nonzero reward at every step in an episode (dense reward), while the best case is that nonzero reward is received only at the terminal step in each episode. Hence

Algorithm 5: Batch VB Inference for DEC-SBPR

Input: Episodes $\mathcal{D}^{(K)}$ and the number of agents N ,
while $\Delta\text{LB} < \epsilon$ **do**
 for $k = 1$ to K , $n = 1$ to N **do**
 Update the global rewards $\{\hat{\nu}_t^k\}$ using (6.5).
 Compute $\{\alpha_\tau^{n,k}\}$ and $\{\beta_{t,\tau}^{n,k}\}$ with (6.10)-(6.36) ;
 end for
 Compute LB using (6.2)
 for $n = 1$ to N **do**
 Compute $\{\xi_{t,\tau}^{n,k}(i,j)\}$ and $\{\phi_{t,\tau}^{n,k}(i)\}$ using (6.8)-(6.9).
 Update the hyper-parameters of Θ_n using (6.15);
 Compute $|\mathcal{Z}_n|$ using (6.16)
 end for
end while
Return: Parameters of the posterior distributions of $\{\Theta_n\}_{n=1}^N$, and controller sizes $\{|\mathcal{Z}_n|\}_{n=1}^N$.

in general the algorithm scales linearly with the number of episodes and the number of agents. The time dependency on T is between linear and quadratic.

Table 6.1: Computational Complexity of Algorithm 5.

VARIABLES	BEST CASE	WORST CASE
α	$O(N \mathcal{Z} ^2KT)$	$O(N \mathcal{Z} ^2KT)$
β	$O(N \mathcal{Z} ^2KT^2)$	$O(N \mathcal{Z} ^2KT^2)$
ν_t^k	$O(K)$	$O(KT)$
Θ	$O(N \mathcal{Z} ^2KT)$	$O(N \mathcal{Z} ^2KT^2)$

6.4.2 Tradeoff between Exploration and Exploitation

Algorithm 5 assumes off-policy batch learning where trajectories are collected using a separate behavior policy. Off-policy learning is efficient if the behavior policy is close to optimal, as in the case when expert information is available to guide the agents. With a random behavior policy, it may take a long time for the policy to converge to optimality; in this case, the agents may exploit the policies learned so far to speed up the learning process.

An important issue concerns keeping a proper balance between exploration and exploitation, such that the action choices are sufficiently explored to prevent premature convergence to a suboptimal policy, and yet a proper level of exploitation is maintained to prevent repeated exploration. To address this issue, we define an auxiliary FSC, $\Psi_n = \langle \mathcal{Y}, \mathcal{O}_n, \mathcal{Z}_n, W_n, \mu_n, \varphi_n \rangle$, to represent the policy of each agent in balancing exploration and exploitation. To avoid confusion, we refer to Θ_n as a primary FSC. The only two components distinguishing Ψ_n from Θ_n are \mathcal{Y} and φ_n , where $\mathcal{Y} = \{0, 1\}$ encodes exploration ($y = 1$) or exploitation ($y = 0$), and $\varphi_n = \{\varphi_y^{n,z}\}$ with $\varphi_y^{n,z}$ denoting the probability of agent n choosing y in z . One can express $p(y_{n,t}|h_{n,t}, \Psi_n)$ in the same way as one expresses $p(a_{n,t}|h_{n,t}, \Theta_n)$, the latter described in the paragraph above Section 6.2.1.

The behavior policy Π_n of agent n is constructed as

$$p^{\Pi_n}(a|h, \Theta_n, \Psi_n) = \sum_{y=0,1} p(a|y, h)p(y|h, \Psi_n), \quad (6.17)$$

where $p(a|y = 0, h) \equiv p(a|h, \Theta_n)$ is the primary FSC policy, and $p(a|y = 1, h)$ is a policy agent n uses to explore action choices. For example, one may let $p(a|y = 1, h) = 1/|\mathcal{A}_n|$, leading to uniformly random exploration. The ϵ -greedy policy [132] is a special of (6.17), corresponding to $p(y = 1|h, \Psi_n) \equiv \epsilon$, $p(y = 0|h, \Psi_n) \equiv 1 - \epsilon$.

The behavior policy in (6.17) has achieved significant success in the single-agent case [26, 82]. Here we extend it to the multi-agent case and show that, under a special construction of φ_n , the optimality of the primary joint FSC $\{\Theta_n\}$ is directly related to the exploration rate..

We define $\varphi_0^{n,z}$ as a random draw from a beta distribution, $\text{Beta}(u_0^{n,z}, u_1)$, and $\varphi_1^{n,z} = 1 - \varphi_0^{n,z}$, where u_1 is kept as a sufficiently large constant, and $u_0^{n,z}$ is updated as

$$u_0^{n,z} = \sum_{k=1}^K \sum_{t=0}^{T_k} \hat{\nu}_t^k \sum_{\tau=0}^t \phi_{t,\tau}^{n,k}(z). \quad (6.18)$$

Recall that $\varphi_1^{n,z}$ is the probability of agent n choosing exploration in z . Because u_1 is a large constant, the agent will always favor exploration in z until $u_0^{n,z}$ has increased to a point where its value exceeds u_1 . Rewriting (6.18) as

$$u_0^{n,z} = \sum_{k=1}^K \sum_{\tau=0}^{T_k} \left(\sum_{t=\tau}^{T_k} \hat{\nu}_t^k \phi_{t,\tau}^{n,k}(z) \right), \quad (6.19)$$

it becomes clear that $u_0^{n,z}$ is the total amount of expected future reward received by agent n in z over all time steps in all episodes. Therefore, u_1 defines, up to a multiplicative constant, the total future reward required in z for an agent to stop exploration in z .

On the other hand, because $p(y|h, \Psi_n, \Theta_n) = \sum_z p(y|z, \Psi_n) p(z|h, \Theta_n) = \sum_z \varphi_y^{n,z} p(z|h, \Theta_n)$, we deduce that $p(y=0|h, \Psi_n, \Theta_n) \gg p(y=1|h, \Psi_n, \Theta_n)$ implies $\varphi_0^{n,z} \gg \varphi_1^{n,z}$ and hence $u_0^{n,z} \gg u_1$, for any $z \in \{z : p(z|h, \Theta_n) \gg 0\}$. However, $u_0^{n,z}$ is the total amount of expected future reward received by agent n in z , as indicated by (6.19). As a result, when u_1 in (12) is sufficiently large, one can conclude

$$\mathcal{H}_{\text{known}}^n = \{h : p(y=0|h, \Psi_n, \Theta_n) \gg p(y=1|h, \Psi_n, \Theta_n)\} \quad (6.20)$$

represents the set of local histories at which, agent n has acquired large rewards or a large number of small rewards, and thus the primary FSC Θ_n has become optimal or close so. Let u_1^{\min} denote the minimum u_1 such that this is true.

6.4.3 Theoretical Analysis of Exploration and Optimality

Let \mathcal{M} denote the true model of the DEC-POMDP. Then

$$V(\mathcal{M}; \Theta) = \sum_{t=0}^{\infty} \sum_{a_{0:t}, o_{1:t}, r_t} \gamma^t r_t p(\vec{a}_{0:t}, \vec{o}_{1:t}, r_t | \Theta, \mathcal{M}), \quad (6.21)$$

is the true value function of Θ , where $r_t = 0, \forall t > T$, for an episode of length T .³ Denote by R_{\max} the maximum immediate reward. The relation between exploration rate and policy optimality in Theorem 16 is proven in the Appendix.

³ After an episode terminates, the agent stays in an absorbing state with zero reward [132].

Theorem 16. *Let Θ^* be the optimal joint FSC for the underlying DEC-POMDP. Let Θ be the joint FSC learned from $\mathcal{D}^{(K)}$, and φ be constructed as in Section 6.4.2, $u_1 \geq u_1^{\min}$, and $\{u_0^{n,z}\}$ be updated as in (6.18). For any $\epsilon \geq 0$, if $V(\mathcal{M}; \Theta) < V(\mathcal{M}; \Theta^*) - \epsilon$, then*

$$P_e = 1 - p(\vec{y}_{0:\infty} = 0 | \sigma, \Theta) > (1 - \gamma)\epsilon / R_{\max}. \quad (6.22)$$

where P_e denotes the probability of at least one agent choosing exploration, and $\vec{y}_{0:t} = 0$ is a shorthand for “ $y_{n,\tau} = 0, \forall \tau \in [0, t], \forall n \in \mathcal{N}$ ”.

Theorem 16 shows that, when the value of the joint FSC is ϵ away from the optimal value, then, with probability of at least $(1 - \gamma)\epsilon / R_{\max}$, at least one agent will perform exploration. Conversely, when all agents perform exploitation with probability of at least $1 - (1 - \gamma)\epsilon / R_{\max} = (R_{\max} - \epsilon + \gamma\epsilon) / R_{\max}$, the value of the joint FSC is guaranteed to be ϵ close to the optimal value.

6.5 Prior parameter selection and initialization

To obtain high value policy more quickly, instead of initializing the FSCs randomly, we use the episodes with highest value to initialize the FSCs, including both the nodes transition probability, local policy as well as the size of nodes. Towards this end, we follow the following procedure: 1) use random policy to generate a number of trajectories; 2) construct a tree for each agent based on the episodes with high values; 3) convert the tree into a graph by redirecting the absorbing nodes to the initial node⁴; 4) merge the nodes if they transit to the same nodes after taking the same action and receiving the same observation and reward. Based on this initialization, we can decide the upper bound of number of the nodes and the initial FSC value. The process for initializing FSCs is illustrated in figure 6.1.

⁴ The problem we are tested all have a goal state associated with. Once the the goal state is reached, the problem is reset.

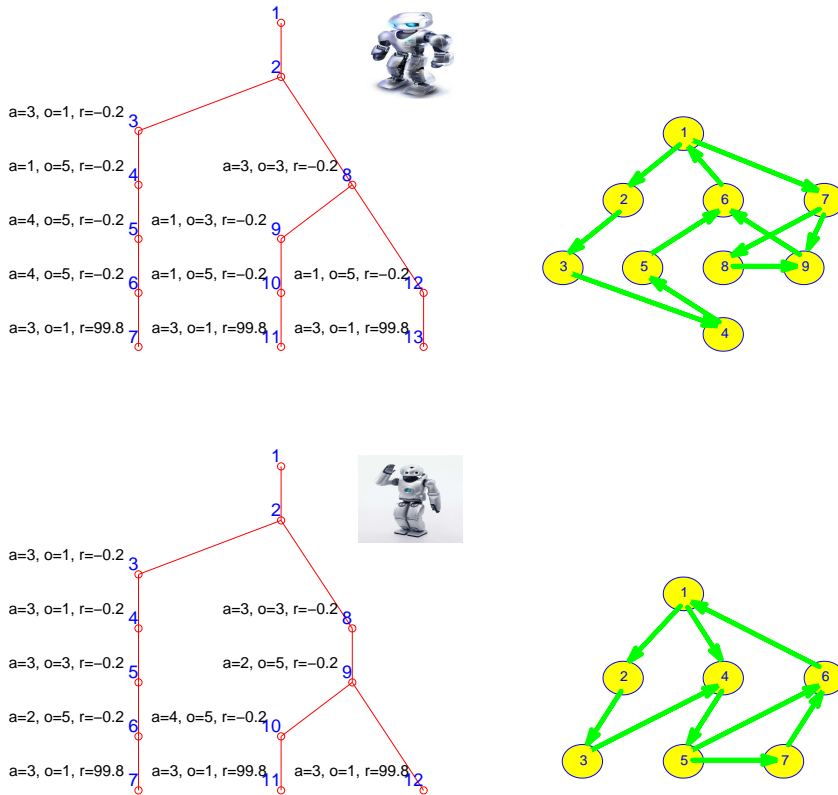


FIGURE 6.1: An illustration of DSBPR and its initialization for two agents. Left: sample trajectories; Right: the initial policy graphs.

6.6 Experiments

We evaluate the performance of proposed algorithms on five benchmark problems⁵ and a large-scale problem (traffic control) [156]. For all results reported here, we have followed the same experimental procedure as used [156]. For DEC-SBPR, the hyper-parameters in (6.13) are set to $c = e = 0.1$ and $d = f = 10^{-6}$ to promote sparse usage of FSC nodes.

Demonstration of convergence and optimality The convergence and optimality of the proposed algorithm is demonstrated on the Marsrover problem, based on using

⁵ Problems are available from <http://rbr.cs.umass.edu/camato/decpomdp/download.html>

$K = 500$ episodes to learn the FSCs and evaluating the policy by the discounted accumulative reward averaged over 100 test episodes of 1000 steps. The results are reported in Figure 6.2, which shows the exact value, its lower bound, and the numbers of FSC nodes for the two agents, as a function of iterations of the VB algorithm. It is seen that, as the the iteration proceeds, the lower bound monotonically increases, while the value itself exhibits an overall improvement and converges to the optimal value. The FSCs initially use a large number of nodes and converges to small numbers around 10.

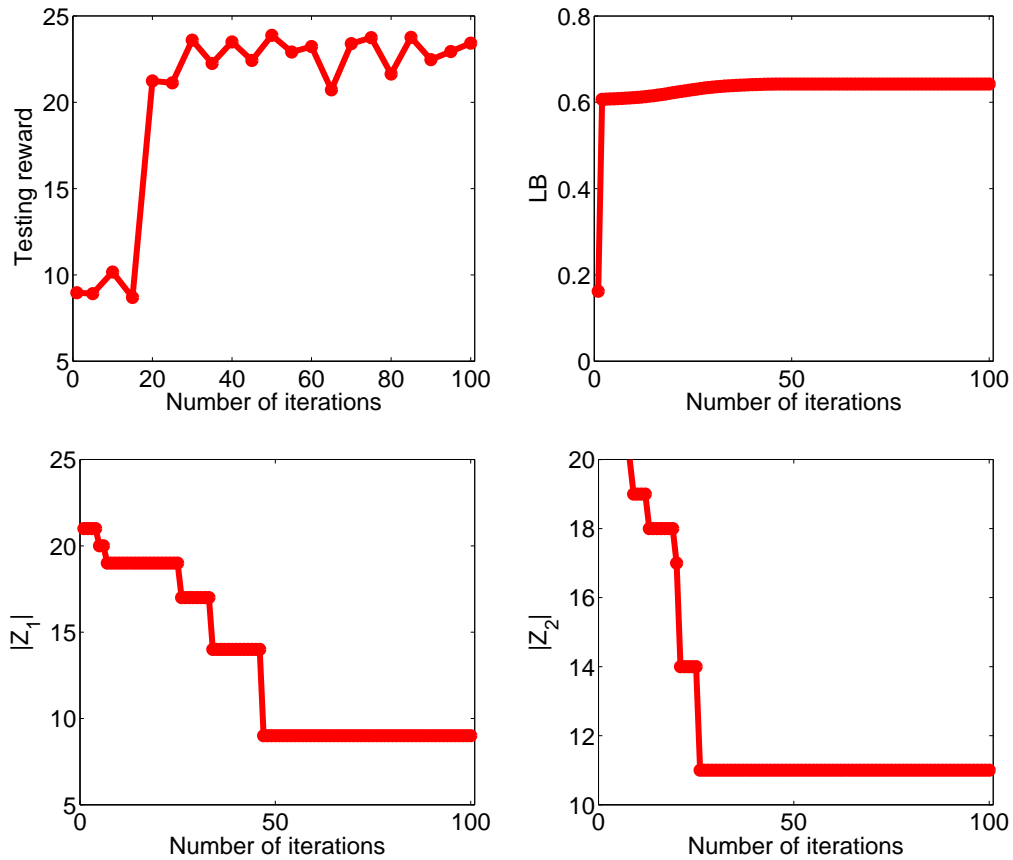


FIGURE 6.2: A demonstration of convergence and optimality of Algorithm 5 on the Marsrover problem.

Table 6.2: Results for DEC-POMDP benchmark problems comparing DEC-SBPR and other state-of-art algorithms. The numbers in the table are values and running times for each algorithm, with higher value indicating better performance. $|Z|$ is the averaged controller size inferred by DEC-SBPR.

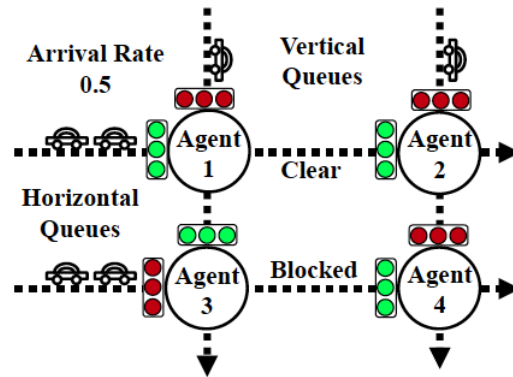
PROBLEMS ($ S , A , \mathcal{O} $)	MODEL-FREE		MODEL-BASED		PERIEM VALUE, TIME
	DEC-SBPR VALUE, TIME, $ Z $	MCEM VALUE, TIME	PBVI-BB-DEC-POMDP VALUE, TIME	PERIEM VALUE, TIME	
DEC-TIGER (2, 3, 3)	-19.14, 96s, 6	-32.31, 20s	13.45, < 5h	9.42, 6540s	
BROADCAST (4, 2, 5)	9.20, 7s, 2	9.15, 24s	9.2710, < 5h	--	
RECYCLING ROBOTS (3, 3, 2)	31.26, 147s, 3	30.78, 19s	31.9291, < 5h	31.80, 272s	
BOX PUSHING (100, 4, 5)	77.65, 290s, 14	59.95, 32s	224.1387, > 5h	106.68, 7164s	
MARS ROVERS (256, 6, 8)	20.62, 1286s, 10	8.16, 160s	--	18.13, 6088s	

Comparison with Other Methods We compare the performance of DEC-SBPR to that of several state-of-art methods, including: Monte Carlo EM (MCEM) [156], PBVI-BB-DecPOMDP [85] and Periodic EM (PeriEM) [98]. Similar to DEC-SBPR, MCEM is also a model-free reinforcement approach. We apply the exploration-exploitation strategy described in section 6.4.2 and follow the same experimental procedure as in [156] to report the results. The testing rewards after the algorithm converges are summarized in Table 6.2. We can see DEC-SBPR achieves comparable or better policy values than MCEM. These results can be explained by the fact that EM algorithm is sensible to initialization and prone to local optima. Moreover, by fixing the size of controllers, the optimal policy from EM algorithms might be over/under represented. While by using Bayesian nonparametric priors, DEC-SBPR learns the policy with variable-sized controllers, hence allows more flexibility for representing the optimal policy (though still prone to local optima).

Finally, we compare to PBVI-BB-DECPOMDP [85], the state-of-art model-based method for generating controllers with the highest value. In this regard, the performance of PBVI-BB-DECPOMDP serves as the upper-bound for those of model-free methods. However PBVI-BB-DECPOMDP is not scalable to large problem sizes, and therefore it is unable to produce results for Mars Rovers [85]. By using a model-free reinforcement learning approach, DEC-SBPR solves this problem without being restricted by the large state space size.

Scaling Up to Larger Domains To demonstrate the scalability to both large problem sizes and large numbers of agents, we test our algorithm on a traffic problem [156], which has 10^{20} states. In this domain, there are 100 agents controlling the traffic flow at 10×10 intersections with one agent located at each intersection. Each agent has 2 actions (aligning horizontally and aligning vertically), and 100 observations which indicate the number of traffic units waiting in the vertical and horizontal queues.

The traffic in one direction can pass through if all the agents along that direction are aligned. The agents receive one unit of reward if one traffic unit passes through, and zero reward if the path is blocked. The goal for this problem is to coordinate the agents to maximize the total reward. Except for MCEM, no previous DEC-POMDPs algorithms are able to solve problems of such a large size. Since [156] uses, with a 10% chance, a hand-coded policy (comparing the traffic flow between two directions) as a heuristic for generating training trajectories, we also use such a heuristic for a fair comparison. Moreover, to examine the effectiveness of the exploration-exploitation strategy described in Section 6.4.2, we also consider the case without using the heuristic for generating training episodes; in this case, the initial behavior policy for each agent is to take random actions. From Figure 6.3, we can see that, with the help of the heuristic, DEC-SBPR can achieve the best performance. While without using the heuristic, by just using our exploration-exploitation strategy, in a few iterations, DEC-SBPR is able to produce a higher quality policy than MCEM. In addition, the inferred number of FSC nodes (averaged over all agents) is smaller than the number preselected by MCEM.



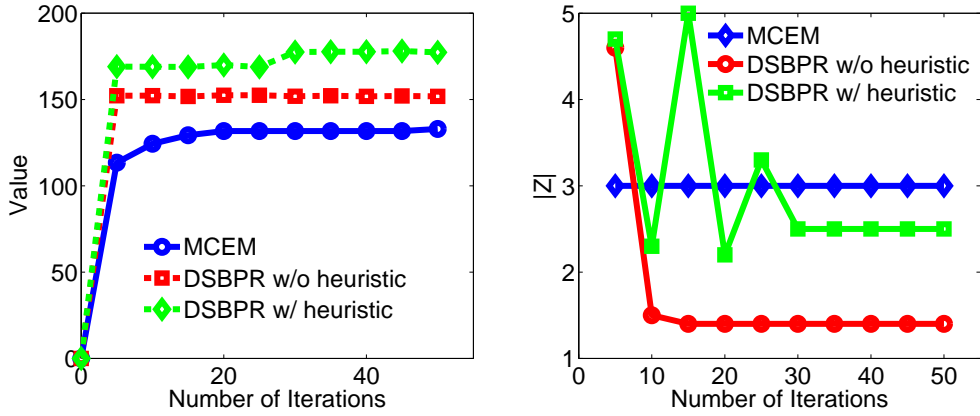


FIGURE 1 6.3: Performance on the traffic control domain with 10^{20} states and 100 agents. Top: Domain illustration; Bottom left: test reward; Bottom right: Inferred decision state number.

6.7 Discussion

In this chapter, we introduced a scalable nonparametric Bayesian policy representation, an associated learning framework (DEC-SBPR) for generating decentralized policies in DEC-POMDPs, and an exploration-exploitation method that continues to explore while the solution is suboptimal. DEC-SBPR infers the parameters of the policy graph (FSCs) *a posteriori* number of nodes, to match policy complexity dynamically to the experiences. In addition to the use of variable-sized FSCs, the proposed methods are further distinguished from previous EM-based algorithms in that we manipulate the DEC-POMDP directly, without transforming it into a mixture of dynamic Bayes nets (DBNs). As a result, the EM algorithm in our case operates on the value function of a DEC-POMDP, instead of the likelihood function of a DBN mixture. This approach was originally developed for POMDPs [76], and has shown several advantages: (i) it leads to a nested EM which achieves exact policy iteration; in particular, the outer E-step recomputes the rewards such that the updated rewards sum up to the new value of the policy improved in the previous outer M-step (policy evaluation), and the outer M-step uses an inner EM to improve the policy; (ii) the inner EM maximizes a weighted log-likelihood, where each time step is weighted by the discounted sum of future rewards updated in the outer E-

step (policy improvement). The EM algorithm in [71] does not update rewards and therefore achieves only one step of policy iteration⁶.

6.8 Appendix

6.8.1 Proof of Theorem 16

Proof. Define

$$\begin{aligned}
V_f(\mathcal{M}; \Theta, \Psi) &= \sum_{t=0}^{\infty} \sum_{\vec{a}_{0:t}, \vec{o}_{1:t}, r_t} \gamma^t r_t p(\vec{a}_{0:t}, \vec{o}_{1:t}, r_t, \vec{y}_{0:t} = 0 | \Theta, \Psi, \mathcal{M}) \\
&+ \sum_{t=0}^{\infty} \gamma^t R_{\max} \sum_{\vec{a}_{0:t}, \vec{o}_{1:t}, r_t} \sum_{\vec{y}_{0:t} \neq 0} p(\vec{a}_{0:t}, \vec{o}_{1:t}, r_t, \vec{y}_{0:t} | \sigma, \Theta, \mathcal{M}),
\end{aligned} \tag{6.23}$$

where $\vec{y}_{0:t} = 0$ is an abbreviation for “ $y_{n,\tau} = 0, \forall \tau \in [0, t], \forall n \in \mathcal{N}$ ” and $\vec{y}_{0:t} \neq 0$ for “ $y_{n,\tau} \neq 0, \exists \tau \in [0, t], \exists n \in \mathcal{N}$ ”. By construction, the agent receives R_{\max} at t when $\vec{y}_{0:t} \neq 0$; thus the second term of V_f is no smaller than the corresponding term of $V(\mathcal{M}; \Theta^*)$. When $\vec{y}_{0:t} = 0$ is true, one can deduce, for any n , that $\{h_{n,\tau} : \tau \in [0, t]\} \subset \mathcal{H}_{\text{known}}^n$, in which Θ_n is optimal, using the arguments below (6.20). Thus, the first term of V_f is equal to the corresponding term of $V(\mathcal{M}; \Theta^*)$. Combining the two cases, we have $V(\mathcal{M}; \Theta^*) \leq V_f(\mathcal{M}; \Theta, \Psi)$ which, alongside the premise of the theorem, implies $\epsilon < V_f(\mathcal{M}; \Theta, \Psi) - V(\mathcal{M}; \Theta)$. Substituting (6.21), (6.23), the equation $p(\vec{a}_{0:t}, \vec{o}_{1:t}, r_t | \Theta, \mathcal{M}) = \sum_{\vec{y}_{0:t}} p(\vec{a}_{0:t}, \vec{o}_{1:t}, r_t, \vec{y}_{0:t} | \Theta, \Psi, \mathcal{M})$, and integrating out \vec{a} 's and \vec{o} 's, one obtains

$$\begin{aligned}
\epsilon &< \sum_{t=0}^{\infty} \sum_{r_t} \gamma^t (R_{\max} - r_t) \sum_{\vec{y}_{0:t} \neq 0} p(r_t, \vec{y}_{0:t} | \Theta, \Psi), \\
&< \sum_{t=0}^{\infty} \sum_{r_t} \gamma^t R_{\max} \sum_{\vec{y}_{0:t} \neq 0} p(r_t, \vec{y}_{0:t} | \Theta, \Psi), \\
&= \sum_{t=0}^{\infty} \gamma^t R_{\max} \sum_{\vec{y}_{0:t} \neq 0} p(\vec{y}_{0:t} | \Theta, \Psi), \\
&= \sum_{t=0}^{\infty} \gamma^t R_{\max} (1 - p(\vec{y}_{0:t} = 0 | \Theta, \Psi))
\end{aligned}$$

⁶ [156] does not explicitly update rewards, but its importance sampling may play a similar role (the exact relations are subject to future investigation).

$$\begin{aligned}
&< \sum_{t=0}^{\infty} \gamma^t R_{\max}(1 - p(\vec{y}_{0:\infty} = 0 | \Theta, \Psi)) \\
&= \frac{R_{\max}}{1 - \gamma} (1 - p(\vec{y}_{0:\infty} = 0 | \Theta, \Psi)), \tag{6.24}
\end{aligned}$$

from which (6.22) follows. \square

6.8.2 (Mean-field) variational Bayesian Inference for DEC-SBPR

Under the standard variational theory [16], minimizing the KL divergence between $q(\Theta, z)$ and $p(\Theta, z | \mathcal{D})$ is equivalent to maximizing the lower bound of log marginal likelihood (empirical value function for our case). Using Jensen's inequality, we can obtain the following lower bound of the log marginal value function

$$\begin{aligned}
\ln \hat{V}(\mathcal{D}^{(K)}) &= \ln \frac{1}{K} \sum_{k=1}^K \sum_{t=0}^{T_k} \sum_{z_0^k, \dots, z_t^k} \int q_t^k(z_{0:t}^k) q(\Theta) q(\Sigma) \frac{\tilde{r}_t^k p(\Theta) p(\Sigma) p(\vec{a}_{0:t}^k, z_{0:t}^k | \vec{o}_{1:t}^k, \Theta)}{q_t^k(z_{0:t}^k) q(\Theta) q(\Sigma)} d\Theta d\Sigma \\
&\geq \frac{1}{K} \sum_{k=1}^K \sum_{t=0}^{T_k} \sum_{z_0^k, \dots, z_t^k} \int q_t^k(z_{0:t}^k) q(\Theta) q(\Sigma) \ln \frac{\tilde{r}_t^k p(\Theta) p(\Sigma) p(\vec{a}_{0:t}^k, z_{0:t}^k | \vec{o}_{1:t}^k, \Theta)}{q_t^k(z_{0:t}^k) q(\Theta) q(\Sigma)} d\Theta d\Sigma \\
&= \ln \hat{V}(\mathcal{D}^{(K)}) - KL(\{q_t^k(z_{0:t}^k) q(\Theta) q(\Sigma)\} \parallel \frac{\xi_t^k}{\ln \hat{V}(\mathcal{D}^{(K)})} p(z_{0:t}^k, \Theta, \Sigma | \vec{a}_{0:t}^k, \vec{o}_{1:t}^k)) \tag{6.25} \\
&\stackrel{def}{=} \text{LB}(\{\{q_t^k(z_{n,0:t}^k)\}_{k,t}, q(\Theta_n), q(\Sigma_n)\}_{n=1, \dots, N})
\end{aligned}$$

We rewrite the lower bound in equation (6.25) as follows

$$\begin{aligned}
\text{LB}(\{\{q_t^k(z_{n,0:t}^k)\}_{k,t}, q(\Theta_n), q(\Sigma_n)\}_{n=1 \dots N}) &= - \int q(\Theta) \ln \frac{q(\Theta)}{p(\Theta)} d\Theta - \int q(\Sigma) \ln \frac{q(\Sigma)}{p(\Sigma)} d\Sigma \\
&\quad - \frac{1}{K} \sum_{k=1}^K \sum_{t=0}^{T_k} \sum_{z_0^k, \dots, z_t^k} \prod_{n=1}^N q_t^k(z_{n,0:t}^k) \ln (\prod_{n=1}^N q_t^k(z_{n,0:t}^k)) \\
&\quad + \frac{1}{K} \sum_{k=1}^K \sum_{t=0}^{T_k} \sum_{z_0^k, \dots, z_t^k} \prod_{n=1}^N \int q_t^k(z_{n,0:t}^k) q(\Theta_n) q(\Sigma_n) \\
&\quad \times \ln (\tilde{r}_t^k \prod_{n=1}^N p_t^k(a_{n,0:t}^k, z_{n,0:t}^k | o_{n,1:t}^k, \Theta_n)) d\Theta_n d\Sigma_n \tag{6.26}
\end{aligned}$$

The VB Inference algorithm for DEC-SBPR is based on maximizing LB w.r.t. the distribution of the joint DEC-SBPR parameters $\{\{q_t^k(z_{n,0:t}^k)\}_{k,t}, q(\Theta), q(\alpha)\}_{n=1, \dots, N}$, which can be achieved by alternating the following steps:

Updating the distribution of hidden nodes (VB E-step) Keeping $q(\Theta)$ and $q(\alpha)$ fixed, solve $\max_{\{q_t^k(z_{n,0:t}^k)\}} \text{LB}(\{\{q_t^k(z_{n,0:t}^k)\}_{k,t}, q(\Theta_n), q(\Sigma_n)\}_{n=1, \dots, N}), \forall n$ subject

to the normalization constraint for $q_t^k(z_{n,0:t}^k)$. In this step, we construct the Lagrangian

$$F_{q_t^k(z_{n,0:t}^k)} = \text{LB}(\{\{q_t^k(z_{n,0:t}^k)\}_{k,t}, q(\Theta_n), q(\Sigma_n)\}_{n=1,\dots,N}) - \lambda(K - \sum_{k,t,z} \prod_{n=1}^N q_t^k(z_{n,0:t}^k)),$$

then take derivative w.r.t $q_t^k(z_{n,0:t}^k)$ and set the result to zero

$$\begin{aligned} \frac{\partial F_{q_t^k(z_{n,0:t}^k)}}{\partial (q_t^k(z_{n,0:t}^k))} &= \frac{1}{K} \int p(\Theta_n) \ln \tilde{r}_t^k p(a_{n,0:t}^k, z_{n,0:t}^k | o_{1:t}^k, \Theta_n) d\Theta_n \\ &+ \frac{1}{K} \sum_{k=1}^K \sum_{t=0}^{T_k} \sum_{z_0^k, \dots, z_t^k} \prod_{i \neq n} \int q_t^k(z_{i,0:t}^k) q(\Theta_n) q(\Sigma_n) \\ &\quad \times \ln(\tilde{r}_t^k \prod_{n=1}^N p_t^k(a_{n,0:t}^k, z_{n,0:t}^k | o_{n,1:t}^k, \Theta_n)) d\Theta_n d\Sigma_n \\ &- \frac{1}{K} \sum_{k=1}^K \sum_{t=0}^{T_k} \sum_{z_0^k, \dots, z_t^k} \prod_{i \neq n} q_t^k(z_{i,0:t}^k) \ln(\prod_{n=1}^N q_t^k(z_{n,0:t}^k)) \\ &- \frac{1}{K} \sum_{k=1}^K \sum_{t=0}^{T_k} \sum_{z_0^k, \dots, z_t^k} \prod_{i \neq n} q_t^k(z_{i,0:t}^k) \\ &+ \lambda \sum_{k=1}^K \sum_{t=0}^{T_k} \sum_{z_0^k, \dots, z_t^k} \prod_{i \neq n} q_t^k(z_{i,0:t}^k) = 0, \end{aligned} \quad (6.27)$$

which is solved to give the distribution of latent nodes $z_{n,0:t}^k$ for the n th agent

$$\begin{aligned} q_t^k(z_{n,0:t}^k) &= \frac{\tilde{r}_t^k}{C_z} \exp \left\{ \int q(\Theta_n) \ln p(a_{n,0:t}^k, z_{n,0:t}^k | o_{n,1:t}^k, \Theta_n) d\Theta_n \right\} \\ &= \frac{\tilde{r}_t^k}{C_z} \exp \left\{ \sum_{\tau=0}^t \langle \ln \pi(z_{n,\tau}^k, a_{n,\tau}^k) \rangle_{p(\pi|\hat{\rho})} + \langle \ln \mu(z_{n,0}^k) \rangle_{p(\mu|\hat{\lambda}, \hat{\Lambda})} \right. \\ &\quad \left. + \sum_{\tau=1}^t \langle \ln W(z_{n,\tau-1}^k, a_{n,\tau-1}^k, o_{\tau}^k, z_{n,\tau}^k) \rangle_{p(W|\hat{\sigma}, \hat{\Sigma})} \right\} \\ &= \frac{\tilde{r}_t^k}{C_z} \tilde{\mu}_{z_{n,0}^k} \tilde{\pi}_{a_{n,0}^k}^{z_{n,0}^k} \prod_{\tau=1}^t \tilde{W}_{z_{n,\tau}^k}^{z_{n,\tau-1}^k, a_{n,\tau-1}^k, o_{\tau}^k} \tilde{\pi}_{a_{n,\tau}^k}^{z_{n,\tau}^k} \end{aligned} \quad (6.28)$$

where

$$\begin{aligned} \tilde{\pi}_a^{n,i} &= \exp \left\{ \langle \ln \pi_a^{n,i} \rangle_{p(\pi|\hat{\rho})} \right\} = \exp \left\{ \langle \psi(\hat{\rho}_a^{n,i}) - \psi(\sum_{m=1}^{|\mathcal{A}|} \hat{\rho}_m^{n,i}) \rangle \right\} \\ \tilde{W}_{i,1}^{n,a,o} &= \exp \left\{ \langle \ln W_{i,1}^{n,a,o} \rangle_{p(W|\hat{\sigma}, \hat{\Sigma})} \right\} = \exp \left\{ \langle \ln V_{n,1}^{n,a,o} \rangle_{p(V|\hat{\sigma}, \hat{\Sigma})} \right\}, \end{aligned}$$

$$\begin{aligned}\widetilde{W}_{i,j}^{n,a,o} &= \exp \left\{ \langle \ln W_{i,j}^{n,a,o} \rangle_{p(W|\hat{\sigma}, \hat{\Sigma})} \right\} = \exp \left\{ \langle \ln V_{i,j}^{n,a,o} \rangle_{p(V|\hat{\sigma}, \hat{\Sigma})} \right. \\ &\quad \left. + \sum_{m=1}^{j-1} \langle \ln(1 - V_{i,m}^{n,a,o}) \rangle_{p(V|\hat{\sigma}, \hat{\Sigma})} \right\} \quad \text{for } j = 2, \dots, |\mathcal{Z}_n| - 1\end{aligned}$$

$$\widetilde{W}_{i,|\mathcal{Z}_n|}^{n,a,o} = \exp \left\{ \langle \ln W_{i,|\mathcal{Z}_n|}^{n,a,o} \rangle_{p(W|\hat{\sigma}, \hat{\Sigma})} \right\} = \exp \left\{ \sum_{m=1}^{|\mathcal{Z}_n|} \langle \ln(1 - V_{i,m}^{n,a,o}) \rangle_{p(V|\hat{\sigma}, \hat{\Sigma})} \right\}$$

with

$$\begin{aligned}\langle \ln V_{i,j}^{n,a,o} \rangle_{p(V|\hat{\sigma}, \hat{\Sigma})} &= \psi(\sigma_{i,j}^{n,a,o} + \omega_{i,j}^{n,a,o}) - \psi(\sigma_{i,j}^{n,a,o} + \langle \Sigma_{i,j}^{n,a,o} \rangle + \sum_{l=j}^{|\mathcal{Z}_n|} \omega_{i,l}^{n,a,o}), \\ \langle \ln(1 - V_{i,j}^{n,a,o}) \rangle_{p(V|\hat{\sigma}, \hat{\Sigma})} &= \psi(\langle \Sigma_{i,j}^{n,a,o} \rangle + \sum_{l=j+1}^{|\mathcal{Z}_n|} \omega_{i,l}^{n,a,o}) - \psi(\sigma_{i,j}^{n,a,o} + \langle \Sigma_{i,j}^{n,a,o} \rangle + \sum_{l=j}^{|\mathcal{Z}_n|} \omega_{i,l}^{n,a,o}), \\ \langle \Sigma_{i,j}^{n,a,o} \rangle &= \hat{c}_{i,j}^{n,a,o} / \hat{d}_{i,j}^{n,a,o}\end{aligned}$$

where $\psi(\cdot)$ is digamma function, $\omega_{i,j}^{n,a,o}$ is the reward(soft-count) allocated to the transition from state i to j , $\hat{c}_{i,j}^{n,a,o}$ and $\hat{d}_{i,j}^{n,a,o}$ are the posterior parameters of $\Sigma_{i,j}^{n,a,o}$.

Updating the sufficient statistics (VB-M Step) In VB-M Step, the distribution of hidden nodes $\{q_t^k(z_{0..t}^k)\}$ are fixed, the objective is to solve $\max_{q(\Theta), q(\Sigma)}$ LB subject to the normalization constraint that $\int q(\Theta) d\Theta = 1$. First we consider finding $q(\Theta)$. To that end, we construct the Lagrangian

$$F_{q\Theta} = \text{LB}(\{q_t^k\}, g(\Theta), q(\Sigma)) - \lambda \left(1 - \int q(\Theta) d\Theta \right) \quad (6.29)$$

Then

$$\begin{aligned}\frac{\partial F_{q\Theta}}{\partial (q(\Theta))} &= \frac{1}{K} \sum_{k,t,z} \int \prod_{n=1}^N q(\Sigma_n) q_t^k(z_{n,0:t}^k) \ln \tilde{r}_t^k \prod_{n=1}^N p(\Sigma_n) p(a_{n,0:t}^k, z_{n,0:t}^k | o_{1:t}^k, \Theta_n) d\Sigma, \\ &\quad - 1 - \ln \frac{q(\Theta)}{p(\Theta)} + \lambda = 0 \Rightarrow\end{aligned}$$

$$\begin{aligned}q(\Theta) &= \prod_{n=1}^N \frac{p(\Theta_n)}{e^{1-\lambda}} \exp \left\{ \frac{1}{K} \sum_{k,t,z} q_t^k(z_{n,0:t}^k) \ln p(a_{n,0:t}^k, z_{n,0:t}^k | o_{n,1:t}^k, \Theta_n) \right\} \\ &= \prod_{n=1}^N \frac{p(\Theta_n)}{e^{1-\lambda}} \exp \left\{ \frac{1}{K} \sum_{k,t} K \mathcal{V}_t^k \left[\phi_{t,0}^{n,k} \ln \mu_i^n + \sum_{\tau=1}^t \sum_{i=1}^{|\mathcal{Z}_n|} \phi_{t,\tau}^{n,k}(i) \ln \pi_i^{n,a_{n,\tau}^k} \right] \right\}\end{aligned}$$

$$\begin{aligned}
& + \sum_{\tau=1}^t \sum_{i,j=1}^{|\mathcal{Z}_n|} \xi_{t,\tau}^{n,k}(i,j) \ln W_{i,j}^{n,a_{\tau-1}^k, o_{\tau}^k} \Big\} \\
& = \prod_{n=1}^N \frac{p(\Theta)}{e^{1-\lambda}} \prod_{i=1}^{|\mathcal{Z}_n|} [\mu_i]^{\sum_{k,t} \nu_t^k \phi_{t,0}(i)} \prod_{i=1}^{|\mathcal{Z}_n|} [\pi_a^i]^{\sum_{k,t,\tau} \nu_t^k \phi_{t,\tau}^k(i)} \\
& \quad \times \prod_{i,j=1}^{|\mathcal{Z}_n|} [W_j^{i,a_{\tau-1}, o_{\tau}}]^{\sum_{k,t,\tau} \nu_t^k \xi_{t,\tau-1}^{n,k}(i,j)}
\end{aligned}$$

(from the relation between stick-breaking weights \mathbf{p} and independent beta random
(variables in \mathbf{V} (2.54))

$$\begin{aligned}
& = \prod_{n=1}^N \frac{p(\Theta)}{e^{1-\lambda}} \prod_{i=1}^{|\mathcal{Z}_n|} \left[U_i^n \prod_{m=1}^{i-1} (1 - U_m^n) \right]^{\sum_{k,t} \nu_t^k \phi_{t,0}^{n,k}(i)} \prod_{i=1}^{|\mathcal{Z}_n|} [\pi_i^{n,a_{\tau}^k}]^{\sum_{k,t,\tau} \nu_t^k \phi_{t,\tau}^{n,k}(i)} \\
& \quad \times \prod_{i,j=1}^{|\mathcal{Z}_n|} \left[V_{i,j}^{n,a_n,\tau-1,o_n,\tau} \prod_{m=1}^{j-1} (1 - V_{i,m}^{n,a_n,\tau-1,o_n,\tau}) \right]^{\sum_{k,t,\tau} \nu_t^k \xi_{t,\tau-1}^{n,k}(i,j)} \\
& = \prod_{n=1}^N GDD(\hat{\boldsymbol{\lambda}}^n, \hat{\boldsymbol{\Lambda}}^n) \prod_{i=1}^{|\mathcal{Z}_n|} Dir(\hat{\boldsymbol{\rho}}_i^n) \prod_{a=1}^{|\mathcal{A}_n|} \prod_{o=1}^{|\mathcal{O}_n|} GDD(\hat{\boldsymbol{\sigma}}^{n,a,o}, \hat{\boldsymbol{\Sigma}}^{n,a,o}) \quad (6.30)
\end{aligned}$$

where

$$\begin{aligned}
\hat{\boldsymbol{\lambda}}_i^n &= \boldsymbol{\lambda} + v_i^n = \boldsymbol{\lambda}_i^n + \frac{1}{K} \sum_{k=1}^K \sum_{t=0}^{T_k} \hat{\nu}_t^k \phi_{t,0}^{n,k}(i) \\
\hat{\boldsymbol{\Lambda}}_i^n &= \boldsymbol{\Lambda}_i^n + \sum_{j=i+1}^{|\mathcal{Z}_n|} u_j^n = \boldsymbol{\Lambda}_i^n + \frac{1}{K} \sum_{j=i+1}^{|\mathcal{Z}_n|} \sum_{k=1}^K \sum_{t=0}^{T_k} \hat{\nu}_t^k \phi_{t,0}^{n,k}(i) \\
\hat{\boldsymbol{\rho}}_a^{n,i} &= \boldsymbol{\rho}_a^{n,i} + \frac{1}{K} \sum_{k=1}^K \sum_{t=0}^{T_k} \sum_{\tau=1}^t \nu_t^k \phi_{t,\tau-1}^{n,k}(i) \mathbb{I}(a_{n,\tau-1}^k = a) \\
\hat{\boldsymbol{\sigma}}_{i,j}^{n,a,o} &= \boldsymbol{\sigma}_{i,j}^{n,a,o} + \omega_{i,j}^{n,a,o} \\
&= \boldsymbol{\sigma}_{i,j}^{n,a,o} + \frac{1}{K} \sum_{k=1}^K \sum_{t=0}^{T_k} \sum_{\tau=1}^t \nu_t^k \xi_{t,\tau-1}^{n,k}(i,j) \mathbb{I}(a_{n,\tau-1}^k = a) \mathbb{I}(o_{n,\tau}^k = o) \\
\hat{\boldsymbol{\Sigma}}_{i,j}^{n,a,o} &= \boldsymbol{\Sigma}_{i,j}^{n,a,o} + \sum_{l=j+1}^{|\mathcal{Z}_n|} \omega_{i,l}^{n,a,o} \\
&= \boldsymbol{\Sigma}_{i,j}^{n,a,o} + \frac{1}{K} \sum_{l=j+1}^{|\mathcal{Z}_n|} \sum_{k=1}^K \sum_{t=0}^{T_k} \hat{\nu}_t^k \sum_{\tau=1}^t \xi_{t,\tau-1}^{n,k}(i,j) \mathbb{I}(a_{n,\tau-1}^k = a) \mathbb{I}(n, o_{\tau}^k, o),
\end{aligned} \quad (6.31)$$

To find $q(\Sigma)$, we construct the Lagrange,

$$F_{q_{\Sigma}} = \text{LB}(\{q_t^k\}, g(\Theta), q(\Sigma)) - \lambda(1 - \int q(\Sigma) d\Sigma) \quad (6.32)$$

$$\frac{\partial F_{\Sigma}}{\partial(q(\Sigma))} = \frac{1}{K} \int q(\Theta) \ln p(\Sigma) p(\Theta) d\Theta - \frac{1}{K} \ln q(\Sigma) + \lambda = 0 \Rightarrow \quad (6.33)$$

$$q(\Sigma) \propto \exp \left\{ \int q(\Theta) \ln p(\Sigma) p(\Theta|\Sigma) d\Theta \right\}$$

$$\begin{aligned}
&= \exp \left\{ \int q(\Theta) \ln p(\Theta|\Sigma) d\Theta + \ln p(\Sigma) \right\} \\
&= p(\Sigma) \exp \left\{ \langle \ln p(\Theta) \rangle \right\} \tag{6.34} \\
&= \prod_{a=1}^{|\mathcal{A}_n|} \prod_{j=1}^{|\mathcal{O}_n|} \prod_{i=1}^{|\mathcal{Z}_n|} \prod_{j=1}^{|\mathcal{Z}_n|} \text{Ga}(\Sigma_{i,j}^{n,a,o}; c, d) \frac{\Gamma(\sigma_{i,j}^{n,a,o} + \Sigma_{i,j}^{n,a,o})}{\Gamma(\sigma_{i,j}^{n,a,o})\Gamma(\Sigma_{i,j}^{n,a,o})} V_{i,j}^{n,a,o} \sigma_{i,j}^{n,a,o-1} (1 - V_j^{n,a,o})^{\Sigma_{i,j}^{n,a,o}-1} \\
&\approx \prod_{a=1}^{|\mathcal{A}_n|} \prod_{j=1}^{|\mathcal{O}_n|} \prod_{i=1}^{|\mathcal{Z}_n|} \prod_{j=1}^{|\mathcal{Z}_n|} \frac{\Gamma(\sigma_{i,j}^{n,a,o} + \Sigma_{i,j}^{n,a,o})}{\Gamma(\Sigma_{i,j}^{n,a,o})} \{\Sigma_{i,j}^{n,a,o}\}^{c-1} \exp \left\{ -\Sigma_{i,j}^{n,a,o} (d - \ln(1 - V_j^{n,a,o})) \right\}
\end{aligned}$$

One can set $\sigma_{i,j}^{n,a,o} = 1$, in this case, the VB approximation of $q(\sigma)$ is a product of independent gamma distributions, however, $\sigma_{i,j}^{n,a,o}$ has both practical meaning and direct relation to the value of a particular state. Note that when $\sigma_{i,j}^{n,a,o} \neq 1$, equation (6.34) is not a gamma distribution (the prior and likelihood are not conjugate). To solve this issue, one might consider the VB inference method for non-conjugate priors [148], by which we consider a point estimate of Σ , such at $q(\Sigma)$ is maximized. One way to obtain the maximum estimate of Σ is to solve $\frac{\partial q(\Sigma)}{\partial \Sigma} = 0$, however this operation involves taking derivative w.r.t gamma functions, which does not have a simple form of solutions. To circumvent this difficult, we use grid search. To make the search more efficient, we use the bounds of $\frac{\Gamma(a+x)}{\Gamma(x)}$ to give an initial estimate of the searching range. The bounds are from Wendel's double Inequality [113].

$$\frac{x}{(x+a)^{1-a}} \leq \frac{\Gamma(a+x)}{\Gamma(x)} \leq x^a \tag{6.35}$$

where $x, a > 0$.

To illustrate the difference, we plot the ratio between two gamma functions $\frac{\Gamma(x+a)}{\Gamma(x)}$ along with its lower and upper bounds (denoted by lb and ub respectively) on the left of figure 6.4. We also plot the absolute differences $\text{diff} - lb = \frac{\Gamma(x+a)}{\Gamma(x)} - \frac{x}{(x+a)^{1-a}}$ and $\text{diff} - lb = \frac{\Gamma(x+a)}{\Gamma(x)} - x^a$ on the right of figure 6.4. We can see that when $0 < a < 1$, the lower bound is more tighter than the upper bound for approximating $\frac{\Gamma(x+a)}{\Gamma(x)}$. When

$a < 10^{-4}$, we use the following approximation

$$\frac{\Gamma(x+a)}{\Gamma(x)} \approx \begin{cases} 1 & \text{if } x \gg a < 10^{-4} \\ \frac{x}{x+a} & \text{if } x \leq a < 10^{-4} \end{cases} \quad (6.36)$$

6.8.3 Some Basics of Stick-breaking Priors

Stick-breaking prior and generalized Dirichlet distribution We denote $\mathbf{p} \sim \text{SB}(\mathbf{v}, \mathbf{w})$ as constructing \mathbf{p} as an infinite process ($d \rightarrow \infty$) as (2.54), and $\mathbf{p} \sim \text{GDD}(\mathbf{v}, \mathbf{w})$ when \mathbf{p} is finite. Here GDD stands for generalized Dirichlet distribution. To see the connection between SBP and GDD, set the truncation level (number of occupied states) to d with $p_{d+1} = 1 - \sum_{i=1}^d p_i$, then we can write down the density function of $\mathbf{V} = (V_1, \dots, V_d)$ as

$$f(\mathbf{V}) = \prod_{i=1}^d f(V_i) = \prod_{i=1}^d \frac{\Gamma(v_i + w_i)}{\Gamma(v_i)\Gamma(w_i)} V_i^{v_i-1} (1 - V_i)^{w_i-1} \quad (6.37)$$

By changing variables from \mathbf{V} to \mathbf{p} and using the relation between \mathbf{V} and \mathbf{p} in (2.54), we can show the density of \mathbf{p} as follows,

$$f(\mathbf{p}) = \left| \frac{\partial \mathbf{V}}{\partial \mathbf{p}} \right| f(\mathbf{V}) = \prod_{i=1}^d \left(\frac{\Gamma(v_i + w_i)}{\Gamma(v_i)\Gamma(w_i)} p_i^{v_i-1} \left(1 - \sum_{j=1}^i p_j \right)^{w_i - (v_{i+1} + w_{i+1})} \right) p_{d+1}^{w_d-1} \quad (6.38)$$

which has a mean and variance for an element p_i ,

$$\mathbb{E}[p_i] = \frac{v_i \prod_{l=1}^{i-1} w_l}{\prod_{l=1}^i (v_l + w_l)}, \quad \mathbb{V}[p_i] = \frac{v_i(v_i + 1) \prod_{l=1}^{i-1} w_l(w_l - 1)}{\prod_{l=1}^i (v_l + w_l)(v_l + w_l + 1)} \quad (6.39)$$

when $w_i = \sum_{j=i+1}^K v_j$ for $i < d$, and keeping $w_d = w_d$, the GDD is equivalent to the standard Dirichlet distribution.

As a concrete example, consider the case $d = 3$, we have

$$\begin{aligned}
p_1 &= V_1 & V_1 &= p_1 \\
p_2 &= (1 - V_1)V_2 & V_2 &= \frac{p_2}{1 - p_1} \\
p_3 &= (1 - V_1)(1 - V_2)V_3 & \iff & V_3 &= \frac{p_3}{1 - p_1 - p_2} \\
p_4 &= (1 - V_1)(1 - V_2)(1 - V_3)V_4 & & & V_4 &= \frac{p_4}{1 - p_1 - p_2 - p_3} = 1 \\
&= (1 - V_1)(1 - V_2)(1 - V_3) & & & &
\end{aligned}$$

and plug these relations into (6.38),

$$\begin{aligned}
f(\mathbf{p}) &= \left| \frac{\partial \mathbf{V}}{\partial \mathbf{p}} \right| f(\mathbf{V}) \\
&= \prod_{i=1}^4 (1 - \sum_{j=1}^{i-1} p_j) \frac{\Gamma(v_i + w_i)}{\Gamma(v_i)\Gamma(w_i)} \times p_1^{v_1-1} (1 - p_1)^{w_1-1} \cdot \left(\frac{p_2}{1-p_1} \right)^{v_2-1} \left(1 - \frac{p_2}{1-p_1} \right)^{w_2-1} \\
&= \left(\frac{p_3}{1-p_1-p_2} \right)^{v_3-1} \left(1 - \frac{p_2}{1-p_1-p_2} \right)^{w_3-1} \cdot \left(\frac{p_4}{1-p_1-p_2-p_3} \right)^{v_4-1} \left(1 - \frac{p_4}{1-p_1-p_2-p_3} \right)^{w_4-1} \\
&= \prod_{i=1}^4 \frac{\Gamma(v_i + w_i)}{\Gamma(v_i)\Gamma(w_i)} p_i^{v_i-1} (1 - \sum_{j=1}^i p_j)^{w_i - (v_{i+1} + w_{i+1})} \tag{6.40}
\end{aligned}$$

Bayesian inference for GDD Given a set of discrete observations $\{X_n\} \stackrel{i.i.d.}{\sim} \text{Discrete}(\mathbf{p})$, and the prior $\mathbf{p} \sim SB(v, w)$, the posterior of \mathbf{p} is

$$\begin{aligned}
p(\mathbf{p} | \{X_n\}) &\propto \prod_{n=1}^N \prod_{i=1}^d p_i^{\mathbb{I}(X_n, i)} \prod_{i=1}^d \left(\frac{\Gamma(v_i + w_i)}{\Gamma(v_i)\Gamma(w_i)} p_i^{v_i-1} (1 - \sum_{j=1}^i p_j)^{w_i - (v_{i+1} + w_{i+1})} \right) p_{d+1}^{w_{d+1}-1} \\
&\propto \prod_{i=1}^d (V_i \prod_{j=1}^{i-1} (1 - V_j))^{\sum_{n=1}^N \mathbb{I}(X_n, i)} V_i^{v_i-1} (1 - V_i)^{w_i-1} \\
&\propto \prod_{i=1}^d V_i^{v_i + \sum_{n=1}^N \mathbb{I}(X_n, i) - 1} (1 - V_i)^{w_i + \sum_{j>i} \sum_{n=1}^N \mathbb{I}(X_n, i) - 1} \tag{6.41} \\
&\propto GDD(\mathbf{v}', \mathbf{w}')
\end{aligned}$$

where the posterior hyper parameters updated as $v'_i = v_i + \sum_{n=1}^N \mathbb{I}(X_n = i)$ and $w'_i = w_i + \sum_{j>i} \sum_{n=1}^N \mathbb{I}(X_n = i)$, where $\mathbb{I}(\cdot)$ is an indicator function with value equal to one when the argument is true and zero otherwise.

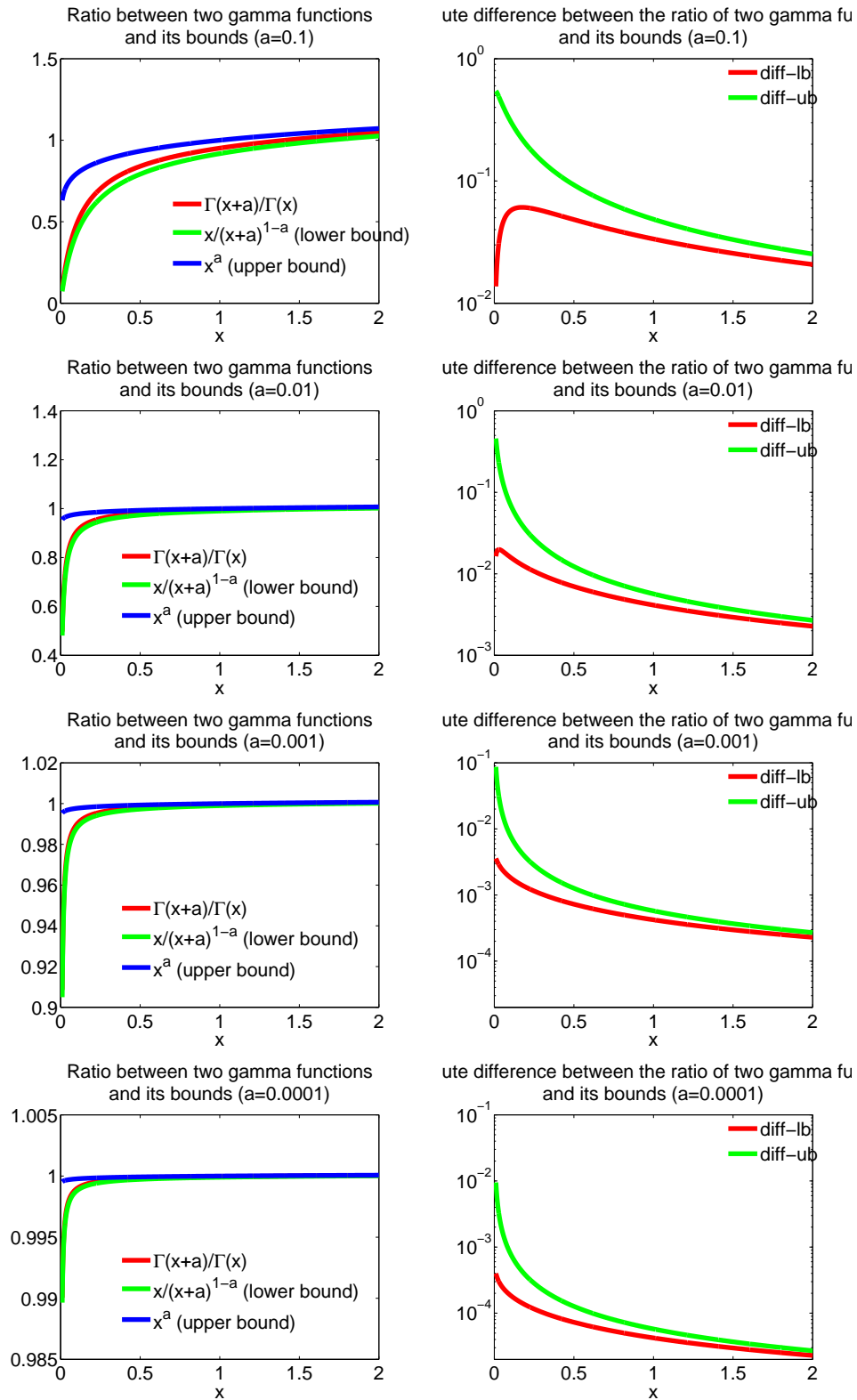


FIGURE 6.4: Left: The ratio between two gamma functions $\frac{\Gamma(x+a)}{\Gamma(x)}$ and its lower and upper bounds. Right: The absolute difference between $\frac{\Gamma(x+a)}{\Gamma(x)}$ and its lower and upper bound.

Conclusions

Reinforcement learning (RL) is an important and thriving research area, which aims at enabling agents to teach themselves to make smart decisions and improve task performance through interacting with stochastic environments. When designing a RL algorithm, there are a number of practical and theoretical issues that need to be considered. These issues include i) how to choose control policy representation (graph vs tree, parametric vs nonparametric); ii) which RL approach should follow (model-free or model-based); iii) which type of learning methods to use (online vs batch, on-policy vs off-policy); and iv) how to balance exploration and exploitation (direct vs indirect). To allow persistent RL in complex tasks when the domain models are unavailable, we applied Bayesian nonparametric methods (BNPMs) in model-free RL settings and demonstrated the advantages of BNPMs, including: i) allowing automatic adjustment of features based on observed data; ii) allowing inference for both the policy representation parameters as well as representation complexity; iii) providing principled ways to balance exploration and exploitation; iv) allowing encoding expert knowledge and prior information for efficient policy learning; v) applicability for both batch and online learning settings; vi) applicability for centralized and

decentralized decision making problems. In this chapter, we summarize the technologies developed in this thesis and discuss some of the open questions that are of interest for future research.

7.1 Summary of Contributions

We started by presenting a nonparametric Bayesian framework, termed GPQ, for approximate RL in MDPs. GPQ uses a Gaussian process model to approximate the value function without using a planner. In addition to presenting the GPQ framework, sufficient conditions for convergence of GPQ to the best achievable optimal Q-function given the data (Q^*) are presented in both the batch and online setting, and it is shown that these properties hold even as new features are added or less important features removed to maintain computational feasibility. Our work also contributes to off-policy approximate RL in general, because unlike other recent papers on off-policy RL with fixed-parameter linear function approximation, our approach allows the basis functions to be automatically identified from data. Furthermore, our condition for convergence reveals why in the batch case GPQ or kernel base fitted Q-Iteration could lead to divergence in the worst case, and how the divergence can be prevented by tuning a regularization-like parameter of the GP. We designed a practical online implementation of this framework that makes use of a recent budgeted online sparse GP inference algorithm. Our theoretical and empirical results show off-policy RL using a GP provides provable convergence guarantees and competitive learning speeds.

We then presented a novel online nested EM algorithm for learning the Regionalized Policy Representation (RPR), a parametric policy for RL in POMDPs. The online algorithm uses only the latest learning episode to update the policy value in the outer-loop E-step and update the sufficient statistic in the inner-loop E-step, and computes the improved policy in closed-form given the sufficient statistic. Our online

RPR framework contributes to RL in POMDPs both theoretically and empirically. In theory, i) the online RPR reduces both the time and memory complexity an order of magnitude, in comparison with the corresponding batch-mode algorithm; ii) under standard conditions on the learning rates, the online RPR provably converges to a local optimal batch-mode policy when the number of learning episodes becomes sufficiently large. Empirically, our study shows that: (i) the online RPR produces policies as good as the batch-mode RPR, while using only a small fraction of the time; (ii) the stochastic nature of the online learning algorithm can help it jump out of local optima; (vi) the RPR can produce high-quality policies by using fewer internal memory units than the U-tree; (v) the RPR has performance that improves with the history length, and is a competitive reactive policy when the length is one.

We also discussed iRPR, a nonparametric generalization of the RPR to represent the POMDP policy on an *a priori* unbounded set of decision states. This extension is nontrivial, since we have to deal with particularly challenging issues, including policy representation uncertainty, and exploration under variable sized policy representation. iRPR solves these issues and contributes to model-free RL in POMDPs from three aspects: 1) a hybrid Gibbs-variational algorithm for inferring the parameters of the policy graph and *a posteriori* number of decision states, to match policy complexity dynamically to the experiences, while avoiding costly model comparisons; 2) a principled way to balance exploration and exploitation while inferring the decision states; 3) convergence analysis that guarantees that the iRPR performs exploration with a rate commensurate with the difference from the optimal value.

We finally introduced a nonparametric Bayesian policy representation framework based on a stick-breaking process, termed DEC-SBPR, which contributes to the DEC-POMDP research from three aspects. This framework represents the first time that BNPMs have been applied to RL in DEC-POMDPs. Secondly, it allows one to solve infinite-horizon problems, and it is able to achieve comparable and better results

to the state-of-the-art model-free method, with the additional benefit of inferring the number of nodes that are necessary for representing the optimal policy based on given experiences. Thirdly, the problem of learning decentralized policies is formulated as a Bayesian optimization problem, with a scalable inference algorithm developed. Lastly, we address the issue of exploration under a decentralized setting, providing a relation between the exploration rate and the difference from the optimal value.

In summary, we have developed efficient algorithms for model-free RL for centralized and decentralized decision making under uncertainty. This is accomplished through BNPMs, online learning, and flexible policy representations. The results demonstrate that our methods are able to solve large and realistic problems with infinite horizons in a data-driven fashion, and yield high-quality policies. The increased flexibility and scalability are instrumental for a broad range of applications. Therefore, these contributions offer a set of new tools for intelligent systems to solve problems that can be described by MDPs, POMDPs and DEC-POMDPs.

7.2 Future work

There are many intriguing open questions for applying BNPMs for RL and sequential decision making under uncertainty, as well as many interesting applications involving BNPM-based planning and learning. We here discuss some of these issues and applications.

7.2.1 *Optimal exploration strategies for GPQ*

For GPQ learning, we use the predictive variance of the GP Q value to control exploration, and are able to achieve significantly better empirical results than ϵ -greedy exploration. We are able to demonstrate the convergence of the proposed algorithm. However, there is an important question remaining to be answered, that is, whether there exists a bound for the sample complexity for the proposed GPQ learning meth-

ods. We omitted such a study in this thesis and argued that such a bound depends on the topology of action-state space. Even though there are very few PAC-optimal RL algorithms existing for continuous action-state space, recently Pazis and Parr [104] proposed a model-free, PAC-optimal algorithm for exploration in continuous space, called C-PAC. Under the assumptions that the Q value is Lipschitz continuous and the covering number for the action-state space is finite, the bound of sample complexity of C-PAC is derived. It is foreseeable that the results of C-PAC might be able to shed some light on studying the sample complexity of the GPQ learning algorithm.

7.2.2 Temporal and spatial(state) abstraction and BNPMs

In this thesis, we only considered standard RL frameworks with one-step modeling of the environment, that is the state trajectory is made up of discrete-time transitions without modeling the actual time interval between any two adjacent decisions. There is no notion of a course of actions persisting over a variable period of time. Moreover, the algorithms are designed based on the pre-specified state/observation spaces, the size of which can be too huge to handle in real problems. To accommodate large scale and more realistic problems, besides developing scalable inference algorithms, it is also necessary to abstract states and temporal actions by employing hierarchical state space structures and macro actions. With the hierarchal models, planning, search, and learning can be efficient, with less complex computation in higher(abstract) levels and refined in lower level [103].

Temporal abstraction has been studied for MDPs [134] and POMDPs [141], and has recently been extended to multi-agent settings [6]. However, there are very few applications of BNPMs in these frameworks. In existing hierarchical modes, such as hierarchies of abstract machines (HAM) [103] and hierarchical hidden Markov Model (HHMM) [45], one has to set the number of abstract states and the level of hierarchy

in advance; hence they are unable to handle model uncertainty in a principled way.

These existing issues provide further research opportunities, especially for the application of BNPMs. Recently, there are Bayesian nonparametric extensions of HHMM and Semi-HMM, such as iHHMM [55] and HDP-HSMM [60]. These models are promising candidates to address the issue of state abstraction and temporal abstraction, respectively; hence they are worth further research in the RL context.

7.2.3 *Speed up BNPMs*

Advanced BNPMs, such as iHHMM [55] and HDP-HSMM [60] generally rely on MCMC methods for inference. While these methods are powerful in their capability to capture complex structures in data without requiring explicit model selection, they suffer practical shortcomings. In particular, they are not ideal for use in contexts where performing inference quickly and reliably on large volumes of streaming data is crucial for timely decision making, such as autonomous robotic systems.

There are two main lines of research aiming to scale up the inference for BNPMs, which are possible candidates for sequential decision-making problems. The first is based on stochastic variational Bayesian (VB) inference [56], which combines the idea of VB and stochastic gradient descent. That is, instead of putting all the data into the memory and performing a batch update, stochastic VB processes one min-batch at a time and updates the global parameters based on the linear combination of old sufficient statistics and the sufficient statistics computed from the new data. Recent research has mainly focused on streaming and distributed asynchronous settings [22] and nonconjugate models [148].

The second thrust for accelerating BNPMs is based on the small variance asymptotic analysis. This method was first applied to the (Hierarchical) Dirichlet Processes (HDPs) mixture model [70]. By using small variance asymptotic for Gibbs samplers, it is able to derive an optimization algorithm similar to that of k-means like algorithm,

which enjoys both fast speed and convergence guarantee. It was later generalized to exponential family DP mixture models [59], Beta processes (BPs) [23], dependent Dirichlet processes (DDPs) [27], as well as hidden Markov models (HMMs) [119].

Being able to adapt these inference algorithms for complex BNPMs, such as iHHMM [55] and HDP-HSMM [60], will further scale up current reinforcement learning methods, for more realistic sequential decision models. Therefore, speeding up BNPMs is considered an important direction for future research.

7.2.4 *Unified view of Model-free RL for (DEC)-POMDPs*

We have focused on employing FSC-type of graphical models for representing the policies of (DEC-)POMDPs and employing variational Bayesian (VB) methods for inference. However, the VB algorithm is prone to local optima. Moreover, because of the involvement of hidden variables, the theoretical properties such as sample complexity and convergence rate are difficult to analyze. As it was shown by the experimental results in Chapter 4, it is not necessary to memorize all the history for good decision making; we only need to remember those core histories. This observation motivates us to look at an alternative representation of (DEC-)POMDPs: predictive state representations (PSRs) [78]. PSRs represent the state of a dynamic system by tracking occurrence probabilities of a set of future observable event (called tests or characteristic events) conditioned on past observable events (core histories or indicative events), and has been shown to have the same representation power as (DEC-)POMDPs. It has been shown that PSRs can be used to represent policies instead of environment dynamics by switching the roles of actions and observation [153, 19]. Moreover, efficient spectral methods [18] have been developed to learn PSRs with statistical consistency and avoidance of local optima. It might be interesting to combine these two ideas together and provide theoretically more sound model-free RL for (DEC-)POMDPs in the future.

7.2.5 *Domains with continuous states, observations and actions*

We have focused on solving the (DEC)-POMDP with finite states, observations and observations. Because the underlying state space is not directly related to policy representation, it is possible to extend FSC-type graphical models to accommodate policy representations for domains that involve continuous states and actions. A naive approach would be to quantize the action-observation space into a finite number of bins and apply the method developed in this thesis. More advanced approaches would allow the continuous actions to be represented by parametric or nonparametric functions specified through the local policy, and allow continuous observations to be clustered into finite groups. Therefore, it is foreseeable that BNPMs would play an important role in more realistic problems, that involve uncountable actions and observations.

7.2.6 *Transfer Learning*

We only considered single-task RL. However, in their lifetime, autonomous agents may be required to perform a series of different tasks. In order to accelerate the speed of learning the optimal policies for several different tasks, and to avoid learning from scratch every time a different task is encountered, it is important that knowledge gained from past experiences is transferred and exploited by the agents. A large body of transfer learning (TL) methods have been developed in the machine learning community in the last few decade; an excellent survey is available in [99]. However, designing effective TL algorithm for RL tasks is still a growing research area [137]. Furthermore, in recent years, TL research is gaining significant interest, as it promises to markedly reduce the samples and time required to obtain feasible RL solutions. Particularly, under the advocacy of lifelong learning [142], TL not only accelerates the learning speed for solving similar tasks, but also helps the agents to identify new tasks automatically. BNPMs allow autonomous transfer by allowing the clustering

of different tasks according to their shared features (in reward function, dynamics or value function). Furthermore, BNPMs are flexible in that they are capable of identifying and clustering previously unseen tasks. Therefore, several authors have explored BNPs for solving TL problems in RL, especially for a class of TL problems called multi-task reinforcement learning (MTRL). In the future, it is of interest to further develop BNPM-based RL TL algorithms, such as those described by [80], that facilitate autonomous transfer by updating and transferring features, identifying new tasks, and removing old irrelevant tasks.

Bibliography

- [1] D. Aberdeen and J. Baxter. Scalable internal-state policy-gradient methods for POMDPs. In *ICML*, pages 3–10. Morgan Kaufmann Publishers Inc., 2002.
- [2] C. Amato. *Increasing scalability in algorithms for centralized and decentralized partially observable Markov decision processes: Efficient decision-making and coordination in uncertain environments*. PhD thesis, University of Massachusetts Amherst, 2010.
- [3] C. Amato, D. S. Bernstein, and S. Zilberstein. Optimizing fixed-size stochastic controllers for POMDPs and decentralized POMDPs. *JAAMAS*, pages 819–840, 2009.
- [4] C. Amato, G. Chowdhary, A. Geramifard, N. K. Ure, and M. J. Kochenderfer. Decentralized control of partially observable Markov decision processes. In *Proceedings of the Conference on Decision and Control*, 2013.
- [5] C. Amato, G. Konidaris, G. Cruz, C. A. Maynor, J. P. How, and L. P. Kaelbling. Planning for decentralized control of multiple robots under uncertainty. *CoRR*, abs/1402.2871, 2014.
- [6] C. Amato, G. D. Konidaris, and L. P. Kaelbling. Planning with macro-actions in decentralized POMDPs. 2014.
- [7] J. Asmuth, L. Li, M. L. Littman, A. Nouri, and D. Wingate. A Bayesian sampling approach to exploration in reinforcement learning. In *UAI*, pages 19–26. AUAI Press, 2009.
- [8] L. C. Baird. Residual algorithms: Reinforcement learning with function approximation. In *ICML*, pages 30–37, 1995.
- [9] B. Banerjee. Pruning for monte carlo distributed reinforcement learning in decentralized pomdps. In *AAAI*, 2013.
- [10] B. Banerjee, J. Lyle, L. Kraemer, and R. Yellamraju. Sample bounded distributed reinforcement learning for decentralized POMDPs. In *AAAI*, 2012.

- [11] M. J. Beal. *Variational Algorithms for Approximate Bayesian Inference*. PhD thesis, Gatsby Computational Neuroscience Unit, University College London, 2003.
- [12] A. Benveniste, P. Priouret, and M. Métivier. *Adaptive algorithms and stochastic approximations*. Springer-Verlag New York, Inc., New York, NY, USA, 1990.
- [13] D. S. Bernstein, C. Amato, E. A. Hansen, and S. Zilberstein. Policy iteration for decentralized control of Markov decision processes. *JAIR*, 34:89–132, 2009.
- [14] D. S. Bernstein, R. Givan, N. Immerman, and S. Zilberstein. The complexity of decentralized control of Markov decision processes. *MOR*, 27(4):819–840, 2002.
- [15] D. S. Bernstein, S. Zilberstein, R. Washington, and J. L. Bresina. Planetary rover control as a Markov decision process. In *Proceedings of the The Sixth International Symposium on Artificial Intelligence, Robotics and Automation in Space*, 2001.
- [16] C. M. Bishop et al. *Pattern recognition and machine learning*, volume 1. springer New York, 2006.
- [17] D. Blackwell. Discounted dynamic programming. *Ann. Math. Stat*, 36:226235, 2003.
- [18] B. Boots. *Spectral Approaches to Learning Predictive Representations*. PhD thesis, Carnegie Mellon University, December 2012.
- [19] A. Boularias and B. Chaib-draa. Predictive representations for policy gradient in POMDPs. In *ICML*, pages 65–72. ACM, 2009.
- [20] J. Boyan and A. Moore. Generalization in reinforcement learning: Safely approximating the value function. In *Neural Information Processing Systems 7*, pages 369–376, 1995.
- [21] R. I. Brafman and M. Tennenholtz. R-max - a general polynomial time algorithm for near-optimal reinforcement learning. *JMLR*, 3:213–231, 2003.
- [22] T. Broderick, N. Boyd, A. Wibisono, A. C. Wilson, and M. Jordan. Streaming variational Bayes. In *NIPS*, pages 1727–1735, 2013.
- [23] T. Broderick, B. Kulis, and M. I. Jordan. MAD-Bayes: MAP-based asymptotic derivations from Bayes. In *ICML (3)*, pages 226–234, 2013.
- [24] L. Busoniu, R. Babuska, B. D. Schutter, and D. Ernst. *Reinforcement Learning and Dynamic Programming Using Function Approximators*. CRC Press, Automation and Control Engineering Series, 2010.

- [25] C. Cai, X. Liao, and L. Carin. Learning to explore and exploit in POMDPs. In *NIPS*, pages 198–206, 2009.
- [26] C. Cai, X. Liao, and L. Carin. Learning to explore and exploit in POMDPs. In *NIPS*, pages 198–206, 2009.
- [27] T. Campbell, M. Liu, B. Kulis, J. P. How, and L. Carin. Dynamic clustering via asymptotics of the dependent Dirichlet process mixture. In *NIPS*, pages 449–457, 2013.
- [28] O. Cappé and E. Moulines. Online EM algorithm for latent data models. *Journal of the Royal Statistical Society Series B*, 71(3):593–613, 2009.
- [29] A. R. Cassandra. A survey of POMDP applications. In *Working Notes of AAAI 1998 Fall Symposium on Planning with Partially Observable Markov Decision Processes*, pages 17–24, 1998.
- [30] A. L. Chambers, P. Smyth, and M. Steyvers. Learning concept graphs from text with stick-breaking priors. In *NIPS*, pages 334–342, 2010.
- [31] G. Chowdhary, M. Liu, R. C. Grande, T. J. Walsh, and J. P. How. Off-policy reinforcement learning with Gaussian processes. In *The 1st Multidisciplinary Conference on Reinforcement Learning and Decision Making (RLDM)*, 2013.
- [32] G. Chowdhary, M. Liu, R. C. Grande, T. J. Walsh, J. P. How, and L. Carin. Off-policy reinforcement learning with Gaussian processes. *Acta Automatica Sinica special issue on advances in reinforcement learning and adaptive control*, page (to appear), 2014.
- [33] J. J. Chung, N. R. Lawrance, and S. Sukkarieh. Gaussian processes for informative exploration in reinforcement learning. In *ICRA*, 2013.
- [34] L. Csató and M. Opper. Sparse on-line Gaussian processes. *Neural Computation*, 14(3):641–668, 2002.
- [35] M. P. Deisenroth. *Efficient reinforcement learning using Gaussian processes*. PhD thesis, Karlsruhe Institute of Technology, 2010.
- [36] B. Delyon, M. Lavielle, and Moulines. Convergence of a stochastic approximation version of the EM algorithm. *The Annals of Statistics*, 27(1):94–128, 1999.
- [37] A. Dempster, N. Laird, and D. Rubin. Maximum likelihood from incomplete data via the EM algorithm. *Journal of Royal Statistical Society B*, 39:1–38, 1977.

- [38] F. Doshi-Velez. Nonparametric bayesian policy priors for reinforcement learning. In *NIPS*, pages 532–540. 2010.
- [39] F. Doshi-Velez, D. Wingate, N. Roy, and J. Tenenbaum. The infinite partially observable markov decision process. In *NIPS*, pages 477–485, 2009.
- [40] Y. Engel, S. Mannor, and R. Meir. The kernel recursive least-squares algorithm. *Signal Processing, IEEE Transactions on*, 52(8):2275–2285, 2004.
- [41] Y. Engel, S. Mannor, and R. Meir. Reinforcement learning with Gaussian processes. In *International Conference on Machine Learning (ICML)*, 2005.
- [42] Y. Engel, P. Szabo, and D. Volkinshtein. Learning to control an octopus arm with gaussian process temporal difference methods. In *NIPS 18*, 2005.
- [43] D. Ernst, P. Geurts, and L. Wehenkel. Tree-based batch mode reinforcement learning. *JMLR*, 6:503–556, April 2005.
- [44] T. Ferguson. A Bayesian analysis of some nonparametric problems. *Annals of Statistics*, 1:209–230, 1973.
- [45] S. Fine, Y. Singer, and N. Tishby. The hierarchical hidden Markov model: Analysis and applications. *Machine learning*, 32(1):41–62, 1998.
- [46] D. Grady, M. Moll, and L. E. Kavraki. Automated model approximation for robotic navigation with POMDPs. In *ICRA*, pages 78–84. IEEE, 2013.
- [47] C. Guestrin, D. Koller, and R. Parr. Multiagent planning with factored mdps. *Advances in neural information processing systems*, 14:1523–1530, 2001.
- [48] A. Guez, D. Silver, and P. Dayan. Scalable and efficient Bayes-adaptive reinforcement learning based on Monte-Carlo tree search. *JAIR*, 48:841–883, 2013.
- [49] W. M. Haddad and V. Chellaboina. *Nonlinear Dynamical Systems and Control: A Lyapunov-Based Approach*. Princeton University Press, Princeton, 2008.
- [50] L. Hannah and D. B. Dunson. Approximate dynamic programming for storage problems. In *ICML*, pages 337–344, 2011.
- [51] E. A. Hansen. Solving POMDPs by searching in policy space. In *UAI*, pages 211–219. Morgan Kaufmann Publishers Inc., 1998.
- [52] E. A. Hansen, D. S. Bernstein, and S. Zilberstein. Dynamic programming for partially observable stochastic games. In *AAAI*, volume 4, pages 709–715, 2004.

- [53] E. A. Hansen, D. S. Bernstein, and S. Zilberstein. Dynamic programming for partially observable stochastic games. In *Proceedings of the Nineteenth National Conference on Artificial Intelligence*, pages 709–715, 2004.
- [54] E. A. Hansen, D. S. Bernstein, and S. Zilberstein. Dynamic programming for partially stochastic games. In *NCAI*, pages 709–715, 2004.
- [55] K. A. Heller, Y. W. Teh, and D. Görür. Infinite hierarchical hidden Markov models. In *AISTATS*, pages 224–231, 2009.
- [56] M. D. Hoffman, D. M. Blei, C. Wang, and J. Paisley. Stochastic variational inference. *JMLR*, 14(1):1303–1347, 2013.
- [57] H. Ishwaran and L. F. James. Gibbs sampling methods for stick-breaking priors. *JASA*, 96(453), 2001.
- [58] S. Ji, R. Parr, H. Li, X. Liao, and L. Carin. Point-based policy iteration. In *NCAI*, volume 22, page 1243, 2007.
- [59] K. Jiang, B. Kulis, and M. I. Jordan. Small-variance asymptotics for exponential family Dirichlet process mixture models. In *NIPS*, pages 3167–3175, 2012.
- [60] M. J. Johnson and A. S. Willsky. Bayesian nonparametric hidden semi-Markov models. *JMLR*, 14(1):673–701, 2013.
- [61] T. Jung and P. Stone. Gaussian processes for sample efficient reinforcement learning with rmax-like exploration. In *ECML*, 2012.
- [62] L. P. Kaelbling, M. L. Littman, and A. R. Cassandra. Planning and acting in partially observable stochastic domains. *Artificial Intelligence*, 101:99–134, 1998.
- [63] S. Kakade, M. Kearns, and J. Langford. Exploration in metric state spaces. In *ICML*, volume 3, pages 306–312, 2003.
- [64] R. E. Kalman. A new approach to linear filtering and prediction problems. *Journal of basic Engineering*, 82(1):35–45, 1960.
- [65] M. Kearns and S. Singh. Near-optimal reinforcement learning in polynomial time. *Machine Learning*, 49(2-3):209–232, 2002.
- [66] M. Kearns and S. P. Singh. Near-optimal performance for reinforcement learning in polynomial time. In *ICML*, pages 260–268, 1998.
- [67] H. K. Khalil. *Nonlinear Systems*. Macmillan, New York, 2002.

- [68] J. Z. Kolter and A. Y. Ng. Near-Bayesian exploration in polynomial time. In *ICML*, pages 513–520. ACM, 2009.
- [69] J. Z. Kolter and A. Y. Ng. Regularization and feature selection in least-squares temporal difference learning. In *ICML*, 2009.
- [70] B. Kulis and M. I. Jordan. Revisiting k-means: New algorithms via Bayesian nonparametrics. In *ICML*, 2012.
- [71] A. Kumar and S. Zilberstein. Anytime planning for decentralized POMDPs using expectation. In *UAI*, 2010.
- [72] H. J. Kushner and G. G. Yin. *Stochastic Approximation and Recursive Algorithms and Applications*. Springer-Verlag, New York, 2003.
- [73] M. G. Lagoudakis and R. Parr. Least-squares policy iteration. *Journal of Machine Learning Research (JMLR)*, 4:1107–1149, 2003.
- [74] M. Lázaro-Gredilla, J. Quiñonero-Candela, C. E. Rasmussen, and A. R. Figueiras-Vidal. Sparse spectrum Gaussian process regression. *JMLR*, 11:1865–1881, 2010.
- [75] H. Li, X. Liao, and L. Carin. Region-based value iteration for partially observable Markov decision processes. In *ICML*, pages 561–568. ACM, 2006.
- [76] H. Li, X. Liao, and L. Carin. Multi-task reinforcement learning in partially observable stochastic environments. *JMLR*, 10:1131–1186, 2009.
- [77] L. Li. *A unifying framework for computational reinforcement learning theory*. PhD thesis, Rutgers, The State University of New Jersey, 2009.
- [78] M. L. Littman, R. S. Sutton, and S. Singh. Predictive representations of state. *NIPS*, 2:1555–1562, 2002.
- [79] B. Liu, S. Mahadevan, and J. Liu. Regularized off-policy td-learning. In *NIPS*, Lake Tahoe, NV, Dec 2012.
- [80] M. Liu, G. Chowdhary, J. P. How, and L. Carrin. Transfer learning for reinforcement learning with dependent Dirichlet process and Gaussian process. In *NIPS*, Lake Tahoe, NV, December 2012. workshop on Bayesian Nonparametric Models For Reliable Planning And Decision-Making Under Uncertainty.
- [81] M. Liu, X. Liao, and L. Carin. The infinite regionalized policy representation. In *ICML*, pages 769–776, 2011.
- [82] M. Liu, X. Liao, and L. Carin. Infinite regionalized policy representation. In *ICML*, pages 769–776, 2011.

- [83] M. Liu, X. Liao, and L. Carin. Online expectation maximization for reinforcement learning in POMDPs. In *IJCAI*, pages 1501–1507, 2013.
- [84] D. J. Lizotte. Convergent fitted value iteration with linear function approximation. In *NIPS*, pages 2537–2545, 2011.
- [85] L. Macdermed and C. L. Isbell. Point based value iteration with optimal belief compression for dec-pomdps. In *NIPS*, pages 100–108, 2013.
- [86] O. Madani, S. Hanks, and A. Condon. On the undecidability of probabilistic planning and infinite-horizon partially observable markov decision problems. In *AAAI/IAAI*, pages 541–548, 1999.
- [87] H. R. Maei, C. Szepesvri, S. Bhatnagar, and R. S. Sutton. Toward off-policy learning control with function approximation. In *ICML*, Haifa, Israel, 2010.
- [88] A. K. McCallum. *Reinforcement learning with selective perception and hidden state*. PhD thesis, University of Rochester, 1996.
- [89] F. S. Melo, S. P. Meyn, and M. I. Ribeiro. An analysis of reinforcement learning with function approximation. In *International Conference on Machine Learning (ICML)*, pages 664–671, 2008.
- [90] J. Messias, M. Spaan, and P. Lima. Multi-robot planning under uncertainty with communication: a case study. In *AAMAS 2010*.
- [91] N. Meuleau, K.-E. Kim, L. P. Kaelbling, and A. R. Cassandra. Solving POMDPs by searching the space of finite policies. In *UAI*, pages 417–426. Morgan Kaufmann Publishers Inc., 1999.
- [92] N. Meuleau, L. Peshkin, K.-E. Kim, and L. P. Kaelbling. Learning finite-state controllers for partially observable environments. In *UAI*, pages 427–436. Morgan Kaufmann Publishers Inc., 1999.
- [93] R. Nair, P. Varakantham, M. Tambe, and M. Yokoo. Networked distributed pomdps: A synthesis of distributed constraint optimization and pomdps. In *Proceedings of the National Conference on Artificial Intelligence*, volume 20, page 133. Menlo Park, CA; Cambridge, MA; London; AAAI Press; MIT Press; 1999, 2005.
- [94] F. A. Oliehoek. *Value-Based Planning for Teams of Agents in Stochastic Partially Observable Environments*. PhD thesis, Informatics Institute, University of Amsterdam, Feb. 2010.
- [95] F. A. Oliehoek. Decentralized POMDPs. In M. Wiering and M. van Otterlo, editors, *Reinforcement Learning: State of the Art, Adaptation, Learning, and Optimization*, page 471503. Springer Berlin Heidelberg, Berlin, Germany, 2012.

- [96] F. A. Oliehoek, J. F. P. Kooij, and N. Vlassis. The cross-entropy method for policy search in decentralized POMDPs. *2008*, (32):341–357, 2008.
- [97] J. Paisley and L. Carin. Hidden Markov models with stick-breaking priors. *Signal Processing, IEEE Trans. on*, 57(10):3905–3917, 2009.
- [98] J. Pajarinen and J. Peltonen. Periodic finite state controllers for efficient POMDP and DEC-POMDP planning. In *NIPS*, 2011.
- [99] S. J. Pan and Q. Yang. A survey on transfer learning. *IEEE Trans. Knowl. Data Eng.*, 22(10):1345–1359, 2010.
- [100] C. H. Papadimitriou and J. N. Tsitsiklis. The complexity of Markov decision processes. *Mathematics of operations research*, 12(3):441–450, 1987.
- [101] O. Papaspiliopoulos and G. O. Roberts. Retrospective markov chain monte carlo methods for Dirichlet process hierarchical models. *Biometrika*, 95(1):169–186, 2008.
- [102] R. Parr and S. Russell. Reinforcement learning with hierarchies of machines. *NIPS*, pages 1043–1049, 1998.
- [103] R. E. Parr. *Hierarchical control and learning for Markov decision processes*. PhD thesis, University of California, 1998.
- [104] J. Pazis and R. Parr. PAC optimal exploration in continuous space Markov decision processes. In *AAAI*, 2013.
- [105] T. J. Perkins. Reinforcement learning for POMDPs based on action values and stochastic optimization. In *AAAI/IAAI*, pages 199–204, 2002.
- [106] J. Pineau, G. Gordon, S. Thrun, et al. Point-based value iteration: An anytime algorithm for POMDPs. In *IJCAI*, volume 3, pages 1025–1032, 2003.
- [107] J. Pineau, G. J. Gordon, and S. Thrun. Anytime point-based approximations for large POMDPs. *JAIR*, 27:335–380, 2006.
- [108] J. Pineau, M. Montemerlo, M. Pollack, N. Roy, and S. Thrun. Towards robotic assistants in nursing homes: Challenges and results. *Robotics and Autonomous Systems*, 42(3):271–281, 2003.
- [109] P. Poupart and C. Boutilier. Bounded finite state controllers. In *NIPS*, 2003.
- [110] P. Poupart and N. Vlassis. Model-based Bayesian reinforcement learning in partially observable domains. In *International Symposium on Artificial Intelligence and Mathematics (ISAIM)*, 2008.

- [111] P. Poupart, N. Vlassis, J. Hoey, and K. Regan. An analytic solution to discrete Bayesian reinforcement learning. In *ICML*, pages 697–704. ACM, 2006.
- [112] M. L. Puterman. *Markov Decision Processes*. Wiley, 1994.
- [113] F. Qi and L. Losonczi. Bounds for the ratio of two gamma functions. *Journal of Inequalities and Applications*, 2010:204, 2010.
- [114] L. R. Rabiner. A tutorial on hidden Markov Models and selected applications in speech recognition. *Proceedings of the IEEE*, 77(2):257–286, 1989.
- [115] C. Rasmussen and M. Kuss. Gaussian processes in reinforcement learning. *Advances in Neural Information Processing Systems*, 16:751–759, 2004.
- [116] C. Rasmussen and C. Williams. *Gaussian Processes for Machine Learning*. MIT Press, Cambridge, MA, 2006.
- [117] C. P. Robert and G. Casella. *Monte Carlo statistical methods*, volume 319. Citeseer, 2004.
- [118] S. Ross, J. Pineau, B. Chaib-draa, and P. Kreitmann. A bayesian approach for learning and planning in partially observable Markov decision processes. *JMLR*, 12:1729–1770, 2011.
- [119] A. Roychowdhury, K. Jiang, and B. Kulis. Small-variance asymptotics for hidden Markov models. In *NIPS*, pages 2103–2111, 2013.
- [120] S. J. Russell, P. Norvig, J. F. Canny, J. M. Malik, and D. D. Edwards. *Artificial intelligence: a modern approach*, volume 2. Prentice hall Englewood Cliffs, 1995.
- [121] S. Seuken and S. Zilbersteinn. Improved memory-bounded dynamic programming for decentralized POMDPs. In *UAI*, 2010.
- [122] G. Shani, R. I. Brafman, and S. E. Shimony. Model-based online learning of POMDPs. In *Machine Learning: ECML 2005*, pages 353–364. Springer, 2005.
- [123] D. Silver and J. Veness. Monte-Carlo planning in large POMDPs. In *NIPS*, volume 23, pages 2164–2172, 2010.
- [124] R. D. Smallwood and E. J. Sondik. The optimal control of partially observable Markov processes over a finite horizon. *Operational Research*, 21:1071–1088, 1973.
- [125] T. Smith and R. Simmons. Heuristic search value iteration for POMDPs. In *UAI*, pages 520–527. AUAI Press, 2004.

- [126] E. Snelson and Z. Ghahramani. Sparse Gaussian processes using pseudo-inputs. In *NIPS*, pages 1257–1264, 2006.
- [127] A. Somani, N. Ye, D. Hsu, and W. S. Lee. DESPOT: Online POMDP planning with regularization. In *NIPS*, pages 1772–1780, 2013.
- [128] E. J. Sondik. The optimal control of partially observable Markov processes over the infinite horizon: Discounted costs. *Operations Research*, 26(2):282–304, 1978.
- [129] M. T. Spaan and N. A. Vlassis. Perseus: Randomized point-based value iteration for POMDPs. *JAIR*, 24:195–220, 2005.
- [130] A. L. Strehl, L. Li, E. Wiewiora, J. Langford, and M. L. Littman. PAC model-free reinforcement learning. In *ICML*, pages 881–888. ACM, 2006.
- [131] A. L. Strehl and M. L. Littman. A theoretical analysis of model-based interval estimation. In *International Conference on Machine Learning (ICML)*, pages 856–863, 2005.
- [132] R. Sutton and A. Barto. *Reinforcement Learning: An Introduction*. MIT Press, Cambridge, Massachusetts, 1998.
- [133] R. S. Sutton, H. R. Maei, D. Precup, S. Bhatnagar, D. Silver, C. Szepesvári, and E. Wiewiora. Fast gradient-descent methods for temporal-difference learning with linear function approximation. In *ICML*, 2009.
- [134] R. S. Sutton, D. Precup, S. Singh, et al. Between mdps and semi-mdps: A framework for temporal abstraction in reinforcement learning. *Artificial intelligence*, 112(1):181–211, 1999.
- [135] D. Szer, F. Charpillet, and S. Zilberstein. MMA: A heuristic search algorithm for solving decentralized POMDPs. In *UAI*, 2005.
- [136] G. Taylor and R. Parr. Kernelized value function approximation for reinforcement learning. In *ICML*, pages 1017–1024. ACM, 2009.
- [137] M. E. Taylor and P. Stone. Transfer learning for reinforcement learning domains: A survey. *JMLR*, 10(1):1633–1685, 2009.
- [138] W. L. Teacy, G. Chalkiadakis, A. Farinelli, A. Rogers, N. R. Jennings, S. McClean, and G. Parr. Decentralized bayesian reinforcement learning for online agent collaboration. In *AAMAS*, pages 417–424, 2012.
- [139] Y. W. Teh, M. I. Jordan, M. J. Beal, and D. M. Blei. Hierarchical Dirichlet processes. *JASA*, 101:1566–1581, 2006.

- [140] S. A. Terwijn. On the learnability of hidden Markov models. In *Grammatical Inference: Algorithms and Applications*, pages 261–268. Springer, 2002.
- [141] G. Theodorou, S. Mahadevan, and L. P. Kaelbling. Spatial and temporal abstractions in POMDPs applied to robot navigation. Technical report, DTIC Document, 2005.
- [142] S. Thrun. Is learning the n-th thing any easier than learning the first? In *NIPS*, pages 640–646, 1996.
- [143] S. B. Thrun. Efficient exploration in reinforcement learning. 1992.
- [144] M. Toussaint, L. Charlin, and P. Poupart. Hierarchical pomdp controller optimization by likelihood maximization. In *UAI*, pages 562–570, 2008.
- [145] J. N. Tsitsiklis and V. B. Roy. An analysis of temporal-difference learning with function approximation. *IEEE Transactions On Automatic Control*, 42(5):674–690, 1997.
- [146] N. Vlassis and M. Toussaint. Model-free reinforcement learning as mixture learning. In *ICML*, pages 1081–1088, 2009.
- [147] T. J. Walsh, I. Szita, C. Diuk, and M. L. Littman. Exploring compact reinforcement-learning representations with linear regression. In *UAI*, pages 591–598. AUAI Press, 2009.
- [148] C. Wang and D. Blei. Variational Inference in Nonconjugate Models. *JMLR*, 14:1005–1031, 2013.
- [149] C. Wang, J. Paisley, and D. Blei. Online variational inference for the hierarchical Dirichlet process. In *AISTATS*, 2011.
- [150] C. J. Watkins and P. Dayan. Q-learning. *Machine learning*, 8(3-4):279–292, 1992.
- [151] D. J. White. A survey of applications of Markov decision processes. *Journal of the Operational Research Society*, pages 1073–1096, 1993.
- [152] D. Wierstra and M. Wiering. Utlite distinction hidden Markov models. In *ICML*, 2004.
- [153] E. Wiewiora. Learning predictive representations from a history. In *ICML*, pages 964–971. ACM, 2005.
- [154] D. Wingate, N. D. Goodman, D. M. Roy, L. P. Kaelbling, and J. B. Tenenbaum. Bayesian policy search with policy priors. In *IJCAI*, pages 1565–1570, 2011.

- [155] T.-T. Wong. Generalized dirichlet distribution in bayesian analysis. *Applied Mathematics and Computation*, 97(2):165–181, 1998.
- [156] F. Wu, S. Zilberstein, and N. R. Jennings. Monte-Carlo expectation maximization for decentralized POMDPs. In *IJCAI*, pages 397–403. AAAI Press, 2013.
- [157] S. Young. Using POMDPs for dialog management. In *SLT*, pages 8–13, 2006.
- [158] C. Zhang and V. R. Lesser. Coordinated multi-agent reinforcement learning in networked distributed pomdps. In *AAAI*, 2011.
- [159] L. Zheng and S. Cho. A modified memory-based reinforcement learning method for solving POMDP problems. *Neural Process Lett*, (33):187–200, 2011.

Biography

Miao Liu was born in Wuhan, China, in October 1982. He received the B.S. and M.S. degrees in Electrical Engineering from the Huazhong University of Science and Technology, Wuhan, China, in 2005 and 2007 respectively. He worked in the Department of Electrical and Computer Engineering at Old Dominion University, VA from 2007 to 2009 as a research and teaching assistant. He enrolled in the Department of Electrical and Computer Engineering at Duke University in 2009. He is expected to receive his Ph.D. degree from Duke University in June 2014. He will join the Massachusetts Institute of Technology as a Postdoctoral Associate in July 2014. His research interests include statistical signal and image processing, machine learning and artificial intelligence.