Abstract of "A Theory of Abstraction in Reinforcement Learning"

by David Abel, Ph.D, Brown University, May 2020.

Reinforcement learning defines the problem facing agents that learn to make good decisions through action and observation alone. To be effective problem solvers, such agents must efficiently explore vast worlds, assign credit from delayed feedback, and generalize to new experiences, all while making use of limited data, computational resources, and perceptual bandwidth. Abstraction is essential to all of these endeavors. Through abstraction, agents can form concise models of their environment that support the many practices required of a rational, adaptive decision maker. In this dissertation, I present a theory of abstraction in reinforcement learning. I first offer three desiderata for functions that carry out the process of abstraction: they should 1) preserve representation of near-optimal behavior, 2) be learned and constructed efficiently, and 3) lower planning or learning time. I then present a suite of new algorithms and analysis that clarify how agents can learn to abstract according to these desiderata. Collectively, these results provide a partial path toward the discovery and use of abstraction that minimizes the complexity of effective reinforcement learning.

A Theory of Abstraction in Reinforcement Learning

David Abel

A dissertation submitted in partial fulfillment of the
requirements for the Degree of Doctor of Philosophy in
the Department of Computer Science at Brown University

Providence, Rhode Island
May 2020

This dissertation by David Abel is accepted in its present form by
the Department of Computer Science as satisfying the dissertation requirement
for the degree of Doctor of Philosophy.

Date ______________ ____

Michael L. Littman, Advisor

Recommended to the Graduate Council

Date ______________ ____

George Konidaris, Reader

Date ______________ ____

Stefanie Tellex, Reader

Date ______________ ____

Peter Stone, Reader

Date ______________ ____

Will Dabney, Reader

Approved by the Graduate Council

Date ______________ ____

Andrew G. Campbell, Dean of the Graduate School

VITA

David Abel was born in Portland, Oregon. He received his Bachelors in computer science and philosophy from Carleton College, during which he spent a semester in Budapest, Hungary. He then attended Brown University, where he received a Masters in each of computer science and philosophy en route to his Ph.D. At Brown, he taught CS8: A First Byte of Computer Science and a Summer course he designed for high school students on AI and Society. David also spent time as a research intern at Microsoft Research in New York City, the University of Oxford, and DeepMind London. He received the Open Graduate Fellowship to pursue a Masters in philosophy and the Presidential Award for Excellence in Teaching.

During his Ph.D, David published the following papers: *Value Preserving State-Action Abstractions* [11], *The Efficiency of Human Cognition Reflects Planned Use of Information Processing* [132], *The Value of Abstraction* [131], *The Expected-Length Model of Options* [10], *Finding Options that Minimize Planning Time* [144], *Discovering Options for Exploration by Minimizing Cover Time* [145], *State Abstraction as Compression in Apprenticeship Learning* [9], *State Abstractions for Lifelong Reinforcement Learning* [6], *Policy and Value Transfer in Lifelong Reinforcement Learning* [7], *Bandit-Based Solar Panel Control* [8], and *Near Optimal Behavior via Approximate State Abstraction* [4].

To my fiancée Elizabeth, my brother Mike, and my parents Mark & Diane.

ACKNOWLEDGEMENTS

One of the greatest pleasures of research is working with brilliant and thoughtful individuals from all over the world. I am grateful to my mentors for their encouragement and guidance over the years, my collaborators for sharing their time and ideas, and the support and friendship of many.

MENTORS

First, to my advisor, Michael Littman. From day one Michael was able to blend encouragement with structured advice. He always helped me feel like I was making progress and knew how to point me in the right direction. I regularly left our meetings with a clear objective and a wealth of excitement to learn about a new topic or work toward a new result. On top of this, Michael is both immeasurably wise, funny, and humble. I can recall many occasions where I pitched a new problem to him that I was stuck on—he quickly figured out a solution, and would carefully lead me to the conclusion in a way that nurtured my own sense of curiosity and discovery, and made it fun along the way. Some of my absolute favorite experiences in life are these moments Michael and I stared down a tricky problem on the whiteboard in his office. These conversations eventually grew into the publications and ideas that constitute this dissertation. Ultimately, Michael taught me what good science looks like. He shaped how I ask questions, how to do my best to be unbiased, and how to proceed in the presence of profound challenges and uncertainty. Richard Hamming's "You and Your Research" suggests that great research is done with great courage: through his guidance, Michael has inspired me to try to be courageous in research, to ask big questions, and to methodically seek out principled answers. He has also taught me to embrace my own instincts and curiosities, and to how to balance my own philosophy with existing literature. Michael: thank you for your time, support, and trust, for giving me the freedom to explore, allowing me to make safe mistakes, and being such an outstanding mentor, advisor, and person. I am so profoundly lucky to have had the opportunity to work with you! I will never forget our time doing research together, and I look forward to continuing our collaborations and discussions in the future.

To George Konidaris, thank you for the many fascinating discussions on abstraction and AI, and for the countless pieces of life advice you have offered over the years. I was inspired by your research early on in my Ph.D, so it was a genuine delight to discover that you were coming to Brown, and even more so that I had the good fortunate to collaborate with you for many years. I have such a deep admiration for your research, vision, and methods, and I am honored to have had the chance to work with you. Perhaps the moment in my Ph.D when I laughed the hardest was up against the ICML 2018 submission deadline for our work on transfer learning in RL. I used the term "epistemic guise" in a draft of the paper, and received a note from you "Dave what is an epistemic guise." I have been sure not used this phrase since (until now)! Thank you George for your time and mentorship!

Next, to my Masters advisor, Stefanie Tellex: thank you for taking a chance on me as a researcher, and for inspiring me to pursue AI research. You first taught me how to read papers, how to give talks, how to formulate an appropriate research question, and many other crucial aspects of how to be an effective scientist. I can still remember pitching my first research problem; you were eager to engage with the ideas and gave me the tools I needed to systematize my study. I will always finish my talks with a contributions slide and include slide numbers, thanks to her many essential tips. Moreover, I will always remember the moment in the midst of a paper deadline when you paused, told my co-authors and me to bubble up a level and make sure we were doing okay as people. Thank you, Stefanie!

To Will Dabney, thank you for mentoring me at DeepMind, serving on my committee, and for a both fruitful and enjoyable collaboration—I learned a great deal from working with you and I look forward to our future research together!

To Peter Stone, thank you for your time in serving on my committee, your advice, and for your thoughtful questions that still have me thinking!

To Colin Day, or "D-Day" (In Memoriam), I owe the initial spark of my love of math, elegant theory, and the belief that hard work can translate to deep understanding, even when facing challenges that seem insurmountable. You were an amazing teacher and mentor. Thanks, D-Day!

To Jake Stults (In Memoriam), my first computer science teacher. Thank you for cultivating my initial curiosity about computation, AI, and logic, for sharing music with me, and for your constant encouragement early on in life. Thanks, Jake!

To my other mentors, for their time, patience, advice, and helping me find my path: Jason Decker, Fernando Diaz, Owain Evans, David Liben-Nowell, James MacGlashan, Ana Moltchanova, Joshua Schechter, and Lawson L.S. Wong, thank you all!

FAMILY, FRIENDS, COLLABORATORS, AND COLLEAGUES

To my fiancée, Elizabeth Thiry, I am forever grateful for your love and support throughout these five years. Thank you for listening to countless practice talks, inspiring me to take on new challenges, filling our days with epic experiences all the way from Sweden to Hungary to Boston, and always believing in me. The Ph.D has been such a blast and I look forward to the rest of our life together!

To my parents, Diane and Mark Abel, thank you for your love and support, all the way from 0th grade to 23rd! I am grateful for all you have done—I am so fortunate to have such wonderful parents. From early on in life you were both unwavering in your encouragement, helping me to ask questions and to have fun with learning and life more generally. Thank you for the many insightful conversations, trips, and experiences that have shaped nearly every aspect of my life, and for helping me find my way!

To my brother, Michael Abel, thank you for being such an amazing role model. I still remember the many deep discussions we had about science and philosophy as we grew up together. I vividly recall taking a drive together in the west hills of Portland one night and discussing different kinds of infinity for hours. You were always supportive of these discussions and helped me feel comfortable with wanting to think about philosophy and math (you made it fun, too!). I would not have had the confidence to pursue this degree without you as a role model. Most crucially, Mike is *amazing* at balancing life. Despite the Ph.D being difficult, I always set aside time for friends, family, and personal time because of the example he set. Thank you, Mike!

To Nate Bowditch, thank you for all of the good times, for being an outstanding roommate, person, and friend, and for the many deep discussions and epic adventures we have had together!

To Pablo Leon-Luna, thank you for the many thoughtful conversations and good times, for inspiring me to pursue what I truly love, and for all your support and friendship over the years!

To Ellis Hershkowitz, thank you for helping me to discover the kind of research that I value, for the many productive collaborations, fascinating conversations, and good times!

To those that were especially close collaborators or friends throughout my Ph.D, I am so thankful for your time, support, and friendship: Cam Allen, Enrique Areyan, Dilip Arumugam, Kavosh Asadi, Akshay Balsburamani, Christina Donovan, Chris Grimm, Mark Ho, Yuu Jinnai, Ben LeVeque, Torben Noto, and Greg Yauney, thank you!

Lastly, I am grateful to all of the friends, extended family, colleagues, and collaborators that have helped me on this journey! Thanks to Belle & Sid Abel (In Memoriam), Niko Adamstein, Lori Agresti, Suzanne Alden, Gabriel Barth-Maron, Kathy Billings, Lisa and Jason Bogardus, Ben Breen, Nick Brenner, Evan Cater, Lauren Clarke, Erica Clausen, Genie DeGouveia, Marie desJardins, Katie Franklin, Tomasz Kalbarczyk, Khimya Khetarpal, Brian Kimpson, Akshay Krishnamurthy, Rita & Harry Krych (In Memoriam), Erwan Lecarpentier, Meg & Alexander Leiken, Jason Liu, Jane Martin, Debbie Osterman, Matt

Overlan, Katy Parsons, Bree & Ohm Patel, Robby Plowman, Jesse Polhemous, Emily Reif, Mel Roderick, Mark Rowland, Hannah Roy, Brandon Saranik, Evan Schwed, Gunnar Sigurdson, Satinder Singh, Andy Smith, Katherine & Julius Thiry, Raphael Townshend, Nate Umbanhowar, Ally Wharton, Will Whitney, John Winder, Danfei Xu, and to all of the members of RLAB & Brown CS, thank you all!

CONTENTS

LIST OF TABLES

LIST OF FIGURES

Part 1

PRELIMINARIES

1

INTRODUCTION

Suppose you take a walk in the woods. You find yourself surrounded by pine trees, chirping birds, a peaceful lake, and frogs eating delicious mealworms. A friend returns from a walk and relays a story of a goat miraculously walking up a steep mountain side. Your stomach grumbles and you deliberate over whether to eat an apple from your backpack or to start a campfire and cook a hot meal.

Consider just how many activities are at your finger tips: you could climb a tree, discuss philosophy with your friend, navigate to a nearby stream by listening for rushing water, or create a map of the territory. To engage in any of these practices in this complex and changing environment you must be capable of making hundreds of well-chosen decisions that move you toward a particular objective. Moreover, you must make these decisions while relying on imperfect memory and noisy sensory channels; some light hits your retina indicating the sun has risen, changes in sound pressure are processed through your ears to notify you of an impending thunderstorm, and nerve endings in your feet tell you your boots are wet. Somehow, you map this continual stream of observations to a choice of actions that moves you toward any of the above goals. How is this even remotely possible?

Central to an agent's ability to solve problems is the capacity to reason *abstractly*—walking into a tree will cause pain while moving around it may not. Hence, representing

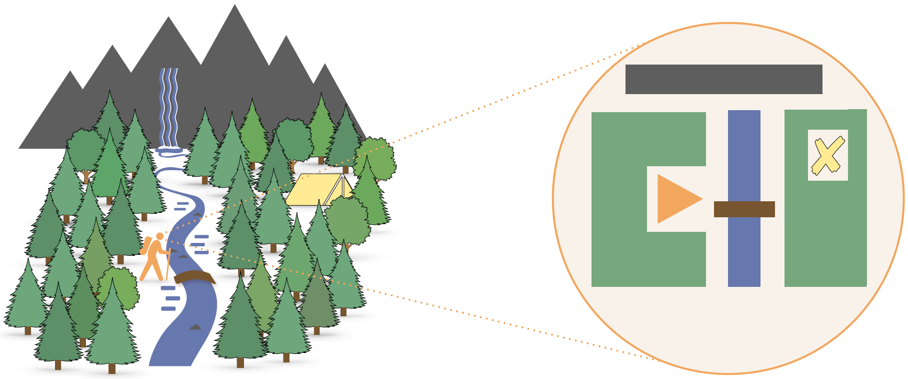


Figure 1.1: The process of abstraction.

particular patterns of visual stimuli as a tree that persists through time is immensely useful. Additionally, conveying to other agents that trees possess the "do-not-walk-into" property is likely to be critical to the overall fitness of the community at large. Indeed, many central practices of agency rely on abstraction: speculating about and learning from hypothetical scenarios, overcoming new challenges because of their similarity to past experiences, and forming high-level plans spanning months or years that inform immediate action; all of these depend on a concise, adaptive, and abstract representational toolkit. For these reasons, the capacity to learn and make use of appropriate abstract representations is likely to be an essential cognitive skill of any intelligent agent, whether biological or artificial.

In the forest, we might imagine that a hiker trying to return to their tent may reason using the abstract representation pictured in Figure 1.1. With this smaller model that still retains relevant information, the hiker can carry out more valuable computation, explore less, draw more robust inferences, and predict further into the future. These benefits ultimately enable the hiker to safely navigate to their tent and to thrive in nature more generally. Where, though, does this model come from? And how can an agent discover such a model solely by interacting with their surroundings? These questions have long stood as a fundamental puzzle in the science of understanding intelligence.

This dissertation is about the study of abstraction and its role in effective agency. I ground this inquiry by concentrating on computational agents that must learn to solve problems from interaction alone, as captured by the reinforcement learning (RL) problem. Such an agent could consist of a finite state machine reacting to discrete symbols on a tape for the purpose of sorting a list, but also a robot or animal observing the world through sensors and a powerful action space that supports movement through and manipulation of the environment. In this remarkably general framework, we will find footing to make the study of abstraction concrete.

1.1 THE REINFORCEMENT LEARNING PROBLEM

RL defines the problem facing an agent that learns to make useful decisions through observation and action alone. The primary objects of interest in RL are computational agents, the worlds they inhabit, and the interactions thereof. An agent is understood as any entity capable of perception and action, where perception involves the receipt and processing of information from the environment, and action defines the process of committing to a choice from a set of alternative courses of behavior. I sharpen our use of the term "world" in the next chapter, but broadly it is to be understood as a set of possible states of affairs, causal laws that move the world between these states, and an agent that makes decisions in the world based on a stream of observation.

Critical to RL is the assumption that one special observation of the world is a numerical *reward signal* that corresponds to the immediate desirability of a given state of affairs. The objective of an RL agent is then simple: maximize future rewards. Richard S. Sutton and Michael L. Littman have articulated what is known as the *reward hypothesis*, or *reinforcement learning hypothesis*, that states the following:

Definition 1.1. *The **reward hypothesis** states, "all of what we mean by goals and purposes can be well thought of as maximization of the expected value of the cumulative sum of a received scalar signal (reward)" (Sutton, 2004).*

Indeed, reward prediction and learning has long played a role in understanding human and animal cognition [281, 84, 38]. For our present purpose, I assume the hypothesis to be valid, and proceed on the basis that the space of agents that effectively learn to maximize reward can be likened to the space of intelligent agents. I note, however, that a more thorough philosophical treatment of this hypothesis is of deep importance.

With these pieces in play, the RL problem is defined at a high level as follows.

Definition 1.2. *The **reinforcement learning problem** is as follows. An RL agent interacts with its environment via the indefinite repetition of the following two discrete steps:*

1. *The agent receives an observation and a reward.*
2. *The agent learns from this interaction and executes an action.*

This process is pictured in Figure 1.2. The goal of the agent during this interaction is to make decisions so as to maximize its long term received reward.

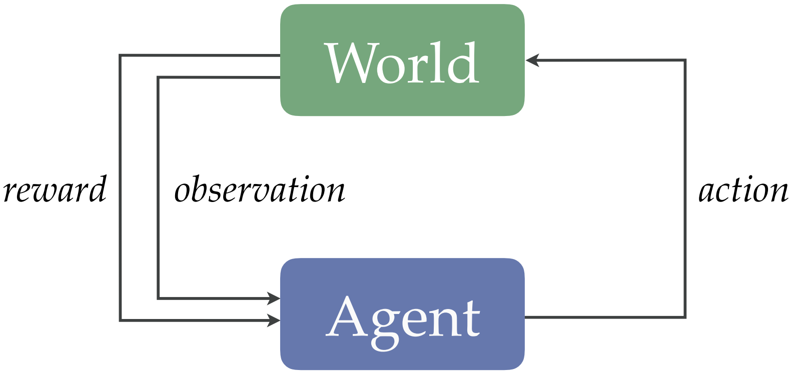


Figure 1.2: The RL problem.

Returning to our peaceful forest, we might imagine that the hiker occupies the world state pictured in Figure 1.1, and is learning about their surroundings to maximize reward. Depending on the reward generating process, the hiker will be incentivized to exhibit different kinds of behavior. For instance, the problem of navigating to a stream might be associated with a reward signal that increases as the hiker gets closer to the water. An effective agent, then, will learn to associate this positive signal with actions that move them toward the stream. Over time, the most effective agents can reach the stream without coming to harm along the way. To define the task of cooking food, we might attach positive reward signal to the experiences of eating tasty food—again, effective agents will be those that can prepare and eat food that is of sufficient levels of tastiness.

It is here that we find the remarkable expressivity of the RL problem: any goal-driven task can be defined in terms of a reward function that is positive when the goal is satisfied, and non-positive otherwise. Moreover, non-terminating behaviors such as controlling an elevator, balancing a pole, survival, or regulating energy on a power grid can *also* be elicited with the right choice of reward function by similar mechanisms.

Richard S. Sutton describes both the appeal and challenge of RL as follows.

> Part of the appeal of reinforcement learning is that it is in a sense the whole [artificial intelligence] problem in a microcosm. The task is that of an autonomous learning agent interacting with its world to achieve a goal. The framework permits the simplifications necessary in order to make progress, while at the same time including and highlighting cases that are clearly beyond our current capabilities, cases that we will not be able to solve effectively until many key problems of learning and representation have been solved. That is the challenge of reinforcement learning.
> (Sutton 1992, p. 2)

I am entirely sympathetic to Sutton's reasoning. Addressing the RL problem is of critical importance to establishing a holistic understanding of intelligence. Even roughly 30 years after the above quote, there are still many "cases that are clearly beyond our current capabilities" at the heart of RL. To be effective, RL agents must address a combination of three classical problems of machine learning:

1. **Generalization:** Given experience from the past, how can an agent better act in the future?

   Example 1: The hiker spots an owl in the woods they have never before seen. How do they know it is an owl?

   Example 2: You approach a door you have never before encountered and manage to open it within seconds.

2. **Exploration:** How can an agent systematically trade off between 1) exploiting what is known to be a reliably good choice with 2) making choices that may lead to new discoveries?

   Example: You visit your favorite restaurant, and deliberate whether to choose your go-to entrée, or to try something new (that you might like even more!).

3. **Credit Assignment:** When feedback is delayed, how can an agent attribute credit to the most causally relevant decisions made previously?

   Example: You study for a test for weeks on end. Also, the night before the test, you eat a bowl of cereal. You ace the test. How can you determine that it was the studying that led to your success, and not the bowl of cereal?

Each of these problems individually is difficult, but in RL, agents must simultaneously address all three. I return to a more technical treatment of some of these problems throughout the dissertation after introducing the mathematical tools of RL in Chapter 2. Fortunately, however, abstraction can help address each of these challenges.

1.2 ABSTRACTION

Indeed, understanding abstraction and its role in agency has long stood as one of the fundamental questions of artificial intelligence (AI), dating back to the famous workshop at Dartmouth that founded the field:

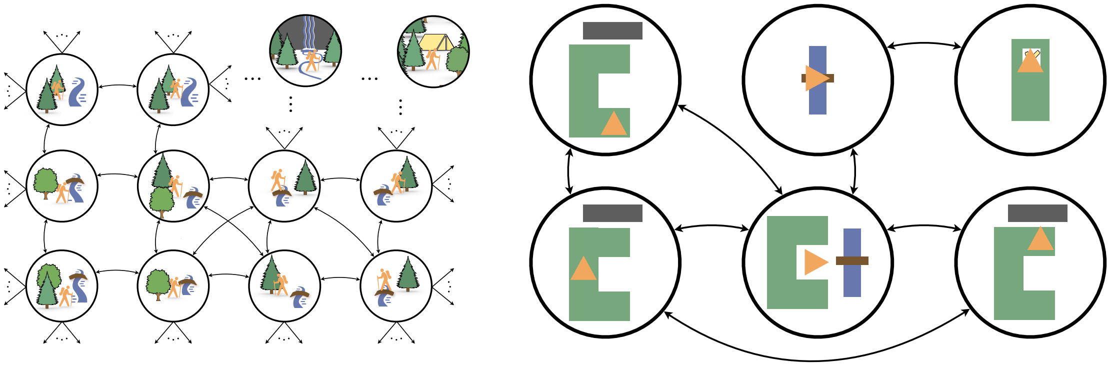
> The study is to proceed on the basis of the conjecture that every aspect of learning or any other feature of intelligence can in principle be so precisely described that a machine can be made to simulate it. An attempt will be made to find how to make machines...form abstractions and concepts.
> (McCarthy et al. 2006, p. 1)

Since this workshop, the study of abstraction in AI and related fields has led to a profound appreciation for the role abstraction can play in both artificial and biological creatures. The focus of this dissertation is naturally on the former, though a large body of research in the cognitive, neuro, and psychological sciences examines the prevalence of abstraction in the representational and decision making practices of humans [50, 158, 321, 297, 48, 338, 267, 97].

The RL problem is perfectly suited to a scientific study of abstraction. Observation, on its own, is far too complicated for an agent to reason with while acting in a changing world. Thinking takes time. Processing, understanding, and reacting to every detail of a history of observations is computationally intractable. Additionally, in deliberating over possible futures, the space of sensible changes to the world that are worth considering is dramatically smaller than that of the possible future observation stream. Behavior, too, can often be defined at multiple levels of abstraction. For instance, an ant may act so as to follow its friend, or choose which precise muscles to twitch to propel its legs.

At a high level, the process of *abstraction* can be divided into two broad categories: 1) state abstraction, which defines the practice of representing only the most relevant properties of the world, and 2) action abstraction, which defines the practice of forming a relevant set of long horizon behaviors available to an agent. In both cases, following Giunchiglia and Walsh [119], I understand abstraction as "the process of mapping a representation of a problem onto a new representation" (1992, p. 1).

Let us return to the woods. Suppose our hiker is trying to navigate back to their camp, and can choose to represent this problem in great detail, or in the abstract, pictured in Figure 1.3. In each case, the hiker deliberates over possible future courses of action. In the first, however, the hiker's actions are modeled in terms of the smallest possible execution



(a) Reasoning in the environment.

(b) Reasoning in the abstract.

Figure 1.3: A comparison of the problem posed to a hiker navigating to their tent in a forest in fine detail (left) and in the abstract (right).

of behavior—a slight step in one direction, or a tilt of the head. In the second, decisions are considered that only change something *substantive* about the environment—previously, the hiker was west of the bridge, and now, they are to the east of the bridge. Depending on the problem, different degrees and types of abstraction will be most effective.

A **state abstraction** determines which changes to the environment count as substantive. As the agent walks toward the bridge, for instance, the clouds shift overhead. The breeze picks up slightly, and a bird flies by. These small changes are likely to be irrelevant to the hiker's objective of crossing the bridge. Conversely, when the hiker has reached the river and can see the bridge, the state has changed in a relevant way. This strategy for reasoning in terms of abstract states of affairs alone is pictured in Figure 1.3b. Note that only six states are required; the hiker might occupy the three distinct regions of the woods west of the river, the bridge, and their camp. These properties obscure many nuances such as weather, precise orientation, and even subtleties about the agent's physical or mental state. For some problems, these six states along would be entirely insufficient for permitting the representation of good behavior.

In Figure 1.3a, we instead see the diversity of state representations available if any slight change to the environment is perceived as a substantive change to world state. The agent could be next to the river, north of the bridge by five paces as opposed to four—or, the agent could be immediately under a tree as opposed to standing near it. For some problems, it is crucial to represent detailed aspects of the environment. For others, however, a state representation that mirrors the figure on the right is more effective.

An **action abstraction** determines the arrows moving between nodes in Figure 1.3b. From the hiker's perspective, abstract actions are simply those behaviors that should be considered in choosing a course of action. Again supposing the hiker is trying to arrive safely at their camp, they may choose to navigate to a nearby hill to gain a view of the surroundings, or may navigate to a known landmark such as the bridge (from which they can quickly return to camp). In contrast, of course, the "primitive" actions define the smallest possible choices available to the hiker—moving a toe, leg, or hand, for instance. Action abstraction has appeared under a variety of names such as skills, temporal abstraction, or macro-actions. I here use the general term of *action abstraction* to capture each of these despite their differences.

Naturally, the two types of abstraction are intimately connected. As presented in Figure 1.3b, there is an explicit sense in which they are related: by some accounts, an abstract action is just a behavior that takes an agent from one abstract state to another. This particular perspective connecting the two abstraction types has a rich history in AI, and will resurface several times throughout this dissertation. However, it is not the only sense in which the two forms of representation might be connected. Konidaris et al. [180] proves that algorithms that build an abstract state representation based on a given collection of abstract actions can preserve desirable properties. Indeed, dating back to the early work of Dietterich [88], the two types of abstraction have been tied together. Near the end of the dissertation, I will return to a technical analysis of state-action abstractions (Part 4).

Finally, repeated application of state or action abstractions can induce **hierarchical abstraction**, through which entities can be represented at varying levels of granularity. In the forest, hierarchical abstraction may permit the hiker to reason using both the left *and* the right representation, depending on the task at hand. As thinking time or data becomes more readily available, a more detailed representation may be used. If, however, a quick decision needs to be made, or if the world is fundamentally unpredictable in great detail at certain time horizons (for instance, it remains difficult to predict the weather a few days away), the hiker may opt for the representation on the right. Many of the fundamental open questions in the area center around hierarchical abstraction, with state and action abstraction serving as palatable chunks that can be analyzed independently. As with state-action abstraction, I will return to hierarchical abstraction near the end of the dissertation in Part 4.

1.3 THESIS STATEMENT

With the main conceptual framework in play, I now highlight the central question addressed by this work:

How do reinforcement learning agents discover and make use of good abstractions?

I answer this question by advancing the following thesis:

Thesis

By drawing on insights from computational complexity theory, decision-theoretic planning, and information theory, it is possible to design efficient algorithms for discovering abstractions that reduce the amount of experience or thinking time an RL agent requires to find a good solution.

To defend this thesis, I introduce three desiderata that articulate which abstractions are useful in RL. At a high level, these desiderata state the following.

$$\textit{Good abstractions for RL are } \underbrace{\textit{easy to discover}}_{\textit{(D1)}} \textit{ and } \underbrace{\textit{enable efficient learning}}_{\textit{(D2)}} \textit{ of } \underbrace{\textit{high value policies.}}_{\textit{(D3)}}$$

I present more detail and justification for these desiderata in Section 2.4.

1.4 CONTRIBUTIONS

The remaining defense of this thesis is organized as follows.

PART 1. In Chapter 2, I provide necessary background on RL (Section 2.1) along with state abstraction (Section 2.2) and action abstraction (Section 2.3). Then, I introduce and motivate the abstraction desiderata in more detail (Section 2.4).

PART 2. The next part is dedicated to **state abstraction**. I present new algorithms and three intimately connected sets of analysis, each targeting the discovery of state abstractions that satisfy the introduced desiderata. In Chapter 3, I develop a formal framework for reasoning about state abstractions that preserve near-optimal behavior. This framework is summarized by Theorem 3.1, which highlights four such sufficient conditions for value-preserving state abstractions. Then, in Chapter 4, I extend this analysis to the *lifelong* RL setting, in which an agent must continually interact with and solve different tasks. The main insight of this chapter is the introduction of PAC state abstractions for the lifelong learning setting, along with results clarifying how to efficiently compute them. Theorem 4.4 illustrates the sense in which these abstractions are guaranteed to preserve good behavior, and Theorem 4.5 shows how many previously solved tasks are sufficient to compute a PAC state abstraction. I highlight results from simulated experiments that illustrate the utility of the introduced types of state abstractions to accelerate learning and planning. Lastly, Chapter 5 brings the tools of information theory to bear on state abstraction. I develop a tight connection between state abstraction and Rate-Distortion theory [284, 43] and the Information Bottleneck Method [318], and exploit this connection to design new algorithms for efficiently constructing state abstractions that elegantly trade off between *compression* and *representation of good behavior*. I extend this algorithmic framework in a variety of ways, illustrating its power for discovering state abstractions that afford sample-efficient learning of good behavior.

PART 3. I then turn to **action abstraction**. In Chapter 6, I present analysis from Jinnai et al. [144] studying the problem of finding abstract actions that make planning as fast as possible—the main result states that this problem is NP-hard in general (under appropriate simplifying assumptions), and is even hard to approximate in polynomial time. Then, in Chapter 7, I address the problem of constructing the *predictive model* that accompanies high level behaviors in planning. Such a model enables an agent to estimate the outcome of executing the behavior in a given state (what will the world look like after I open this door?). In this chapter I introduce and analyze a new model for these high level behaviors, and prove that this simpler alternative is still useful under mild assumptions. I provide empirical evidence that indicates the new predictive model can serve as a suitable substitute for its more complicated counterpart. Lastly, in Chapter 8, I examine the potential for abstract actions to improve the exploration process. I describe an algorithm developed by Jinnai et al. [145] that is based around the notion of constructing abstract actions that can easily reach all parts of the environment, and demonstrate that this algorithm can accelerate exploration on benchmark tasks.

PART 4. Finally, I turn to the joint process of **state-action abstraction**. In Chapter 9, I present a simple mechanism for combining state and action abstractions together. Using this scheme, I then prove which combinations of state and action abstraction can preserve representation of good behavioral policies in any finite MDP, summarized by Theorem 9.1. I next study the repeated application of these joint abstractions as a mechanism for constructing hierarchical abstractions. Under mild assumptions about the construction of the hierarchy and the underlying state-action abstractions, I prove that these hierarchies can *also* preserve representation of globally near-optimal behavioral policies, as stated in Theorem 9.3. I then conclude in Chapter 10 with reflections and directions forward.

Collectively, these results articulate a theory of abstraction in reinforcement learning. Figure 1.4 presents a visual overview of this dissertation.

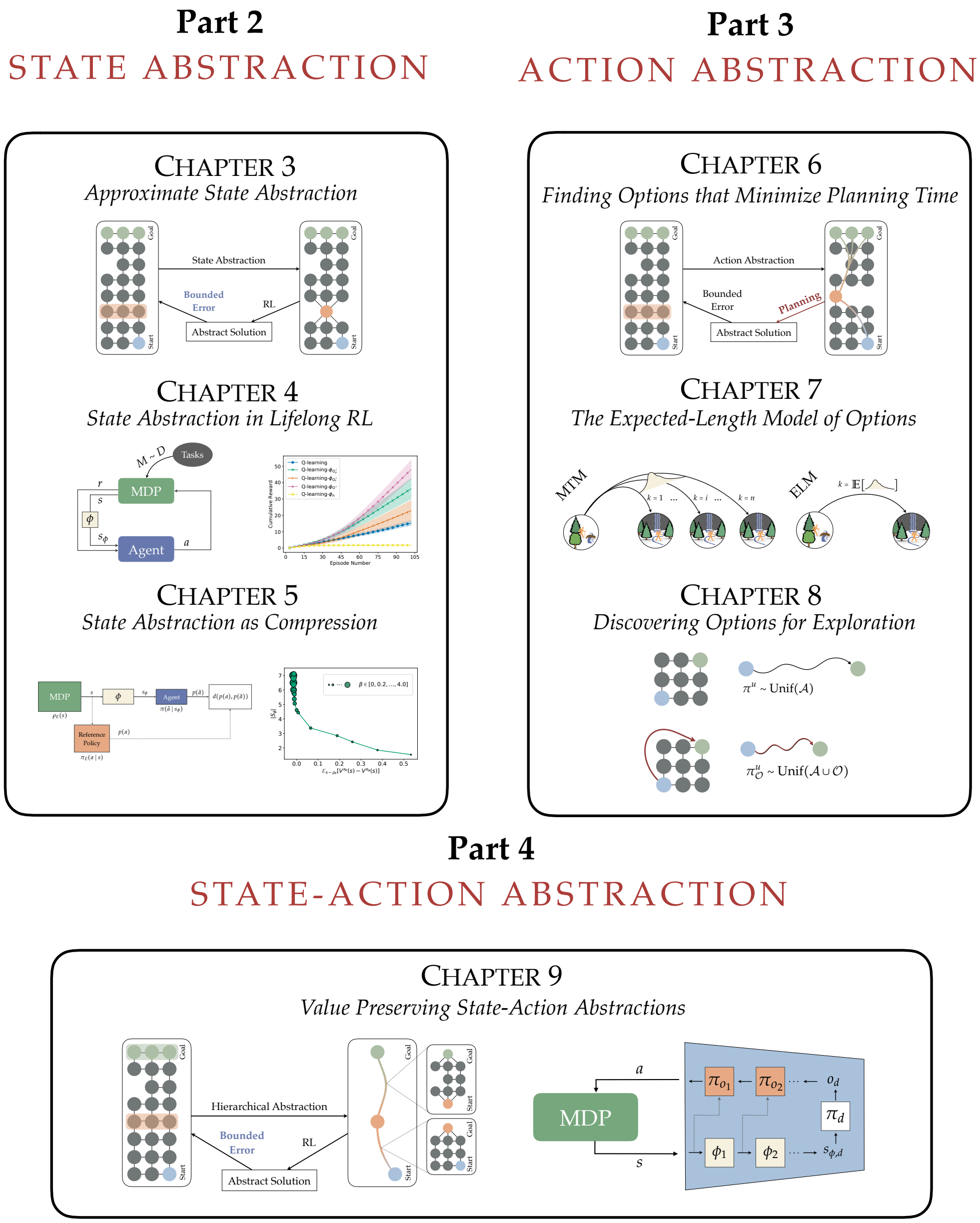


Figure 1.4: A visual overview of this dissertation.

I now turn to providing necessary background and notation on RL and abstraction. For those familiar with RL, I recommend skipping to Section 2.2.

2

BACKGROUND

Parts of this chapter are based on "Concepts in Bounded Rationality: Perspectives from Reinforcement Learning" [2] and "A Theory of State Abstraction for Reinforcement Learning" [3].

In this chapter, I bring clarity to the concepts of *agent*, *reward signal*, *world*, and *abstraction* by introducing the RL problem. In particular, I survey the key definitions and notation of the RL problem (Section 2.1) along with state (Section 2.2) and action abstraction (Section 2.3).

2.1 REINFORCEMENT LEARNING

There are many possible choices for formalizing the agent-environment interaction. How is time to proceed—continuously, or in discrete rounds? What is the space of observations? Are all worlds of interest necessarily *spatial* or filled with objects, at least in some capacity? With so many choices, it is not clear how to restrict attention to a suitable set of worlds. One natural response might be the space of *computable* worlds, or perhaps those with polynomial-time laws that transition the world from one state to the next. Indeed, it is challenging to identify a set of worlds that is both suitably general while remaining restricted enough to be useful.

In computational RL, the space of relevant environments are those that may be modeled as a discrete-time Markov Decision Process (MDP) [266]. At a high level, the space of MDPs defines worlds in which the next reward and the probability of arriving at the next state of the world can be fully predicted by the *current* world state (and perhaps, an agent's choice of action). Formally, an MDP is defined as follows.

Definition 2.1. *A discrete-time* ***Markov Decision Process*** *is a six tuple,* $(\mathcal{S}, \mathcal{A}, R, T, \gamma, \rho_0)$*, where:*

- $\mathcal{S}$*: A set of states describing the possible configurations of the world.*
- $\mathcal{A}$*: A set of actions describing the possible choices available to an agent.*
- $R : \mathcal{S} \times \mathcal{A} \times \mathcal{S} \to [\textsc{RMin}, \textsc{RMax}]$*: A reward function.*
- $T : \mathcal{S} \times \mathcal{A} \to \Delta(\mathcal{S})$*: A transition function denoting the probability of arriving in the next state of the world after an action is executed in the current state.*
- $\gamma \in [0, 1)$*: A discount factor, indicating an agent's preference between near-term and long-term rewards.*
- $\rho_0 \in \Delta(\mathcal{S})$*: The probability of starting in each state.*

The "Markov" in **M**DP indicates that the transition function, T, and reward function, R, both depend only on the current state of the world (and action), and *not* the full state history. That is,

$$T(s_{t+1} \mid s_t, a_t) = p(s_{t+1} \mid s_t, a_t) = p(s_{t+1} \mid s_0, a_0, \ldots, s_t, a_t), \tag{2.1}$$

$$R(s_t, a_t, s_{t+1}) = R(s_t, a_t, s_{t+1} \mid s_0, a_0, \ldots, s_t, a_t). \ldots, s_t, a_t, s_{t+1}). \tag{2.2}$$

Here, and throughout this dissertation, I use $p(x)$ as shorthand for a probability mass function $\mathbb{P}(X = x)$, where X is a discrete random variable taking on values $x \in \mathcal{X}$.

Equation 2.1 and Equation 2.2 state that there exist functions that fully characterize the next state distribution and next reward from the current state and action alone. This assumption is remarkably useful for simplifying analysis while still retaining appropriate generality. Moreover, if any environment is *not* Markov, it is typically feasible to roll the last $k \in \mathbb{N}$ steps of the world into a new memory-rich state representation, thereby yielding a Markov model. In this way, MDPs generalize Markov chains [55] and Markov reward processes [274] by allowing an agent to influence the state distribution $T(s' \mid s, a)$ and reward $R(s, a, s')$ according to the agent's choice of action.

There are a few things to note about the reward function. First, there are three natural ways it may be expressed: $R(s)$, $R(s, a)$, and $R(s, a, s')$. Naturally, the third form is the most general, fully subsuming the first two. For this reason I introduce reward functions in the most general form, but will occasionally use $R(s)$ or $R(s, a)$ for brevity. Note that either of these are just shorthand or cases where all actions or all next states have the same reward for the given state. Second, while I have defined R as a deterministic function, it can in general be a probability distribution with support $[\text{RM}\textsc{in}, \text{RM}\textsc{ax}]$. Throughout the dissertation I will tend to treat R as deterministic both in analysis and experiments unless otherwise noted. Lastly, I will sometimes assume the initial reward r_0 is sampled from some initial reward distribution $R(s_0)$ with the same support mentioned previously.

The central operation of RL is the repeated interaction between an agent and an MDP in discrete time steps. It is common to assume that the agent knows everything about the current *state* of world: the agent has no uncertainty regarding which state it occupies, only what the reward and transition functions are. A more general formalism *also* models hidden information, called the Partially Observable MDP (POMDP) [154]. In both POMDPs and MDPs, the agent interacts indefinitely with its environment with the goal of learning how to take actions that maximize long-term discounted reward. Throughout this dissertation, I make the standard assumption that the environment can be accurately modeled by an MDP, rather than a POMDP. Other work has considered a more general variant of the RL problem in non-Markovian settings [280, 134, 331, 197]. I focus only on agents learning in Markovian environments, though note that there is interesting and important work to be done in clarifying the role of abstraction in these general settings. I will often restrict attention only to finite MDPs, too, in which the state and action space are assumed to be finite.

From a methodological perspective, MDPs occupy an appropriate middle ground between simplicity and generality. I take it to be of fundamental importance to address prominent open questions in the context of simple formalisms for which those questions still remain open. By providing principled answers in these restricted settings, we can systematically build up our understanding and guide future research into richer settings rooted in first principles.

Under the assumption that RL agents will interact with an MDP, the RL problem can be stated more precisely as follows.

Definition 2.2. *The **RL problem** is formalized as follows. An RL agent interacts with an MDP $M = (\mathcal{S}, \mathcal{A}, R, T, \gamma, \rho_0)$ by repeating the following four steps, letting $t = 0$:*

1. *The agent receives a state $s_t \in \mathcal{S}$ and a reward $r_t \in \mathbb{R}$ from M.*
2. *The agent learns from this interaction and outputs an action, $a_t \in \mathcal{A}$.*
3. *The MDP outputs the next state, $s_{t+1} \sim T(s_{t+1} \mid s_t, a_t)$, and reward $r_{t+1} = R(s_t, a_t, s_{t+1})$.*
4. *Increment t.*

The goal of an RL agent interacting with an MDP is to make decisions that maximize long term discounted reward:

$$\sum_{t=0}^{\infty} r_t \gamma^t. \tag{2.3}$$

The standard objective of an RL agent is to solve for behavior that will prescribe what to do from *any* state the agent might occupy. Note, though, that this is a stronger notion than what is strictly necessary. If the agent starts in state $s_0 \sim \rho_0$, then there may be some states of the environment that are difficult to reach. In this sense, it might be more effective to focus attention on those states that are likely to be visited during the agent's lifetime. This insight will emerge shortly when we discuss the quality of an agent's decision.

We ground this notion of behavior in all states in the MDP with a policy:

Definition 2.3. *A* ***policy****, $\pi : \mathcal{S} \rightarrow \Delta(\mathcal{A})$, is a prescription for behavior for any state in the given MDP.*

Note that in the general case a policy can be stochastic. To make a decision, then, the agent samples $a \sim \pi(\cdot \mid s)$. Given that deterministic policies are often of interest, I will also use $\pi(s)$ to denote a deterministic policy.

To characterize the notion of expected long term expected discounted reward, we next introduce the value (V) and action-value (Q) functions.

Definition 2.4. *The* ***value function*** *$V^\pi : \mathcal{S} \rightarrow \mathbb{R}$, under a policy π of a state $s \in \mathcal{S}$ is denoted*

$$V^\pi(s) := \sum_{a \in \mathcal{A}} \pi(a \mid s) \sum_{s' \in \mathcal{S}} R(s,a,s') + \gamma T(s' \mid s,a) V^\pi(s'). \tag{2.4}$$

Definition 2.5. *The* ***action-value function****, $Q^\pi : \mathcal{S} \times \mathcal{A} \rightarrow \mathbb{R}$, under a policy π of a state $s \in \mathcal{S}$ and action $a \in \mathcal{A}$ is denoted*

$$Q^\pi(s,a) := \sum_{s' \in \mathcal{S}} R(s,a,s') + \gamma T(s' \mid s,a) V^\pi(s'). \tag{2.5}$$

I denote the value (V) and action-value (Q) functions under the optimal policy as V^* and Q^* respectively, which are determined by applying the max operator to the Bellman Equation [41]:

$$V^*(s) := \max_a \left(\sum_{s'} R(s,a,s') + \gamma T(s' \mid s,a) V^*(s') \right). \tag{2.6}$$

Since the support of R is the real valued interval $[\textsc{RMin}, \textsc{RMax}]$, I will denote

$$\textsc{QMax} = \textsc{VMax} \leq \frac{\textsc{RMax}}{1-\gamma}, \tag{2.7}$$

$$\textsc{QMin} = \textsc{VMin} \geq \frac{\textsc{RMin}}{1-\gamma}, \tag{2.8}$$

as upper and lower bounds on the value achievable in a discounted, infinite horizon RL problem. That is, for any state s in any MDP,

$$\textsc{VMin} \leq V^*(s) \leq \textsc{VMax}. \tag{2.9}$$

In general, the value of a policy will serve as our primary method for evaluating an agent's behavior, and in many cases, for determining learning progress. Recently, Bellemare et al. [40] propose an extension to the classical Bellman Equation that translates the expected future returns into a distribution over future returns. Later work has developed RL algorithms that learn relative to this distributional objective to great effect [80, 79, 130], and has given rise to new explanatory models of the role dopamine neurons play in reward error prediction [81]. While I do not attend to these directions, there is interesting work to be done in combining the ideas of abstraction and distributional RL.

EXAMPLE. Let us now consider an example: the Russell and Norvig grid world, used by the classic AI textbook [277]. The Russell and Norvig grid world is a discrete, 4×3 two-dimensional grid in which each state corresponds to the agent inhabiting one of the eleven empty grid cells (Figure 2.1). For the purpose of clarity, I adopt a factored representation for states. Specifically, each state will be defined as (x,y), for $x \in \mathbb{N}_{[1:4]}$, $y \in \mathbb{N}_{[1:3]}$. Here, $(1,1)$ denotes the state in the bottom left corner with $x = 1$ and $y = 1$, with x increasing as the agent moves to the right and y increasing as the agent moves up. Naturally, an enumerated state space representation could be adopted, too, according to which the states are represented by a single number (and thus not imposing any notion of "space" onto the problem). The grid world MDP is then defined as follows.

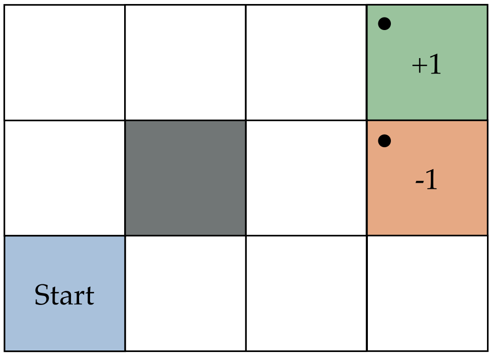


Figure 2.1: The classical grid world by Russell and Norvig [277].

1. $\mathcal{S} = \{(1,1),(2,1),(3,1),(4,1),(1,2),(3,2),(4,2),(1,3),(2,3),(3,3),(4,3)\}$,

2. $\mathcal{A} = \{\uparrow,\ \rightarrow,\ \downarrow,\ \leftarrow\}$,

3. $R(s,a,s') = \begin{cases} 1 & s' = (4,3), \\ -1 & s' = (4,2), \\ 0 & \text{otherwise}, \end{cases}$

4. $T(s' \mid s,a) = \mathbb{1}\{s' = \texttt{grid\_move}(s,a)\}$ where,

$$\texttt{grid\_move}(s,a) = \begin{cases} (s.x, \min(s.y+1,3)) & a = \uparrow \text{ and } s.x \neq 2, \\ (\min(s.x+1,4), s,y) & a = \rightarrow \text{ and } s \neq (1,2), \\ (s.x, \max(s.y-1,1)) & a = \downarrow \text{ and } s.x \neq 2, \\ (\max(s.x-1,1), s,y) & a = \leftarrow \text{ and } s \neq (3,2), \\ s & \text{otherwise}, \end{cases}$$

5. $\rho_0(s) = \mathbb{1}\{s = (1,1)\}$,

6. $\gamma = 0.99$.

That is, the reward function outputs zero for every transition unless the agent enters state (4,3) or (4,2), in which case it receives +1 and -1 respectively. The four actions move the agent in each cardinal direction with the exception of moving into the wall or edge of the environment, which yields no effect. Lastly, when the agent arrives in either (4,3) or (4,2), the episode ends and the agent moves back to s_0 to start the next episode.

From the perspective of the learning agent, it does not know that this grid world has a Cartesian coordinate system, and that the action associated with the symbol "→" will typically increase its x coordinate. Instead, the agent must repeatedly experiment with the execution of different behaviors. With the initial state distribution ρ_0 only assigning mass to $(1,1)$, the agent is first presented with choosing an action in the lower left state. Without prior information, each state is equally informative, and so the agent might choose ←. Upon execution of this action, the MDP samples $s_1 \sim T(\cdot \mid s_0 = (1,1), a_0 = \leftarrow)$, and $r_1 = R(s_0 = (1,1), a_0 = \leftarrow, s_0' = (1,1))$. Now, after this single action execution, the agent occupies state $(1,1)$, since the ← action moved the agent into the wall. After receiving the first bit of signal from the environment the agent has the opportunity to learn something. What effect did applying the "←" action in $s = (1,1)$ have? How much reward was received? As more data is gathered, agents will be better positioned to give high confidence answers to these questions for different (s,a) pairs throughout the MDP.

This process continues indefinitely: the agent receives a state and reward pair from the environment (s_t, r_t), and chooses an action a_t. The MDP then transitions to the next state and generates the next reward. This process is pictured in Figure 2.2.

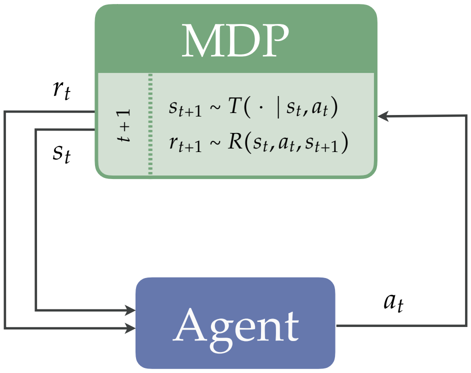

MDP
$t+1$
$s_{t+1} \sim T(\cdot \mid s_t, a_t)$
$r_{t+1} \sim R(s_t, a_t, s_{t+1})$
r_t
s_t
a_t
Agent

Figure 2.2: The RL problem when an agent interacts with an MDP.

It is often useful to allow the agent to periodically *reset* by resampling $s_0 \sim \rho_0$. Such a system typically fixes a finite horizon $H \in \mathbb{N}$ and allows the agent to execute up to H actions before resetting to a start state $s_0 \sim \rho_0$. This setting is referred to as *episodic* RL, with each episode consisting of at most H steps, as is the case in the grid world above. In some cases, arriving at special goal or trap states (such as (4,3) and (4,2) in the present example) can also have the effect of resetting the agent. All of this is in contrast to continual learning in which the agent repeatedly interacts with its environment and is never allowed to reset.

2.1.1 *Canonical RL Algorithms*

What, then, does it look like to *solve* the RL problem? A typical solution comes in the form of a learning algorithm that captures a particular strategy for mapping a history of experiences, $(r_0, s_0, a_0, r_1, s_1, a_1, \ldots)$ to an action. In this way, the space of RL algorithms is roughly the space of all functions that map arbitrary length histories of experience to a choice of action. Good algorithms are those whose action selection becomes better with time, as measured by the sum of discounted rewards received.

RL algorithms can be divided into three broad categories: policy-based, model-free, and model-based. Each category is an answer to the question: "which functions are being estimated during learning?". If an RL algorithm maintains estimates of the transition and reward functions, $\hat{T}$ and $\hat{R}$, then it is said to be model-based. If the reward and transition functions are not estimated, but the action-value function Q is, then the algorithm is model-free. Lastly, if all that is estimated is the policy directly, then it is policy-based. These are relatively loose boundaries, however, as many algorithms often compute partial solutions to different functions or carry out implicit computational work that resembles construction of one of these functions [328]. A rough division between these three approaches is pictured in Figure 2.3.

In model-based RL, the transition and reward functions are typically estimated explicitly. Then, using these estimates $\hat{T}$ and $\hat{R}$, the agent often constructs a simulated MDP,

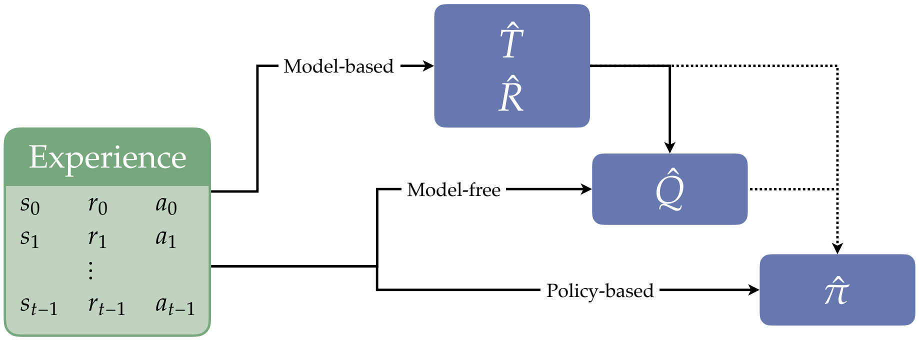


Figure 2.3: The different families of RL algorithms.

$\hat{M}$, which can be used to do explicit search for good behavior, or to evaluate different policies. That is, given simulation access to an MDP $\hat{M}$ that is sufficiently similar to the environmental MDP M, the agent can perform computations on $\hat{M}$ to construct $\hat{\pi}^*$ or perhaps $\hat{Q}^*$, which can induce a policy by choosing the action with highest value.

In model-free and policy-based RL, the agent typically maintains an estimate of the action-value function Q or a policy π directly. Various mechanisms are adjusted in order to learn these functions faster by assigning credit more efficiently, generalizing more robustly, or exploring more elegantly.

There are good arguments to adopt each style of algorithm depending on the context. For instance, recent work by Sun et al. [303] illustrates a gap in the efficiency between model-based and model-free approaches under a particular method of dividing the two families. On the other hand, it has proven difficult to estimate accurate models. For instance, even nearly-accurate one-step models are known to lead to an exponential increase in the error of n-step predictions as a function of the horizon [157, 54], though recent approaches show how to diminish this error through smoothness assumptions [23]. Moreover, composing an accurate one-step model into an n-step model is known to give rise to predictions of states dissimilar to those seen during training of the model, leading to poor generalization [314]. Notably, model-free and policy-based methods have enjoyed a great deal of success when combined with deep neural networks, giving rise to so called "deep RL" methods that learn effectively in a variety of challenging domains from Atari

[238] to robotics [199]. For more discussion on the relationship and interplay of these two families, see work by Asadi [21]. To build further intuition, I now introduce two classical RL algorithms, one model-free and one model-based.

model-based: r-max. One of the earliest successful model-based algorithms was that of *Explicit Explore-Exploit* or E^3, developed by Kearns and Singh [157]. This pioneering work established early theoretical guarantees for the efficiency E^3, which has since paved the way for a long and fruitful sequence of new and improved algorithms. Around the same time, Brafman and Tennenholtz [54] introduced the algorithm R-Max, which uses a similar exploration strategy to E^3, and will be used in analysis and empirical study throughout this dissertation.

R-Max makes decisions by exploring so as to seek out every opportunity for high-performing behavior. That is, R-Max initially supposes it inhabits the maximally rewarding MDP: all rewards are believed to be RMax, the maximal possible reward, and all transitions are assumed to be self-loops. Then, the algorithm acts according to the optimal policy in this optimistic MDP to explore and collect experience. Consequently, the algorithm will effectively try *new* actions that it does not yet know about. R-Max uses all

Algorithm 2.1 R-Max

```
Input: $\varepsilon$, $\delta$, $\gamma$, $s_0$

$m = f(\varepsilon, \delta)$
initialize $\hat{T}, \hat{R}, n$
$\hat{Q}$ = solve_mdp($\hat{R}, \hat{T}, \gamma$)
$t = 0$
while True do
    $a_t = \arg\max_b \hat{Q}(s_t, b)$
    $r_t, s_{t+1}$ = act($s_t, a_t$)
    $n(s_t, a_t) = n(s_t, a_t) + 1$
    if $n(s_t, a_t) = m$ then
        $\hat{Q}, \hat{R}, \hat{T}$ = update_model()
    end if
    $t = t + 1$
end while
```

collected experience to inform empirical estimates of the reward and transition function for each (s, a) pair it encounters. Once enough data is collected for a particular (s, a) pair, the empirical estimate of the reward and transition replace the optimistic estimate, and the behavioral policy is recomputed based on this mixture of known and optimistic reward and transitions. Simplified pseudocode for R-Max is presented in Algorithm 2.1, where $\texttt{act}(s_t, a_t)$ defines a single interaction with the MDP. For more details on R-Max, see Brafman and Tennenholtz [54] or Chapter 2.1 of Strehl et al. [301].

MODEL-FREE: Q-LEARNING. The second and perhaps most canonical RL algorithm is called Q-learning, first introduced by Watkins and Dayan [337]. Q-learning maintains an estimate of the Q function for each state-action pair, and proceeds on the basis of performing one simple update to this Q function estimate based on the last experience, $(s_{t-1}, a_{t-1}, r_{t-1}, s_t)$, and a learning rate $\alpha \in [0, 1]$. That is, we first initialize a Q-function according to some protocol, such as choosing Q values uniformly at random from the interval $[\text{QMin}, \text{QMax}]$. Or, more commonly, the initial Q function is set to be either 0, or is set optimistically where $\hat{Q}_0(s, a) = \text{QMax}$ for all s, a. Then, actions are chosen according to the *greedy policy*, defined as follows.

Definition 2.6. *The* ***greedy policy*** *with respect to some Q function is:*

$$\pi_Q(a \mid s) = \mathbb{1}\left\{a = \arg\max_{a'} Q(s, a')\right\}, \tag{2.10}$$

where I assume that $\arg\max$ *returns a single entity, breaking ties consistently according to any fixed method.*

Note that if the given Q is in fact Q^*, then π_{Q^*} will be optimal. However, prior to learning Q^*, acting greedily will not sufficiently explore the environment as the agent may over commit to locally promising but globally poor solutions. Indeed, the main guarantee provided about Q-learning is that its asymptotic performance is optimal subject to the assumption that every state-action pair is experienced infinitely often in the limit (among other assumptions) [337]. Clearly, for many choices of initialization $\hat{Q}_0$, the greedy policy may force Q-learning to only experience a small subset of the state-action space, thus violating the conditions necessary to ensure convergence to optimal behavior.

The most common method of overcoming this difficulty is to pair Q-learning with an ε-greedy policy, which chooses an action uniformly at random with some small $\varepsilon \in [0,1]$ probability, defined as follows.

Definition 2.7. *The* ***ε-greedy policy****, for* $\varepsilon \in [0,1]$*, with respect to some* Q *function is:*

$$\pi_{Q,\varepsilon}(a \mid s) = \begin{cases} 1-\varepsilon & a = \arg\max_{a'} Q(s,a'), \\ \frac{\varepsilon}{|\mathcal{A}|-1} & \textit{otherwise}. \end{cases} \tag{2.11}$$

Using an ε-greedy policy (or another choice of stochastic policy, such as a softmax [22]) Q-learning makes decisions that are greedy with respect to its current estimate of the Q function and slowly updates this estimate to be more accurate over time. Specifically, when an experience $(s_{t-1}, a_{t-1}, r_{t-1}, s_t)$ takes place, the following update is applied:

$$\hat{Q}_t(s_{t-1}, a_{t-1}) = (1-\alpha)\hat{Q}_{t-1}(s_{t-1}, a_{t-1}) + \alpha(r_{t-1} + \max_{a' \in \mathcal{A}} \hat{Q}_{t-1}(s_t, a')). \tag{2.12}$$

The full pseudocode of Q-learning is presented in Algorithm 2.2. For more detail on Q-learning, see the original work by Watkins and Dayan [337].

Algorithm 2.2 Q-Learning

INPUT: α, γ, s_0, Q_0, ε

```
$t = 0$
while True do
    $a_t \sim \pi_{Q_t,\varepsilon}(\cdot \mid s_t)$
    $r_t, s_{t+1} = \texttt{act}(s_t, a_t)$
    $Q_{t+1}(s_t, a_t) = (1-\alpha)Q_t(s_t, a_t) + \alpha(r_t + \max_{a' \in \mathcal{A}} Q_t(s_{t+1}, a'))$
    $t = t + 1$
end while
```

I will shortly contrast how these two algorithms behave in the grid world described earlier in the section. First, we must attend to the broader question: what does it look like to compare different RL algorithms?

2.1.2 *Evaluation in RL*

As with other areas of machine learning, there are two broad approaches for evaluating and understanding an RL algorithm. First, theoretical guarantees may be established about an algorithm, often presented in the form of convergence to desirable fixed points, a bound on the sample complexity of exploration [155], the algorithm's (Bayesian or frequentist) regret [28], or a KWIK bound on the algorithm [204]. Second, empirical investigations of hypotheses relating to algorithms and their properties, or analysis on explanatory benchmark tasks. These may include visuals of learned policies, value functions, or representations, or, most commonly, learning curves illustrating the agents learning process.

In this section I provide a brief overview of these two approaches to evaluation. The focus of this dissertation is on algorithms (and the abstractions they use) that can reliably and quickly find near-optimal behavioral policies in any MDP. Naturally, there is much more to the RL problem that I can not cover here. In particular, we might also care about robustness, explainability, how an algorithm handles failures, safety, generalization, and many other properties of interest.

sample complexity of exploration. After conducting an experiment that involves a learning algorithm interacting with a chosen MDP, what is useful to have learned about the algorithm? There are many possible answers. The most pressing is typically related to the sample efficiency of the algorithm; how many experiences are needed until the algorithm will achieve a satisfactory level of performance? This notion is grounded in several different measures in RL, with the first being the sample complexity of exploration introduced by Kakade [155], based in part by the analysis done by Kearns and Singh [157] and Brafman and Tennenholtz [54].

Definition 2.8. *Let $\varepsilon \in \mathbb{R}_{\geq 0}$ denote accuracy and $\delta \in \mathbb{R}_{\geq 0}$ denote an allowed failure probability. The expression,*

$$\zeta\left(\frac{1}{\varepsilon}, \frac{1}{\delta}, |\mathcal{S}|, |\mathcal{A}|, \frac{1}{1-\gamma}, \text{RMax}\right) \tag{2.13}$$

is a ***sample complexity*** *bound for a learning algorithm $\mathscr{A}$ if the following holds. For any finite MDP M, RMax > 0: let $\mathscr{A}$ interact with M, starting in $s_0 \sim \rho_0$, resulting in the process $s_0, r_0, a_0, s_1, \ldots$, Then, with probability at least $1-\delta$, the number of time steps such that $V^{\mathscr{A}}(s_t) < V^*(s_t) - \varepsilon$, is at most $\zeta(\frac{1}{\varepsilon}, \frac{1}{\delta}, |\mathcal{S}|, |\mathcal{A}|, \frac{1}{1-\gamma}, \text{RMax})$.*

The sample complexity captures how many mistakes we expect an algorithm to make in any finite MDP, if left to run indefinitely. That is, a sample complexity of $|\mathcal{S}|$ will tell us that the agent might make one mistake per state in the MDP. The sample complexity is intended to clarify an RL algorithm's effectiveness for exploring its environment while also learning to make good decisions in that environment. It determines, for a given ε and δ, how many mistakes the agent is expected to make before acting in a near-optimal way. The Probably Approximately Correct in Markov Decision Processes (PAC-MDP) [301] criterion expresses a desirable guarantee about an algorithm's sample complexity, inspired by the seminal work establishing the learnability of concepts in supervised learning introduced by Valiant [324]. PAC-MDP algorithms are those that achieve a polynomial sample complexity *and* computational complexity with respect to ε, δ and the MDP parameters. For an early survey of PAC-MDP approaches, see work by Strehl et al. [301].

REGRET. The regret compares an agent's total expected accumulated reward to that of the optimal policy from the time of the agent's first action execution.

Definition 2.9. *Let $\mu^*(M)$ denote the average reward of the optimal policy in MDP M:*

$$\mu^*(M) = \mu^*(M, s) = \max_{\pi \in \Pi}(M, \pi, s). \tag{2.14}$$

Further let $G(M, \mathscr{A}, s_0, H)$ denote the total accumulated reward by the agent after H steps in MDP M with start state s_0:

$$G(M, \mathscr{A}, s_0, H) = \sum_{t=0}^{H-1} r_t \tag{2.15}$$

*Then the following is the **regret** of an algorithm, $\mathscr{A}$, with finite horizon H, on MDP M.*

$$Regret(M, \mathscr{A}, s_0, H) := H\mu^*(M) - G(M, \mathscr{A}, s_0, H). \tag{2.16}$$

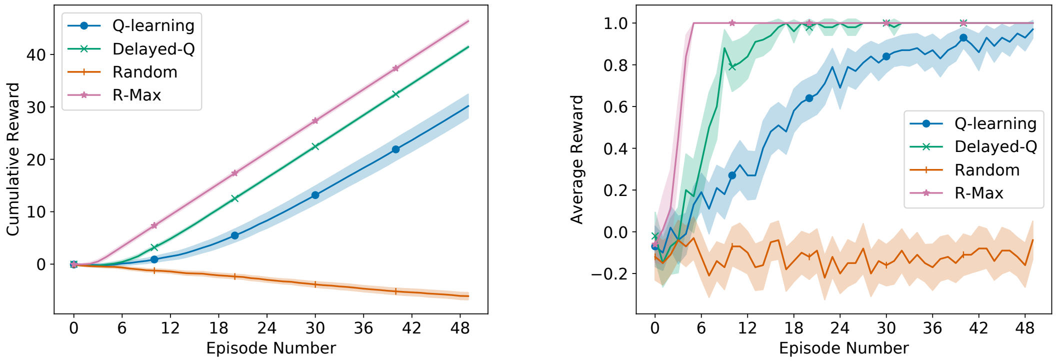
Regret differs from the sample complexity of exploration in several critical ways. First, the magnitude of each mistake the agent makes matters. While sample complexity counts the number of mistakes, the size of the mistake made will contribute to an agent's overall regret. Second, in measuring regret, the agent's long term behavior must approach the global optimum, and not a locally optimal policy for the region of the state space the agent is in. Regret is also commonly presented in two slightly different forms: *Bayesian* regret, in which the agent is compared to optimal behavior relative to its prior, and *Frequentist* regret, in which the agent is compared to the true optimal behavior. For more on these two measures and their relationship, see work by Dann et al. [82].

KWIK. Li et al. [204] introduced the Knows What It Knows (KWIK) criterion, which captures prominent elements of the PAC objective, but also incorporates an adversarial element. In KWIK, we suppose there exists an input set $\mathcal{X}$ and output set, $\mathcal{Y}$. A given hypothesis class, $\mathcal{H}$, contains a subset of possible functions from $\mathcal{X}$ to $\mathcal{Y}$. The agent's goal is to learn some target function, $h^* \in \mathcal{H}$ during the following repeated process:

- The agent and an adversary are given $\varepsilon \in \mathbb{R}_{\geq 0}$, $\delta \in \mathbb{R}_{\geq 0}$, and $\mathcal{H}$.
- The adversary selects the target function $h^* \in \mathcal{H}$.
- Repeat:
  - The adversary selects $x \in \mathcal{X}$ and gives it to the learner.
  - The learner predicts an output, $\hat{y} \in \mathcal{Y} \cup \perp$.
  - If $\hat{y} \neq \perp$, then it must be accurate: $|\hat{y} - h^*(x)| \leq \varepsilon$, otherwise, the run is a failure.
  - If $\hat{y} = \perp$, then the learner observes $z \in \mathcal{Z}$ of the output, where $z = y$ in the deterministic case, and has noise determined by nature in the stochastic case.
- The probability of a failed run must be bounded by δ.
- Over the course of a run, the total number of steps on which $\hat{y} = \perp$ must be bounded by $B(\varepsilon, \delta)$.

KWIK has been used as a further evaluative criteria in the RL to bound the number of experiences needed for an agent to accurately learn the model, T and R [93, 334, 335, 336]. KWIK typically deals with learning the transition model or rewards directly, and so is naturally suited for model-based RL. It provides stronger guarantees than sample complexity, since both samples and the target function are chosen in an adversarial way.

EMPIRICAL EVALUATION IN RL. Empirical evaluation in RL is naturally diverse. The most standard experiments assess aspects of the sample efficiency of a given RL algorithm. A typical experiment proceeds as follows. Choose an MDP, M, and a collection of learning algorithms, $\mathscr{A}_1, \ldots, \mathscr{A}_n$. Allow each of the n algorithms to interact with the MDP for some number of steps, H. Then, compare the total cumulative reward received by each algorithm. In this way, we test the relationship between the amount of experience the algorithm has and its overall cumulative reward.



(a) Cumulative

(b) Average

Figure 2.4: Example learning curves showing cumulative (left) and average (right) reward for a variety of RL algorithms.

The results from such experiments are often represented as learning curves where the X-axis denotes experience and the Y-axis is some measure of performance on the task, either total reward accumulated or average reward accumulated per time step.

Example learning curves for each of the two algorithms discussed previously (R-Max and Q-learning) acting in the Russell & Norvig grid world from Figure 2.1 are presented in Figure 2.4. I additionally include two other approaches for further contrast. First, a random actor (in orange) that does no learning, but simply always samples from the uniform distribution over actions. Second, a more sophisticated version of Q-learning called Delayed-Q (green) [300], that enjoys similar theoretical guarantees to R-Max (both are PAC-MDP).

The two curves present the same results from a slightly different perspective. On the left, I show that mean cumulative reward over episodes, presented with 95% confidence intervals from 100 runs of the experiment. On the right, I show the average reward per episode, again with 95% confidence intervals (from the same 100 runs).

These plots give us insight into how the different learning algorithms perform on this grid world MDP. First note that on the first handful of episodes, all of the algorithms perform nearly identically, on average. This is because nearly any algorithm designed to learn on any MDP will require some number of samples before finding a reasonable behavioral policy. If an algorithm were to behave well from the outset, it is likely overfit to the task of interest and is susceptible to learn slowly in other MDPs, by No Free Lunch [346]. Second, note that R-Max and Delayed-Q ultimately find the optimal policy in the sample budget allotted. From the definition of the problem, recall that the maximum reward receivable in any given episode is one. Therefore, note that the average plot indicates that both R-Max and Delayed-Q are able to achieve one reward per episode after the first ten or so episodes—this indicates that they have both found near-optimal policies, and in a relatively sample efficient manner. In contrast, the random approach unsurprisingly never finds a reasonable policy. Since the -1 square in the grid is easier to reach on average than the $+1$ square, the random approach tends to lose reward over time. Lastly, we find that Q-learning ultimately does find a reasonable policy, but it requires more samples in order to produce high reward behavior.

These are roughly the questions we ask in running such an experiment: 1) What is the initial performance like? 2) How long does it take the agent to converge to its eventual best policy? 3) What is the value of the best policy the algorithm discovers? Learning curves like the ones presented can help answer each of these questions, thereby giving insight into the performance of the algorithm on the MDP of interest. Naturally, other questions are of interest as well, such as how well an algorithm can perform when given a new task, or its robustness to different hyper-parameter settings.

Beyond learning curves, there are many other facets of an RL algorithm that determine its effectiveness. Naturally this space is far too vast to summarize, but one important focus is on how well algorithms address each of the *subproblems* of RL discussed briefly in the previous chapter. That is, in confronting the RL problem, algorithms must by necessity tackle several subproblems of deep interest to the broader machine learning and AI communities, including generalization, the explore-exploit dilemma, credit assignment, and planning. I next provide detail on one subproblem of special interest, planning. For more on the explore-exploit dilemma, see the recent book on bandits by Lattimore and

Szepesvári [189]. For more on credit assignment, see the recent work by Harutyunyan et al. [127]. Lastly, for more on generalization as understood in supervised learning, see the classic works by Valiant [324], Vapnik and Chervonenkis [330], or the book by Shalev-Shwartz and Ben-David [283].

PLANNING. A computational practice critical to RL is *planning*. The key difference between RL and planning is that the full model of the environment is given as input in planning, so there is no uncertainty around T or R. Planning was originally studied early on in AI by Bellman [41] and Newell et al. [248] among others. The planning problem is commonly formalized in the context of a language (such as STRIPS [105]) that provides a high level scheme for expressing the consequences of decisions available to an agent in a given domain. Under the assumption that environments of interest may be modeled as MDPs, there is a decision-theoretic version of the planning problem tightly connected to RL.

Definition 2.10. *The **MDP planning problem** is defined as follows:* `given` *an MDP M as input,* `return` *a sequence of actions that achieves maximal expected discounted reward when executed in M,*

$$\sum_{t=1}^{\infty} r_t \gamma^t. \tag{2.17}$$

In the above version of the problem the required solution is a sequence of actions. Of course, other possible solutions may be of interest as well, such as a policy or value function, though naturally these tend to be harder to compute.

Depending on the constraints placed on the problem, the general (propositional) planning problem is known to be PSPACE-Complete [61, 62], but solving for optimal behavior in an MDP is known to be P-Complete in the size of the environment [257, 209]. Many

Algorithm 2.3 Value Iteration

INPUT: M, Δ, V_0

```
while True do
    for $s \in \mathcal{S}$ do
        $V_{t+1}(s) = \max_{a \in \mathcal{A}} \sum_{s' \in \mathcal{S}} R(s,a,s') + \gamma T(s' \mid s,a) V_t(s')$
    end for
    if $\max_{s \in \mathcal{S}} |V_{t+1}(s) - V_t(s)| \leq \Delta$ then
        return $V_{t+1}$
    end if
    $t = t + 1$
end while
```

problem representations are known to grow super-polynomialy with the number of variables that characterize the domain and robust action spaces are often large or even continuous, making many planning problems on the scale of the real world computationally intractable.

The standard algorithm in MDP planning is a dynamic programming algorithm called Value Iteration (VI), first introduced by Bellman [42]. As per the name, the idea of VI is to repeatedly propagate value to adjacent states, starting from some arbitrary initialized value function V_0 and terminating when the optimal value function is realized. Pseudocode for VI is presented in Algorithm 2.3.

This dissertation is ultimately about RL, rather than planning. Hence, VI will be treated a general purpose tool for planning in finite MDPs that can be called as a subprocedure by RL algorithms. In the context of model-based RL, such subprocedures are often called to produce a policy $\pi^*_{\hat{M}}$ that is optimal in the simulated MDP $\hat{M}$. Abstraction, as we will see, is particularly effective for accelerating planning, since planning with a well structured state-action space can be dramatically faster.

For further background on Markov Decision Process, see the text by Puterman [266], and for more background on reinforcement learning, see texts by Kaelbling et al. [153], Bertsekas and Tsitsiklis [46] and Sutton and Barto [309, 310].

2.2 STATE ABSTRACTION

I next introduce the formalisms for state abstraction in RL, followed by a survey of prior research in the area.

In an MDP, a state fully describes the current configuration of the environment down to the last detail. In a finite, discrete-time MDP, the default state representation is the set containing the states $s_1, s_2, \ldots, s_{|\mathcal{S}|}$. The actions, $\mathcal{A}$, change the state of the MDP according to the transition dynamics defined by T. However, this view on states is quite limiting, as the state representation lacks structure. For instance, we may instead suppose that two states can be deemed as similar or dissimilar to one another (standing near the bridge and on the bridge in the forest, for instance). It is precisely these similarities that underlie ontologies supporting *objects*, *properties*, *relations*, and *universals*. To facilitate this more general notion of *state*, an MDP state is sometimes defined as a vector of variables or features, formalized as a Factored MDP [167, 123], similar to the typical supervised and unsupervised learning settings. Other types of MDPs have been introduced that leverage some kind of implicit ontological structure, such as Relational MDPs [159, 113], and Object-Oriented MDPs [93], which explicitly carve the world into objects, their classes, and functions thereof.

Regardless of the mechanism for representing states, the goal of state abstraction is to reduce the size or complexity of the state space by grouping together similar states in a way that doesn't change anything important about the underlying problem being solved. Concretely, a state abstraction is defined as follows.

Definition 2.11. *A* ***state abstraction*** *is a function, $\phi : \mathcal{S} \rightarrow \mathcal{S}_\phi$, that maps each true environmental state $s \in \mathcal{S}$ into an abstract state $s_\phi \in \mathcal{S}_\phi$.*

The abstract state corresponds to the agent's *representation* of the current configuration of the environment; it is often not a perfect characterization in that the *abstracted* state may throw away some information. In some cases, the underlying state space $\mathcal{S}$ may be continuous, and the abstracted space, discrete. For instance, we might let $\mathcal{S} = \mathbb{R}_{\geq 0}$, with $\mathcal{S}_\phi = \mathbb{N}$ induced by the abstraction function $\phi(s) = \lceil s \rceil$. The function ϕ may also just reduce a finite space $\mathcal{S}$ to a smaller one $\mathcal{S}_\phi$. In this sense, state representations that sharpen sensory observations into features or objects are carrying out a particular form of abstraction.

Determining what information to throw away is the central question behind the theory of state abstraction: how do effective agents come up with an appropriate abstract understanding of the environments they inhabit? I choose to systematize this question through the introduced state abstraction functions, sometimes called state *aggregation* functions.

Why study such a broad question using such a specific and simple formalism? As with our choice of MDPs as the model of the environment, I take it to be important to attend to our question's simplest unanswered form. That way, any new results build a foundation upon which subsequent inquiry can take place. There are many other functions of interest that change the state representation, and it is important to understand each of them. These simple aggregation functions, ϕ, are perhaps the simplest function that allow analysis and study, and for which new insights can bring clarity to the process of state abstraction in RL more generally. A natural direction for future work will investigate the more general classes of state abstraction that are expressive enough to include features, objects, and their kin.

In general, an RL agent makes use of a state abstraction function as follows. Each time the MDP produces a state s_t, it is first passed through ϕ yielding the abstracted state $s_{\phi,t}$. Then, the agent takes $s_{\phi,t}$ as input, learns, and outputs an action a_t. This process is pictured in Figure 2.5. In this way, the agent never needs to know or confront the environmental state space. Moreover, this division between ϕ and the agent allows for the study of ϕ in a way that is agnostic to choice of RL algorithm.

It is also possible to define a new abstract MDP that is tightly connected to the original MDP. I refer to this new MDP as the *abstract* MDP, M_ϕ, drawing from the rich history of abstraction in MDPs [165, 87]. The abstract MDP is defined by three components: the

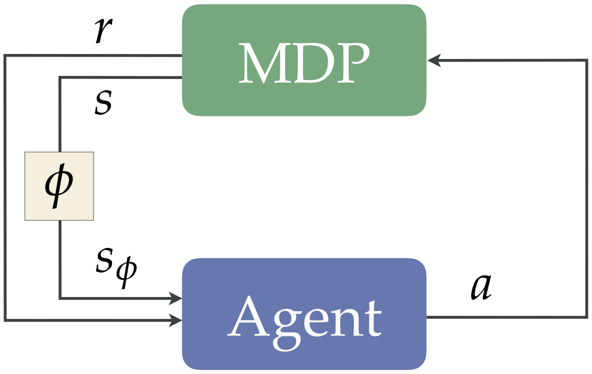


Figure 2.5: RL with a state abstraction.

original MDP M, a state abstraction ϕ, and a *weighting* function [273, 327, 203], $w : \mathcal{S} \rightarrow [0, 1]$, such that:

$$\forall_{s_\phi \in \mathcal{S}_\phi} : \sum_{s \in s_\phi} w(s) = 1, \tag{2.18}$$

where I use $s \in s_\phi$ as shorthand for $s \in \{\bar{s} \in \mathcal{S} : \phi(\bar{s}) = s_\phi\}$.

Together, these three components induce an abstract reward and transition function as follows.

Definition 2.12. *The **abstract reward function**, $R_\phi : \mathcal{S}_\phi \times \mathcal{A} \times \mathcal{S} \rightarrow [\text{RMIN}, \text{RMAX}]$, is a weighted sum of the rewards of each of the ground states that map to the same abstract state:*

$$R_\phi(s_\phi, a, s'_\phi) = \sum_{s \in s_\phi} \sum_{s' \in s'_\phi} R(s, a, s')w(s). \tag{2.19}$$

Definition 2.13. *The **abstract transition function**, $T_\phi : \mathcal{S}_\phi \times \mathcal{A} \rightarrow \Delta(\mathcal{S}_\phi)$, is a weighted sum of the transitions of each of the ground states that map to the same abstract state:*

$$T_\phi(s'_\phi \mid s_\phi, a) = \sum_{s \in s_\phi} \sum_{s' \in s'_\phi} T(s' \mid s, a)w(s). \tag{2.20}$$

With these two components in place, the abstract MDP is defined as follows.

Definition 2.14. *An **abstract MDP** is induced by the triple (M, ϕ, w), yielding:*

$$M_\phi := (\mathcal{S}_\phi, \mathcal{A}, R_\phi, T_\phi, \gamma, \rho_0^\phi), \tag{2.21}$$

where ρ_0^ϕ is the start state distribution ρ_0 projected into the abstract state space.

Of special interest is the best policy representable in this abstract state space. Here, "best" is understood in terms of the *environmental* value function, which defines the actual problem being solved. More formally,

$$\pi_\phi^* = \arg\max_{\pi_\phi \in \Pi_\phi} \mathbb{E}_{s_0 \sim \rho_0} \left[V^{\pi_\phi}(\phi(s_0)) \right]. \tag{2.22}$$

The policy π_ϕ is really a mapping from abstract states to actions, but can easily be turned into a policy over ground states and actions when paired with ϕ. That is, for a given state s, the function composition $\pi_\phi(\phi(s))$ outputs an action.

This policy is particularly informative as it represents the best solution an RL algorithm reasoning with a state abstraction may try to discover. At the end of learning, we may ask about the *value loss* incurred by the state abstraction, which corresponds to the gap in value between the true optimal policy and this optimal abstract policy. For instance, if we were interested in the policy that maximizes the expected ground value function under the start state distribution, it would be desirable to minimize the following quantity:

$$\min_{\pi_\phi \in \Pi_\phi} \mathbb{E}_{s_0 \sim \rho_0} \left[V^*(s_0) - V^{\pi_\phi}(s_0) \right]. \tag{2.23}$$

Much of the technical work of this dissertation focuses on ensuring that abstract policies that still achieve high value in the original problem still exist. While this property is not *sufficient* to ensure an RL algorithm using ϕ can eventually learn good behavior, it is *necessary*—if a good policy cannot even be represented, it certainly cannot be learned.

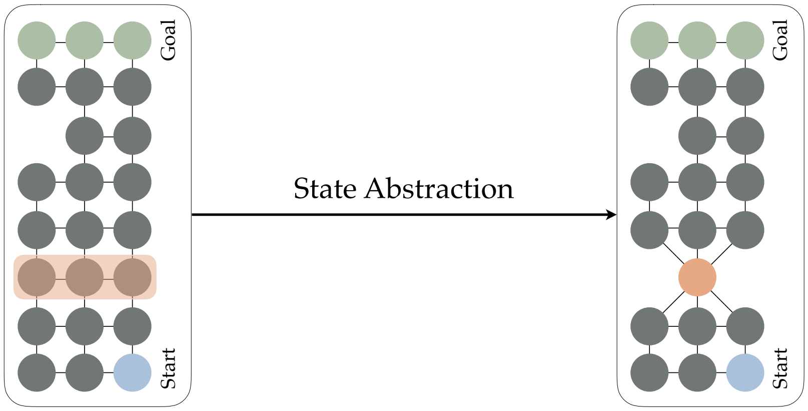


Figure 2.6: A simple grid world problem (left) and the abstracted problem induced by the state abstraction (right).

A few brief comments regarding notation and language are in order. Throughout the dissertation, I will use the term *abstract state* or *cluster* to refer to states in the abstract MDP, and *ground* or *environmental* state to refer to states in the original MDP M. I will occasionally abuse notation and allow $\pi_\phi(s)$ to act as shorthand for $\pi_\phi(s_\phi)$, and similarly $V^{\pi_\phi}(s)$ to abbreviate $V^{\pi_\phi}(\phi(s))$, and so on. Where it is needed for clarity, I include the use of ϕ. Additionally, I let Π_ϕ denote the set containing all policies defined over the abstract state space $\mathcal{S}_\phi$ induced by a particular ϕ. Finally, I use Φ_{all} to denote the space of all state abstraction functions.

As a motivating example, let us suppose an agent is placed into a wide hallway with the goal of reaching the exit, which is placed at the far end of the hall. A traditional representation for this problem might yield a Cartesian grid: the agent has an `x` and a `y` coordinate, and the `up`, `down`, `left`, and `right` actions, and must navigate until its `y` coordinate is sufficiently large (and so has reached the exit of the hallway). This MDP and a corresponding abstract MDP, with a single row abstracted, is pictured in Figure 2.6.

What is an effective state abstraction in this domain? Of course, there are many possible groupings. However, given the structure inherent in the problem, the agent's `x` coordinate is actually irrelevant for computing optimal behavior, or for computing the optimal value function V^* and action-value function, Q^*, or (depending on the algorithm) for efficiently learning any of these quantities. Thus, consider the function ϕ that projects the ground state to an abstract state that only tracks the y coordinate. By this abstraction, all ground states with the same y coordinate belong to the same abstract state. In this way, the true state space of size $N \times M$ can be reduced to an abstract state space of size equivalent to the length of the hallway, M. Intuitively, imagine you are walking down a hallway without obstacles; if your mission was truly to make forward progress toward the exit of the hall, then there is little need to pay attention to horizontal movement.

A single application of this state abstraction is pictured in the right of Figure 2.6. Again, this abstract state space throws away information. For instance, there is no longer a Markov transition model in the abstract space that will track perfectly with every trajectory in the ground MDP. When the agent moves into the clustered state (pictured on the right in orange), what happens when the agent executes the ↑ action? This is determined by choice of weighting function, w, that induces the abstract MDP. Hence, while there is no permissible Markov T_ϕ that can predict $T(s' \mid s, a = \uparrow)$, there surely exist Markov transition functions T_ϕ that can still support representation and discovery of good policies. Intuitively, such a state space will inevitably induce a *wrong* but potentially *useful* model, reminiscent of the classical adage of Box [51].

To summarize, state abstraction is about translating the environmental state space into a new, more well behaved space. The central operation is *aggregation*, which can apply to both continuous-to-discrete or large-to-small transformations. Different kinds of state abstractions are guaranteed to preserve certain properties, and are thus desirable for RL algorithms to discover and use. With our notation and concepts in place, I now turn to a survey of prior literature on state abstraction in RL.

2.2.1 *Prior Work on State Abstraction*

The study of state abstraction in RL has a rich history, dating back to early work on approximating dynamic programs by Fox [107], Whitt [340, 341], Axsäter [29], and Bertsekas and Castanon [45]. Additionally, much of the literature has been heavily influenced by research on abstraction in planning [279, 165, 86], and hierarchical RL [85, 152, 342, 259, 88, 129]. Indeed, the literature on state abstraction is vast. I here survey prior research that focuses on state abstraction in the context of RL in MDPs.

BISIMULATION. The work of Fox [107] and Whitt [340, 341] paved the way for understanding the value loss of state abstraction in MDPs. Fox and Whitt establish the first principles regarding state aggregation and its impact on value function representation and dynamic programming. Building on this work, Dean and Givan [86] developed an algorithm for finding states that resemble one another via the *bisimulation* property [188]. Concretely, Dean and Givan [86] introduce the approximate bisimulation metric for partitioning an MDP's state space into clusters of states whose transition model and reward function are within $\varepsilon \in \mathbb{R}_{\geq 0}$ of each other. In later work, Givan et al. [120] use these ideas to develop an algorithm called Interval Value Iteration (IVI) that converges to the correct bounds on a family of abstract MDPs called Bounded MDPs, which summarize the space of possible MDPs the agent *could* be in, given the agent's current knowledge of the MDP.

Since then, many approaches have adapted bisimulation in a variety of contexts related to state abstraction. Ferns et al. [102, 103] develop state similarity metrics for MDPs by bounding the value difference of ground states and abstract states for several bisimulation metrics that induce an abstract MDP. More recent work has since extended these bisimulation metrics to cooperate with action abstraction [65] and high-dimensional state spaces [114, 64]. Taylor et al. [315] further analyze the performance loss associated with using a bisimulation as part of an MDP homomorphism [272]. In a similar vein, Even-Dar and Mansour [99] study different distance metrics used in identifying state space partitions subject to ε-similarity, providing value loss bounds for ε-homogeneity subject to the L_∞ norm. Even et al. also prove that the problem of finding the maximally compressing state abstraction is NP-hard, a result I return to in Chapter 4. In more recent work, Lehnert and Littman [194, 195] combine ideas from bisimulation with the successor representation

[83] in the form of successor features [34]. Lehnert and Littman present a novel combination of these two concepts in the form of the Linear Successor Feature Model (LSFM) and establish how LSFMs can underlie effective transfer, generalization, and model-based RL more generally.

SOFT STATE AGGREGATION. One of the earliest studies of state abstraction in RL was carried out by Singh et al. [294], who introduce *soft state aggregation*. These soft forms of aggregation generalize the typical aggregation function to the class of stochastic functions $\tilde{\phi} : \mathcal{S} \to \Delta(\mathcal{S}_\phi)$. Using $\tilde{\phi}$ to act, an RL agent samples $s_\phi \sim \tilde{\phi}(s)$, and learns functions based on abstract s_ϕ, rather than s. Singh et al. present convergence guarantees for Q-learning with a fixed aggregation $\tilde{\phi}$, and a new heuristic method for adapting such an aggregation online during learning.

MODEL SELECTION. A related and important body of work studies the problem of *selecting* a state abstraction from a given class [173, 216, 251, 142, 254]. Ortner et al. [253] developed an algorithm for learning partitions in an online setting by taking advantage of the confidence bounds for T and R provided by UCRL [28], a model-based RL algorithm that explores its environment efficiently. In earlier work, Konidaris and Barto [173] formulate the abstraction selection problem as a model selection problem, making use of the Bayesian Information Criterion [282] to inform which abstraction to choose. This work focuses primarily on the selection of action abstractions, however, and will be discussed in more length in this later survey (subsection 2.3.1) Later, Jiang et al. [142] analyze the problem of choosing between two candidate abstractions for model-based RL. The core analysis again studies an algorithm that treats the choice of abstraction as a model selection problem, and analyzes the trade off between approximation error and estimation error produced by each abstraction. van Seijen et al. [329] study a similar problem in the context of Factored MDPs, in which the agent is explicitly given actions that move between candidate abstractions. Diuk et al. [94] study the closely related problem of *feature* *selection* in MDPs, which has also received careful attention in prior work [260, 184, 296]. Diuk et al. make use of the adaptive *k*-meteorologists problem to learn an appropriate set of features. Most recently, Ortner et al. [254] also study the problem of choosing an appropriate state abstraction from a given library during RL in MDPs without trap states. The key assumption is that at least one of the abstractions in the library induces an MDP—subject to this assumption, they develop an online algorithm that enjoys bounded regret relative to this best state abstraction in the library.

LEARNING WHAT IS IRRELEVANT. The early work of McCallum [225] proposed the U-Tree algorithm that learns to represent state only in terms of its most relevant factors. The algorithm proceeds by carrying out statistical tests to determine which factors can be ignored, and which must be included in the tree. This notion of identifying irrelevant factors has been a key component of many abstraction methods, such as the work by Jong and Stone [147]. Here, Jong and Stone propose the property of policy irrelevance, which states roughly that states can be grouped together if they have the same optimal action. They then present statistical tests that can be deployed to determine which state variables may be safely ignored. Menashe and Stone [236] introduce an algorithm for abstracting continuous state, with the goal of inducing a small, tractable decision problem. They present Recursive Cluster-based Abstraction Synthesis Technique (RCAST), a technique for constructing a state abstraction that maps continuous states to discrete ones. Like the other tree-based methods discussed, RCAST uses *k*-d-trees to partition the state space. The key insight at the core of RCAST is to partition based on the ability to predict different *factors* that characterize the state space.

OTHER ADAPTIVE AND ONLINE METHODS. Several related approaches introduce algorithms for adaptively updating a state-space partitioning scheme, but do not base the clustering on notions of irrelevance. Lee and Lau [192] use a form of adaptive vector quantization to repeatedly partition a continuous state space into a discrete one, thereby enabling classical RL algorithms like Q-learning to learn in a continuous space. To carry out this partitioning, Lee et al. make use of a form of vector quantization to rapidly cluster a continuous state space online in a computationally efficient manner using a Voronoi like tesselation of the state space. Then, the resulting state space can be used by Q-learning like algorithms (such as Temporal Difference or TD learning [305]). Krose and Van Dam [185] develop a similar approach in the context of controlling a robot to avoid collisions. Here, the state space is discretized using a Voronoi tesselation of the input state space, similar to Lee and Lau [192]. Nicol and Chadès [250] build on these approaches by extending the ideas of state-space quantization to the problem of state estimation in POMDPs, focused on applications in conservation biology. Lastly, Cobo et al. [71, 72] study methods for finding state abstractions based on a *demonstrator's* behavior—their method constructs abstract states that can be used to *predict* what a demonstrator will do in each cluster. This idea will partially inform the study of Chapter 5 as well. Similarly, Akrour et al. [12] proposes a method to simultaneously learn a clustering and learn behavior within each cluster. This coupled learning process is shown to be effective, as the existence of a sufficiently useful policy within each cluster is precisely the property needed to determine how to assign the clusters.

Most recently, Du et al. [96] and Misra et al. [237] study the process of learning a state abstraction online when learning in an observation-rich environment. Both approaches develop and analyze an algorithm for learning a state abstraction online assuming that the environment can be well modelled by a small, well-behaved state space, as modeled by a Block MDP introduced by Du et al. [96]. A Block MDP, roughly, is an MDP that is describable in terms of a small, well behaved state space, but is *observed* through rich observations that are uniquely determined by their underlying latent state. Then, the algorithm of Du et al. [96] focuses on learning a mapping from this rich observation space to an estimate of this latent state space. The algorithm of Misra et al. [237] follows a similar approach, but searches for abstract state spaces in which all ground states share forward *and* backward transitions are shared, called "kinematic inseparability". Both algorithms enjoy guarantees on the sample complexity of RL while learning and exploiting these state abstractions.

A UNIFIED FRAMEWORK OF STATE ABSTRACTION IN MDPS. Li et al. [203] presented a unifying framework for state abstraction in MDPs. They define five types of state abstractions that each ensure some property must hold between all the ground states in each abstract state. Formally, a state abstraction *type* is defined with respect to a two-argument predicate on state pairs.

Definition 2.15. *A* ***state abstraction type*** *is a collection of functions* $\Phi_p \subseteq \Phi_{all}$ *associated with a fixed predicate on state pairs,* $p : \mathcal{S} \times \mathcal{S} \to \{0, 1\}$*, such that when any* $\phi_p \in \Phi_p$ *clusters state pairs, the predicate must be true for that state pair:*

$$\phi_p(s_1) = \phi_p(s_2) \implies p(s_1, s_2). \tag{2.24}$$

Several candidate types introduced in previous work are presented in Table 2.1, along with two notable properties I discuss in Chapter 3 and Chapter 4. For simplicity, I abuse notation and let ϕ_p denote the type Φ_p for any predicate p.

Name	Predicate	Value Loss	Transitive
ϕ_{Q^*}	$\max_a \lvert Q^*(s_1, a) - Q^*(s_2, a)\rvert = 0$	0	yes
ϕ_{a^*}	$a_1^* = a_2^*$ and $V^*(s_1) = V^*(s_2)$	0	yes
$\phi_{\pi*}$	$\pi^*(s_1) = \pi^*(s_2)$	0	yes
$\phi_{Q_\varepsilon^*}$	$\max_a \lvert Q^*(s_1, a) - Q^*(s_2, a)\rvert \leq \varepsilon$	$\frac{2\varepsilon \text{RMax}}{(1-\gamma)^2}$	no
ϕ_{mult}	$\max_a \left\lvert \frac{Q^*(s_1,a)}{\sum_b Q^*(s_1,b)} - \frac{Q^*(s_2,a)}{\sum_b Q^*(s_2,b)} \right\rvert \leq \varepsilon$	$2\varepsilon \frac{\lvert\mathcal{A}\rvert \text{RMax}+k}{(1-\gamma)^2}$	no
ϕ_{bolt}	$\max_a \left\lvert \frac{e^{Q^*(s_1,a)}}{\sum_b e^{Q^*(s_1,b)}} - \frac{e^{Q^*(s_2,a)}}{\sum_b e^{Q^*(s_2,b)}} \right\rvert \leq \varepsilon$	$2\varepsilon \frac{(\lvert\mathcal{A}\rvert \text{RMax}+\varepsilon k+k)}{(1-\gamma)^2}$	no
$\phi_{Q_d^*}$	$\forall_a : \lceil \frac{Q^*(s_1,a)}{d} \rceil = \lceil \frac{Q^*(s_2,a)}{d} \rceil$	$\frac{2d \text{RMax}}{(1-\gamma)^2}$	yes

Table 2.1: A summary of several existing state abstraction types.

Li et al. analyze five state abstraction types, in many cases drawing from state abstractions introduced in prior work. These five classes are as follows.

1. ϕ_{model}: The ground reward function and the ground transition function into abstract states are the same,

$$\phi_{model}(s_1) = \phi_{model}(s_2) \implies \forall_{a \in \mathcal{A}} : R(s_1, a) = R(s_2, a), \tag{2.25}$$

and

$$\forall_{s_\phi \in \mathcal{S}, a \in \mathcal{A}} : \sum_{s' \in s_\phi} T(s' \mid s_1, a) = \sum_{s' \in s_\phi} T(s' \mid s_2, a). \tag{2.26}$$

2. ϕ_{Q^π}: The Q function under any policy is the same,

$$\phi_{Q^\pi}(s_1) = \phi_{Q^\pi}(s_2) \implies \forall_{a \in \mathcal{A}, \pi \in \Pi} : Q^\pi(s_1, a) = Q^\pi(s_2, a). \tag{2.27}$$

3. ϕ_{Q^*}: The Q values for each action are the same,

$$\phi_{Q^*}(s_1) = \phi_{Q^*}(s_2) \implies \forall_{a \in \mathcal{A}} : Q^*(s_1, a) = Q^*(s_2, a). \tag{2.28}$$

4. ϕ_{a^*}: The optimal action is the same and the Q value is the same for that action in each state.

$$\phi_{a^*}(s_1) = \phi_{a^*}(s_2) \implies \arg\max_{x \in \mathcal{A}} Q^*(s_1, x) = \arg\max_{y \in \mathcal{A}} Q^*(s_2, y), \tag{2.29}$$

and

$$|V^*(s_1) - V^*(s_2)| = 0. \tag{2.30}$$

5. ϕ_{π^*}: The optimal policy chooses the same action in each state,

$$\phi_{\pi^*}(s_1) = \phi_{\pi^*}(s_2) \implies \pi^*(s_1) = \pi^*(s_2). \tag{2.31}$$

Li et al. further introduce the ordering operator, $\Phi_X \geq \Phi_Y$, which states that any instance $\phi_X \in \Phi_X$ is also an instance of Φ_Y. They prove the following ordering among the five introduced classes:

Theorem 2.1. *(Theorem 2 from [203]) For any MDP,* $\Phi_0 \geq \Phi_{model} \geq \Phi_{Q^\pi} \geq \Phi_{Q^*} \geq \Phi_{a^*} \geq \Phi_{\pi^*}$.

That is, any element of Φ_{model} is also an element of Φ_{Q^π}, and so on. This result is particularly useful for understanding planning performance in the abstracted models and for clarifying convergence conditions of different algorithms. They further prove that certain types of state abstraction preserve representation of optimal behavior.

Theorem 2.2. *(Theorem 3 from [203]) Any state abstraction function of type* $\Phi_{model}, \Phi_{Q^\pi}, \Phi_{Q^*}$ *or* Φ_{a^*} *preserves the optimal policy. That is, if the abstraction* ϕ *is any of the above types, then the value loss is 0. Conversely, instances of* Φ_{π^*} *do not always preserve the optimal policy.*

Subsequent study by Walsh et al. [333] investigates the power of state abstraction functions for supporting efficient transfer learning. That is, in the multitask RL setting, they seek to maximize the speedup ratio, which measures the reduction in time needed to find a good policy on the current MDP given that information may be transferred from a collection of source MDPs. Walsh et al. introduce the General Abstraction Transfer Algorithm (GATA), an algorithm for carrying out effective transfer of state abstractions across these MDPs by choosing a ϕ that maximizes this speedup ratio.

REPRESENTING STATE IN OTHER FRAMEWORKS. Many previous works have proposed different kinds of state representation (beyond aggregation) for MDPs. Dietterich [88] introduced the MAXQ framework for decomposing value functions into a hierarchy represented as an acyclic graph over subtask policies. In subsequent work, [89] further investigate the impact of state abstraction on learning under MAXQ. Dietterich highlights five conditions for safe state abstraction, under which a specific RL algorithm operating in the MAXQ framework is guaranteed to discovery a globally optimal policy. These five conditions primarily deal with picking up on the right notions of irrelevance, similar to those methods discussed above. Andre and Russell [17] also investigated a method for state abstraction in hierarchical reinforcement learning leveraging a programming language called ALISP that supports *safe* state abstraction. Agents programmed using ALISP can ignore irrelevant parts of the state, achieving abstractions that maintain optimality.

STATE REPRESENTATION AND VALUE FUNCTION APPROXIMATION. A variety of other approaches to state representation learning have been proposed that parallel the objectives of state abstraction. For instance, Whiteson et al. [339] build an adaptive variant of the classical process of tile coding [307] for use in value function approximation. A related body of work studies the formation and discovery of appropriate basis functions for use in linear function approximation, typically applied to estimating Q^*. Lagoudakis and Parr [187], Mahadevan and Maggioni [214, 215], and Konidaris et al. [177] propose different kinds of basis functions (the polynomial, Laplacian, and Fourier, respectively) for use in value function approximation, each offering different desirable characteristics. Similarly, Liang et al. [206] present empirical evidence that classical learning algorithms can achieve competitive performance to many deep RL algorithms. Their approach constructs features that are well suited to the structure of Atari games, including properties like relative object and color locations. The main result of the work shows that with a well crafted set of features, even a simple learning algorithm can achieve competitive scores in Atari games.

VALUE PRESERVATION. Since the work of Fox [107], a long line of work has brought continued clarity the conditions under which state abstractions preserve value in MDPs. Van Roy [327] provide bounds on the suboptimality achieved by approximate VI in a well-behaved class of MDPs, in line with work on model minimization [86, 120, 270, 272, 273]. Recent analysis extends these insights to non-Markovian environments [135, 136, 217], invoking similar classes of state abstraction to those surveyed by Li et al. [203] but adapted to a more general class of environments. Again, the focus is on determining which classes of state abstraction are guaranteed to preserve representation of high value policies, a property that will be of central focus throughout the dissertation.

Li [202] also analyze a form of approximate state abstraction as applied to Delayed Q-learning (see Corollary 1 in Section 8.2.3). The main result here presents a modified sample complexity for Delayed Q-learning that only depends on the size of the abstract state space rather than the true state space. The key property to note, however, is that the analysis assumes Delayed Q-learning is interacting directly with an abstract MDP M_ϕ, rather than interacting with M and projecting states through ϕ. This difference will return again in Chapter 4, and particularly Equation 4.1. Additionally, Lehnert et al. [196] explore the impact of horizon length on representation of value functions, with close ties to the diameter of an abstract state. They find that the value loss of any policy that optimizes with respect to an artificially-short horizon can achieve value similar to that of policies that take into account the full horizon, building on the results of Jiang et al. [143].

STATE ABSTRACTION AND EXPLORATION. Mandel et al. [219] focus on the exploration-exploitation dilemma in the context of state abstraction. In particular, they introduce a Bayesian method for clustering states to facilitate effective exploration while generalizing across the state space appropriately. The core contribution is the algorithm Thompson Clustering for Reinforcement Learning (TCRL), which addresses the large space of possible abstract state spaces by exploiting explicit structure present in the environment. As a result, TCRL narrows the search delicately to improve learning speed, enjoying Bayesian regret guarantees. A slight variant of TCRL achieves regret similarly to that of PSRL [255]. Separately, Taïga et al. [313] study the exploration-exploitation dilemma from the perspective of approximate state abstraction.

Moore [240] introduced the Parti-Game algorithm, which uses a decision tree to dynamically partition a continuous state space based on the need for further exploration. That is, as data about the underlying environment is collected, state partitions are refined depending on a minimax score with respect to an adversary that prevents the learning algorithm from reaching the goal (and knows the current partitioning scheme). Parti-Game applies in tasks where 1) the transition function is deterministic, 2) the MDP is goal-based and the goal state is known, and 3) a local greedy controller is available. Feng et al. [101] also make use of a tree-based approach—this time, k-d-trees [110]—to dynamically partition a continuous MDP's state space into discrete regions. In contrast to Parti-Game, partitions are chosen based on value equivalence, thereby enabling a form of closure under the Bellman Equation.

Chapman and Kaelbling [66] study tree-based partitioning as a means of generalizing knowledge in RL. Specifically, Chapman and Kaelbling propose the G algorithm, which constructs a data-dependent tree of Q-value partitions based on which Q value can adequately summarize different regions of the state space. Over time, the tree will grow to sufficiently represent the needed distinctions in states. Further work uses decision trees of different forms to partition complex (and often continuous) state spaces into discrete models [323]. Asmuth et al. [25] introduce the Best of Sampled Set (BOSS) algorithm, a Bayesian approach to exploration in RL that accommodates priors for clustering states. The algorithm itself resembles PSRL: maintain a posterior on models, sample from the posterior, and use the samples to inform decision making.

STATE REPRESENTATION LEARNING. A separate but relevant body of literature investigates learning state *representations* in the context of control and deep RL. For instance, Jonschkowski and Brock [150] proposed learning state representations through a set of well chosen ontological priors, catered toward robotics tasks, including a simplicity prior and a causality prior (among others). These priors are then encoded into an optimization problem that seeks to jointly optimize over each of their prescribed properties. Similarly, Karl et al. [156] developed a variational Bayes method for learning a latent state-space representation of a Markov model, given high dimensional observations. Critically, this state space is of a simple Markov model, and does not involve decision making or rewards, which are critical aspects of learning state representations in MDPs [252]. For a full survey of recent state representation schemes for deep RL, see text by Lesort et al. [198], or the recent survey by Bertsekas [44].

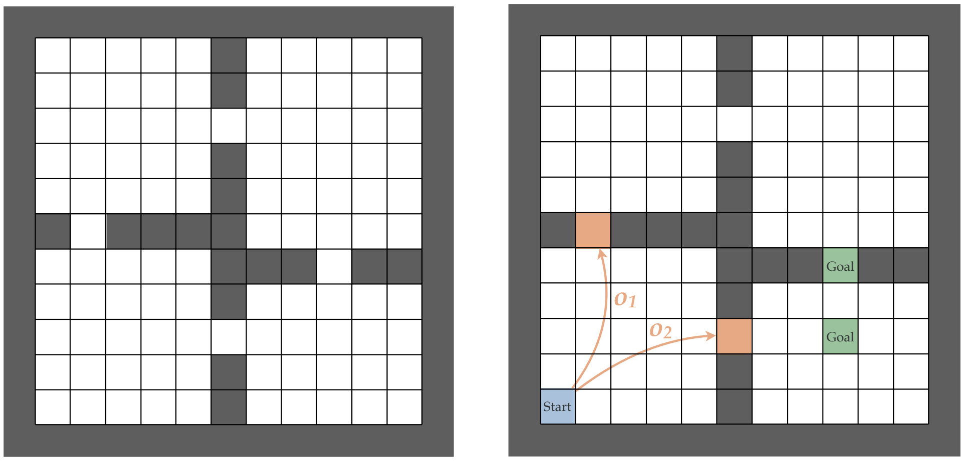
STATE ABSTRACTION AND PLANNING As a final note, state abstraction has also been applied extensively in the context of planning. Given the breadth of planning as a field, I again highlight several methods that are tightly connected to MDPs and RL. Hostetler et al. [133] apply state abstraction to Monte Carlo Tree Search [166, 74, 139] and expectimax search, giving value bounds of applying the optimal abstract action in the ground tree(s). Other work develops similar methods for incorporating state abstraction into Monte Carlo style planning algorithms [15, 16, 141]. Dearden and Boutilier [87] also examine state abstraction for planning, focusing on abstractions that are quickly computed and offer bounded value. The primary analysis is on abstractions that remove negligible literals from the planning domain description, yielding value bounds and a mechanism for incrementally improving abstract solutions to planning problems.

As is hopefully apparent, the literature on state abstraction in RL is exceptionally both broad and deep. I have chosen to exclude those approaches that are better considered as joint state-action abstraction, as they will be discussed in subsection 2.3.2.

I now turn to a formal introduction of action abstraction.

2.3 ACTION ABSTRACTION

Action abstractions describes methods that empower the *action space* of an RL agent. With a well structured action space, decision making agents can probe more deeply in search, plan efficiently by focusing on progress toward a subgoal, or prune away irrelevant primitive action sequences based on knowledge of action-optimality correlations. Of course, as with structuring state, there are many possible operations available to organize the action space. We might add new long horizon sequences of actions, prune away actions, or add



(a) Four Rooms MDP

(b) Four Rooms MDP with Options

Figure 2.7: The classical Four Rooms domain (left) extended by options (right).

actions that specifically try to satisfy some property or reach a subgoal with high probability. I here concentrate on what has become the most canonical formalism for action abstraction, the options framework introduced by Sutton et al. [311].

To introduce options, let us consider the example domain pictured in Figure 2.7. This problem is known as Four Rooms [311] and will be used as a canonical grid world problem for simple experiments and visuals throughout this dissertation. As with the original Russell & Norvig grid world, the agent may move up, down, left, and right, with the goal of getting from a particular cell in the grid to another. In this case there are long walls that change the structure of the problem. The flow of movement throughout the environment is thought to be suggestive of certain types of action abstractions, such as those that take the agent to the doorways between rooms, or that transition between the rooms directly. These kinds of additional high level behaviors may be naturally expressed in terms of options. At a high level, an option is one prescription for an abstract behavior—in the language of our forest example from Chapter 1, options will constitute the behaviors at the level of "move to the waterfall" or "navigate back to camp", rather than "rotate head".

More formally, an option is defined as follows.

Definition 2.16. *An **option** is a triple $o = (I_o, \beta_o, \pi_o)$, where:*

- *$\mathcal{I}_o \subseteq \mathcal{S}$ is a subset of states denoting in which states the option is available to be executed,*
- *$\beta_o : \mathcal{S} \rightarrow \Delta(\{0,1\})$, assigns a Bernoulli random variable to each state denoting the probability that the option terminates upon arriving in that state,*
- *$\pi_o : \mathcal{S} \rightarrow \Delta(\mathcal{A})$ is a behavioral policy.*

Options denote abstract actions; the three components indicate where the option o can be executed ($\mathcal{I}_o$), where it terminates (β_o), and what to do in between these two conditions (π_o). Options are known to aid in transfer [172, 174, 58, 321], encourage better exploration [227, 290, 291, 58, 30, 111, 211, 319, 145], and make planning more efficient [222, 223, 144].

Action abstraction has historically been treated as a generic class of operations that change the action space of an agent. Throughout this thesis, I will formalize the process of action abstraction as a replacement of the primitive actions of an agent with some set of options, $\mathcal{O}$. I again take this class of operations to be of sufficient generality so as to characterize the important questions about abstraction, but not to be so general so as to limit analysis, understanding, and progress. Concretely, I define action abstraction as follows.

Definition 2.17. *An **action abstraction** is a function $\omega : \mathcal{A} \mapsto \mathcal{O}$ that replaces the primitive actions $\mathcal{A}$ with a set of options $\mathcal{O}$.*

An RL algorithm paired with an action abstraction chooses from among the available options, denoted $\Omega(s)$, at each time step. That is,

$$\Omega(s) := \{o \in \mathcal{O} : s \in \mathcal{I}_o\}. \tag{2.32}$$

Then, the agent runs the option until it terminates in some state s' according to $\beta_o(s)$. Finally, the algorithm again chooses its next option from among the set $\Omega(s')$ and repeats this process indefinitely.

With $\mathcal{O}$ *replacing* the primitive action space, it is not necessarily the case that every policy over $\mathcal{S}$ and $\mathcal{A}$ may be represented. That is, the action abstraction may destroy an agent's ability to ever discover a near-optimal policy. Note, however, that the formalism is expressive enough to describe the case where the primitive actions are redefined in terms of options. For instance, the action a_1 can be translated into an option by constructing the option that initiates in every state, terminates with probability 1 in every state, and executes the policy $\pi_1 : \mathcal{S} \rightarrow \{a_1\}$, effectively encodes a_1 as an option. However, by including options *and* primitive actions, learning algorithms face a larger branching factor, and must search the full space of policies which can hurt learning performance [149]. Hence, it is often prudent to restrict the action space *only* to a set of options to avoid blowing up the search space.

Then, when the agent chooses from among available options, the agent commits to executing the policy associated with the option until the sampled terminating condition is true for a state the agent arrives in. For example, if the termination condition assigns zero probability to all states except s_4 (in which $\beta(s_4) = 1$), then the agent will execute the option's policy indefinitely until arriving in s_4. When the agent reaches s_4, the agent will stop executing the option policy, and will make its next choice of action or option. So, options facilitate action pruning of a certain form; when an option is selected, every state the agent arrives in up until termination, the actions not chosen by the option policy are effectively pruned. The resulting decision making problem slightly loses out on the Markov property, too, as any state encountered while executing the option will induce a different policy according to which option is currently being run. This process is pictured in Figure 2.8. As with state abstraction, I will occasionally use $\mathcal{O}_{\text{all}}$ to denote the set of all options to simplify notation.

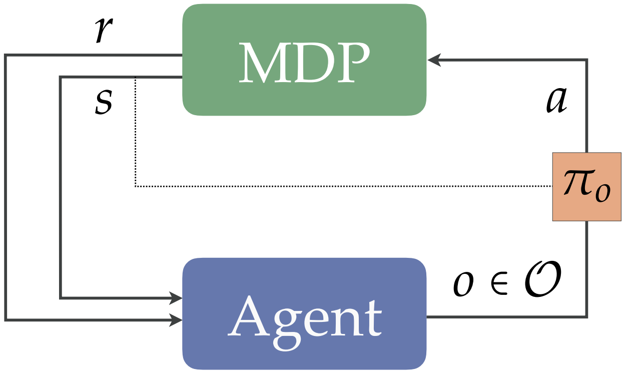


Figure 2.8: RL with action abstraction.

Since the primary objects of interest in an action abstraction are the options introduced, $\mathcal{O}$, I will largely talk about action abstraction in terms of which options are added.

Options also give rise to new transition and reward functions based on where the options will terminate and the trajectory taken by the option's policy. The model was originally proposed for options by Precup and Sutton [263, 264], and is defined as follows.

Definition 2.18. *For a given γ and option o, the* ***multi-time model*** *(MTM) defines the transition and reward function by:*

$$T_\gamma(s' \mid s,o) := \sum_{k=0}^{\infty} \gamma^k \beta_o(s') \mathbb{P}(S'_k = s' \mid S_0 = s, O = o) = \sum_{k=0}^{\infty} \gamma^k \beta_o(s') p(s', k \mid s, o), \tag{2.33}$$

$$R_\gamma(s,o) := \mathop{\mathbb{E}}_{k, s_{1\ldots k}} \left[r_1 + \gamma r_2 \ldots + \gamma^{k-1} r_k \,\middle|\, s, o \right]. \tag{2.34}$$

This model allows for straightforward application of many standard RL and optimization algorithms to settings that incorporate options. In Chapter 6, I will study how these models can be exploited to make planning faster, and in Chapter 7, I motivate a simpler alternative to these models. As with state abstraction, it is natural to define an abstracted form of the environmental MDP in terms of the available options (and their models) at a given time step. Without further modification, the induced model is in fact a *semi-*Markovian decision process, as the situation in which the agent enters state s actually differs depending on whether the agent is executing some option o, or not [311]. In Chapter 9, I will avoid the semi-Markovian nature of options by studying a restricted subclass of options that can only describe policies that are Markov in the original MDP.

In the Four Rooms MDP, let us consider two options that terminate in hallways, pictured in the right of Figure 2.7. These two options are defined as follows.

$$o_1 = \begin{array}{ll} \mathcal{I}(s) & s = (1,1), \\ \beta(s) & \mathbb{1}\{s = (3,6) \text{ or } s = (6,3)\}, \\ \pi(s) & \arg\max_{a \in \mathcal{A}} \left(\mathbb{1}\{s = (3,6) \text{ or } s = (6,3)\} + \gamma \sum_{s' \in \mathcal{S}} T(s' \mid s, a) V^*_{\mathbb{1}\{s=(3,6) \text{ or } s=(6,3)\}}(s') \right), \end{array}$$

and,

$$o_2 = \begin{array}{ll} \mathcal{I}(s) & s = (1,1), \\ \beta(s) & \mathbb{1}\{s = (3,6) \text{ or } s = (6,9)\}, \\ \pi(s) & \arg\max_{a \in \mathcal{A}} \left(\mathbb{1}\{s = (3,6) \text{ or } s = (6,9)\} + \gamma \sum_{s' \in \mathcal{S}} T(s' \mid s, a) V^*_{\mathbb{1}\{s=(3,6) \text{ or } s=(6,9)\}}(s') \right). \end{array}$$

The first option o_1 will initiate in the bottom left corner, and will only terminate when the agent arrives in one of the two hallways leaving the lower left room, and executes the policy that moves the agent to one of the hallways as quickly as possible. So, the option might induce the following true trajectory:

$$(1,1), \rightarrow, (2,1), \uparrow, (2,2), \rightarrow, (3,2), \rightarrow, (4,2), \rightarrow, (5,2), \uparrow, (5,3), \rightarrow, (6,3). \tag{2.35}$$

When the agent enters a new state while executing either option it draws a sample from the Bernoulli distribution $\beta_o(s)$. If the sample is 1, then the agent *stops* following the option policy, and is again in a position to choose among its active options (and perhaps primitive actions). In the trajectory above, we define the option o_1 such that $\beta_o((6,3)) = 1$, and so the agent will stop executing the option at $(6,3)$. At this point, the agent resumes its regular action selection process.

With the basic notation and formalisms for action abstraction established, I next survey previous research in the area.

2.3.1 *Prior Work on Action Abstraction*

As with state abstraction, research on action abstraction in RL has a long and deep history. Early work concentrated on incorporating macro-operators [182, 183] that characterized relevant sequences of actions to accelerate both planning and RL [137, 293, 229, 228, 235, 298], often expressed through hierarchical structures [85, 152, 128]. In the planning literature, it has long been understood that behavioral abstractions in the form of hierarchies can greatly accelerate planning speed, as in Hierarchical Task Networks [76, 98]. For these reasons, the primary focus of work on action abstraction in RL has been to unlock this same degree of highly efficient decision making.

In the late 1990s and early 2000s, three formalisms emerged for capturing similar notions of abstract behavior. Parr [258] proposed Hierarchies of Abstract Machines (HAMs) to specify abstract behaviors in terms of partially specified *programs*. Around the same time, Dietterich [88] proposed the MAXQ framework for hierarchical RL discussed briefly in subsection 2.2.1. Finally, Sutton et al. [311] developed the options framework that much of this section has been focused on. Each of these three methods systematizes the study of behavioral abstractions in a slightly different way. Given the focus of this dissertation on options (for reasons discussed earlier in the chapter), I concentrate this survey on prior research on the options framework.

The initial options work grew out of the dissertation by Precup [262]. Since then, options have explored for their capacity to address many aspects of the RL problem, from aiding in transfer, to representation learning, to off-policy evaluation, to planning. This survey decomposes the use of options into these different desired effects, though in many cases options are intended to aid in more than one of these processes.

OPTION DISCOVERY. In contrast to planning, incorporating options into RL algorithms typically requires that options are learned online. It has thus long stood as an open question as to what precisely constitutes a good option for RL, and more generally how to learn good options through interaction alone. The option discovery literature studies variations of this problem. The emphasis has tended to be on forming objectives that give rise to algorithms that learn options during RL, though naturally the literature is diverse.

One of the most popular strategies for option discovery is based around the discovery of subgoals as a mechanism for inspiring options—once a set of subgoals (or often abstract states) is fixed, options may be defined that move the agent to the subgoal as quickly as possible. McGovern and Barto [228] propose one of the earliest option discovery methods based on estimating useful subgoals from a series of past trajectories. The main idea is to those states that always appear in successful trajectories, and not on any unsuccessful trajectories—if the number of trajectories is sufficiently large, then intuitively these states are likely to be useful subgoals. These subgoals are then used to inform options, which are shown to accelerate RL in benchmark tasks. Digney [90] and Stolle and Precup [298] introduce similar approaches that determine subgoals based on which states have high visitation count on past successful runs. By similar reasoning, Menache et al. [235] propose the Q-cut algorithm for discovering useful subgoals in RL. Here, the subgoals are identified by estimating the MDP's transition graph and solving a min-cut problem online to identify bottleneck states. Mannor et al. [224] later developed an algorithm inspired by similar principles; the algorithm maintains an estimate of the MDP's transition graph and applies a form of state abstraction to determine clusters of states that have similar value functions. Using these clusters, options are then naturally defined as those policies that transition between the clusters, thus moving the RL agent between relevant subregions of the MDP. In a similar vein, Castro and Precup [65] construct an algorithm for discovering options that connects disparate states, with state-distance defined according to the bisimulation metrics discussed earlier in the chapter. In all of these works, options are empirically shown to accelerate RL on collections of benchmark tasks. Also along these lines, Provost et al. [265] learn discrete abstract state features in a continuous state environment. Then, options are constructed that explicitly target nearby feature changes, thereby again moving the agent through the state space. A separate but closely related pair of approaches by Thrun and Schwartz [317] and Pickett and Barto [261] searches for high value policies in related tasks that share decisions in select states. These policies are then merged to form a general purpose option that is likely to accelerate learning across the different tasks.

Later, Şimşek and Barto [290] build on the above approaches by explicitly studying the property of movement throughout the MDP, perhaps most closely related to the work of Mannor et al. [224]. Here, Şimşek and Barto use the property of *relative novelty* to identify those states that are hard for the RL agent to reach from its current region of the state space. These hard to reach states (referred to as "access states") closely resemble many of the bottleneck or subgoal discovery methods discussed above, but are unique in that they are defined so as to allow the agent to reach a new portion of the state space. The relative novelty of a state s_t is said to be the ratio of novelty between the states before $s_{t-k}, \ldots, s_{t-1}$ and after $s_{t+1}, \ldots$. Hence, identifying novel subgoals is reduced to a classification problem in which subgoals are learned that have relative novelty above a particular well chosen threshold. This classification problem is solved in both a batch offline and online setting. Using the estimated subgoals, options are generated that allow RL agents to move more fluidly throughout the MDP. As with many of the approaches surveyed thus far, experiments are conducted contrasting learning with and without the learned options on grid world variations and the Taxi task introduced by Dietterich [88]. This was shortly followed by Şimşek et al. [292], which blends the notion of access states with the min-cut method by Menache et al. [235] but focuses on local rather than global cuts.

In follow up work, Şimşek and Barto [291] analyzes the graph theoretic property of *betweenness centrality* [109], and argues for its application in formalizing what it means to be a good option. The betwenness centrality assigns a real number to each vertex of a graph measuring roughly how easy it is to reach all other vertices from that vertex. Subgoal states, then, are defined as those states with high betweenness centrality in their local region of the state space. That is, if a state s has considerably higher betweenness then all states reachable in some chosen $n \in \mathbb{N}$ steps from s, it is a good candidate for a subgoal. Options are then defined based around this subgoal as in prior work.

Konidaris and Barto [174] present one of the first algorithms for option discovery in goal-based MDPs with continuous state based on the idea of *chaining*. Concretely, the algorithm focuses on identifying options whose termination conditions always lie inside of at least one other option's initiation condition, thereby ensuring sequential application of options. The key idea is that the chain of options discovered ensure that the goal is eventually contained at the end of a chain, thus allowing an execution of the options to lead to the goal. More generally, the initiation and termination conditions of the options can be framed around arbitrary target events, such as those subgoals or bottlenecks discussed in prior work, rather than just the goal state. The chained options are shown to dramatically improve learning on the challenging continuous state Pinball domain, both when the ideal options are given up front and when the chained options are learned online. Bagaria and Konidaris [31] recently extend these ideas to coordinate with deep neural networks to great effect in environments with rich observations.

As discussed briefly earlier in the chapter, Konidaris and Barto [173] frame the problem of option discovery as one of model selection—given a collection of experiences collected during exploration, the objective is to determine the best option from a predetermined library of candidate options. In this work, Konidaris and Barto in fact study the joint process of selecting a state and action abstraction. The criteria used to determine the best abstraction is that of the Bayesian Information Criterion [282], which offers an elegant principle for determining the simplest, but most explanatory abstraction given the data. Moreover, it allows for the incorporation of prior knowledge or preference about the abstractions through the prior, as is standard in Bayesian methods. In experiments, the selected abstractions are shown to accelerate RL performance relative to learning without abstraction, providing strong evidence for the effectiveness of the approach. In follow up work, Konidaris et al. [175] study the process of learning options given access to data generated by a demonstrator. Here, Konidaris et al. present CST, an algorithm for learning how to segment existing trajectories into options, again using the idea of target events from previous work. That is, given a list of target events, the agent will repeatedly try to formulate options for realizing these events when they occur. Given access to a trajectory generated by a demonstrator, the problem is to identify relevant change points throughout the trajectory that should be broken into target events, and hence, options. These ideas were later extended to the application of autonomous option discovery and use on a mobile-manipulator robot, which learned to pull levers to open doors and navigate through rooms [176]. For more on these approaches, see the dissertation by Konidaris [168].

More recently, Machado et al. [211, 213] develop a suite of algorithms for discovering options based around the graph Laplacian of the MDP's transition graph. The resulting options, called "eigenoptions", are behaviors defined by Proto-value functions (PVFs), a spectral approach to representation learning in RL [215]. The PVF captures, roughly, a reward-agnostic representation of the diffusion structure of transitions in an MDP. By similar reasoning to earlier work on subgoal discovery, the PVF may then be used to identify those regions of the state space that are distant in transition space. Machado et al. introduce the "eigenpurpose", a mechanism for defining an intrinsic reward function that increases as the agent moves toward disparate regions of the state space, as defined by the PVF. Eigenoptions, then, are those options whose policies are optimal with respect to this intrinsic reward function. In subsequent work, Machado et al. [213] extend these ideas to richer settings, allowing for the discovery of eigenoptions in stochastic MDPs and MDPs with high-dimensional state input. In a similar vein, Eysenbach et al. [100] propose learning options such that the diversity of the trajectories produced by the set of options is maximized, thereby generating options that may explore infrequently visited states. Option discovery has also been studied in select other settings, including in inverse RL [249] by Ranchod et al. [268], and in active learning [73] by da Silva et al. [78].

understanding the impact of options. Alongside the option discovery problem, it has also long been of interest to characterize how options impact the RL problem. As much of the aforementioned work shows, when the right options are used, RL algorithms can be empowered in a dramatic way. Jong et al. [149] address this question by taking a close look at the impact options have on RL. Much of the focus is on inspecting the empirical effect different options have on RL algorithms. In particular, Jong et al. set out to clarify when and why options can help learning. One experiment conducted contrasts the learning performance of traditional Q-learning with two variants: 1) Q-learning with subgoal-based options, and 2) Q-learning with experience replay [207], a mechanism for improving an agent's capacity to assign credit across long time horizons. The results suggest that both variant (1) and (2) perform nearly identically, suggesting that options are serving the same role that experience replay can. In other experiments, results suggest that options can negatively affect learning performance, or have no change at all. In particular, when options are paired with the principle of optimism under uncertainty for exploration, learning time is increased. The conclusion from this study is that it is not always straightforward that intuitively useful options will have the desired effect, and that sometimes they can even *negatively* impact RL.

options and transfer learning. Similarly to state abstractions, options have long been studied as a mechanism for facilitating transfer across tasks. A collection of options can summarize many things about previous experience, including good default policies for exploration, areas of an environment to pursue or avoid, or which actions should appear in sequence. Early on, Konidaris and Barto [172] developed a method for building options that are transferable between similar tasks. A core practice of the method is to separate the problem representation into two types, a global problem-space representation, and a local *agent-space* representation [171] that captures an agent-relative perspective on the environment's features. Each MDP the agent inhabits is characterized entirely by its problem-space representation, but the agent adopts the agent-space representation for use in transferring options across relevant situations. For instance, in facing several MDPs with keys and doors, a sensible agent-space option is one that collects the nearest agent-space key and takes it to the nearest door. In contrast, the problem-space representation defines these constituents by their absolute coordinates, and thus prohibits this same form of seamless transfer. Konidaris and Barto make use of given and learned options for accelerating learning in these key-door grid world MDPs to great effect.

Similarly, da Silva et al. [77] study the acquisition of *parameterized* options. That is, each option policy is associated with some parameter that allows the option to adapt flexibly to specific aspects of the given domain. The proposed approach is cast as a series of regression problems, given data gathered on a collection of training MDPs. In particular, this data is used to estimate the geometry of policy space in terms of some number of low-dimensional manifolds. Then, a series of regression problems are solved that give rise to the parameters of each option. Experiments are conducted in a challenging dart throwing game, with results providing substantial evidence for the effectiveness of the learned parameterized options.

Separately, Brunskill and Li [58] study the option discovery problem in the lifelong setting. In particular, adopt the perspectives of Probably Approximately Correct (PAC) learning introduced in the seminal work by Valiant [324]. The first result is a highly general form of a PAC-MDP [301] guarantee adapted to the case of learning with options. Recall that a PAC-MDP algorithm is one that is said to have a polynomial bound on the number of mistakes made by the algorithm with high probability. Brunskill and Li first present (their Theorem 1) a PAC-SMDP guarantee, suited to the case where an agent inhabits an SMDP. Using this result, an extension of R-Max, SMDP-R-Max, is developed and analyzed that is again well suited to the case of learning with options. Finally, these insights are applied to the lifelong learning setting in which an agent will face a series of MDPs sampled from the same distribution, each sharing a state-action space [344].

Thrun and Schwartz [317] performed one of the earliest studies of transferring abstract actions in RL. The work centeres around the SKILLS algorithm, which identifies correlated action sequences to group together into macro-operators called skills. In a similar vein, Pickett and Barto [261] propose Policy Blocks, an approach for generating useful options in lifelong RL. Given a set of optimal policies for some initial number of sampled MDPs, all possible policy combinations are enumerated and scored according to the size of their intersection with the solution policies. The n best options found this way are then kept alongside the primitive actions during learning. Topin et al. [321] extend Policy Blocks to Object-Oriented MDPs [93], thereby allowing for transfer to take place across tasks that share object structure. The main advantage is similar to the agent-space approach of Konidaris and Barto [172]: the components of each option can be defined in terms of objects and their relations that are guaranteed to exist across different tasks, thereby enabling high-level behaviors to be immediately applicable in new domains. MacGlashan [210] develops a suite of algorithms based on similar ideas, targeting the transfer of policies across tasks in continuous state settings, or when the tasks require different state representations.

Most recently, Barreto et al. [35] propose the option keyboard, a framework for combining options together into compositions of novel behavior. The main perspective is to consider each option as a single key on a keyboard—then, when confronted with a new task that is a composition of previous tasks [326], a combination of keys can compose a "chord" that can solve the new task. There are two key technical ideas underlying the option keyboard. First, a new perspective that views the process of combining distinct options as one of combining intrinsic reward functions that induce the options; Second, to make use of of *generalized policy improvement* (GPI) first presented by Barreto et al. [34], to inform the creation of new option policies. GPI is a mechanism for constructing a policy, π, from a collection of source policies, $\pi_1, \ldots, \pi_n$, such that π is guaranteed to be no worse than any in the collection. Hence, the option keyboard first learns a collection of options designed to maximize independent intrinsic reward streams (also treated as generic cumulants). Then, these different options can be easily composed to form a new option that corresponds to any linear combination of the chosen intrinsic reward streams.

options and exploration. As suggested by earlier work, options can, in ideal circumstances, dramatically improve the sample complexity of RL. Beyond the discovery and transfer work already surveyed, a recent line of work has explicitly concentrated on understanding how options impact exploration. For instance, Fruit and Lazaric [111] develop an algorithm for minimizing regret of learning options, building on SMDP-R-Max introduced Brunskill and Li [58]. Fruit and Lazaric propose SMDP-UCRL, an option-based RL algorithm that builds around UCRL [28]. The main result of the work provides upper and lower bounds on the regret of this algorithm, along with additional analysis proving which cases the regret of option learning can be lower than that of traditional RL. However, SMDP-UCRL requires prior knowledge in the form of the distribution of reward and expected run time of each option. In follow up work, Fruit and Lazaric [111] build on SMDP-UCRL with Free-SMDP-UCRL, which no longer requires this prior knowledge, but still matches the original regret bound up to an additive constant.

options and neural networks. The focus of much the present survey has been on options in the context of finite MDPs, with occasional extensions into continuous state spaces. In many cases, to confront the complexity of rich state spaces, deep neural networks are exploited for their power in function approximation [191]. To this end, Bacon et al. [30] established an elegant adaptation of the policy gradient theorem [312] to the problem of learning options. This result underlies the Option-Critic, a neural network architecture that supports end-to-end training of option policies and their termination conditions. Critically, learning options in the Option-Critic does not require any use of intrinsic rewards, which differentiates it from many of the option discovery approaches surveyed previously. Bacon et al. present strong empirical evidence that the Option-Critic can accelerate learning on challenging RL domains, even those with rich state spaces such as games from the Arcade Learning Environment (ALE) [39].

The Option-Critic work was later extended in several ways to cooperate with policy optimization methods [347], and to generate hierarchies of arbitrary depth [275]. Harb et al. [124] incorporate the notion of *deliberation cost* into the training objective, drawing on ideas from bounded rationality [286]. Here, as with the work of Şimşek and Barto [291], the goal is to clarify what is meant by a good option. Harb et al. answer this question by arguing that options may be used to help resource-bounded agents make decisions efficiently under harsh computational constraints. This perspective leads to the introduction of the deliberation cost that acts as a regularizer to encourage options that execute for longer periods of time, following similar reasoning to Mankowitz et al. [220]. Additionally, Tiwari and Thomas [319] offered pathways for incorporating the natural gradient [14] into the Option-Critic, yielding performance gains on benchmark tasks.

THE INITIATION AND TERMINATION OF OPTIONS. The Option-Critic is primarily concerned with learning the option policies and the termination condition, but assumes that the options are available everywhere (so $\mathcal{I}_o = \mathcal{S}$, for all o). Khetarpal et al. [160] extend the Option-Critic to also incorporate the a generalization of the initiation condition called the interest function of each option. Similarly to the Option-Critic, a gradient-based update rule is developed that is suitable for learning options, their interest functions, and their termination conditions, resulting in the Interest-Option-Critic. On the termination side, Harutyunyan et al. [126] propose options that terminate in an "off-policy" way, enabling unification of typical off-policy TD updates and option updates. This gives rise to a new option learning algorithm, $Q(\beta)$, that enables faster convergence by learning β in an off-policy manner. Similarly, Mankowitz et al. [220] study interrupting options, a means of improving a given set of options during planning. Their idea is to alter a given option's predefined termination condition based on information computed during planning. In this way, options can be iteratively improved via a Bellman like update (with interruption added). They demonstrate that these new options also lead to a contraction-mapping that ensures convergence of the option value function to a fixed point. Their main contribution is to build regularization into this framework by encouraging their operator to choose longer options. Later, Mankowitz et al. [221] propose Adaptive Skills Adaptive Partitions (ASAP), a framework for learning when to apply options and what they should do. ASAP is well suited for continuous state domains, and comes along with strong properties, including the correction of a misspecified model, and convergence to a locally optimal set of options and partitions.

OPTIONS AND PLANNING. Perhaps the greatest potential of options is their capacity to empower the planning capabilities of RL agents. Silver and Ciosek [285] develop compositional option models, which enable recursive nesting of option models through a generalization of the Bellman operator. Using this new operator, Silver and Ciosek present an algorithm designed for goal-based MDPS that estimates the transition model and termination condition for both the goal and subgoals simultaneously. These options and their models are shown to greatly accelerate planning on the classical planning problems of Towers of Hanoi and and a navigation task. Separately, Mann and Mannor [222], Mann et al. [223] analyze the convergence rate of approximate dynamic programming with and without options. The main result of the work proves that options can improve the convergence rate of approximate value iteration. The degree of improvement depends on how long the options run for, whether the value function is initialized pessimistically, and the value of the policies associated with the options.

In a different vein, Konidaris et al. [178, 179] and James et al. [140] use options to generate a well behaved abstract state representation, even if the underlying environment is continuous. Building on the chaining work discussed earlier, the main idea is again to draw on sequential composition from robotics [60]; that is, for each option, the termination condition of the option must lie entirely inside of the initiation condition of at least one other option. Using this idea, Konidaris et al. construct an abstract symbolic state representation based on the initiation-termination relationships in a given set of options. The result is an algorithm that can translate a complicated domain and a set of options into a domain that is representable in a classical planning language like Strips. Notably, the resulting discretized state space, along with the given options, is proven to have the property that any plan consisting of these options is *feasible*—that the sequence of operators is in fact executable. Tom Silver [320] study a related problem, focusing on how to learn options that may then be exploited by a Strips planner given new, more complicated goals than those seen during the learning of the options.

Indeed, existing research on options is broad, exciting, and growing. Many open fundamental questions remain, a few of which I will study in Part 3.

2.3.2 *Other Forms of Abstraction*

State and action abstraction on their own have each been of long lasting interest in the RL literature. Many of the surveyed methods were in fact designed to carry out both types of abstraction simultaneously, or focus on one while carrying out the other implicitly. This is particularly true in the broader study of hierarchical abstraction, which allows for the representation of phenomena at different levels of granularity.

While my review of prior work concentrates on methods that focus on just state or action abstraction, there has been a rich history on both hierarchical abstraction and joint state-action abstraction.

I will differentiate between the process of state-action abstraction from hierarchical abstraction as follows. A single application of a state *and* action abstraction will be defined as state-action abstraction, whereas a hierarchical abstraction is the repetition of $n > 1$ applications of state or action abstraction. Ultimately, this difference is purely for convenience, as a single state-action abstraction is effectively a shallow hierarchy.

STATE-ACTION ABSTRACTION. Together, state and action abstractions can distill complex problems into simple ones [151, 70]. As with the other types of abstraction, the literature on joint state-action abstraction is too broad to cover in its entirety. I instead highlight select works that are particularly relevant to the objectives of this dissertation.

Perhaps most relevant is are those approaches that inform state-action abstraction through *MDP homomorphisms* Ravindran and Barto [270, 271, 272, 273], Ravindran [269]. MDP homomorphisms form a compressed representation of a given MDP by collapsing state-action pairs that can be treated as equivalent. First, Ravindran and Barto [270] adapt homomorphisms as used in finite state automata [125] for application to MDPs, building on the model minimization techniques of Dean and Givan [86]. The main idea is to search through the state-action space for symmetries that allows for the formation of a functionally identical MDP. This tool is then exploited for the purpose of constructing homomorphisms and options that induce more compact representation of the original problem that is similar functionally.

In subsequent work, these ideas are expanded, building toward the more general family of SMDP-homomorphisms Ravindran and Barto [272] that allows for the discovery of symmetries between an MDP and SMDP. Later, Ravindran and Barto [273] generalize the previous frameworks to account for *similarity* of state-action pairs, rather than equivalence, a move similar to the one I make in Chapter 3. In this work, Ravindran and Barto introduce approximate MDP homomorphisms and prove the conditions under which they are guaranteed to preserve representation of good behavior—this result is one of the strongest of its kind, and heavily inspires the work of this dissertation. For more on MDP homomorphisms as a framework for abstraction, see the dissertation by Ravindran [269]. Lastly, in later years, Majeed and Hutter [218] extend the analysis of Ravindran and Barto to non-Markovian settings, proving the existence of several classes of value preserving homomorphisms—these classes closely resemble some of the families of state abstraction discussed earlier: for instance, one family studied groups histories of states together that induce similar value functions.

Mugan and Kuipers [242, 243, 244] develop a holistic approach called Qualitative Learner of Action and Perception (QLAP) that autonomously discovers state and action abstractions, even in MDPs with an underlying rich state and action space. The main idea is to first learn a *qualitative* state representation supposing that the agent can observe the value of different variables changing over time. By executing effectively random actions, a this qualitative representation learns measures such as *magnitude* and *change* variables of the observable quantities. Once enough data is collected, options are defined that explicitly modify the qualitative variables captured by the learned state representation. Experiments are conducted in the robotic simulator BREVE [164] in which a simulated humanoid robot is asked to manipulate objects on a tabletop. QLAP is able to successfully learn in a variety of tasks involving activities like pushing a block to a particular location picking up a block.

Finally, Bai and Russell [32] develop a Monte Carlo planning algorithm that incorporates state and action abstractions to efficiently solve the Partially Observable MDP [154] induced by these abstractions. That is, given an MDP with state space $\mathcal{S}$, the state-action abstraction induce a POMDP with observation space $\mathcal{S}_\phi$. The main results present guarantees on the performance of the algorithm: Theorem 1 shows that the value loss of their approach is bounded as a function of the state aggregation error [133], and Theorem 2 shows their algorithm converges to a recursively optimal policy for given state-action abstractions.

hierarchical abstraction. Hierarchical abstraction captures methods that form representations—either of state, action, or both—at different levels of granularity. Like the other forms of abstraction, hierarchy has a rich history in RL, dating back to early work on feudal learning by Dayan and Hinton [85], hierarchical Q-learning by Wiering and Schmidhuber [342], HAMs by Parr and Russell [259], and the MAXQ framework by Dietterich [88]. Since then, research has continued to establish the core principles of hierarchical abstractions, including the study of bayesian hierarchical RL [63], model-based hierarchical RL [148, 205, 108], model-free hierarchical RL [186, 106, 332, 246, 247, 200], learning hierarchies in imitation learning [190], from demonstration [233], for transfer [230, 232, 231], in multi-agent RL [116], and for planning [169, 121, 180, 345]. For more on the early works of hierarchical RL, see the survey by Barto and Mahadevan [37].

In summary: the literature on understanding abstraction and its role in RL is expansive, and far too broad to cover in this dissertation. I will return to direct comparisons where appropriate in subsequent chapters, again drawing the distinction between the four abstraction types: 1) state, 2) action, 3) state-action, and 4) hierarchical. A visual illustrating the intuitive difference between these four types of abstraction is presented in Figure 2.9.

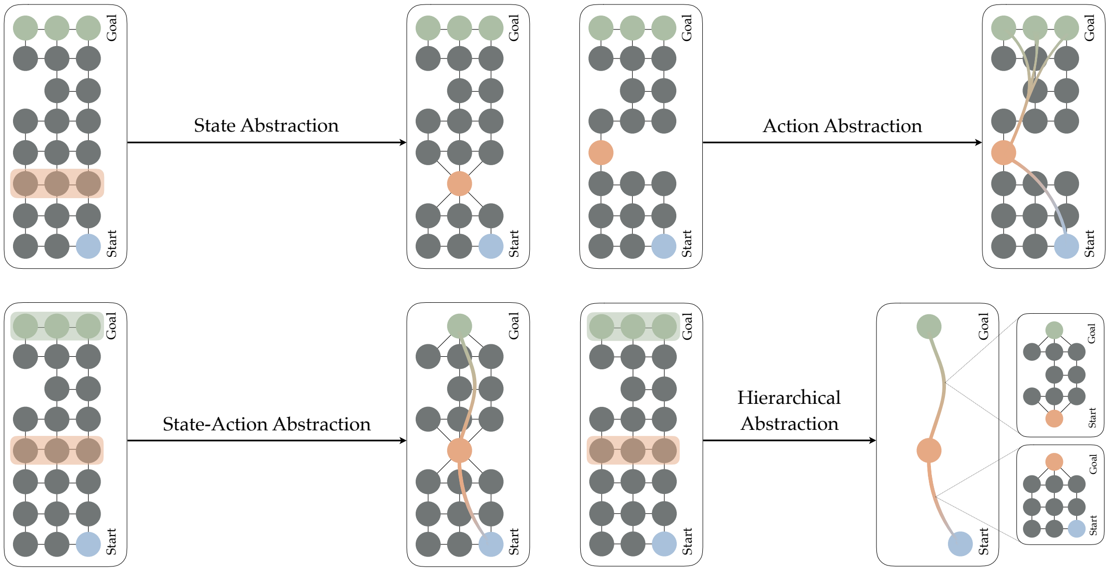


Figure 2.9: The different forms of abstraction in MDPs.

2.4 ABSTRACTION DESIDERATA

What is a *good* abstraction? A natural route to answering this question is to measure an abstraction's utility in terms of the quality of the representations the abstraction induces, with a focus on how these representations change the RL problem. These considerations could be made with respect to a particular choice of RL algorithm (or perhaps family, such as model-free), or, in contrast, may be agnostic to the choice of RL algorithm. Indeed, the right abstraction for one type of algorithm may be entirely useless to another. Similarly, some abstractions may be effective in certain kinds of environments—those that abstract aggressively may be most appropriate in highly uncompressed worlds, for instance. Further, it might be the case that the properties underlying useful state abstraction differ from action.

Throughout this dissertation, I advocate for three simple properties that both state and action abstractions should have regardless of the choice of environment or RL algorithm. For this reason, these desiderata are not intended to be exhaustive. There are surely other properties we might hope abstractions possess depending on the broader context, domain, or resource requirements. Still, it is useful to highlight an initial set of properties that capture at least some of what is meant by "good" abstraction—this is precisely the purpose of these desiderata. They are as follows.

(D1) Efficient-Creation: Computing or learning the abstraction should not be prohibitively difficult.

Measurement: The most natural evaluation for D1 is to provide sample bounds or computational complexity results that illustrate what resources are required to accurately learn or construct the abstraction.

(D2) Efficient Decision Making: An abstraction should enable efficient decision making. That is, planning or learning with a good abstraction should be faster than planning or learning without it.

Measurement: It is natural to measure such quantities in terms of the speed with which RL or its subproblems can be solved on MDPs of relevance. Concretely, an abstraction should lower the computational complexity of planning, or the sample complexity of RL. Throughout this dissertation I will sometimes use the *size* of the induced abstract model as a proxy for this measure, as most worst case sample complexity, computational complexity, and regret bounds depend on the size of the MDP being solved.

(D3) NEAR OPTIMALITY: An abstraction should enable agents to discover policies that solve the original problem to a satisfactory degree.

Measurement: I measure this property based on some variant of a bound on the *value loss* of an abstraction discussed throughout the chapter. Such a bound captures the suboptimality of the best abstract policy in the environmental MDP. In the case of state abstraction, value loss is defined in a straightforward way.

Definition 2.19. *A* ***value loss*** *bound of a state abstraction $\phi \in \Phi_{all}$ is any value $\tau \in \mathbb{R}$ such that*

$$\min_{\pi_\phi \in \Pi_\phi} \max_{s \in \mathcal{S}} V^*(s) - V^{\pi_\phi}(s) \leq \tau, \tag{2.36}$$

with V the ground MDP's value function and Π_ϕ the set of all policies over abstract states.

Note that to extend this definition to an action abstraction ω, we will require some extra machinery to define the ground value function under a policy over options, as it is not necessarily well defined due to the semi-Markovian nature of option execution. I remedy this fact in Chapter 9 with the introduction of joint state-action abstractions that are guaranteed to yield a policy class $\Pi_{\mathcal{O}_\phi}$ for which every entity has a well defined ground value function.

Other measures of optimality include *recursive* optimality and *hierarchical* optimality introduced by Dietterich [88]. The value loss mentioned above is a bound on the global optimality, and so is stronger than either recursive or hierarchical optimality.

Abstraction, broadly speaking, reduces the dimensionality of an entity. In the context of sequential decision making, abstraction reduces the representational complexity needed to support efficient learning in complex decision making problems. This process is captured by D2 (Efficient Decision Making) and D3 (Near-Optimality), with D1 further requiring that abstractions should be easy to create, given the computational and statistical budget available.

I take these three statements as guiding principles that help govern which abstractions to learn in RL. Collectively, they state the following:

$$\textit{Good abstractions for RL are } \underbrace{\textit{easy to discover}}_{\textit{(D1)}} \textit{ and } \underbrace{\textit{enable efficient learning}}_{\textit{(D2)}} \textit{ of } \underbrace{\textit{high value policies.}}_{\textit{(D3)}}$$

I now show that satisfying any *one* or any *two* desiderata is trivial. In each of the below remarks, I let M denote the environmental MDP and $\tilde{M}$ denote the abstract MDP induced by a pair $\phi, \mathcal{O}$, with components $(\mathcal{S}_\phi, \mathcal{O}, R_{\phi,\omega}, T_{\phi,\omega}, \gamma, \rho_0^\phi)$ and optimal policy $\tilde{\pi}^*$. The reward and transition functions resulting from ϕ and ω are defined as a straightforward combination of the MTM with the weighted average mechanism that underlies R_ϕ and T_ϕ. For more detail on the construction of this abstract MDP, and particularly the components $R_{\phi,\omega}, T_{\phi,\omega}$, see Chapter 9. Further, I let $I_{\phi,\omega}$ denote the identity abstraction, such that $I_{\phi,\omega}(M) = M$. More formally, $I_{\phi,\omega}$ is the pair (ϕ_I, ω_I), where $\phi_I : s \mapsto s$, and $\omega_I : \mathcal{A} \mapsto \mathcal{O}_\mathcal{A}$, with $\mathcal{O}_\mathcal{A}$ the primitive actions redefined as options as per the scheme described earlier in the chapter.

Remark 2.1. *All three desiderata are trivial to satisfy individually.*

Proof of D1.

> For D1 (efficient abstraction discovery), consider $I_{\phi,\omega}$. The abstraction is the identity function, and so requires no computation or learning. □

Proof of D2.

> For D2 (supports efficient decision making), suppose we replace the ground state and actions space with a single state and single action: $|\mathcal{S}_\phi| = |\mathcal{O}| = 1$. Clearly, such

a resulting MDP satisfies the first desiderata—it is trivial to plan or learn in the resulting MDP. □

Proof of D3.

For D3, consider $I_{\phi,\omega}$. The optimal policy for $M_{I_{\phi,\omega}}$ is exactly the optimal policy for M, thus preserving representation of high value policies. □

I now show that any pair of desiderata are trivial to satisfy.

Remark 2.2. *Any two desiderata are trivial to satisfy.*

Proof of D1 & D2.

For D1 and D3, we again consider the abstraction that induces an abstract MDP consisting of a single state and action. Planning and learning in this MDP are trivial, and the abstraction can be created without any computation or data. □

Proof of D2 & D3.

For D2 and D3, suppose we solve for the optimal policy π^* in M and abstract according to the π^*-irrelevance abstraction that clusters states based on optimal action in each state [147, 203]. The resulting abstract MDP is as small as can be without losing the optimal policy, per the result of Li et al. [203], and so may be said to support quick learning (under the assumption that MDP size may be treated as a proxy for learning difficulty). Further, the abstract policy $\tilde{\pi}^*$ is guaranteed to be optimal when applied in the ground: $\max_{s \in \mathcal{S}} V^*(s) - V^{\tilde{\pi}^*}(s) = 0$. □

Proof of D1 & D3.

For D1 and D3 we again invoke $I_{\phi,\omega}$. Clearly, the identity function is easy to compute and the optimal policy of $\tilde{M}$ will necessarily preserve optimality. □

The case of interest is an abstraction that satisfies *all three* desiderata. Really, though, none of the properties expressed by the desiderata are themselves boolean functions. They can each be satisfied to a different degree. Depending on the situation, it might be prudent to represent an near-optimal policy, or to ensure learning is as fast as possible. Thus, when we look for abstractions, our attention will be on those that achieve an appropriate trade off between the different properties.

In general, it is unclear whether there is a single optimal abstraction—it will largely depend on the broader objectives guiding the agent. Do we care about sample efficiency, safety, asymptotic performance, or reliability? Depending on these criteria, and on the resources available to the RL agent, different abstractions may be better suited to the given context. For this reason, much of the analysis in this dissertation is focused on understanding the interplay between these desiderata.

I now turn to the primary technical contributions of this work, beginning with state abstraction.

Part 2

STATE ABSTRACTION

3

APPROXIMATE STATE ABSTRACTION

This chapter is based on "Near Optimal Behavior via Approximate State Abstraction" [4], jointly led by D. Ellis Hershkowitz, also in collaboration with Michael L. Littman.

In this chapter I study which kinds of state abstraction are capable of preserving representation of good policies. Intuitively, abstraction of almost any kind throws away some amount of information. However, in RL, it is desirable (as per our third desiderata) that abstractions retain enough relevant information so as to allow RL agents to eventually learn to solve problems of interest. In light of this, I here introduce and analyze classes of *approximate state abstraction* that are guaranteed to support representation of near-optimal behavior, as pictured in Figure 3.1. Concretely, approximate state abstractions aggregate states based on degrees of similarity in terms of relevant functions like Q, R, and T. As we will see, these approximate state abstractions can jointly preserve representation of near-optimal behavioral policies while simultaneously reducing the size of the represented state space. In this way, this chapter is about state abstractions that satisfy desiderata D2 (efficient decision making) and D3 (near optimality). In the subsequent two chapters I show how to translate the main conceptual framework here introduced to accommodate all three desiderata.

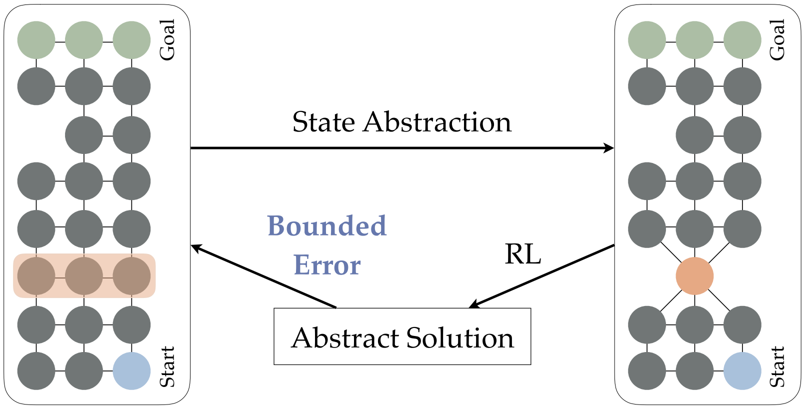


Figure 3.1: This chapter studies families of approximate state abstraction that induce abstract MDPs whose optimal policies have bounded value in the original MDP.

I will first prove that approximate state abstraction can still preserve near-optimal behavior. The main result (Theorem 3.1) shows that, by relaxing state aggregation criteria from equality to similarity, it is possible to achieve bounded value loss while offering three benefits. First, approximate abstractions make use of the kind of knowledge that we might expect a planning or learning algorithm to obtain without fully solving the MDP, and are thus more in line with the first desiderata. In contrast, exact abstractions often require solving for optimal behavior, thereby defeating the purpose of abstraction. Second, because of their relaxed criteria, approximate state abstractions can achieve greater degrees of compression than exact abstractions. This difference is particularly important in environments where no two states are identical. With a more compressed state space, many subprocedures critical to the overall effectiveness of an RL algorithm can be accelerated. For instance, planning in an abstract model $\hat{M}_\phi$ to compute an estimate of the optimal policy, $\hat{\pi}^*_\phi$, tends to be faster with a smaller state space. Third, because the state aggregation criteria are relaxed to near equality, approximate abstractions are able to tune the aggressiveness of abstraction by adjusting what they consider sufficiently similar states. I explicitly build an algorithmic framework around this idea in Chapter 5.

Furthermore, I empirically demonstrate the relationship between the degree of compression and error incurred on a variety of MDPs, illustrating a general trade off between compression and value-preservation that will serve as the focus of Chapter 5.

3.1 FOUR CLASSES OF APPROXIMATE STATE ABSTRACTION

I next describe four different types of approximate state abstractions that preserve near-optimal behavior by aggregating states on different criteria: $\phi_{Q^*_\varepsilon}$, on similar optimal Q-values; $\phi_{model,\varepsilon}$, on similarity of rewards and transitions; $\phi_{bolt,\varepsilon}$, on similarity of a Boltzmann distribution over optimal Q-values; and $\phi_{mult,\varepsilon}$, on similarity of a multinomial distribution over optimal Q-values. These four predicates are defined as follows.

$$p_{Q^*_\varepsilon}(s_1, s_2) \equiv \max_{a \in \mathcal{A}} |Q^*(s_1, a) - Q^*(s_2, a)| \leq \varepsilon, \tag{3.1}$$

$$p_{model,\varepsilon}(s_1, s_2) \equiv \max_{a \in \mathcal{A}} |R(s_1, a) - R(s_2, a)| \leq \varepsilon_R \text{ and}$$

$$\forall_{a \in \mathcal{A}, s'_\phi \in \mathcal{S}_\phi, s' \in s'_\phi} : \left| T(s' \mid s_1, a) - T(s' \mid s_2, a) \right| \leq \varepsilon_R, \tag{3.2}$$

$$p_{mult}(s_1, s_2) \equiv \max_{a \in \mathcal{A}} \left| \frac{Q^*(s_1, a) - Q^*(s_2, a)}{\sum_b Q^*(s, b)} \right| \leq \varepsilon, \tag{3.3}$$

$$p_{bolt}(s_1, s_2) \equiv \max_{a \in \mathcal{A}} \left| \frac{e^{Q^*(s_1, a)} - e^{Q^*(s_2, a)}}{\sum_b e^{Q^*(s, b)}} \right| \leq \varepsilon. \tag{3.4}$$

I now introduce the main theorem of this chapter that shows for each of the four classes of approximate abstraction, for *any* finite MDP, the abstracted model preserves near-optimal behavior. More formally:

Theorem 3.1. *There exist at least four types of approximate state aggregation functions, $\phi_{Q^*_\varepsilon}$, $\phi_{model,\varepsilon}$, ϕ_{mult} and ϕ_{bolt}, for which the optimal policy in the resulting abstract MDP, applied to the environmental MDP, has suboptimality bounded by a function of ε:*

$$\forall_{s \in \mathcal{S}} : V^*(s) - V^{\pi^*_\phi}(s) \leq 2\varepsilon \textsc{RMax} \eta_p, \tag{3.5}$$

where η_p depends on the predicate associated with abstraction function types:

$$\eta_{Q^*} = \frac{1}{(1-\gamma)^2}, \tag{3.6}$$

$$\eta_{model} = \frac{1 + \gamma\left(|\mathcal{S}_\phi| - 1\right)}{(1-\gamma)^3}, \tag{3.7}$$

$$\eta_{bolt} = \frac{\left(\frac{|\mathcal{A}|}{1-\gamma} + \varepsilon k_{bolt} + k_{bolt}\right)}{(1-\gamma)^2}, \tag{3.8}$$

$$\eta_{mult} = \frac{\left(\frac{|\mathcal{A}|}{1-\gamma} + k_{mult}\right)}{(1-\gamma)^2}. \tag{3.9}$$

For η_{bolt} and η_{mult}, I also assume that the difference in the normalizing terms of each distribution is each bounded by some non-negative constant, $k_{mult} \in \mathbb{R}_{\geq 0}$, $k_{bolt} \in \mathbb{R}_{\geq 0}$ of ε:

$$\left|\sum_i Q^*(s_1, a_i) - \sum_j Q^*(s_2, a_j)\right| \leq k_{mult}\varepsilon, \tag{3.10}$$

$$\left|\sum_i e^{Q^*(s_1,a_i)} - \sum_j e^{Q^*(s_2,a_j)}\right| \leq k_{bolt}\varepsilon. \tag{3.11}$$

Further, I note that η_{model} of the original theorem has since been improved by a factor of $1/(1-\gamma)$ through Lemma 4 by Taïga et al. [313].

Naturally, the value bound of Equation 3.5 is meaningless for $2\varepsilon\eta_f \geq \frac{\text{RMax}}{1-\gamma} = \frac{1}{1-\gamma}$, since this is the maximum possible value achievable in any MDP (assuming $\text{RMin} = 0$). In light of this, observe that for $\varepsilon = 0$, all of the above bounds are exactly 0. Any value of ε spanning between these two points achieves different degrees of abstraction, with different degrees of bounded loss. The degree of approximation (choice of ε) changes the compression-value trade off made by the abstraction. In the closing section of the chapter (Section 3.3), I present an empirical study of this relationship in a variety of benchmark MDPs. In each experiment the finding is consistent: as ε increases, the size of the abstract state space is reduced, and the value loss increases, though the rate at which this trade off is made differs across tasks. I return to a more technical treatment of this trade off in Chapter 5.

3.2 ANALYSIS

I now introduce each approximate state aggregation family in more technical detail and prove the main result of the chapter. The proof strategy consists of proving a specific value loss bound for each of the four function types.

Let us consider an approximate version of Li et al.'s [203] ϕ_{Q^*}. In this abstraction, states are aggregated together when their optimal Q-values are within ε.

Definition 3.1. *An* ***approximate Q* abstraction*** *($\phi_{Q^*_\varepsilon}$) has the following form:*

$$\phi_{Q^*_\varepsilon}(s_1) = \phi_{Q^*_\varepsilon}(s_2) \implies \forall_{a\in\mathcal{A}} : |Q^*(s_1, a) - Q^*(s_2, a)| \leq \varepsilon. \tag{3.12}$$

Lemma 3.1. *When a $\phi_{Q^*_\varepsilon}$ type abstraction is used to create the abstract MDP:*

$$\forall_{s\in\mathcal{S}} : V^{\pi^*}(s) - V^{\pi_\phi}(s) \leq \frac{2\varepsilon \text{RM}\textsc{ax}}{(1-\gamma)^2}. \tag{3.13}$$

Proof of Lemma 3.1.

We first demonstrate that Q-values in the abstract MDP are close to Q-values in the ground MDP (Claim 3.1). We next use Claim 3.1 to demonstrate that the optimal action in the abstract MDP is nearly optimal in the ground MDP (Claim 3.2). Lastly, Claim 3.3 shows that Lemma 3.1 follows from Claim 3.2.

Claim 3.1. *Optimal Q-values in the abstract MDP closely resemble optimal Q-values in the ground MDP:*

$$\forall_{s\in\mathcal{S}, a\in\mathcal{A}} : |Q^*(s, a) - Q^*_\phi(\phi_{Q^*_\varepsilon}(s), a)| \leq \frac{\varepsilon}{1-\gamma}. \tag{3.14}$$

Consider a non-Markovian decision process of the same form as an MDP, $M_N = \langle \mathcal{S}_N, \mathcal{A}, R_N, T_N, \gamma, \rho_0 \rangle$, parameterized by non-negative integer an $N \in \mathbb{Z}_{\geq 0}$, such that for the first N time steps the reward function, the transition function, and state space are those of the abstract MDP, M_ϕ, and after N time steps the reward function, transition dynamics and state spaces are those of M. Thus,

$$\mathcal{S}_N = \begin{cases} \mathcal{S} & \text{if } N = 0 \\ \mathcal{S}_\phi & \text{o/w} \end{cases} \tag{3.15}$$

$$R_N(s,a) = \begin{cases} R(s,a) & \text{if } N = 0 \\ R_\phi(s,a) & \text{o/w} \end{cases} \tag{3.16}$$

$$T_N(s' \mid s,a) = \begin{cases} T_\phi(s' \mid s,a) & \text{if } N = 0 \\ \sum_{g \in G(s)} T_\phi(s' \mid g,a)w(g) & \text{if } N = 1 \\ T_\phi(s' \mid s,a) & \text{o/w} \end{cases} \tag{3.17}$$

The Q-value of state s in $\mathcal{S}_N$ for action a is:

$$Q_N^*(s,a) = \begin{cases} Q^*(s,a) & \text{if } N = 0 \\ \sum_{g \in G(s)} Q^*(g,a)w(g) & \text{if } N = 1 \\ R_\phi(s,a) + \gamma \sum_{s_\phi' \in \mathcal{S}_\phi} T_\phi(s_\phi \mid s,a) \max_{a'} Q_{N-1}^*(s_\phi', a'). & \text{o/w} \end{cases} \tag{3.18}$$

We proceed by induction on N to show that:

$$\forall_{N,s \in \mathcal{S},a} |Q_N^*(s_N,a) - Q^*(s,a)| \leq \sum_{n=0}^{N-1} \varepsilon\gamma^n, \tag{3.19}$$

where $s_N = s$ if $N = 0$ and $s_N = \phi_{Q_\varepsilon^*}(s)$ otherwise.

(Base Case: N* = 0*)

When $N = 0$, $Q^*_N = Q^*$, so this base case trivially follows.

(Base Case: N = 1)

By definition of Q^*_N, for any s, a,

$$Q^*_1(s,a) = \sum_{g \in G(s)} \left[Q^*(g,a)w(g)\right]. \tag{3.20}$$

Since all co-aggregated states have Q-values within ε of one another and $w(g)$ induces a convex combination,

$$Q^*_1(s_N,a) \leq \varepsilon\gamma^n + \varepsilon + Q^*(s,a) \tag{3.21}$$

$$\therefore |Q^*_1(s_N,a) - Q^*(s,a)| \leq \sum_{n=0}^{1} \varepsilon\gamma^n. \tag{3.22}$$

(Inductive Case: N > 1)

We assume as our inductive hypothesis that:

$$\forall_{s \in \mathcal{S},a} |Q^*_{N-1}(s_N,a) - Q^*(s,a)| \leq \sum_{n=0}^{N-2} \varepsilon\gamma^n. \tag{3.23}$$

Consider a fixed but arbitrary state, $s \in \mathcal{S}$, and fixed but arbitrary action a. Since $N > 1$, s_N is $\phi_{Q^*_\varepsilon}(s)$. By definition of $Q^*_T(s_N,a)$, R_ϕ, T_ϕ:

$$Q^*_N(s_N,a) = \sum_{s \in G(s_N)} w(g) \times \left[R(s,a) + \gamma \sum_{s' \in \mathcal{S}} T_\phi(s' \mid s,a) \max_{a'} Q^*_{N-1}(s',a')\right].$$

Applying our inductive hypothesis yields:

$$Q^*_N(s_N,a) \leq \sum_{s \in G(s_N)} w(g) \times \left[R(s,a) \ + \gamma \sum_{s' \in \mathcal{S}} T(s' \mid s,a) \max_{a'}(Q^*(s',a') + \sum_{n=0}^{N-2} \varepsilon\gamma^n)\right].$$

Then,

$$Q_N^*(s_N, a) \leq \gamma \sum_{n=0}^{N-2} \varepsilon\gamma^n + \sum_{s \in G(s_N)} \left[w(s)\ Q^*(s,a) \right]. \tag{3.24}$$

Since all aggregated states have Q-values within ε of one another:

$$Q_N^*(s_N, a) \leq \gamma \sum_{n=0}^{N-2} \varepsilon\gamma^n + \varepsilon + Q^*(s,a), \tag{3.25}$$

$$\therefore\ |Q_N^*(s_N, a) - Q^*(s,a)| \leq \gamma \sum_{n=0}^{N-1} \varepsilon\gamma^n. \tag{3.26}$$

Since s is arbitrary we conclude Equation 3.19. As $N \to \infty$, $\sum_{n=0}^{N-1} \varepsilon\gamma^n \to \frac{\varepsilon}{1-\gamma}$ by the sum of infinite geometric series and $Q_N^* \to Q_\phi^*$. Thus, Equation 3.19 yields Claim 3.1.

Claim 3.2. *Consider a fixed but arbitrary state, $s \in \mathcal{S}$ and its corresponding abstract state $s_\phi = \phi_{Q^*\varepsilon}(s)$. Let a^* stand for the optimal action in s, and a_ϕ^* stand for the optimal action in s_ϕ:*

$$a^* = \arg\max_a Q^*(s,a), \quad a_\phi^* = \arg\max_a Q_\phi^*(s_\phi, a). \tag{3.27}$$

The optimal action in the abstract MDP has a Q-value in the ground MDP that is nearly optimal:

$$V^*(s) \leq Q^*(s, a_\phi^*) + \frac{2\varepsilon}{1-\gamma}. \tag{3.28}$$

By Claim 3.1,

$$V^*(s) = Q^*(s,a^*) \leq Q_\phi^*(s_\phi, a^*) + \frac{\varepsilon}{1-\gamma}. \tag{3.29}$$

By the definition of a_ϕ^*, we know that

$$Q_\phi^*(s_\phi, a^*) + \frac{\varepsilon}{1-\gamma} \leq Q_\phi^*(s_\phi, a_\phi^*) + \frac{\varepsilon}{1-\gamma}. \tag{3.30}$$

Lastly, again by Claim 3.1, we know

$$Q^*_\phi(s_\phi, a^*_\phi) + \frac{\varepsilon}{1-\gamma} \leq Q^*(s, a^*_\phi) + \frac{2\varepsilon}{1-\gamma}. \tag{3.31}$$

Therefore, Equation 3.28 follows.

Claim 3.3. *Lemma 3.1 follows from Claim 3.2.*

Consider the policy for M of following the optimal abstract policy π^*_ϕ for t steps and then following the optimal ground policy π^* in M:

$$\pi_{\phi,n}(s) = \begin{cases} \pi^*(s) & \text{if } n = 0 \\ \\ \pi_\phi(s) & \text{if } n > 0 \end{cases} \tag{3.32}$$

For $n > 0$, the value of this policy for $s \in \mathcal{S}$ in the ground MDP is:

$$V^{\pi_{\phi,n}}(s) = R(s, \pi_{\phi,n}(s)) + \ \gamma \sum_{s' \in \mathcal{S}} T_\phi(s, a, s') V^{\pi_{\phi,n-1}}(s').$$

For $n = 0$, $V^{\pi_{\phi,n}}(s)$ is simply $V^*(s)$.

We now show by induction on t that

$$\forall_{n \in \mathbb{Z}_{\geq 0}, s \in \mathcal{S}} : V^*(s) \leq V^{\pi_{\phi,n}}(s) + \sum_{i=0}^{n} \gamma^i \frac{2\varepsilon}{1-\gamma}. \tag{3.33}$$

(Base Case: $n = 0$)

By definition, when $n = 0$, $V^{\pi_{\phi,n}} = V$, so our bound trivially holds in this case.

(Inductive Case: $n > 0$)

Consider a fixed but arbitrary state $s \in \mathcal{S}$. We assume for our inductive hypothesis that

$$V^*(s) \leq V^{\pi_{\phi,n-1}}(s) + \sum_{i=0}^{n-1} \gamma^i \frac{2\varepsilon}{1-\gamma}. \tag{3.34}$$

By definition,

$$V^{\pi_{\phi,n}}(s) = R(s, \pi_{\phi,n}(s)) + \gamma \sum_{s' \in \mathcal{S}} T(s' \mid s, a) V^{\pi_{\phi,n-1}}(s').$$

Applying our inductive hypothesis yields:

$$V^{\pi_{\phi,n}}(s) \geq R(s, \pi_{\phi,n}(s)) + \gamma \sum_{s' \in \mathcal{S}} T(s' \mid s, \pi_{\phi,n}(s)) \left(V^*(s') - \sum_{i=0}^{n-1} \gamma^i \frac{2\varepsilon}{1-\gamma} \right).$$

Therefore,

$$V^{\pi_{\phi,n}}(s) \geq -\gamma \sum_{i=0}^{n-1} \gamma^i \frac{2\varepsilon}{1-\gamma} + Q^*(s, \pi_{\phi,n}(s)). \tag{3.35}$$

Applying Claim 3.2 yields:

$$V^{\pi_{\phi,n}}(s) \geq -\gamma \sum_{i=0}^{n-1} \gamma^i \frac{2\varepsilon}{1-\gamma} - \frac{2\varepsilon}{1-\gamma} + V^*(s) \tag{3.36}$$

$$\therefore V^*(s) \leq V^{\pi_{\phi,n}}(s) + \sum_{i=0}^{n} \gamma^i \frac{2\varepsilon}{1-\gamma}. \tag{3.37}$$

Since s was arbitrary, we conclude that our bound holds for all states in $\mathcal{S}$ for the inductive case. Thus, from our base case and induction, we conclude that

$$\forall_{n \in \mathbb{N}, s \in \mathcal{S}} : V^{\pi^*}(s) \leq V^{\pi_{\phi,n}}(s) + \sum_{i=0}^{n} \gamma^i \frac{2\varepsilon}{1-\gamma}. \tag{3.38}$$

Note that as $n \to \infty$, $\sum_{i=0}^{n} \gamma^i \frac{2\varepsilon}{1-\gamma} \to \frac{2\varepsilon}{(1-\gamma)^2}$ by the sum of infinite geometric series and $\pi_{\phi,n} \to \pi_\phi$. Thus, we conclude Lemma 3.1. □

Now, consider an approximate version of Li et al. [203]'s ϕ_{model}, where states are aggregated together when their rewards and transitions are within ε.

Definition 3.2. *We let $\phi_{model,\varepsilon}$ define a type of state abstraction that, for fixed $\varepsilon_R \in [\textsc{RMin}, \textsc{RMax}]$ and $\varepsilon_T \in [0,1]$ satisfies:*

$$\phi_{model,\varepsilon}(s_1) = \phi_{model,\varepsilon}(s_2) \implies \forall_{a \in \mathcal{A}} : |R(s_1,a) - R(s_2,a)| \leq \varepsilon_R$$

and

$$\forall_{s_\phi \in \mathcal{S}_\phi, a \in \mathcal{A}} : \left| \sum_{s' \in s_\phi} \left[T(s' \mid s_1, a) - T(s' \mid s_2, a) \right] \right| \leq \varepsilon_T. \quad (3.39)$$

Lemma 3.2. *When $\mathcal{S}_\phi$ is created using a $\phi_{model,\varepsilon}$ type:*

$$\forall_{s \in \mathcal{S}} : V^{\pi^*}(s) - V^{\pi_\phi}(s) \leq \textsc{RMax} \frac{2\varepsilon_R + 2\gamma\varepsilon_T\left(|\mathcal{S}_\phi| - 1\right)}{(1-\gamma)^3}. \quad (3.40)$$

Proof of Lemma 3.2.

Let B be the maximum Q-value difference between any pair of ground states in the same abstract state for $\phi_{model,\varepsilon}$:

$$B = \max_{s_1,s_2,a} |Q^*(s_1,a) - Q^*(s_2,a)|,$$

where $s_1, s_2 \in s_\phi$. First, we expand:

$$B = \max_{s_1,s_2,a} \left| R(s_1,a) - R(s_2,a) \ + \gamma \sum_{s' \in \mathcal{S}} \left[(T_\phi(s' \mid s_1,a) - T_\phi(s' \mid s_2,a)) \max_{a'} Q^*(s',a') \right] \right| \quad (3.41)$$

Since difference of rewards is bounded by ε_R:

$$B \leq \max_{s_1,s_2,a} \left| \varepsilon_R + \gamma \sum_{s_\phi \in \mathcal{S}_\phi} \sum_{s' \in s_\phi} \left[(T(s' \mid s_1,a) \ - T(s' \mid s_2,a)) \max_{a'} Q^*(s',a') \right] \right|. \quad (3.42)$$

By similarity of transitions under $\phi_{model,\varepsilon}$:

$$B \leq \varepsilon_R + \gamma \text{QM}\textsc{ax} \sum_{s_\phi \in \mathcal{S}_\phi} \varepsilon_T \leq \varepsilon_R + \gamma |\mathcal{S}| \varepsilon_T \text{QM}\textsc{ax}. \tag{3.43}$$

Recall that QMAX $\leq \frac{\text{RM}\textsc{ax}}{1-\gamma}$. Hence:

$$B \leq \text{RM}\textsc{ax} \frac{\varepsilon_R + \gamma(|\mathcal{S}| - 1)\varepsilon_T}{1-\gamma}. \tag{3.44}$$

Since the Q-values of ground states grouped under $\phi_{model,\varepsilon}$ are strictly less than B, we can understand $\phi_{model,\varepsilon}$ as a type of $\phi_{Q^*,B}$. Applying Lemma 3.1 yields Lemma 3.2. □

As mentioned previously, the above bound has since been tightened by Lemma 4 of Taïga et al. [313]. The new bound is

$$\forall_{s \in \mathcal{S}} : V^{\pi^*}(s) - V^{\pi_\phi}(s) \leq \text{RM}\textsc{ax} \frac{2\varepsilon_R + 2\gamma\varepsilon_T \left(|\mathcal{S}_\phi| - 1\right)}{(1-\gamma)^2}. \tag{3.45}$$

Boltzmann over Optimal Q

Next we introduce $\phi_{bolt,\varepsilon}$, which aggregates states with similar Boltzmann distributions on Q-values. This type of abstractions is appealing as a Boltzmann distribution over Q-values often shows up in exploration methods [309]. We find this type particularly interesting for abstraction purposes as, unlike $\phi_{Q^*_\varepsilon}$, it allows for aggregation when Q-value ratios are similar but their magnitudes are different.

Definition 3.3. *We let $\phi_{bolt,\varepsilon}$ define a type of state abstraction that, for fixed ε, satisfies:*

$$\phi_{bolt,\varepsilon}(s_1) = \phi_{bolt,\varepsilon}(s_2) \implies \forall_a \left| \frac{e^{Q^*(s_1,a)}}{\sum_b e^{Q^*(s_1,b)}} - \frac{e^{Q^*(s_2,a)}}{\sum_b e^{Q^*(s_2,b)}} \right| \leq \varepsilon. \tag{3.46}$$

We also assume that the difference in normalizing terms is bounded by some non-negative constant, $k_{\text{bolt}} \in \mathbb{R}_{\geq 0}$, of ε:

$$\left| \sum_{b \in \mathcal{A}} e^{Q^*(s_1,b)} - \sum_{b \in \mathcal{A}} e^{Q^*(s_2,b)} \right| \leq k_{\text{bolt}} \times \varepsilon. \tag{3.47}$$

Lemma 3.3. *When $\mathcal{S}_\phi$ is created using a function of the $\phi_{bolt,\varepsilon}$ type, for some non-negative constant $k \in \mathbb{R}$:*

$$\forall_{s \in \mathcal{S}} : V^{\pi^*}(s) - V^{\pi_\phi}(s) \leq \text{RMax}\frac{2\varepsilon\left(\frac{|\mathcal{A}|}{1-\gamma} + \varepsilon k_{bolt} + k_{bolt}\right)}{(1-\gamma)^2}. \tag{3.48}$$

Proof of Lemma 3.3.

To prove the result, we make use of the approximation for e^x, with δ error:

$$e^x = 1 + x + \delta \approx 1 + x. \tag{3.49}$$

We let δ_1 denote the error in approximating $e^{Q^*(s_1,a)}$ and δ_2 denote the error in approximating $e^{Q^*(s_2,a)}$.

By the approximation in Equation 3.49 and the assumption in Equation 3.47:

$$\left| \frac{1 + Q^*(s_1,a) + \delta_1}{\sum_j e^{Q^*(s_1,a_j)}} - \frac{1 + Q^*(s_2,a) + \delta_2}{\sum_j e^{Q^*(s_1,a_j)} \underbrace{\pm k\varepsilon}_{ⓐ}} \right| \leq \varepsilon \tag{3.50}$$

Either term ⓐ is positive or negative. First suppose the former. It follows by algebra that:

$$-\varepsilon \leq \frac{1 + Q^*(s_1,a) + \delta_1}{\sum_j e^{Q^*(s_1,a_j)}} - \frac{1 + Q^*(s_2,a) + \delta_2}{\sum_j e^{Q^*(s_1,a_j)} + \varepsilon k_{\text{bolt}}} \leq \varepsilon \tag{3.51}$$

Moving terms:

$$\begin{aligned}
& -\varepsilon\left(k_{\text{bolt}}\varepsilon + \sum_j e^{Q^*(s_1,a_j)}\right) - \delta_1 + \delta_2 \leq \\
& \qquad \varepsilon k_{\text{bolt}}\left(\frac{1 + Q^*(s_1,a) + \delta_1}{\sum_j e^{Q^*(s_1,a_j)}}\right) + Q^*(s_1,a) - Q^*(s_2,a) \leq \\
& \qquad\qquad \varepsilon\left(\varepsilon k_{\text{bolt}} + \sum_j e^{Q^*(s_1,a_j)}\right) - \delta_1 + \delta_2
\end{aligned} \tag{3.52}$$

When ⓐ is the negative case, it follows that:

$$-\varepsilon \leq \frac{1 + Q^*(s_1,a) + \delta_1}{\sum_j e^{Q^*(s_1,a_j)}} - \frac{1 + Q^*(s_2,a) + \delta_2}{\sum_j e^{Q^*(s_1,a_j)} - \varepsilon k_{\text{bolt}}} \leq \varepsilon \tag{3.53}$$

By similar algebra that yielded Equation 3.52:

$$\begin{aligned}
& -\varepsilon\left(-\varepsilon k_{\text{bolt}} + \sum_j e^{Q^*(s_1,a_j)}\right) - \delta_1 + \delta_2 \leq \\
& \qquad -k_{\text{bolt}}\varepsilon\left(\frac{1 + Q^*(s_1,a) + \delta_1}{\sum_j e^{Q^*(s_1,a_j)}}\right) + Q^*(s_1,a) - Q^*(s_2,a) \leq \\
& \qquad\qquad \varepsilon\left(\varepsilon k_{\text{bolt}} + \sum_j e^{Q^*(s_1,a_j)}\right) - \delta_1 + \delta_2
\end{aligned} \tag{3.54}$$

Combining Equation 3.52 and Equation 3.54 results in:

$$|Q^*(s_1,a) - Q^*(s_2,a)| \leq \varepsilon\left(\frac{|\mathcal{A}|}{1-\gamma} + \varepsilon k_{\text{bolt}} + k_{\text{bolt}}\right). \tag{3.55}$$

Consequently, we can consider $\phi_{bolt,\varepsilon}$ as a special case of the $\phi_{Q^*_\varepsilon}$ type, with Q^* similarity of

$$B = \varepsilon\left(\frac{|\mathcal{A}|}{1-\gamma} + \varepsilon k_{\text{bolt}} + k_{\text{bolt}}\right). \tag{3.56}$$

Lemma 3.3 then follows from Lemma 3.1. □

Multinomial over Optimal Q: $\phi_{mult,\varepsilon}$

Lastly, I consider a variant derived from a multinomial distribution over Q^* for similar reasons to the Boltzmann distribution, with the multinomial offering the added appeal of simplicity.

Definition 3.4. *Let $\phi_{mult,\varepsilon}$ define a type of abstraction that, for fixed ε, satisfies*

$$\phi_{mult,\varepsilon}(s_1) = \phi_{mult,\varepsilon}(s_2) \implies \forall_a \left| \frac{Q^*(s_1,a)}{\sum_b Q^*(s_1,b)} - \frac{Q^*(s_1,a)}{\sum_b Q^*(s_1,b)} \right| \le \varepsilon. \tag{3.57}$$

As with the Boltzmann class, I again assume that the difference in normalizing terms is bounded by some non-negative constant, $k_{\text{mult}} \in \mathbb{R}_{\geq 0}$, of ε:

$$\left| \sum_i Q^*(s_1,a_i) - \sum_j Q^*(s_2,a_j) \right| \le k_{\text{mult}} \times \varepsilon. \tag{3.58}$$

Lemma 3.4. *When $\mathcal{S}_\phi$ is created using a function of the $\phi_{mult,\varepsilon}$ type, for some non-negative constant $k_{mult} \in \mathbb{R}$:*

$$\forall_{s \in \mathcal{S}_M} V^{\pi^*}(s) - V^{\pi_\phi}(s) \le \text{RMax} \frac{2\varepsilon\left(\frac{|\mathcal{A}|}{1-\gamma} + k_{mult}\right)}{(1-\gamma)^2}. \tag{3.59}$$

Proof of Lemma 3.4.

> The proof follows an identical strategy to that of Lemma 3.3, but without the approximation $e^x \approx 1 + x$. □

3.3 EXPERIMENTS

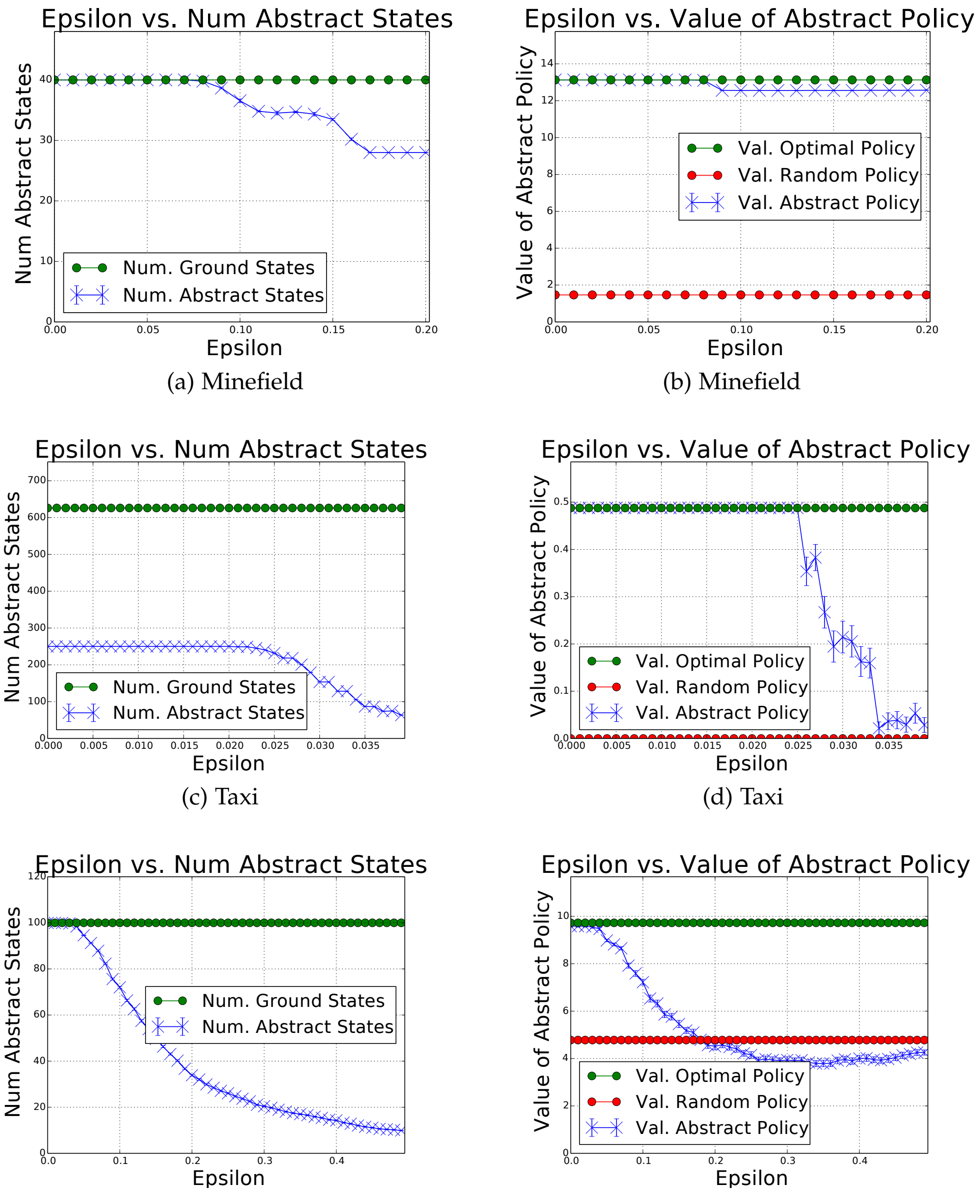
I next conduct experiments to highlight the impact state abstractions of the $\phi_{Q^*_\varepsilon}$ type can have. I provide results for only $\phi_{Q^*_\varepsilon}$ because, as per Lemma 3.2, Equation 3.3, and Equation 3.4, the other three functions are reducible to particular $\phi_{Q^*_\varepsilon}$ functions. The code for running these experiments is publicly available.[3.1]

I first explicitly construct an approximate state abstraction instance by approximating Q^* through dynamic programming, then greedily aggregating ground states into abstract states that satisfy the $\phi_{Q^*_\varepsilon}$ criteria. Since this approach represents an order-dependent approximation to the maximum amount of abstraction possible, I randomize the order in which states are considered across trials. Every ground state is equally weighted in its abstract state (that is, $w(s) = 1/|\phi(s)|$).

For each domain, I report the average number of abstract states and the value of the best abstract policy in the ground MDP, each with 95% confidence intervals. First, I compare the number of states in the abstract MDP for different values of ε, shown in the left column of Figure 3.2. Second, I report the value under the abstract policy of the initial ground state, shown in the right column of Figure 3.2. In the Taxi and Random domains, 200 trials were run for each data point, whereas 20 trials were sufficient in Minefield.

These empirical results corroborate the main finding of this chapter—approximate state abstractions can decrease state space size while retaining bounded error. In Minefield, observe that as ε increases from 0, the number of abstract states is reduced, and optimal behavior is very nearly maintained. Similarly, in Taxi, when ε is between .02 and .025, we observe a reduction in the number of states in the abstract MDP while value is fully maintained. For values of $\varepsilon \geq .025$, increased reduction in state space size comes at a cost of value. Lastly, as ε is increased in the Random domain, there is a smooth reduction in the number of abstract states with a corresponding cost in the value of the derived policy.

[3.1] https://github.com/david-abel/state_abstraction



(a) Minefield

(b) Minefield

(c) Taxi

(d) Taxi

(e) Random

(f) Random

Figure 3.2: ε vs. Num States (left) and ε vs. Abstract Policy Value (right).

When $\varepsilon = 0$, there is no reduction in state space size whatsoever (the ground MDP has 100 states), because no two states have identical optimal Q-values.

These experimental results also highlight a noteworthy characteristic of approximate state abstraction in goal-based MDPs. Taxi exhibits relative stability in state space size and behavior for ε up to .02, at which point both fall off dramatically. I attribute the sudden fall off of these quantities to the goal-based nature of the domain; once information critical for achieving optimal behavior is lost in the state aggregation, solving the goal—and so acquiring any reward—is impossible. Conversely, in the Random domain, a great deal of near optimal policies are available to the agent. Thus, even as the information for optimal behavior is lost, there are many near optimal policies available to the agent that remain available.

To summarize, in this chapter I motivated and introduced the family of approximate state abstraction. Each of the four types analyzed is guaranteed to preserve representation of good behavioral policies, with the degree of suboptimality induced a direct function of the amount of knowledge used to inform the state abstraction.

4

STATE ABSTRACTION IN LIFELONG RL

This chapter is based on 'State Abstractions for Lifelong Reinforcement Learning" [6], joint with Dilip Arumugam, Lucas Lehnert, and Michael L. Littman.

In the previous chapter, I motivated the family of *approximate* state abstraction, which can preserve representation of good behavior without requiring a perfect solution to the MDP of interest. In this chapter, I extend these results to the case where an agent is presented with a continuous stream of tasks to solve. That is, I study state abstraction in the context of *lifelong* RL.

Indeed, a long standing goal of AI is to understand how autonomous agents can accumulate and make use of knowledge across a variety of related tasks, or perhaps from a

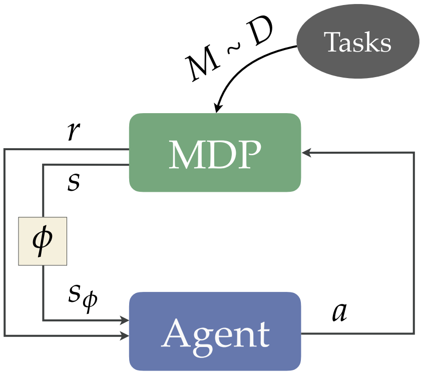


Figure 4.1: Lifelong Reinforcement Learning with State Abstraction.

continual stream of experience. This setting (and its kin) has appeared under a variety of names including multitask learning, lifelong learning, transfer learning, or continual learning. The premise of each of these settings is to agents that must interact with and solve many different tasks over the course of a lifetime, as studied by Thrun [316], Wilson et al. [344], Isele et al. [138], Walsh et al. [333] and Wilson et al. [344]. There are important nuances that separate each of these specific settings, but the spirit is largely the same. I will henceforth use the term *lifelong* learning to capture the many variations of this setting.

Lifelong *reinforcement learning* presents a particularly difficult set of challenges as it forces agents not only to generalize within an MDP, but also across MDPs. Drawing from prior literature, I offer the following definition of the lifelong RL setting.

Definition 4.1. *In **lifelong reinforcement learning**, the agent receives $\mathcal{S}, \mathcal{A}, \rho_0$, horizon H, discount factor γ, and query access to a fixed but unknown probability distribution over reward-transition function pairs, D. The agent samples $(R_i, T_i) \sim D$, and interacts with the MDP $(\mathcal{S}, \mathcal{A}, R_i, T_i, \gamma, \rho_0)$ for H time steps, starting in state $s_0 \sim \rho_0$. After H time steps, the agent resamples from D and repeats.*

The tools of abstraction are particularly well-suited to assist in lifelong RL, as state abstraction can capture relevant task structure across MDPs that can aid in information transfer and accelerate learning. Equipped with the right state abstraction, then, an agent might be able to learn in an extremely sample efficient manner, even on unseen tasks. It is precisely this insight that I investigate in this chapter. For more motivation and background on lifelong RL, see work by Brunskill and Li [59].

4.1 TRANSITIVE PAC STATE ABSTRACTIONS

Concretely, I propose two new classes of state abstraction that together are easy to compute in the lifelong setting. That is, the joint family of 1) transitive, and 2) PAC state abstractions are efficient to compute, can be estimated from a finite number of sampled and solved tasks, and preserve expected near-optimal behavior in lifelong RL. These are the first state abstractions to satisfy this collection of properties. I close the chapter with a negative result, however: PAC-MDP algorithms [301] such as R-Max [54] are not guaranteed to interact effectively with an abstracted MDP, suggesting that additional work is needed to leverage this idea to yield sample efficient learning. Finally, I conduct several simple experiments with standard algorithms to empirically corroborate the effects of state abstraction on lifelong RL.

I will make use of two PAC-MDP algorithms from prior literature: R-Max [54] and Delayed Q-learning [300]. As discussed in Chapter 2, a PAC-MDP algorithm comes with a guarantee that it will only make a polynomial number of mistakes with high probability, thereby ensuring that it explores the MDP efficiently. Delayed Q-learning is a model free algorithm that also makes heavy use of optimism to inform its exploration strategy.

In general, computing the approximate state abstraction that induces the smallest possible abstract state space for that predicate is known to be NP-Hard [99]. Indeed, this result limits the potential utility of state abstractions, as reducing the size of the abstract state space is the main goal of state abstraction; a smaller state space is desirable as the reduced MDP is (typically) easier to solve.

I first introduce *transitive* state abstractions, a restricted class of approximate state abstractions that can be computed efficiently. Intuitively, transitive state abstractions induce an equivalence class on ground states; transitivity guarantees that the predicate p associated with the type satisfies the implication $[p(s_1, s_2) \text{and } p(s_2, s_3)] \implies p(s_1, s_3)$. Many existing state-abstraction types are transitive. However, the *approximate* abstraction types introduced in the previous chapter are not transitive. Thus, I next introduce a transitive modification of each of the approximate state-abstraction types.

Definition 4.2. *For a given $d \in [0, \text{VMax}]$, $\boldsymbol{\phi}_{Q^*_d}$ denotes a state-abstraction type with predicate:*

$$p^d_M(s_1, s_2) := \forall_{a \in \mathcal{A}} : \left\lceil \frac{Q^*_M(s_1, a)}{d} \right\rceil = \left\lceil \frac{Q^*_M(s_2, a)}{d} \right\rceil. \tag{4.1}$$

Intuitively, the abstraction discretizes the interval from $[0, \text{VMax}]$ by buckets of size d. Then, a pair of states satisfy the predicate if the Q^*-values for all actions fall in the same discrete buckets. Note that this predicate is transitive by the transitivity of being-in-the-same-bucket. As I will show in the next section, the above type affords representation near-optimal behavior as a function of d.

The second new family of state abstractions introduced are those that are suitable for application to a distribution of MDPs. The motivation for these state abstractions is to identify a mechanism for extending a state abstraction that preserves representation of good policies in one MDP to the case of many MDPs. In particular, I will later show that PAC abstractions ensure, with high probability over the task distribution D, that a high value policy is representable. The family is defined as follows, inspired by PAC learning [324].

Definition 4.3. ϕ_p^δ *is **a PAC state abstraction** belonging to type ϕ_p such that, for a given $\delta \in (0,1]$, and a given distribution over MDPs D, the abstraction groups together nearly all state pairs for which the predicate p holds with high probability over the distribution. More formally, for an arbitrary state pair (s_1, s_2), let ρ_x^p denote the predicate that is true if and only if p is true over the distribution with probability $1 - x$:*

$$\rho_x^p(s_1, s_2) := \mathop{\mathbb{P}}_{M \sim D}\{p_M(s_1, s_2) = 1\} \geq 1 - x. \tag{4.2}$$

Then ϕ_p^δ is a PAC state abstraction if there exists an $\varepsilon \in (-\delta, \delta)$ such that, for all s_1, s_2:

$$\mathbb{P}\left\{\rho_{\delta+\varepsilon}^p(s_1, s_2) \equiv \phi_p^\delta(s_1) = \phi_p^\delta(s_2)\right\} \geq 1 - \delta. \tag{4.3}$$

4.2 ANALYSIS

I now present our main theoretical results on each of the two new abstraction types. These results summarize how to bring efficiently computable, value-preserving state abstractions into lifelong RL. I first analyze transitive abstraction, then PAC abstractions.

4.2.1 *Transitive State Abstractions*

I first show that transitive state abstractions can be computed efficiently.

Theorem 4.1. *Consider any transitive predicate on state pairs, p, that takes computational complexity c_p to evaluate for a given state pair. The state abstraction type ϕ_p that induces the smallest abstract state space can be computed in $\mathcal{O}(|\mathcal{S}|^2 \cdot c_p)$.*

Proof of Theorem 4.1.

Let c_p denote the computational complexity associated with computing the predicate p for a given state pair. Consider the algorithm consisting of the following four rules for constructing abstract clusters (which define the abstract states) using queries to each of the $|\mathcal{S}|^2$ state pairs. Let (s_i, s_j) denote the current state pair:

1. If $p(s_i, s_j)$ is true, and neither state is in an abstract cluster yet, make a new cluster consisting of these two states.
2. If $p(s_i, s_j)$ is true and only one of the states is already in a cluster, add the other state to the existing cluster.
3. If $p(s_i, s_j)$ is true and both s_i and s_j are in different cluster, merge the clusters.
4. If $p(s_i, s_j)$ is false, add each state not yet in a cluster to its own cluster.

Running this algorithm makes one query per state pair, of which there are $|\mathcal{S}|^2$. Thus, the complexity is $O\left(|\mathcal{S}|^2 \cdot c_p\right)$.

From steps 1-3, after iterating through the possible state pairs, there cannot exist a state pair (s_x, s_y) such that $p(s_x, s_y)$ is true but s_x and s_y are in different clusters. Further, by transitivity, when we apply the cluster merge in step 3, we are guaranteed that every state pair in the resulting cluster necessarily satisfies the predicate. Thus, we compute the smallest clustering definable by p. □

The intuition here is that we can avoid many computations by relying on transitivity. Any one query made of a state pair predicate can yield information about all state pairs in the equivalence class. Critically, the complexity of c_p dictates the overall complexity of computing ϕ_p.

Recall that most known approximate state-abstraction types are *not* transitive (see Table 2.1, for instance). Hence, I next show that there exists an approximate state-abstraction type—with a transitive predicate—with bounded value loss:

Theorem 4.2. *The $\phi_{Q_d^*}$ abstraction type is a subclass of $\phi_{Q_\varepsilon^*}$ introduced in the previous chapter, with $d = \varepsilon$, and therefore, for a single MDP, the optimal abstract policy π_ϕ^* resulting from $\phi_{Q_d^*}$ ensures*

$$V^*(s_0) - V^{\pi_\phi^*}(s_0) \leq \frac{2d\text{RMAX}}{(1-\gamma)^2}. \tag{4.4}$$

Proof of Theorem 4.2.

For any two state-action pairs that satisfy the predicate p_M^d, we know by definition of the predicate that for each action a, there exists a Q_{lower} such that:

$$Q_{lower} \leq Q^*(s_1, a) \leq Q_{lower} + d,$$
$$Q_{lower} \leq Q^*(s_2, a) \leq Q_{lower} + d.$$

Therefore, for each action a:

$$|Q^*(s_1, a) - Q^*(s_2, a)| \leq d. \tag{4.5}$$

Therefore, $\phi_{Q_d^*}$ is a subclass of $\phi_{Q_\varepsilon^*}$. □

Thus, the $\phi_{Q_d^*}$ class represents a reasonable candidate for state abstractions as it can be computed efficiently and posses a value loss that scales according to a free parameter, d. When $d = 0$, the value loss is zero, and the abstraction collapses to the typical ϕ_{Q^*} irrelevance abstraction from Li et al. [203]. Note that predicates defining other existing abstraction types, such as ϕ_{a^*} [203], also have natural translations to transitive predicates using the same discretization technique. While most of the main theoretical results are agnostic to choice of predicate, I concentrate on Q based abstractions due to their simplicity and utility. Notably, none of these state abstractions require exact knowledge of Q^*: I always approximate it based on knowledge of prior tasks. Our results shed light on when it is possible to employ approximate knowledge of this kind for use in decision making.

Recall, however, that the primary goal of state abstraction is to reduce the size of the agent's representation over problems of interest. A natural question arises: if one were to solve the full NP-Hard problem of computing the maximally compressing state abstraction of a particular class, how much more compression can be achieved over the transitive approximation? Intuitively: Is the transitive abstraction going to compress the state space? The following result addresses this question.

Theorem 4.3. *For a given d, the function belonging to the transitive abstraction type $\phi_{Q_d^*}$ that induces the smallest possible abstract state space size is at most $2^{|\mathcal{A}|}$ times larger than that of the maximally compressing instance of type $\phi_{Q_\varepsilon^*}$, for $d = \varepsilon$. Thus, letting $\mathcal{S}_d$ denote the abstract state space associated with the maximally compressing $\phi_{Q_d^*}$, and letting $\mathcal{S}_\varepsilon$ denote the abstract state space associated with the maximally compressing $\phi_{Q_\varepsilon^*}$,:*

$$|\mathcal{S}_\varepsilon| \cdot 2^{|\mathcal{A}|} \geq |\mathcal{S}_d|. \tag{4.6}$$

Proof of Theorem 4.3.

Let M be an arbitrary MDP. Consider a set of states $\tilde{S} \subset \mathcal{S}$ clustered together under $\phi_{Q_\varepsilon^*}$ and, in particular, consider the Q-values of all states in $\tilde{S}$ for a particular action, $a \in \mathcal{A}$. Note that, by construction of $\phi_{Q_\varepsilon^*}$, for any

$$\forall_{s,s' \in \tilde{S}} : |Q^*(s,a) - Q^*(s',a)| \leq \varepsilon,$$

Recall that, intuitively, $\phi_{Q_d^*}$ is a discretization of the interval $[0, \text{VM}\textsc{ax}]$ where d controls the placement of boundaries, forming buckets of Q-values. The Q-values for all states in $\tilde{S}$ and for action a reside in a single sub-interval of length ε.

Letting $d = \varepsilon$, the placement of boundaries that form $\phi_{Q_d^*}$ could break the ε-interval of Q-values for the non-transitive cluster $\tilde{S}$ no more than once, resulting in the creation of at most two new state clusters in $\phi_{Q_d^*}$. Repeating the process for each action, these separations within the original cluster compound, resulting in at most $2^{|\mathcal{A}|}$ such subdivisions and, accordingly, $2^{|\mathcal{A}|}$ clusters in $\phi_{Q_d^*}$ for each cluster in $\phi_{Q_\varepsilon^*}$. □

The above result shows that the non-transitive, maximally compressing state space size can in fact be quite smaller than the transitive approximation (by a factor of $2^{|\mathcal{A}|}$).

4.2.2 *PAC Abstractions*

Next, I analyze PAC abstractions for the purpose of extending state abstractions to the lifelong setting. I first show that, for any abstraction type ϕ_p, its PAC variant achieves bounded value loss (with high probability) as a function of the single task loss of ϕ_p:

Theorem 4.4. *Consider any state-abstraction type ϕ_p with value loss τ_p. That is, in the traditional single task setting, letting π_p^* denote $\pi_{\phi_p}^*$:*

$$\forall_{s \in \mathcal{S}} : V^*(s) - V^{\pi_p^*}(s) \leq \tau_p. \tag{4.7}$$

Then, the PAC abstraction ϕ_p^δ, in the lifelong setting, induces a policy $\pi^*_{p,\delta}$ with expected value loss:

$$\forall_{s \in \mathcal{S}} : \mathop{\mathbb{E}}_{M \sim D}\left[V^*_M(s) - V_M^{\pi^*_{p,\delta}}(s)\right] \le \varepsilon(1 - 3\delta)\tau_p + 3\delta\text{VMax}. \tag{4.8}$$

Proof of Theorem 4.4.

> By definition of PAC abstractions, with probability $1 - \delta$, the abstraction function ϕ_p^δ aggregates if and only if $\rho^p_{\delta+\varepsilon}$, for some small $\varepsilon \in (-\delta, \delta)$.
>
> Then, with probability $1 - \delta$, there is at least a $1 - \delta - \varepsilon$ chance that the predicate holds for a particular state, by definition of ρ^p_δ. Thus, by definition of ρ^p_δ, with probability $(1 - \delta)(1 - \delta - \varepsilon)$, the state abstraction correctly aggregates, and consequently the inherited value loss τ_p bound holds. If the abstraction incorrectly aggregates, the value loss can be up to VMax.
>
> Letting $\varepsilon = \delta$, we see that the PAC loss is at worst upper bounded by a convex mixture of τ_p with probability $(1 - 3\delta)$, and with probability 3δ, is VMax. Thus, the expected value loss of ϕ_p^δ is:
>
> $$\forall_{s \in \mathcal{S}} : \mathop{\mathbb{E}}_{M \sim D}\left[V^*_M(s) - V_M^{\pi^*_{p,\delta}}(s)\right] \le \varepsilon(1 - 3\delta)\tau_p + 3\delta\text{VMax}. \tag{4.9}$$
>
> □

The value loss may be quite high, as up to 3δVMax value can be lost in the worst case. Accordingly, it is important to be cautious in selection of δ. This bound is not tight, however, so in practice the value loss is likely to be lower.

Next, I show how to compute PAC abstractions from a finite number of sampled tasks.

Theorem 4.5. *Let $\mathscr{A}_p$ be an algorithm that given an MDP $M = (\mathcal{S}, \mathcal{A}, R, T, \gamma)$ as input can determine if $p(s_1, s_2)$ is true for any pair of states, for any state abstraction type. Then, for a given $\delta \in (0, 1]$ and $\varepsilon \in (-\delta, \delta)$, it is possible to compute a PAC abstraction $\hat{\phi}_p^\delta$ after $m \ge \frac{\ln(\frac{2}{\delta})}{\varepsilon^2}$ sampled MDPs from D.*

Proof of Theorem 4.5.

We are given as input a $\delta \in (0,1]$, a distribution over MDPs D, and the algorithm $\mathscr{A}_p$ which, given an MDP M and a state pair outputs $p_M(s,s')$.

Consider an arbitrary pair of states s and s'. For m sampled MDPs, the algorithm $\mathscr{A}_p$ can produce a sequence of m predicate evaluations:

$$p_1(s,s'), \cdots, p_m(s,s'). \tag{4.10}$$

Let $\hat{p}$ be the empirical mean over the predicate sequence:

$$\hat{p} = \frac{1}{m}\sum_{i=1}^{m} p_i(s,s'). \tag{4.11}$$

The clustering algorithm is quite simple: for our input $\delta \in (0,1]$, cluster all state pairs (s,s') such that $\hat{p}(s,s') \geq 1-\delta$ after m samples.

We now prove that, for a particular setting of m, the resulting cluster assignments constitute a state abstraction that clusters a pair of states only if the predicate is true with high probability.

First, let $\overline{p}$ denote the probability that p is true over the distribution:

$$\overline{p}(s,s') = \mathbb{P}_{M\sim D}\{p(s,s') = 1\}. \tag{4.12}$$

Using Hoeffding's Inequality, we upper bound the probability that $\hat{p}$ deviates from $\overline{p}$ by more than some small $\varepsilon \in (0,\delta)$:

$$\mathbb{P}\left\{\left|\hat{p}(s,s') - \mathbb{E}\left[\hat{p}(s,s')\right]\right| \geq \varepsilon\right\} = \mathbb{P}\left\{|\hat{p}(s,s') - \overline{p}(s,s')| \geq \varepsilon\right\} \leq 2e^{-2m\varepsilon^2}. \tag{4.13}$$

Thus, for $\delta = 2e^{-2m\varepsilon^2}$:

$$\mathbb{P}\left\{|\hat{p}(s,s') - \overline{p}(s,s')| < \varepsilon\right\} > 1-\delta. \tag{4.14}$$

Rewriting:

$$\mathbb{P}\left\{|\hat{p}(s,s') - \overline{p}(s,s')| < \varepsilon\right\} > 1 - \delta \tag{4.15}$$

$$\iff$$

$$\mathbb{P}\left\{-\varepsilon < \hat{p}(s,s') - \overline{p}(s,s') < \varepsilon\right\} > 1 - \delta, \tag{4.16}$$

By algebra, note that, when $m \geq \frac{\ln \frac{2}{\delta}}{\varepsilon^2}$, the condition of Equation 4.14 holds.

Let ρ_δ^p denote the predicate that is true if and only if p is true over the distribution with high probability for a given δ:

$$\rho_\delta^p(s_1, s_2) = \begin{cases} 1 & \overline{p} \geq 1 - \delta \\ 0 & \text{otherwise.} \end{cases} \tag{4.17}$$

Now, we form our state abstraction under the following rule:

$$\hat{\phi}_p^\delta(s_1) = \hat{\phi}_p^\delta(s_2) \equiv \hat{p}(s,s') > 1 - \delta. \tag{4.18}$$

If, after m samples, $\hat{p}$ were identical to $\overline{p}$, then we would have:

$$\forall_{s,s'} : \mathbb{P}_{M \sim D}\{\rho_\delta^p(s,s') \equiv \hat{\phi}_p^\delta(s_1) = \hat{\phi}_p^\delta(s_2)\} \geq 1 - \delta. \tag{4.19}$$

Hence, $\hat{p}$ deviates from $\overline{p}$ by at most ϵ with probability $1 - \delta$. Thus, for some $\varepsilon \in (-\delta, \delta)$, $\hat{p} + \varepsilon = \overline{p}$. Therefore, the clustering rule defined by Equation 4.18 ensures there exists an ε such that, with high probability, we cluster according to:

$$\forall_{s_1,s_2} : \rho_{\delta+\varepsilon}^p(s_1, s_2) \equiv \phi_p^\delta(s_1) = \phi_p^\delta(s_2). \tag{4.20}$$

We conclude that, for $m \geq \frac{\ln \frac{2}{\delta}}{\varepsilon^2}$ sampled and solved MDPs, we compute a lifelong PAC state abstraction $\hat{\phi}_p^\delta$. □

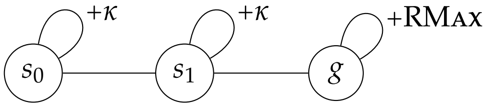
Note that this result assumes oracle access to the true predicate, $p(s_1, s_2)$, during the computation of $\hat{\phi}_p^{\delta}$. The analogous case in which p can only be estimated via an agent's interaction with its environment is a natural next step for future work.

Given the ability to compute PAC abstractions from a finite number of samples, I now study the interplay between state abstractions and PAC-MDP algorithms for efficient RL.

Theorem 4.6. *Consider an MDP M and an instance of R-Max [54] that breaks ties using round-robin selection over actions. Now, consider R-Max paired with a state-abstraction function ϕ with value loss bounded by $\varepsilon_\phi \in \mathbb{R}_{\geq 0}$. If R-Max interacts with M by projecting any received state s through ϕ, then R-Max is no longer guaranteed to be PAC-MDP in M (even relative to mistakes defined by ε_ϕ). In fact, the number of mistakes made by R-Max can be arbitrarily large.*

Proof of Theorem 4.6.

Consider the simple three state chain:



The agent has three actions, `left`, `right`, and `loop`, associated with their natural effects (`left` in s_0 is a self loop with reward 0, while `right` moves the agent to s_1, and so on).

In states s_0 and s_1, let the reward for `loop` be some small constant κ, and let the `loop` action in s_3 yield RMax reward.

Let $\gamma = 0.95$, s_0 define the initial state, and $\kappa = 0.001$. Suppose the agent reasons using an instance of $\phi_{Q_\varepsilon^*}$ with $\varepsilon = 0.1$. Then, observe that all three states may be clustered, since:

$$\forall_{s \in \{s_0, s_1, s_2\}} : \max_{a_1, a_2} Q^*(s, a_1) - Q^*(s, a_2) \leq \varepsilon. \tag{4.21}$$

That is, note that the Q^* values of each state-action pair are roughly as follows:

$$\begin{array}{lll} Q^*(s_0,\texttt{right}) \approx 0.90 & Q^*(s_1,\texttt{right}) \approx 0.95 & Q^*(s_2,\texttt{right}) \approx 0.95 \\ Q^*(s_0,\texttt{left}) \approx 0.86 & Q^*(s_1,\texttt{left}) \approx 0.86 & Q^*(s_2,\texttt{left}) \approx 0.90 \\ Q^*(s_0,\texttt{loop}) \approx 0.87 & Q^*(s_1,\texttt{loop}) \approx 0.91 & Q^*(s_2,\texttt{loop}) \approx 1.0 \end{array}$$

Therefore, for $\varepsilon = 0.1$, a valid clustering assigns $\phi(s_0) = \phi(s_1)$.

To break ties, we suppose R-Max chooses actions according to a *round-robin* policy, starting with action `left`. Thus, in the abstract, R-Max first chooses left, then right, then self loop, then left, right, self loop, and so on, until each state-action pair is known.

In the above problem, this sequence of actions will *never* lead the agent out of state s_0 or s_1. Let m denote the parameter given to R-Max that determines how many samples per state-action pair are sufficient for the pair to be considered known. Therefore, after m executions of these three actions across states s_0 and s_1, R-Max with ϕ will compute a transition model that assigns zero probability to arriving in g from the aggregated state $\phi(s_0)$. Further, the action `loop` will have the largest reward associated with it—κ, a reward chosen to be arbitrarily small—which is thus arbitrarily worse than the goal reward. So, R-Max with ϕ will make an unbounded number of mistakes even when ϕ is a state abstraction that ensures bounded value loss. □

The above result is a surprising *negative* result—it suggests that there is more to the abstraction story than simply projecting states into the abstract. Specifically, it is indicative of future work that clarifies how to form abstractions that preserve the right kinds of guarantees.

To communicate this piece more directly, I conduct a simple experiment in the 3-chain problem introduced in the proof of Theorem 4.6. Here I run R-Max and Delayed Q-learning with and without $\phi_{Q_\varepsilon^*}$, with abstraction parameter $\varepsilon = 0.01$. Each agent is given 250 steps to interact with the MDP. The results are shown in Figure 4.2. R-Max, paired with abstraction $\phi_{Q_\varepsilon^*}$, fails to learn a anywhere close to a near-optimal policy. In fact, it is

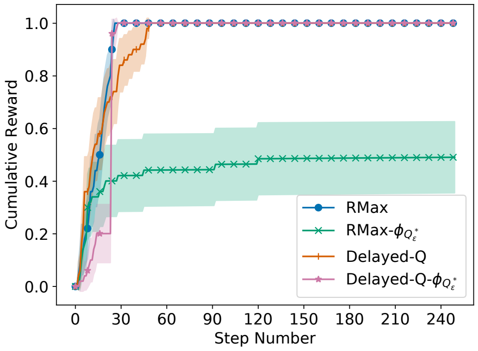


Figure 4.2: Results averaged over 50 runs on the pathological three chain MDP introduced in the proof of Theorem 4.6.

possible to control a parameter in the MDP such that R-Max performs arbitrarily bad. It remains an open question as to whether ϕ preserves the PAC-MDP property for Delayed Q [300].

To highlight this point further, I next show that projecting an MDP to the abstract state space via ϕ and learning with M_ϕ is non-identical to learning with M and projecting states through ϕ:

Corollary 4.1. *Consider any RL algorithm $\mathscr{A}$ whose policy updates during learning and a fixed choice of state abstraction, ϕ. Let $\mathscr{A}_\phi$ denote the algorithm yielded by projecting all incoming states to $\phi(s)$ before presenting them to $\mathscr{A}$, and let $M_\phi = (\mathcal{S}_\phi, \mathcal{A}, T_\phi, R_\phi, \gamma, \rho_0^\phi)$, denote the abstract MDP induced by ϕ on M, with $w(s)$ the uniform weighting function.*

There exists an MDP M such that the process yielded by $\mathscr{A}_\phi$ interacting with M is not identical to $\mathscr{A}$ interacting with M_ϕ, even if $\pi^t_{\mathscr{A}_\phi}(s) = \pi^t_{\mathscr{A}}(s)$ for all t and s. That is, there exists choice of ϕ and M such that the expected trajectory of these two processes is non-identical. Formally, there exists a time step $t \in \mathbb{N}$, MDP M, and $\phi \in \Phi_{all}$ such that

$$\mathop{\mathbb{E}}_{\mathscr{A}, s_0 \sim \rho_0^\phi} [s_t \mid s_0] \neq \mathop{\mathbb{E}}_{\mathscr{A}_\phi, s_0 \sim \rho_0^\phi} [s_t \mid s_0], \tag{4.22}$$

where s_t is the state the agent arrives in after t time steps.

Proof of Corollary 4.1.

Note that when M_ϕ is computed directly, the functions R_ϕ and T_ϕ assume a fixed weighting function $w(s)$.

Again let us consider the three state chain MDP from the previous proof. During typical interaction between M and $\mathscr{A}_\phi$, no such fixed weighting function exists *for any algorithm $\mathscr{A}$ that updates its policy*. That is, the distribution of states the agent finds itself in will change as its policy changes, and therefore, $w(s)$ must change, too, thereby updating T_ϕ. Conversely, in the true MDP, T remains fixed.

Thus, the process of $\mathscr{A}_\phi$ interacting with M induces a sequence of interactions with abstract MDPs whose transition and rewards change along with the policy the agent follows. Thus, for some non-identity ϕ, for any algorithm $\mathscr{A}$ whose policy changes over time, the resulting expected interaction may be different. □

These results illustrate a peculiarity to using state abstractions in RL: abstracting during interaction is distinct from offline abstraction. This result is reminiscent of Theorem 4 and Theorem 5 by Li et al. [203] that describe the impact ϕ can have on convergence guarantees of well known RL algorithms. An important direction for future work is to provide a cohesive framework that preserves both PAC and convergence guarantees, whether the abstractions are used offline or online.

To summarize the analysis in this chapter: any state abstraction that belongs to both the transitive class *and* the PAC class is: (1) efficient to compute, (2) can be estimated from a polynomial number of sampled and solved problems, (3) and preserves near-optimal behavior in the lifelong RL setting. The identification of such a class of desirable state abstractions for lifelong RL is the main contribution of this chapter, gesturing toward state abstractions that can trade off between all three desiderata. Alongside these positive results, I have highlighted shortcoming of state abstractions in the final two results, raising open questions about how to generalize state abstractions to work well with existing PAC-MDP algorithms.

I now move on to an empirical study evaluating the utility of these abstractions.

4.3 EXPERIMENTS

I conduct two sets of simple experiments with the goal of illuminating how state abstractions of various forms impact learning and decision making. The code for running these experiments is publicly available.[4.1]

- *Learning with and without ϕ_p^δ*: I investigate the impact of different types of abstractions on Q-Learning [337] and Delayed Q-learning [300] in different lifelong RL task distributions.
- *Planning with and without ϕ_p^δ*: Second, I explore the impact of planning via VI (Algorithm 2.3) with and without a state abstraction, intended to be suggestive of the potential to accelerate model-based algorithms with good state abstractions.

In each case, I compute various types of ϕ_p^δ according to the sample bound from Theorem 4.5, with $\delta = 0.1$, and the PAC parameter $\varepsilon = 0.1$ (the worst case ε). I experiment with ($\phi_{Q^*}^\delta$), approximate ($\phi_{Q_\varepsilon^*}^\delta$), and transitive ($\phi_{Q_d^*}^\delta$) state abstractions from the Q similarity classes across each of the above algorithms. I experiment with probably approximate Q based abstractions because their value loss bound is known, tight, and a small function of the approximation parameter, and (2) They have known transitive variants and are thus simple to compute, as shown in Theorem 4.1. Further, if a Q^* based abstraction presents no opportunity to abstract (the reward or transition function change too dramatically across tasks), then Theorem 4.5 indicates that ϕ will abstain from abstracting.

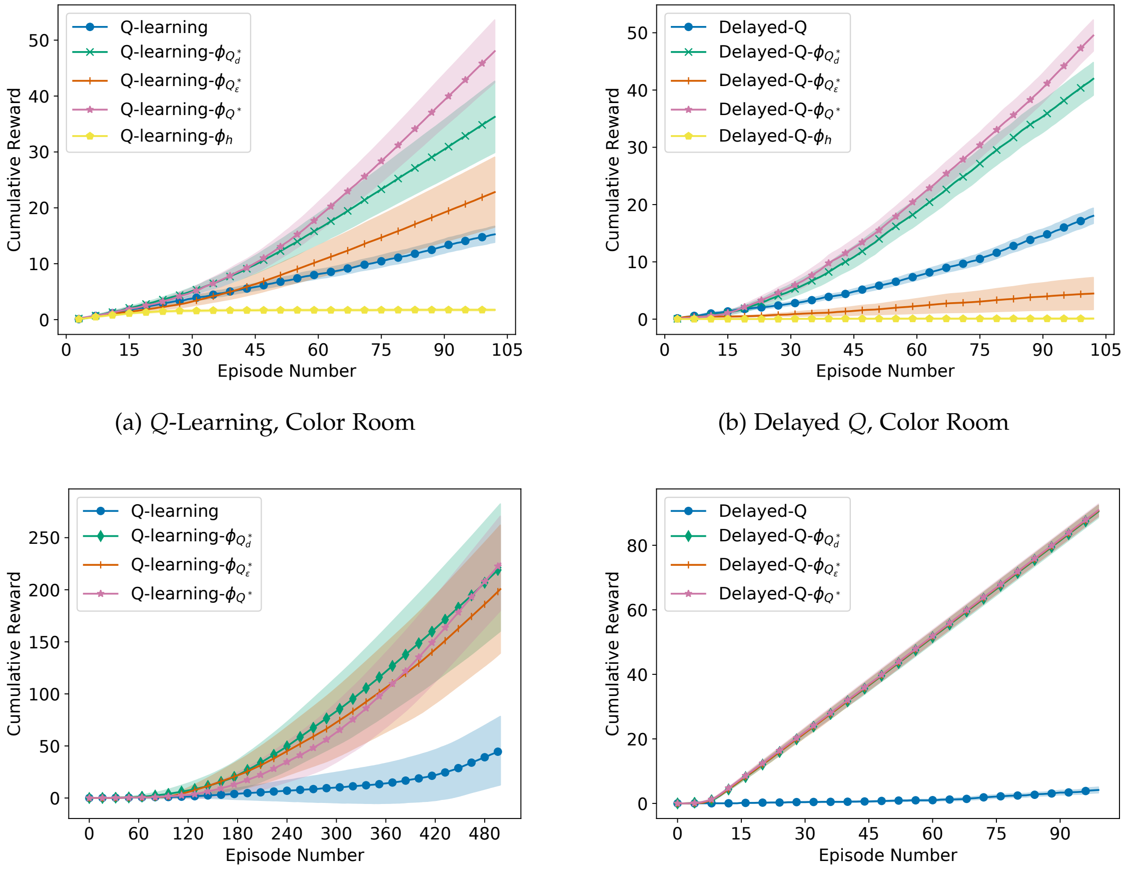
Lifelong RL

Each learning experiment proceeds as follows. For each agent, at time step zero, sample a reward function from the distribution. Then, let the agent interact with the resulting MDP for 100 episodes. When the last episode finishes, reset the agent to the start state s_0, and repeat. All learning curves are averaged over samples from the distribution. Thus,

[4.1] https://github.com/david-abel/rl_abstraction improvements to learning from each ϕ are improvements *averaged over the task distribution*. In all learning plots we report 95% confidence intervals over the sampling process (both samples from the distribution and runs of each agent in the resulting MDP).

Color Room: I first conduct experiments testing Q-learning and Delayed Q learning on an 11×11 Four Rooms variant, adapted from Sutton et al. [311]. In this task distribution, goal states can appear in exactly one of the furthest corner of each of the three non-starting rooms (that is, there are three possible goal locations) uniformly at random. Transitions into a goal state yield $+1$ reward with all other transitions providing $+0$. Goal states are set to terminal. To explore the impact of abstraction, I augment the problem representation by introducing an irrelevant state feature: *color*. Specifically, each cell in the grid can have a color *red, blue, green*, or *yellow*. All cells are initially *red*. The agent is given another action, `paint`, that paints the entire set of cells to one of the four colors uniformly at random. No other action can change the color of a cell. The color has no impact on either reward or transitions, and so is fundamentally irrelevant in decision making. We are thus testing the hypothesis as to how effectively the sample based PAC abstractions can pick up on the irrelevant characteristics and still support efficient but high performance learning. Given the inherent structure of the Four Rooms domain we also experiment with an intuitively useful hand-coded state abstraction, ϕ_h, that assigns an abstract state to each room for a total of four abstract states. The agents all start in the bottom left cell.

The top row of Figure 4.3 shows results for algorithms run on the Colored Four Rooms task distribution. For Q-learning the data suggest that all three PAC abstractions achieve improvement in mean cumulative reward, averaged across 100 task samples. Notably, the slope of the learning curves are similar as well, suggesting that the policies discovered after 100 episodes are comparable in value. Notably, the variants with the abstraction tend to find better policies more quickly. In the case of Delayed-Q, the new transitive PAC abstraction (green) finds even further improvement over the baseline algorithm, both in terms of learning speed and the value of the policy used near the end of learning.



(a) Q-Learning, Color Room

(b) Delayed Q, Color Room

(c) Q-Learning, Upworld

(d) Delayed Q, Upworld

Figure 4.3: Cumulative reward averaged over 100 task samples from the Colored Four Rooms task distribution (top) and the Upworld task distribution (bottom).

The bottom row of Figure 4.3 present results for the same learning set up with the Upworld MDP distribution. This domain is an extremely simple 30 × 11 grid world, where one of the 30 possible goals in the top row is active at any given time. The agent always starts in the bottom left corner. As expected, the data suggest that the improvement from the abstraction in this domain are dramatic, as there is great opportunity to abstract. The performance of both baseline algorithms is dominated by any of the approaches that use state abstraction.

Four Rooms: I next conduct an experiment in a larger 15×15 Four Rooms variant in which color and `paint` are removed to explore the degree to which the irrelevant variables

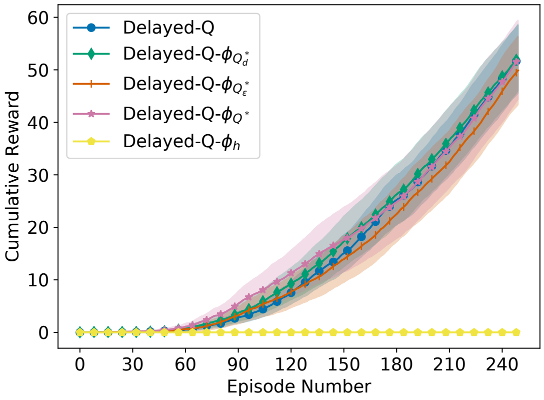
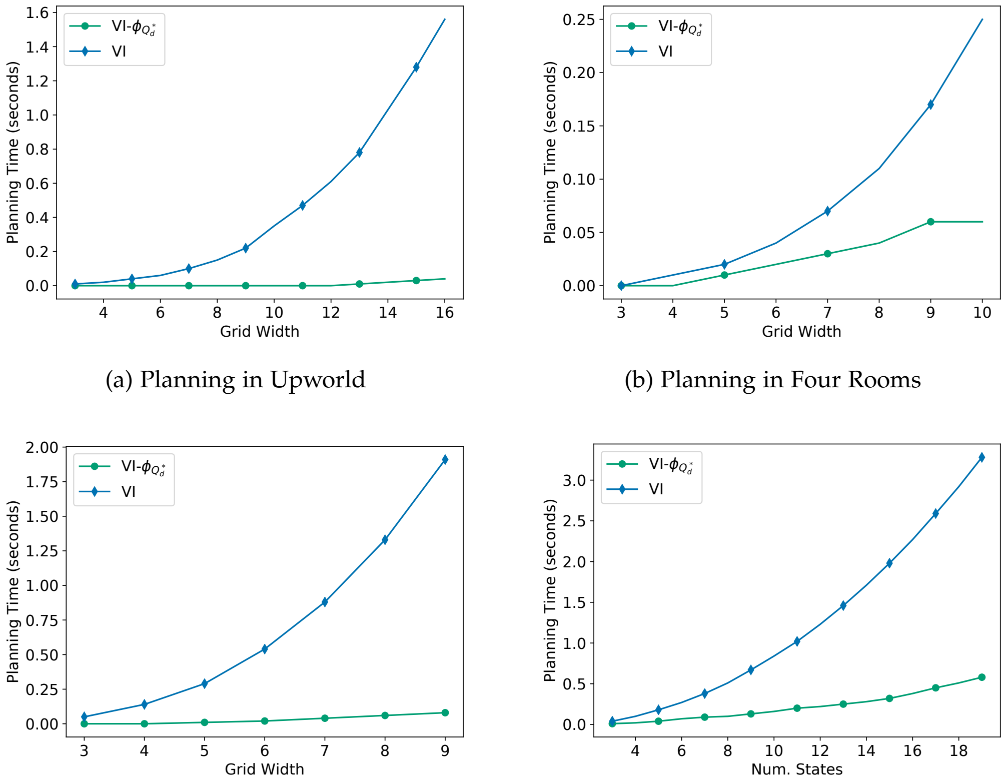


Figure 4.4: Delayed Q-learning on a 15 × 15 Four Rooms task distribution.

explain the learning improvement found in the previous experiment. I evaluate Delayed Q-learning again with the same process for constructing state abstractions, again with 250 episodes, with 100 steps per episode. Here, the results suggest that the abstractions do effectively nothing to change learning—no irrelevant variables are included in the problem, and so only a few states are clustered. The original MDP has around 200 states, with the abstract state spaces averaging around 150 states. Consequently, learning is largely unchanged. However, not that in the case of ϕ_h, when each state in the same room is clustered together, learning is devastated. This again highlights the importance of delicately choosing a state abstraction. Even when the other state abstractions did not accelerate learning, they at least did not negatively impact it, either.

Planning

To give further evidence of the potential benefits offered by state abstraction I next contrast the time taken to plan with and without the state abstraction. Indeed, the benefits of state abstraction to planning have been well studied [165, 133, 141, 15, 16]. I next study the impact of giving VI (Algorithm 2.3) a state abstraction in four simple problems. The first is the 10×30 Upworld grid problem from Chapter 3, the second is the standard Four Rooms domain, the third is the Color Rooms domain from the previous experiment, and



(a) Planning in Upworld

(b) Planning in Four Rooms

(c) Planning in Color Rooms

(d) Planning in a Random MDP

Figure 4.5: Planning time for Value Iteration with and without a state abstraction as the environmental state space grows.

the final is the Random MDP from Chapter 3. In each MDP, I vary the size of the underlying state space and contrast the time taken for VI to converge with and without a state abstraction.

Results are presented in Figure 4.5. The findings are expected: in both Upworld and the Color Rooms MDPs, there are opportunities to abstract aggressively, thereby significantly lowering the computation needed to run VI to convergence. In the other two domains, there is some opportunity to abstract, but not as much, and consequently the benefits to planning time are not as dramatic. In all cases, the ground value of the computed policy is identical. Thus, the data suggest that planning can be accelerated when there is opportunity to abstract. Consequently, model-based RL algorithms employing the appropriate state abstractions may be able to plan more efficiently.

In this chapter, I focused on bringing state abstraction theory out of the traditional single task setting and into lifelong RL. I introduce two new complementary families of state abstractions, (1) transitive state abstractions, and (2) PAC abstractions. Together, they characterize state abstractions that can be feasibly obtained (satisfying D1) while still preserving near-optimal behavior (satisfying D3). Additionally, I drew attention to several shortcomings of learning with abstractions, building on those studied by Li et al. [203] and Gordon [122], suggesting pathways for realizing the full potential of abstraction in RL. Moreover, the experimental evidence suggests that both planning and learning can be made more efficient when these abstractions are used—in this sense, these state abstractions target the satisfaction of all three desiderata.

5

STATE ABSTRACTION AS COMPRESSION

This chapter is based on "State Abstraction as Compression in Apprenticeship Learning" [9] joint with Dilip Arumugam, Kavosh Asadi, Lawson L.S. Wong and Michael L. Littman, and "Learning State Abstractions for Transfer in Continuous Control" [24] led by Kavosh Asadi, also with Michael L. Littman.

In the previous two chapters, I analyzed classes of state abstraction functions that can reduce the size of the underlying state space while simultaneously preserving representation of good policies. This dual-objective closely parallels the mission of *information theory*, which presents a rigorous formalism for understanding communication in the presence of noise. The key results of information theory are centered around the act of *compression*—how an entity can be reduced in size while preserve its essence. There is striking similarity between this process and that of abstraction. A natural line of reasoning, then, seeks to establish more explicit contact between the tools of information theory and the process of abstraction. Indeed, cognitive neuroscience has suggested that perception and generalization are tied to efficient compression [27, 288, 289], termed the "efficient coding hypothesis" by Barlow [33].

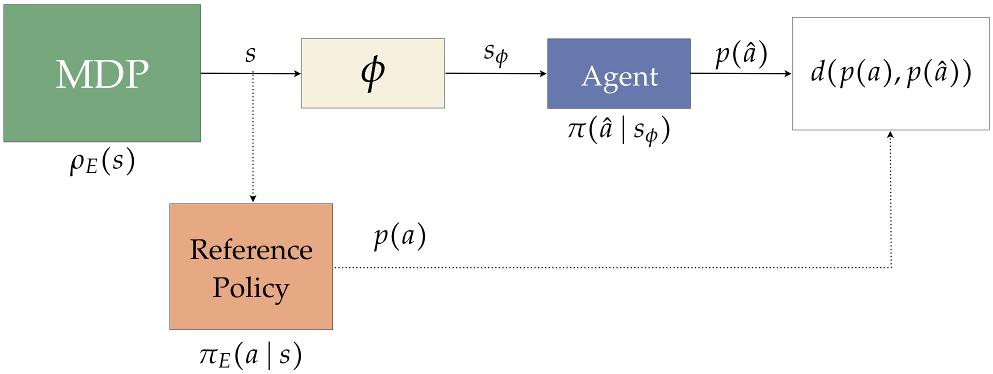


Figure 5.1: The proposed framework for trading off compression with value via state abstraction.

The goal of this chapter is to adopt the viewpoint that state abstraction for sequential decision making can be understood as a process of compression. From this new perspective, I will introduce a new algorithm for constructing state abstractions that imports many of the desirable characteristics enjoyed by some of the key algorithms of information theory. Precisely, I draw a parallel between *state abstraction* as used in reinforcement learning and *compression* as understood in information theory. This parallel is heavily inspired by the seminal work of Shannon [284], Blahut [47], Arimoto [20] and Tishby et al. [318], and draws on insights from related work on understanding the relationship between abstraction and compression [49, 297].

While the perspective I here introduce is intended to be general, I will restrict the initial study by concentrating on the learning problem when a *demonstrator* is available, as in Apprenticeship Learning [26, 1, 19], which simplifies aspects of the model. I will later build toward the regular RL setting after this initial framework is established.

Concretely, I introduce a new objective function that explicitly balances state-compression and performance. The main result of this chapter proves this objective is upper bounded by a variant of the Information Bottleneck objective adapted to sequential decision making. I introduce Deterministic Information Bottleneck for State abstraction (Dibs), an algorithm that outputs a *lossy* state abstraction optimizing the trade off between compressing the state space and preserving the capacity for performance in that compressed state space. I present empirical results that showcase the relationship between compression and performance captured by the algorithm in a traditional grid world, along with an extension to high-dimensional observations via experiments with the Atari game Breakout. Then, I introduce several extensions to this framework that relax critical assumptions, allowing for more general application of the proposed methods.

First, I present a brief survey of information theory.

5.1 INFORMATION THEORY

Information theory offers foundational results about the limits of compression [284]. The core of the theory clarifies how to communicate in the presence of noise, culminating in seminal results about the nature of communication and compression that helped establish the science and engineering practices of computation. In Shannon's words: "The fundamental problem of communication is that of reproducing at one point either exactly or approximately a message selected at another point" (1948, p. 1). One focus of information theory is on constructing coder-decoder pairs that can faithfully communicate messages with zero or low error, even in the presence of noise.

The seminal results all center around the definition of *entropy*, sometimes called the Shannon entropy:

Definition 5.1. *The **entropy** of a discrete random variable X, with alphabet $\mathcal{X}$, is given by:*

$$H(X) := -\sum_{x \in \mathcal{X}} p(x) \log p(x). \tag{5.1}$$

The entropy measures, roughly, the surprise inherent in a random variable. Throughout this chapter, I use log as shorthand for $\log_2$.

Entropy may be extended to account for joint and conditional probability distributions as follows.

Definition 5.2. *The **joint entropy** of two discrete random variables X and Y, with alphabets $\mathcal{X}$ and $\mathcal{Y}$, is given by:*

$$H(X,Y) := -\sum_{x\in\mathcal{X}}\sum_{y\in\mathcal{Y}} p(x,y)\log p(x,y). \tag{5.2}$$

Definition 5.3. *The **conditional entropy** of X given Y (again two discrete random variables with alphabets $\mathcal{X}$ and $\mathcal{Y}$) is given by:*

$$H(X\mid Y) := -\sum_{x\in\mathcal{X}}\sum_{y\in\mathcal{Y}} p(x,y)\log p(x\mid y). \tag{5.3}$$

An additional quantity of relevance is the Mutual Information between two discrete random variables:

Definition 5.4. *The **mutual information** of two discrete random variables X and Y is given by:*

$$I(X;Y) := \sum_{x\in\mathcal{X}}\sum_{y\in\mathcal{Y}} p(x,y)\log\frac{p(x,y)}{p(x)p(y)}. \tag{5.4}$$

Together with the joint and conditional entropy, the theory offers an elegant, interlocking set of relations between these basic quantities. Their relations are pictured in Figure 5.2.

A final quantity that is of special interest is the *relative entropy*, also called the Kullback-Leibler divergence (KL divergence). The KL divergence expresses the error associated with choosing the probability distribution $p(\tilde{x})$ to approximate the probability distribution $p(x)$.

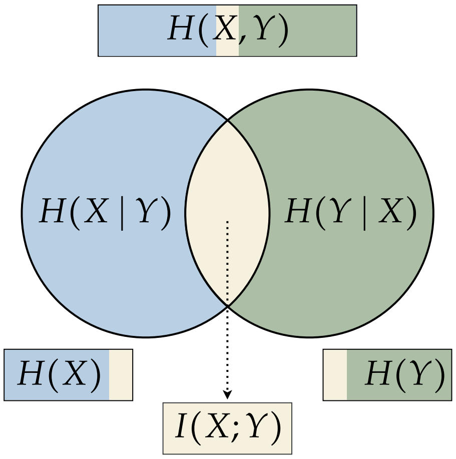


Figure 5.2: The basic quantities of information theory and their relations.

Definition 5.5. *The* ***KL divergence*** *between two probability distributions $p(x)$ and $q(x)$ is given by:*

$$D_{KL}(p \parallel q) = \sum_{x \in \mathcal{X}} p(x) \log \frac{p(x)}{q(x)}. \tag{5.5}$$

Observe that another interpretation for the mutual information of two random variables X and Y is that it expresses the KL divergence between the joint $p(x,y)$ and the independent: $p(x)p(y)$ distributions:

$$I(X;Y) = D_{\mathrm{KL}}(p(x,y) \parallel p(x)p(y)) = \sum_{x \in \mathcal{X}} \sum_{y \in \mathcal{Y}} p(x,y) \log \frac{p(x,y)}{p(x)p(y)}. \tag{5.6}$$

Further note two other properties of the KL divergence: 1) The D_{KL} between two probability distributions that do not have overlapping support is ∞, and 2) D_{KL} is not a metric, because there exist choices of p and q such that $D_{\mathrm{KL}}(p \parallel q) \neq D_{\mathrm{KL}}(q \parallel p)$. Still, the KL divergence is an exceptionally useful measure.

For more background on information theory, see the book by Cover and Thomas [75].

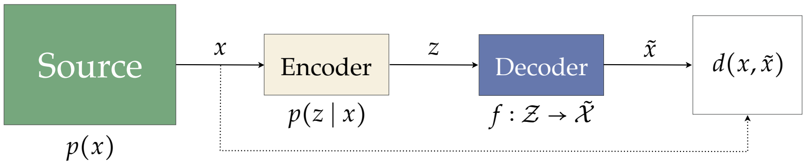


Figure 5.3: The usual Rate-Distortion setting.

5.1.1 *Rate-Distortion Theory*

Of particular relevance to state abstraction is Rate-Distortion (RD) theory, which is a subfield of information theory that studies the trade off between a code's ability to compress (rate) and represent the original signal (distortion) [284, 43].

The typical RD setting is pictured in Figure 5.3: an information source generates $x \in \mathcal{X}$, which is coded via $p(z \mid x)$ to $z \in \mathcal{Z}$, and decoded via a deterministic function $f : \mathcal{Z} \to \tilde{\mathcal{X}}$. *Distortion* is defined with respect to a chosen distortion metric, $d : \mathcal{X} \times \tilde{\mathcal{X}} \to \mathbb{R}_{\geq 0}$, where typically $\mathcal{X} = \tilde{\mathcal{X}}$. The information *rate*, R, denotes the number of bits in each code word. So, with a coding alphabet $\tilde{\mathcal{Z}} = \{0, 1\}^n$, the rate is n. Shannon and Kolmogorov (see Berger [43] for more background) offer a lower bound on the trade off between Rate and Distortion: given a level of distortion, D, the following function defines the smallest rate that achieves expected distortion of at most D:

$$R(D) = \min_{p(\tilde{x}|x) : \mathbb{E}[d(x,\tilde{x})] \leq D} I(X; \tilde{X}). \tag{5.7}$$

Intuitively, Equation 5.7 tells us that as bits are added to the code, algorithms can more faithfully reconstruct the original source messages.

For a given information source, it is natural to consider how to compute a coder-decoder pair that achieves one of the minimal points defined by the Rate-Distortion function. Finding this point presents the following optimization problem:

$$\min_{p(\tilde{x}|x)} \underbrace{I(X; \tilde{X})}_{Rate} + \beta \underbrace{\mathbb{E}_{p(x,\tilde{x})} \left[d(x, \tilde{x}) \right]}_{Distortion}, \tag{5.8}$$

with a Lagrange multiplier $\beta \in \mathbb{R}_{\geq 0}$ expressing the relative preference between minimizing rate and distortion. As β gets closer to 0, rate becomes more important, while as β approaches ∞, minimizing distortion is prioritized. Note that it is desirable to identify coder-decoder pairs that live exactly on this curve, as any point living above indicates that either more compression can take place (lower rate) or more accurate reconstruction can take place (lower distortion).

Blahut-Arimoto (BA) is a simple iterative algorithm that converges to the global optimum of this optimization problem [20, 47]. BA alternates between the following two steps, for a given $\beta \in \mathbb{R}_{\geq 0}$:

$$p_{t+1}(\tilde{x}) = \sum_{x \in \mathcal{X}} p(x) p_t(\tilde{x} \mid x), \tag{5.9}$$

$$p_{t+1}(\tilde{x} \mid x) = \frac{p_{t+1}(\tilde{x}) \exp(-\beta d(x, \tilde{x}))}{\sum_{x' \in \tilde{\mathcal{X}}} p_{t+1}(x') \exp(-\beta d(x, x'))}. \tag{5.10}$$

BA is known to converge to the global optimum with convergence rate:

$$O\left(|\mathcal{X}||\tilde{X}|\sqrt{\log(|\tilde{X}|)/\varepsilon}\right), \tag{5.11}$$

for ε error tolerance [20]. The computational complexity of finding the exact solution for a discrete, memoryless channel is unknown. For a continuous memoryless channel, the problem is an infinite-dimensional convex optimization which is known to be NP-hard [304].

5.1.2 *The Information Bottleneck Method*

Note, however, that for us to make use of the Blahut-Arimoto algorithm for computing an optimal coder-decoder pair (for a given β), a distortion metric d is required. However, as discussed by Tishby et al. [318], this requirement places *all* of the burden of relevant information onto choice of metric; RD theory defines "relevant" information by choice of a distortion function—codes are said to capture relevant information if they achieve

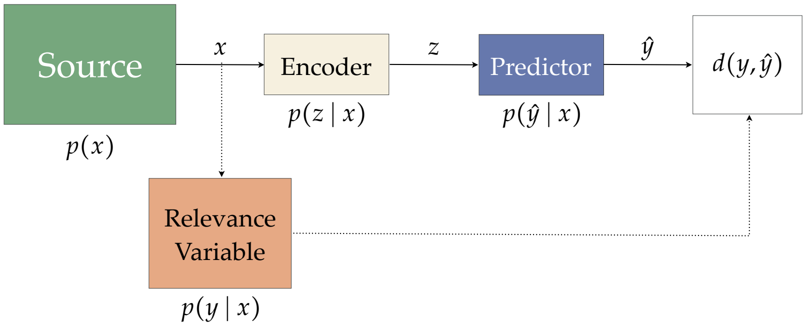


Figure 5.4: The Information Bottleneck.

low distortion. How, though, should such a metric be chosen if the signal being transmitted were to represent an image? It is not obvious whether precise pixel values are the important thing to preserve, rather than the overall contents of the image at the level of objects, scenes, and relations. A simple pixel inversion or image rotation will surely yield extremely high increase for many natural choices of metric, but in many contexts such transformations don't actually destroy relevant information.

In this sense, the choice of metric determines what counts as relevant information. In general, it is desirable to allow for a less restrictive choice of relevant information.

The Information Bottleneck (IB) Method is one possible remedy to this problem. The IB defines relevant information according to how well a random variable Y can be predicted from each $\tilde{x} \in \tilde{\mathcal{X}}$, as pictured in Figure 5.4. For the IB to make sense, we must suppose that $I(X;Y) > 0$, and that the coder–decoder scheme has access to the joint probability mass function (pmf) $p(x,y)$. IB then recasts the RD lower bound in Equation 5.7 in terms of prediction of Y given $\tilde{X}$. The optimal assignment to the distribution $p(\tilde{x} \mid x)$ is then given by minimizing:

$$\mathcal{L}[p(\tilde{x} \mid x)] = I(\tilde{X};X) - \beta I(\tilde{X};Y), \tag{5.12}$$

where $\beta \in \mathbb{R}_{\geq 0}$ is again a Lagrange multiplier attached to the meaningful information. Like BA, choice of β determines the relative preference between compression (rate) and predicting Y (distortion); when $\beta = 0$, the coder can ignore Y entirely, and so is free to compress arbitrarily. Conversely, as $\beta \to \infty$, the coder must prioritize prediction of Y, requiring more bits in the coding alphabet.

Tishby et al. [318] offer a convergent algorithm for solving the above optimization problem.

Theorem 5.1. *(Appears as Theorem 5 by Tishby et al. [318]) Equation 5.12 yields the following optimization problem:*

$$\min_{p(\tilde{x}\mid x)} \mathcal{L}_{IB}\left[p(\tilde{x} \mid x); p(\tilde{x}); p(y \mid \tilde{x})\right] = \min_{p(\tilde{x}\mid x)} \left(I(X;\tilde{X}) + \beta \underset{p(x,\tilde{x})}{\mathbb{E}} \left[D_{KL}(p(y \mid x) \parallel p(y \mid \tilde{x})\right]\right). \quad (5.13)$$

The algorithm consists of the following three steps, which, when repeated, converge to a local minima of the above optimization problem, with $Z(\beta, x)$ a normalizing term:

$$\begin{cases} p_t(\tilde{x} \mid x) \leftarrow \frac{p_t(\tilde{x})}{Z(\beta,x)} \exp(-\beta D_{KL}(p(y \mid x) \parallel p_t(y \mid \tilde{x}))), \\ p_{t+1}(\tilde{x}) \leftarrow \sum_x p(x) p_t(\tilde{x} \mid x), \\ p_{t+1}(y \mid \tilde{x}) \leftarrow \sum_y p(y \mid x) p_t(x \mid \tilde{x}). \end{cases} \quad (5.14)$$

It is important to note that the algorithm only presents a *locally optimal* solution to the above optimization problem. To the best of our knowledge, there is no known efficient algorithm for computing the global optimum. Mumey and Gedeon [245] show that a closely related problem to finding the global optimum in the above is in fact NP-hard, suggesting that local convergence or approximation is likely our best option. Additionally, if the support of $p(y \mid x)$ and $p(y \mid \tilde{x})$ does not *exactly* overlap, then D_{KL} is trivially infinity, leading to vacuous updates. It is thus important that application of their algorithm be applied in a context with overlapping supports.

THE DETERMINISTIC IB (DIB). Strouse and Schwab [302] extend IB by focusing on deterministic coding functions where $p(\tilde{x} \mid x) \equiv f : \mathcal{X} \to \tilde{\mathcal{X}}$. Given the equality $I(X;Y) =$

$H(X) - H(X \mid Y)$, note that when the coder is a deterministic function f, we can replace the mutual information term in the objective by the entropy of the latent space:

$$\min_{f(x)} \mathcal{L}_{\text{DIB}}\left[f(x); p(\tilde{x}); p(y \mid x)\right] = \min_{f(x)} \left(H(\tilde{X}) + \beta \mathop{\mathbb{E}}_{p(x)} \left[D_{\text{KL}}(p(y \mid x) \,\|\, p(y \mid \tilde{x}))\right] \right). \tag{5.15}$$

Given that state abstractions are often deterministic, I will primarily be focused on this extension.

5.2 ANALYSIS: STATE ABSTRACTION AS COMPRESSION

I now adapt the Information Bottleneck to construct state abstractions for sequential decision making problems. The proposed framework is pictured in Figure 5.1, with the information generating source defined as ρ_E, the stationary distribution in the given MDP induced by the expert policy, π_E. That is, the source distribution is defined as the γ-discounted stationary distribution $\rho_E(s)$, for each $s \in \mathcal{S}$, for a given start state distribution ρ_0, as:

$$\rho_E(s) := \sum_{t=0}^{\infty} \gamma^t \mathbb{P}\{s_t = s \mid \rho_0, \pi_E\} \tag{5.16}$$

My goal is to answer the following question: *How many abstract states are needed for an agent to faithfully make similar decisions to an expert demonstrator?* To answer this question, I cast the Rate-Distortion trade off as one between (1) the size of the abstract state space $|\mathcal{S}_\phi|$, and (2) the value of the best policy representable using $\mathcal{S}_\phi$ compared to π_E.

One might wonder why such a question cannot be answered by assigning one abstract state to each action, as is captured by the π^*-irrelevance abstractions studied by Jong and Stone [147] and Li et al. [203]. First, if the demonstrator policy is stochastic, no such abstraction exists. Second, we are ultimately interested in state abstractions that facilitate effective *learning*; if the abstraction were given to an arbitrary RL algorithm, we would like learning to be made easier. Highly aggressive abstraction types like π^* destroy guarantees and make aspects of learning harder [203, 5]. Lastly, π^*-irrelevance only captures lossless abstraction; through RD, we can build a toward theory of lossy compression for RL.

More formally, I introduce and study the following objective:

Definition 5.6. *The objective function, $\mathcal{J}$, for a given Lagrange multiplier $\beta \in \mathbb{R}_{\geq 0}$, is defined as:*

$$\mathcal{J}\left[\phi\right] := |\mathcal{S}_\phi| + \beta \mathop{\mathbb{E}}_{\rho_E(s)} \left[V^{\pi_E}(s) - V^{\pi^*_\phi}(\phi(s)) \right]. \tag{5.17}$$

The goal is to define an algorithm that efficiently minimizes the above objective.

Eventually, I will introduce an algorithm that minimizes an upper bound on the CVA objective. To related this upper bound, we require the following definition, denoting the size of the non-negligibly used portion of an alphabet under a pmf:

Definition 5.7. *The pmf-used alphabet size of $\mathcal{X}$ is the number of elements whose probability under $p(x)$ is greater than some negligibility threshold $\delta_{min} \in (0, 1)$:*

$$|\mathcal{X}|^{\delta_{min}}_{p(x)} := \min\left\{ |\{x \in \mathcal{X} : p(x) > \delta_{min}\}|, |\mathcal{X}| \right\}. \tag{5.18}$$

This notion of alphabet size generalizes the usual method of measuring the size of a state space. When we think about the CVA objective, the state space size will be thought of in relation to this notion of state space size, under a given state distribution.

DIB Upper Bounds the CVA Objective

Recall that the given MDP M paired with the fixed control policy π_E defines an information-generating source. At each time step, a state is sampled from ρ_E and given to a learning agent through a state-abstraction function, $\phi : \mathcal{S} \rightarrow \mathcal{S}_\phi$, which projects each state to each abstract state s_ϕ. I make the additional simplifying assumption that there exists a fixed policy π_E that controls the MDP. The agent's goal is to perform as well as the demonstrator using as small of a state space as possible, as reflected by $\mathcal{J}$. This reference policy π_E may be the optimal policy π^*, but it could also be something else, such as the agent's policy on a previous episode.

I now construct the IB and DIB analogue objectives. First, let $I(S; S_\phi)$ denote the rate, where S is a random variable indicating the probability of arriving in each state under ρ_E, and S_ϕ is a random variable indicating the probability of arriving in each abstract state under ρ_E and projecting each ground state through ϕ. Second, following the IB, let $D_{\mathrm{KL}}(\pi_E(\cdot \mid s) \mid\mid \pi_\phi(\cdot \mid s_\phi))$ denote the distortion for a given state s. The total distortion, then, is the KL in expectation under ρ_E:

$$\mathbb{E}_{s \sim \rho_E}\left[D_{\mathrm{KL}}(\pi_E(\cdot \mid s) \mid\mid \pi_\phi(\cdot \mid \phi(s_\phi)))\right]. \tag{5.19}$$

Further, suppose there exists a fixed, deterministic mapping from $\mathcal{S}_\phi$ to $\tilde{\mathcal{S}}$, with $\tilde{\mathcal{S}} = \mathcal{S}$. Thus, the distribution $p(\tilde{x} \mid x)$ is simply $p(s_\phi \mid s)$, which I henceforth abbreviate as ϕ. Consequently, the following alignments emerge between the present objects of study (abstractions, policies) and those studied by the IB:

$$p(\tilde{x} \mid x) \rightsquigarrow \phi, \quad p(\tilde{x}) \rightsquigarrow \rho_\phi, \quad p(y \mid \tilde{x}) \rightsquigarrow \pi_\phi, \tag{5.20}$$

where ρ_ϕ is the stationary distribution over abstract states induced by π_ϕ and ϕ. Thus, per Theorem 5.1, I next construct an objective function $\hat{\mathcal{J}}$ based on the IB:

$$\hat{\mathcal{J}}_{\mathrm{IB}}\left[\phi; \rho_\phi; \pi_\phi\right] := I(S; S_\phi) + \underset{s \sim \rho_E, s_\phi \sim \phi(s)}{\mathbb{E}}[\beta D_{\mathrm{KL}}(\pi_E(\cdot \mid s) \mid\mid \pi_\phi(\cdot \mid s_\phi))]. \tag{5.21}$$

If we choose to use the DIB instead, then we only consider deterministic state abstraction functions $\phi : \mathcal{S} \to \mathcal{S}_\phi$, and so $H(S_\phi \mid S) = 0$. Therefore, the DIB analogue objective is expressed as:

$$\hat{\mathcal{J}}_{\text{DIB}}\left[\phi; \rho_\phi; \pi_\phi\right] := H(S_\phi) \ + \mathop{\mathbb{E}}_{s \sim \rho_E}\left[\beta D_{\text{KL}}(\pi_E(\cdot \mid s) \,\|\, \pi_\phi(\cdot \mid \phi(s)))\right]. \tag{5.22}$$

With these objectives in place, I now build toward the main theorem of this chapter, which relates $\hat{\mathcal{J}}_{\text{DIB}}$ to $\mathcal{J}$. To prove the theorem, I first introduce two key lemmas. The first relates the entropy of a pmf to the maximum size of the alphabet used by that pmf:

Lemma 5.1. *Consider a discrete random variable X, with alphabet $\mathcal{X}$ and some pmf $p(x)$. For a given threshold $\delta_{min} \in (0,1)$, the pmf-used alphabet size of the alphabet is bounded, relative to some pmf $p(x)$:*

$$|\mathcal{X}|_{p(x)}^{\delta_{min}} \leq \frac{H(X)}{\delta_{min} \log\left(\frac{1}{\delta_{min}}\right)}. \tag{5.23}$$

Proof of Lemma 5.1.

> First, recall the definition of the entropy $H(X)$ of a discrete random variable X, taking on values $x \in \mathcal{X}$,
>
> $$H(X) := -\sum_{x \in \mathcal{X}} p(x) \log_2 p(x). \tag{5.24}$$
>
> Then, for a given maximum entropy $H(X) \leq N$ and $\delta_{min} \in \mathbb{R}_{>0}$, we seek an upper bound on $|\mathcal{X}|_{p(x)}^{\delta_{min}}$.
>
> That is, we would like to upper bound the following quantity:
>
> $$\max_{p(x): H(X) \leq N} |\mathcal{X}|_{p(x)}^{\delta_{min}}. \tag{5.25}$$

Note that this quantity is maximized by a uniform distribution that applies δ_{min} mass to each element in the support, across the largest alphabet where $H(X) = N$:

$$H(X) = -\sum_{x \in \mathcal{X}'} p(x) \log_2 p(x) \tag{5.26}$$

$$= -\sum_{x \in \mathcal{X}'} \delta_{min} \log_2 \delta_{min} \tag{5.27}$$

$$= -|\mathcal{X}'| \delta_{min} \log_2 \delta_{min} \tag{5.28}$$

$$= |\mathcal{X}'| \delta_{min} \log_2 \frac{1}{\delta_{min}}. \tag{5.29}$$

Therefore, for a given pmf $p(x)$ with entropy $H(X)$, and a minimum threshold of probability, the minimum size of the alphabet $\mathcal{X}$ is upper bounded:

$$|\mathcal{X}| \leq \frac{H(X)}{\delta_{min} \log_2 \frac{1}{\delta_{min}}}. \tag{5.30}$$

□

This bound is relatively loose; we know trivially that $H(X) \leq \log_2 |\mathcal{X}|$. Thus, in the worst case, the bound can be up to $\tilde{O}\left(1/\delta_{min}\right)$ times larger than the true alphabet. Still, this result allows us to relate the entropy of a random variable with its used alphabet size. Further, by definition, the entropy of the abstract stationary distribution, $H(\rho_\phi)$, gives us a lower bound on the number of bits needed to represent the used parts of $\mathcal{S}_\phi$. In this way, the entropy as a measure of compression is exploiting the fact that the most probable state can be written as `0`, the second most probable state as `10`, and so on. Thus, a lower entropy is already indicative of reducing $|\mathcal{S}_\phi|$. Further, in experiments, we will find this upper bound is loose relative to the size of the abstract state space the algorithm produces.

Next, I introduce a second lemma that relates the expected KL divergence between two policies to the difference in value achieved by the policies, in expectation under some state distribution:

Lemma 5.2. *Consider two stochastic policies, π_1 and π_2 on state space $\mathcal{S}$, and a fixed probability distribution over $\mathcal{S}$, $p(s)$. If, for some $k \in \mathbb{R}_{\geq 0}$:*

$$\mathop{\mathbb{E}}_{p(s)} \left[D_{KL}(\pi_1(\cdot \mid s) \parallel \pi_2(\cdot \mid s)) \right] \leq k, \tag{5.31}$$

then:

$$\mathop{\mathbb{E}}_{p(s)} \left[V^{\pi_1}(s) - V^{\pi_2}(s) \right] \leq \sqrt{2k}\text{VM}\textsc{ax}, \tag{5.32}$$

where VM<span>ax</span> *is an upper bound on the value-function.*

Proof of Lemma 5.2.

Recall the total variation distance (TVD) between our two policies for a given state s is defined as:

$$TV(\pi_E(\cdot \mid s), \pi_\phi(\cdot \mid s)) = \sup_{a \in \mathcal{A}} |\pi_E(a \mid s) - \pi_\phi(a \mid s)|. \tag{5.33}$$

Furthermore, recall that TVD relates to the L_1 norm and the KL divergence:

$$\begin{aligned} TV(\pi_E(\cdot \mid s), \pi_\phi(\cdot \mid s)) &= \frac{1}{2} \sum_{a \in \mathcal{A}} |\pi_E(\cdot \mid s) - \pi_\phi(\cdot \mid s)| & (5.34) \\ &\leq \sqrt{\frac{1}{2} D_{\text{KL}}(\pi_E(\cdot \mid s) \parallel \pi_\phi(\cdot \mid s))}, & (5.35) \end{aligned}$$

where the inequality in Equation 5.35 is formally known as Pinsker's inequality. With this inequality in place, we expand the expectation in the value bound:

$$\begin{aligned} &\mathbb{E}_{p(s)}[V^{\pi_E}(s) - V^{\pi_\phi}(s)] & (5.36) \\ &\leq \sum_{s \in \mathcal{S}} p(s) \left(\sum_{a \in \mathcal{A}} |\pi_E(a \mid s) - \pi_\phi(a \mid s)| (R(s,a) + \gamma \sum_{s'} T(s' \mid s, a) |V^{\pi_E}(s') - V^{\pi_\phi}(s')| \right) \\ &= \sum_{s \in \mathcal{S}} \sum_{a \in \mathcal{A}} p(s) |\pi_E(a \mid s) - \pi_\phi(a \mid s)| \left(R(s,a) + \gamma \sum_{s'} T(s' \mid s, a) |V^{\pi_E}(s') - V^{\pi_\phi}(s')| \right) \end{aligned}$$

Then, applying the upper bound on the possible value $\text{VMax} = \text{RMax}/(1-\gamma)$ to Equation 5.36:

$$\mathbb{E}_{p(s)}[V^{\pi_E}(s) - V^{\pi_\phi}(s)] \leq \text{VMax}\ \mathbb{E}_{p(s)}\left[|\pi_E(a \mid s) - \pi_\phi(a \mid s)|\right]. \tag{5.37}$$

Then, by Pinsker's inequality, we conclude:

$$\begin{aligned}\mathbb{E}_{p(s)}[V^{\pi_E}(s) - V^{\pi_\phi}(s)] &\leq 2\text{VMax}\ \mathbb{E}_{p(s)}\left[\sqrt{\frac{1}{2}D_{\text{KL}}(\pi_E(a \mid s) \mid\mid \pi_\phi(a \mid s))}\right] \quad (5.38)\\ &\leq \sqrt{2k}\text{VMax}. \quad \square\end{aligned}$$

The above bound relates the distortion measure present in IB to that of the CVA objective. Note that this bound is vacuous for values of $k \geq \frac{1}{2}$.

With these lemmas in place, I now present the theorem.

Theorem 5.2. *A variation of the DIB objective $\hat{\mathcal{J}}_{DIB}$ is an upper bound for the CVA objective, $\mathcal{J}$, where state space size is treated as $|\mathcal{S}_\phi|^{\delta_{min}}_{\rho_\phi(s)}$. Formally, for all $\phi \in \Phi$:*

$$|\mathcal{S}_\phi|^{\delta_{min}}_{\rho_\phi(s)} + \beta \mathop{\mathbb{E}}_{\rho_E(s)}\left[V^{\pi_E}(s) - V^{\pi^*_\phi}(s)\right] \leq \frac{H(\rho_\phi)}{\delta \log \frac{1}{\delta}} + 2\text{VMax}\beta \mathop{\mathbb{E}}_{\rho_E(s)}\left[D_{KL}(\pi_E(s) \mid\mid \pi^*_\phi(s)\right]. \tag{5.39}$$

Proof of Theorem 5.2.

The proof follows from Lemma 5.1 and Equation 5.2. Consider the ϕ that minimizes $\hat{\mathcal{J}}_{\text{DIB}}$, yielding the value of at most $N + \beta k$, where:

$$N := H(S_\phi) \tag{5.40}$$

$$k := \mathop{\mathbb{E}}_{s \sim \rho_E}\left[D_{KL}(\pi_E(a \mid s) \mid\mid \pi_\phi(a \mid \phi(s))\right]. \tag{5.41}$$

Then, by Lemma 5.1, we know:

$$|\mathcal{S}_\phi|_{\rho_\phi(s)}^{\delta_{min}} \leq \frac{N}{\delta_{min}\log_2 \frac{1}{\delta_{min}}}. \tag{5.42}$$

By Equation 5.2, we know:

$$\mathbb{E}_{\rho(s)}\left[V^{\pi_E}(s) - V^{\pi_\phi}(s)\right] \leq \sqrt{2k}\text{VM}\textsc{ax}. \tag{5.43}$$

Therefore, since both quantities are non-negative, we conclude:

$$\mathcal{J}[\phi] = |\mathcal{S}_\phi|_{\rho_\phi(s)}^{\delta_{min}} + \beta\mathbb{E}_{\rho(s)}\left[V^{\pi_E}(s) - V^{\pi_\phi}(s)\right] \tag{5.44}$$

$$\leq \frac{N}{\delta_{min}\log_2 \frac{1}{\delta_{min}}} + \beta\sqrt{2k}\text{VM}\textsc{ax}. \tag{5.45}$$

Thus, we can upper bound the quantities in $\mathcal{J}$ as a function of the quantities in $\hat{\mathcal{J}}_{\text{DIB}}$. □

This theorem tells us that the optimization problem presented by $\hat{\mathcal{J}}_{\text{DIB}}$ can be well approximated by the usual IB method. I thus introduce Determinstic Information Bottleneck for State abstractions (Dibs, presented in Algorithm 5.1), a simple iterative algorithm that adapts the DIB to Apprenticeship Learning with state abstractions. Dibs outputs a state-abstraction–policy pair in finite time that computes a local minimum of $\hat{\mathcal{J}}_{\text{DIB}}$, which we know from Theorem 5.2 is an upper bound on $\mathcal{J}$. The pseudocode presented is for the *deterministic* variant of the IB, as often state abstraction functions are treated as deterministic aggregation functions [203]. The stochastic variant, which I call Sibs, will also be of interest, as soft state aggregation has been explored as well [294].

5.3 EXPERIMENTS

I now describe several experiments that explore the power of Dibs for constructing abstractions that trade off between compression and value. First, I study the traditional

Algorithm 5.1 DIBS

```
INPUT: $\pi_E, \rho_E, M, \beta, \Delta, iters$
OUTPUT: $\phi, \pi_\phi$

$\forall_s : \phi_0(s) = $ random.choice([1, $|\mathcal{S}|$])          ▷ Initialize
$\forall_s : \pi_{\phi_0}(a \mid s_\phi) \sim \text{Unif}(\mathcal{A})$
$\forall_s : \rho_{\phi,0}(s_\phi) \sim \text{Unif}([1, |\mathcal{S}|])$
for $t = 0$ to $iters$ do                                                ▷ Iterative updates
    $j_{t+1}(\phi_t(s)) = \log \rho_{\phi,t}(\phi_t(s)) - \beta D_{\text{KL}}(\pi_E(\cdot \mid s) \,||\, \pi_{\phi,t}(\cdot \mid \phi_t(s)))$
    $\phi_{t+1}(s) = \arg\max_{s_\phi} j_{t+1}(s_\phi)$
    $\rho_{\phi,t+1}(s_\phi) = \sum_{s:\phi_t(s)=s_\phi} \rho_E(s)$
    $\pi_{\phi,t+1}(a \mid s_\phi) = \frac{\sum_{s:\phi_t(s)=s_\phi} \pi_E(a|s)\rho_E(s)}{\sum_{s:\phi_t(s)=s_\phi} \rho_E(s)}$
    if $\max_{f \in \{\pi_\phi, \phi, \rho_\phi\}} \text{L}_1(f_t, f_{t+1}) \leq \Delta$ then      ▷ Check convergence
        break
    end if
end for
return $\phi_{t+1}, \pi_{\phi,t+1}$
```

Four Rooms domain discussed in Chapter 2. Second, I present a simple extension to Sibs that scales to high-dimensional state spaces and evaluate this extension in the Atari game Breakout using the Arcade Learning Environment (ALE) [39]. The code for running these experiments is freely available for reproduction and extension.[5.1]

5.3.1 *Four Rooms*

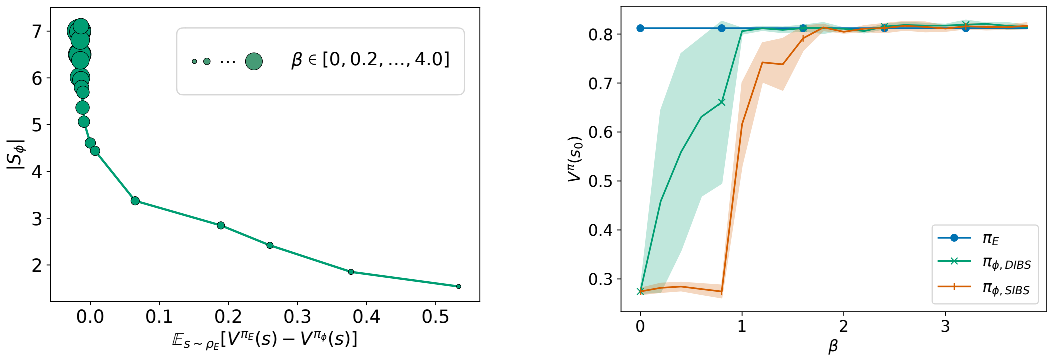
I first investigate the power of Dibs to appropriately trade off between value loss and compression.

The first experiment focuses on the Four Rooms grid world domain discussed in Figure 2.7. Recall that the agent interacts with an 11×11 grid with walls dividing the world into four connected rooms. The agent has four actions, `up`, `left`, `down`, and `right`. Each action moves the agent in the specified direction with probability 0.9 (unless it hits a wall), and orthogonally with probability 0.05. The agent starts in the bottom left corner, and receives +1 reward for transitioning into the top right state, which is terminal. All other transitions receive 0 reward. I set γ to 0.99. For simplicity, I set the expert policy π_E to be

[5.1] https://github.com/david-abel/rl_info_theory the optimal policy, with an additional $\varepsilon = 0.05$ probability of taking an action at random to ensure that an arbitrary stochastic policy over the action space has overlapping support with the expert policy.

In Four Rooms, I run Dibs and Sibs to convergence and compare the value of $\pi_{\phi,\text{Dibs}}$ and $\pi_{\phi,\text{Sibs}}$ to the value of the demonstrator policy for β between 0 and 4, incrementing by 0.2. I determine convergence as per line 9 of Algorithm 5.1: if all updating functions change by no more than Δ, the algorithm has converged. I set Δ to 0.001, an arbitrarily chosen small constant.

Figure 5.5a illustrates the rate-distortion trade off made by 500 different runs of Dibs with different settings of β ranging from 0.0 to 4.0: each point indicates the size of the average abstract state space (y-axis) and value of the abstract policy relative to the demonstrator (x-axis), achieved by the computed state abstraction for the given β. The large values of β correspond to larger circles, with the maximal value of $\beta = 4.0$ appearing in the top left of the curve, and the smallest of $\beta = 0.0$ appearing at the bottom right. Since the demonstrator is in fact suboptimal due to the ε-randomness, it is possible for the abstract policy to do slightly better than the demonstrator in terms of reward, as is the case for all points with x-value less than 0.0. However, there is no incentive in the objective for this



(a) Trade off made by the ϕ's output by Dibs for different values of β.

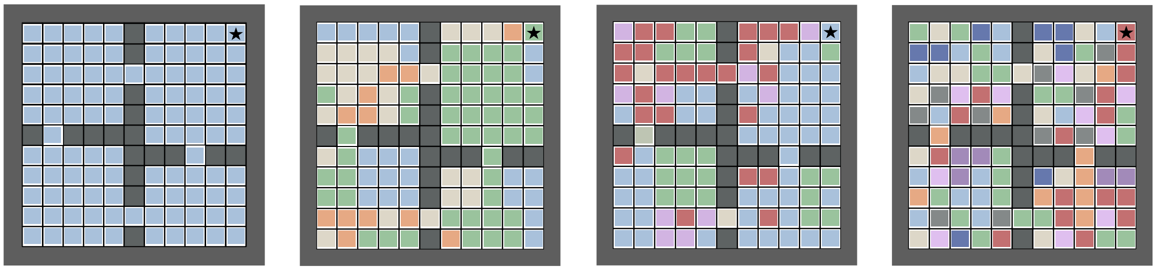
(b) Four Rooms: $V^\pi(s_0)$ vs. β

Figure 5.5: (a) The average rate-distortion trade off made by Dibs as β varies, and (b) The average value of the ϕ, π_ϕ pairs found by Dibs for different values of β.

to be the case, as reward is not yet incorporated into learning in any way. Observe that when $\beta = 0$, DIBS prioritizes compression, yielding a one-state MDP on average (the far right point). As β increases (which moves along the line to the left), the algorithm gradually tips the trade off from prioritizing compression to prioritizing performance. As β increases, we see the abstract policy achieve the same value as the demonstrator. Also of note is that a two-state abstract space is capable of representing a policy (which could be stochastic) that is nearly as effective as the expert policy. With a one state MDP, however, the best policies found by DIBS still yield around 0.45 expected value loss relative to π_E.

Figure 5.5b offers a slightly different perspective on the same results. Here, I present the average value of the abstract policy achieved as a function of β, with β again varying from 0 to 4.0. The results are averaged over the same 500 runs, with the lines indicating the average and shaded regions denoting 95% confidence intervals. Notably, when β is 0, the algorithm tends to find a policy that achieves significantly worse than the expert. With DIBS, as β increases, we see rapid improvement in the quality of the discovered policy, up until $\beta = 1$, at which point the abstract policy achieves almost identical performance to the expert. In contrast, the stochastic variant SIBS sees effectively no improvement in the quality of the policy until $\beta = 1$. I conjecture that this is due to the more difficult optimization problem presented, as the space of probabilistic state abstraction may be much harder to search through than the space of deterministic ones. Still, as β increases past one, both $\pi_{\phi,\text{DIBS}}$ and $\pi_{\phi,\text{SIBS}}$ nearly match the value of the control policy.

Figure 5.6a, Figure 5.6b, Figure 5.6c, and Figure 5.6d show the state abstractions found by DIBS for $\beta = 0$, $\beta = 1$, $\beta = 2$, and $\beta = 20$ respectively. Notably, each of these state abstractions were sufficient for effectively solving the problem escept for the $\beta = 0$ case. For the abstraction in Figure 5.6b, there are four abstract states, which is sufficient for nearly representing a π^*-irrelevance abstraction of the demonstrator policy ("move up" in green, "move right" in tan, and so on). As β is increased, observe that the abstraction yields far more states, though the quality of the optimal policy is already as high as it can go (there is simply less pressure to compress). Note that these experiments only examine



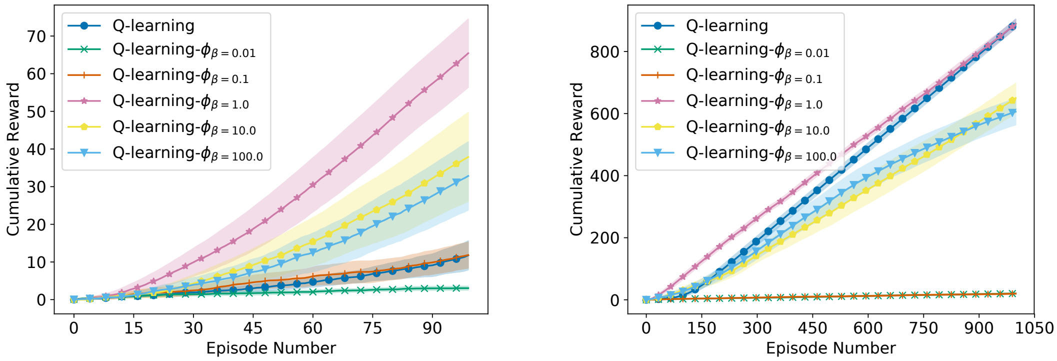
(a) ϕ with $\beta = 0$, $|\mathcal{S}_\phi| = 1$ (b) ϕ with $\beta = 1$, $|\mathcal{S}_\phi| = 4$ (c) ϕ with $\beta = 2$, $|\mathcal{S}_\phi| = 5$ (d) ϕ with $\beta = 20$, $|\mathcal{S}_\phi| = 9$

Figure 5.6: The state abstractions found by Dibs in the Four Rooms domain when (a) $\beta = 0$, (b) $\beta = 1$, (c) $\beta = 2$, and (d) $\beta = 20$.

the effect of the state abstractions found by Dibs on representation of the optimal policy. However, intuitively, the abstract state spaces pictured in Figure 5.6b, Figure 5.6c, and Figure 5.6d will give rise to different degrees of *learning* difficulty, even though they can each represent a near-optimal policy.

I now investigate the impact of these state abstractions on learning. In particular, I study how the resulting state abstractions change learning for simple RL algorithms in the Four Rooms grid world. The experiment proceeds as follows. First, before any RL algorithm interacts with an MDP, I construct state abstractions using Dibs using different values of β, with all other settings as described in the previous experiment. Then, I give the resulting state abstractions to Q-learning on Four Rooms and contrast the different algorithm-ϕ pairs' behavior.

Results are presented in Figure 5.7. Each plot is the result of 25 runs of the experiment, indicating the mean cumulative reward of different learning algorithm–ϕ combinations. The β parameter was chosen to range from 0.01 up to 100.0, with each order of magnitude in between. Observe that, with a smaller data set (left), Q-Learning is able to more quickly find a better policy using state abstractions resulting from middle choices of β. In particular, the best performance within 100 episodes is obtained by the approach using $\beta = 1.0$. As β decreases, performance decays rapidly, and as β increases, performance worsens as well. In contrast, with a *larger* data set—that is, when the number of episodes



(a) Q-learning, 100 episodes

(b) Q-learning, 1,000 episodes

Figure 5.7: A comparison of how ϕ with different choice of β impacts simple RL in the Four Rooms domain.

is set to 1,000—Q-learning is able to ultimately overtake all approaches using a state abstraction, but less efficiently than when β is set to the ideal value of 1 for the domain. This is precisely because Q-learning is guaranteed to eventually find the optimal policy, whereas the variations using ϕ will learn a best-in-class policy, but relative to a restricted class. This is precisely a bias-complexity trade off; choice of β is determining how large the abstract state space is, and thus the space of possible Q functions learnable by the algorithm.

5.3.2 *Breakout*

I next translate the proposed algorithmic framework into domains with high-dimensional observations. To do so, I turn to variational autoencoders (VAEs) [163]. In a VAE, we are concerned with learning a compact latent data representation, z, that captures a high-dimensional observation space, x, through the use of two parameterized functions $q_\psi(z \mid x)$ and $p_\theta(x \mid z)$. The pair represent a probabilistic encoder and decoder, typically captured by two separate neural networks, where the former maps data to a latent representation and the latter maps from z to the original observation. Traditionally, the two models are trained jointly to optimize the evidence lower bound objective (ELBO), which maximizes $\mathbb{E}_{q_\psi(z|x)}[\log p_\theta(x \mid z)]$ to facilitate reconstruction of the original data and minimizes $D_{\mathrm{KL}}(q_\psi(z \mid x) \parallel p(z))$ to keep $q_\psi(z \mid x)$ close to some prior $p(z)$ over latent codes. Both $q_\psi(z \mid x)$ and $p(z)$ are commonly treated as Gaussian to use the Gaussian reparameterization trick [163]. Since the ELBO is optimized in expectation over the data distribution, $p(x)$, we can leverage a known result regarding the KL divergence [161]:

$$\mathbb{E}_{p(x)}[D_{\mathrm{KL}}(q_\psi(\cdot \mid x) \parallel p(z))] = I(X;Z) + D_{\mathrm{KL}}(q(z) \parallel p(z)) \geq I(X;Z), \tag{5.46}$$

where $q(z) = \mathbb{E}_{p(x)}[q_\psi(z \mid x)]$. I treat x as the ground state representation $\mathcal{S}$ and z as the abstract state $\mathcal{S}_\phi$. I derive a new objective function that serves as a variational upper bound to the stochastic IB (Sibs) objective derived in Equation 5.21:

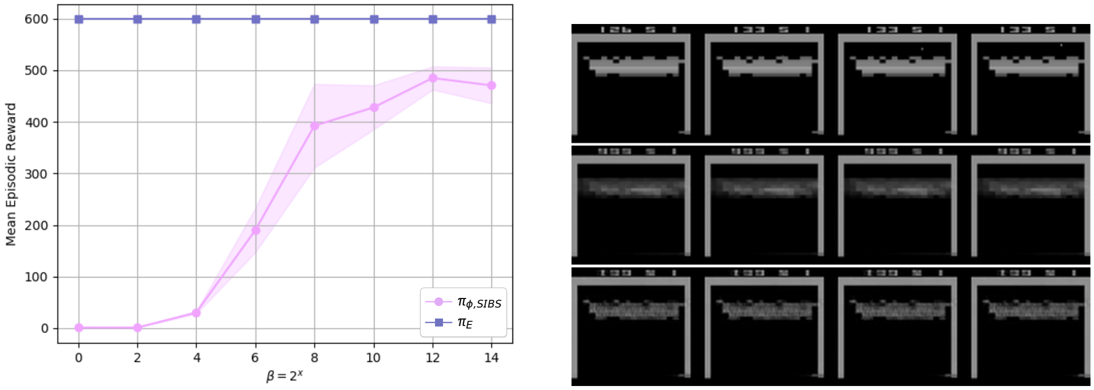
$$\min_{\phi} \mathop{\mathbb{E}}_{s \sim \rho_E} [D_{\mathrm{KL}}(q_\psi(\cdot \mid s) \parallel p(s_\phi)) \ + \beta \mathop{\mathbb{E}}_{s_\phi \sim \phi(s)} [D_{\mathrm{KL}}(\pi_E(\cdot \mid s) \parallel \pi_\phi(\cdot \mid s_\phi))]], \tag{5.47}$$

$$\geq \min_{\phi} I(S; S_\phi) \ + \beta \mathop{\mathbb{E}}_{s \sim \rho_E, s_\phi \sim \phi(s)} [D_{\mathrm{KL}}(\pi_E(\cdot \mid s) \parallel \pi_\phi(\cdot \mid s_\phi))], \tag{5.48}$$

where the upper bound follows directly from Equation 5.46.

To make use of this upper bound, we first create a demonstrator policy π_E for the Atari game Breakout [39] using A2C [239]. A Gaussian VAE agent is then trained with a separate architecture that has the same first four layers as the A2C agent before mapping out to a mean and covariance (in $\mathbb{R}^{25}$). Instead of reconstructing states, this decoder serves as an abstract policy network, mapping to a final distribution over the primitive actions, $\pi_\phi(a \mid s)$, which is really $\pi_\phi(a \mid z)$. The model is trained via Equation 5.47 for 2000 episodes using the Adam optimizer [162] with a learning rate of 0.0001. During training, the expert's policy (π_E) controls the MDP.

Figure 5.8 presents results showcasing the effect of β on compression and performance in. The data suggest a relationship similar to that of Figure 5.5) between choice of β, success in approximating the demonstrator policy, and the nature of the resulting state



(a) Breakout: Mean Reward vs. β

(b) Original (top) and ϕ with $\beta = 2$ (middle) and $\beta = 2048$ (bottom)

Figure 5.8: (a) The mean reward over 100 evaluation episodes of ϕ, π_ϕ combinations found by the VAE-approximation to Sibs for different values of β, and (b) attempted state reconstructions using fixed abstractions found when $\beta = 2$ and $\beta = 2048$.

abstraction. In the visualizations of the abstraction, observe that the quality of state reconstruction (each state is a row of four consecutive game screens) is compromised under a low setting of β (prioritizing compression), whereas a higher value of β preserves more information (paddle position and shape of bricks), leading to higher-quality reconstruction. Seeing that agent performance converges below the expert policy π_E, it is possible that increasing the size of the latent bottleneck may close the gap. Of critical importance to this setup is the use of a reconstruction network to visualize the latent state information, shown in Figure 5.8b. Due to the method of reconstruction, it may be difficult to determined what information is truly represented as opposed to what information is easily captured in the reconstruction.

So far, I have articulated a new formalism for treating state abstraction as compression in MDPs. To make the setting concrete, I adopt the apprenticeship learning perspective and assumed query access to an expert control policy π_E. Then, using this new perspective, I introduced a new objective and proved that it is well-approximated by an IB-like objective, giving rise to a convergent algorithm for computing state abstractions that trade off between compression and value.

5.4 EXTENSIONS

Many questions remain. First, it is natural to be interested in the case when π_E does not control the MDP, but rather the RL agent's actions determine which states are occupied. In this case, unfortunately, no fixed state distribution is likely to be available. Second, the current formulation concentrates on constructing abstract state spaces that can represent high value policies, but do nothing to ensure that such policies are easy for RL algorithms to discover. An important direction going forward is to identify the role that a well structured state space plays in ensuring low sample complexity RL. Third, much of the chapter has focused on discrete state spaces. It is important to understand whether similar ideas can apply to continuous state spaces, too. Lastly, it is natural to consider the lifelong or multitask setting, in which an RL agent must learn to solve a variety of related tasks. I next discuss study some of these extensions.

5.4.1 *Agents Controls the MDP*

Relaxing the assumption that π_E controls the MDP is essential for extending these ideas to traditional RL. I here propose a path toward removing the control policy π_E by focusing on an intermediate goal: define an algorithm with the same properties as Dibs, but with the learning agent's *non-stationary* policy controlling the underlying MDP instead of π_E. Ultimately, this will give rise to an algorithm that, after $T < \infty$ iterations, can produce an abstraction–policy pair such that, for some state distribution $p(s)$:

$$\mathbb{E}_{p(s)}\left[V^*(s) - V^{\pi_\phi^T}(s)\right] \leq f(\beta, T). \tag{5.49}$$

The most challenging aspect of this setup is that the source distribution is no longer fixed, since the agent's policy will change over time as the agent learns and updates both ϕ and π_ϕ. To this end, I present the following lemma that suggests that two policies that deviate by a bounded amount are guaranteed to share similar stationary distributions.

Lemma 5.3. *Given two policies π_1 and π_2, if $\sup_{s\in\mathcal{S}} \sum_{a\in\mathcal{A}} |\pi_1(a \mid s) - \pi_2(a \mid s)| \leq \Delta$, for $\Delta \in \mathbb{R}_{\geq 0}$, then:*

$$\sum_{s\in\mathcal{S}} |\rho_{\pi_1,s_0}(s) - \rho_{\pi_2,s_0}(s)| \leq \frac{\Delta\gamma}{1-\gamma}, \tag{5.50}$$

where ρ_{π,s_0} denotes the stationary distribution over states under π, starting in state s_0.

Proof of Lemma 5.3.

We bound the difference between the two state distributions after t steps as follows. First, expanding:

$$\begin{aligned}\sum_{s'\in\mathcal{S}} |\rho^t_{\pi_1,s_0}(s') - \rho^t_{\pi_2,s_0}(s')| = \sum_{s'\in\mathcal{S}} |\mathbb{P}(S_t = s' \mid s_0, \pi_1) - \mathbb{P}(S_t = s' \mid s_0, \pi_2)| \\ - \sum_{s\in\mathcal{S}} \mathbb{P}(S_{t-1} = s \mid s_0, \pi_2) \sum_{a\in\mathcal{A}} \pi_2(a \mid s) T(s' \mid s, a)|.\end{aligned} \tag{5.51}$$

Then, by algebra:

$$\begin{aligned}
& \sum_{s'\in\mathcal{S}} |\rho^t_{\pi_1,s_0}(s') - \rho^t_{\pi_2,s_0}(s')| && (5.52)\\
\leq\; & \sum_{s'\in\mathcal{S}} |\sum_{s\in\mathcal{S}} \mathbb{P}(S_{t-1} = s \mid s_0, \pi_1) \sum_{a\in\mathcal{A}} \Big(\pi_1(a \mid s) - \pi_2(a \mid s)\Big) T(s' \mid s, a)| && (5.53)\\
+\; & \sum_{s'\in\mathcal{S}} |\sum_{s\in\mathcal{S}} \Big(\mathbb{P}(S_{t-1} = s \mid s_0, \pi_1) - \mathbb{P}(S_{t-1} = s \mid s_0, \pi_2)\Big) \sum_{a\in\mathcal{A}} \pi_2(a \mid s) T(s' \mid s, a)|.
\end{aligned}$$

Continuing,

$$\begin{aligned}
& \sum_{s'\in\mathcal{S}} |\rho^t_{\pi_1,s_0}(s') - \rho^t_{\pi_2,s_0}(s')| && (5.54)\\
\leq\; & \sum_{s\in\mathcal{S}} \mathbb{P}(S_{t-1} = s \mid s_0, \pi_1) \sum_{a\in\mathcal{A}} \Big|\pi_1(a \mid s) - \pi_2(a \mid s)\Big| \sum_{s'\in\mathcal{S}} T(s' \mid s, a) && (5.55)\\
+\; & \sum_{s\in\mathcal{S}} \Big|\mathbb{P}(S_{t-1} = s \mid s_0, \pi_1) - \mathbb{P}(S_{t-1} = s \mid s_0, \pi_2)\Big| \sum_{a\in\mathcal{A}} \pi_2(a \mid s) \sum_{s'\in\mathcal{S}} T(s' \mid s, a) && \\
\leq\; & \Delta + \sum_{s\in\mathcal{S}} \Big|\mathbb{P}(S_{t-1} = s \mid s_0, \pi_1) - \mathbb{P}(S_{t-1} = s \mid s_0, \pi_2)\Big| && (5.56)\\
=\; & \Delta + \sum_{s'\in\mathcal{S}} |\rho^{t-1}_{\pi_1,s_0}(s') - \rho^{t-1}_{\pi_2,s_0}(s')| && (5.57)
\end{aligned}$$

Applying the above bound, and using induction, we have:

$$\sum_{s'\in\mathcal{S}} |\rho^t_{\pi_1,s_0}(s') - \rho^t_{\pi_2,s_0}(s')| \leq t\Delta \tag{5.58}$$

Therefore,

$$\begin{align}
& \sum_{s\in\mathcal{S}} |\rho_{\pi_1,s_0}(s) - \rho_{\pi_2,s_0}(s)| \tag{5.59}\\
= \;& \sum_{s\in\mathcal{S}} |(1-\gamma)\sum_{t\in\mathbb{N}} \gamma^t \rho^t_{\pi_1,s_0}(s) - (1-\gamma)\sum_{t\in\mathbb{N}} \gamma^t \rho^t_{\pi_2,s_0}(s)| \tag{5.60}\\
\leq \;& (1-\gamma)\sum_{t\in\mathbb{N}} \gamma^t \sum_{s\in\mathcal{S}} |\rho^t_{\pi_1,s_0}(s) - \rho^t_{\pi_2,s_0}(s)| \tag{5.61}\\
\leq \;& (1-\gamma)\sum_{t\in\mathbb{N}} \gamma^t t\Delta = (1-\gamma)\frac{\gamma\Delta}{(1-\gamma)^2} = \frac{\gamma\Delta}{1-\gamma}. \quad \square \tag{5.62}
\end{align}$$

Corollary 5.1. *As a simple corollary of Lemma 5.3, observe that,*

$$\begin{align}
|V^{\pi_1}(s) - V^{\pi_2}(s)| &= |\sum_{t\in\mathbb{N}}\sum_{s\in\mathcal{S}} \rho^t_{\pi_1,s_0}(s)R(s) - \sum_{t}\sum_{s\in\mathcal{S}} \rho^t_{\pi_2,s_0}(s)R(s)| \tag{5.63}\\
&\leq \textsc{RMax} \sum_{t\in\mathbb{N}} (1-\gamma) \sum_{s\in\mathcal{S}} |\rho^t_{\pi_1,s_0}(s) - \rho^t_{\pi_2,s_0}(s)| \tag{5.64}\\
&\leq \textsc{RMax}(1-\gamma) \sum_{t\in\mathbb{N}} \Delta\gamma^t t \tag{5.65}\\
&= \textsc{RMax}\frac{\gamma\Delta}{1-\gamma} = \Delta\gamma\textsc{VMax} \tag{5.66}
\end{align}$$

This lemma is useful as it suggests that policies that are similar to one another will have similar stationary state distributions, too. In particular, in the case where π_E no longer controls the MDP, but rather π_ϕ does, the state distribution of relevance will be $\rho_{\phi,t}$. If, however, over time we can ensure that after some $t > N$ updates, $\pi_{\phi,t} \approx \pi_E$, then we can construct a convergent algorithm for the agent-in-control setting. I leave this analysis as an open question.

I offer an initial variant of this algorithm that I call agent-controlled Dibs (AC-Dibs). I conduct an experiment similar to that of the previous section in the Four Rooms domain.

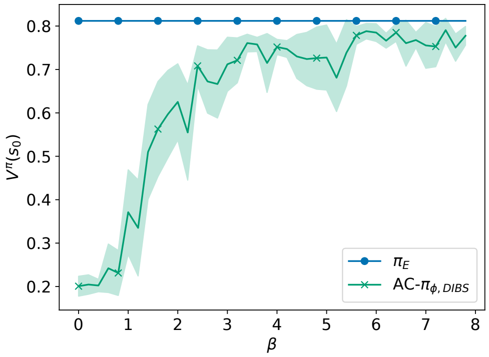


Figure 5.9: The value of the abstract policy found by AC-Dibs for values of β between 0 and 4.

Here, I run the entire process of AC-Dibs for N rounds, each time to convergence, but letting the agent's *initial* policy for that round define the stationary state distribution.

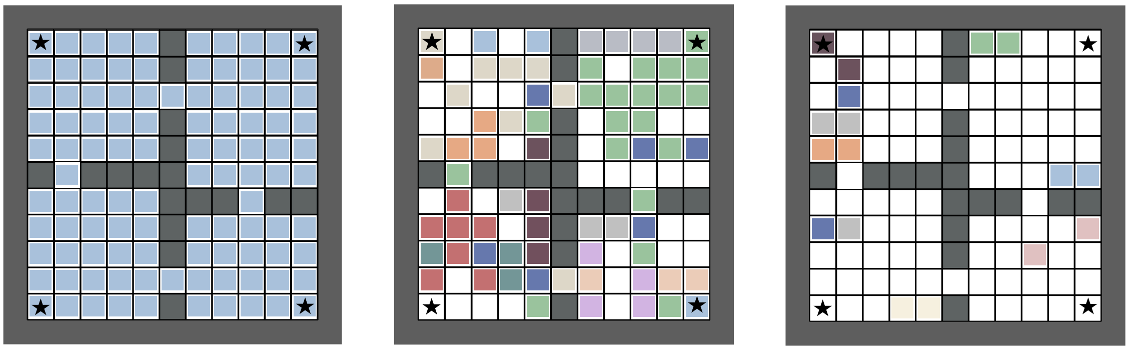
Results are presented in Figure 5.9. Surprisingly, AC-Dibs *always* converges, yielding policies that are similar in value to π_E for $\beta > 2$. This finding supports the previous speculation, along with Lemma 5.3, that there is a feasible route to defining a convergent form of Dibs when the agent's policy controls the underlying MDP.

5.4.2 *Multiple MDPs*

Second, I use our framework to compute a single abstraction sufficient for representing the demonstrator policy across distinct but potentially related tasks. Concretely, I suppose we are given a set of MDPs $\mathcal{M}$, each sharing a state and action space, but are allowed to vary in T, R, and γ.

I conduct an experiment in which $|\mathcal{M}| = 4$, where the four task are defined by a goal being in each of the four corners of the grid world. I run Dibs for each MDP in $\mathcal{M}$, for a fixed β, and form a global abstraction $\phi_{\mathcal{M}}$ by taking the intersection across each computed state abstraction. That is, for any state pair (s_1, s_2), for ϕ_i computed by Dibs on each MDP, I define $\phi_{\mathcal{M}}$:

$$\phi_{\mathcal{M}}(s_1) = \phi_{\mathcal{M}}(s_2) \equiv \bigwedge_{i=1}^{|\mathcal{M}|} \{\phi_i(s_1) = \phi_i(s_2)\}. \tag{5.67}$$



(a) $\beta = 0$

(b) $\beta = 10$

(c) $\beta = 20$

Figure 5.10: State abstractions computed by Dibs for a collection of MDPs using Equation 5.67 for different values of β.

Figure 5.10 shows $\phi_{\mathcal{M}}$ for different values of β. All cells with the same color are grouped into the same state, except for white: all white states are each treated as their true ground state. Note that the abstraction becomes far more detailed as β increases. When β is close to 0, the algorithm prioritizes *compression*, as is reflected by Figure 5.10a, which only has a single state. Conversely, as β increases, the algorithm adds more distinctions between states, only grouping those that are close to one another or the near the same wall. When $\beta = 10$, the abstraction groups large regions of contiguous states together, such as the central group of states in the top right room, and many states in the center of the bottom left room. When $\beta = 20$, we find even less compression, but still see a few small contiguous regions that have some structural similarities. Critically, none of the abstractions are perfect. It is unknown what might constitute an ideal abstraction in this case, nor how well our proposed algorithm might approximate such an ideal.

5.4.3 *Learning ϕ in Continuous State Environments*

The investigation in this part of the dissertation so far has focused on discrete MDPs. It is important, however, to consider whether similar ideas are applicable when the underlying environmental state and action space are continuous, too. In this final section, I study the case when the underlying state space is continuous, summarizing work first presented by Asadi et al. [24].

Concretely, I here introduce a new objective function that may be solved with standard stochastic gradient descent. Thus, we will develop a procedure simple for learning a state abstraction that maps a continuous state space into a discrete one given access to some number of trajectories of behavior on several training tasks. I then provide a generalization error bound on the quality of the learned state abstraction, assuming a fixed distribution over states ρ is used in both training and testing (even if the MDP changes). I then conduct an empirical study to validate the usefulness of the learned state abstraction in two kinds of experiments: 1) using ϕ to accelerate RL on the exact MDP in which ϕ was learned, and 2) using ϕ to accelerate RL on MDPs similar (but not identical to) the MDP in which ϕ was learned. That is, in the most general case, the agent will be allowed to collect data on some set of training MDPs, and then use this data to inform a choice of state abstraction for use in future related tasks.

One important difference from previous settings is that I will here focus on *stochastic* state abstraction functions, which I denote $\tilde{\phi} : \mathcal{S} \to \Delta(\mathcal{S}_\phi)$ that defines a function from ground states $s \in \mathcal{S}$ to a probability distribution over abstract states $\mathcal{S}_\phi$ [294]. Any policy over abstract states, denoted $\pi_{\tilde{\phi}}$ will first sample $s_\phi \sim \tilde{\phi}(\cdot \mid s)$, then act given this s_ϕ.

THE NEW OBJECTIVE. The objective is similar in spirit to the CVA and DIBS introduced earlier in the chapter (Algorithm 5.1) adapted to continuous state spaces. Here, we again look to map ground states in which the optimal policy is similar into a common abstract state. The difficulty is that only finitely many pairs $(s_i, \pi^*(s_i))$ can be sampled, thus making it challenging to determine how to cluster any state s' not seen during training. This is especially challenging when the state space is continuous, as conservatively we expect the agent to never inhabit the same state twice.

The objective is based around the idea of forming a state abstraction and policy pair that are most likely to have generated the set of trajectories seen during training. Concretely, we introduce the following objective that measures the probability of trajectories τ^i generated by an estimate of the optimal policy $\hat{\pi}^*$. In a single MDP, the goal is to maximize this probability if an agent were to use the function $\tilde{\phi}$ and policy, $\pi_{\tilde{\phi}}$, over abstract states. This optimization problem is formulated as follows.

$$\operatorname*{arg\,max}_{\tilde{\phi},\pi_{\tilde{\phi}}} \prod_{i=1}^{M} p(\tau^i, \tilde{\phi}, \pi_{\phi}) = \operatorname*{arg\,max}_{\tilde{\phi},\pi_{\tilde{\phi}}} \sum_{i=1}^{M} \log p(\tau^i, \tilde{\phi}, \pi_{\tilde{\phi}}), \tag{5.68}$$

where:

$$\begin{aligned}
\log p(\tau^i, \tilde{\phi}, \pi_{\tilde{\phi}}) &= \log \prod_{j=1}^{T(\tau^i)} \pi(a_j^{\tau^i} \mid s_j) p(s_{j+1}^{\tau^i} \mid s_j^{\tau^i}, a_j^{\tau^i}), && (5.69)\\
&= \sum_{j=1}^{T(\tau^i)} \log \sum_{s_\phi \in \mathcal{S}_\phi} \tilde{\phi}(s_\phi \mid s_j^{\tau^i}) \pi_{\tilde{\phi}}(a_j^{\tau^i} \mid s_\phi) + \underbrace{\sum_{j=1}^{T(\tau^i)} \log p(s_{j+1}^{\tau^i} \mid s_j^{\tau^i}, a_j^{\tau^i})}_{ⓐ}. && (5.70)
\end{aligned}$$

Note that term ⓐ is not a function of the optimization variables, and may be dropped, yielding the following.

$$\operatorname*{arg\,max}_{\tilde{\phi},\pi_{\tilde{\phi}}} \sum_{i=1}^{M} \log p(\tau^i, \tilde{\phi}, \pi_{\tilde{\phi}}) = \operatorname*{arg\,max}_{\tilde{\phi},\pi_{\tilde{\phi}}} \sum_{i=1}^{M} \sum_{j=1}^{T(\tau^i)} \log \sum_{s_\phi \in \mathcal{S}_\phi} \tilde{\phi}(s_\phi \mid s_j^{\tau^i}) \pi_{\tilde{\phi}}(a_j^{\tau^i} \mid s_\phi). \tag{5.71}$$

In general, we imagine that the agent is tasked with solving a set of K different MDPs, only some of which are seen during training. This extension changes the objective as follows.

$$\operatorname*{arg\,max}_{\tilde{\phi},\pi_{\tilde{\phi}}} \sum_{k=1}^{K} \sum_{i=1}^{M} \sum_{j=1}^{T(\tau^i)} \log \sum_{s_\phi \in \mathcal{S}_\phi} \tilde{\phi}(s_\phi \mid s_j^{\tau^i}) \pi_{\tilde{\phi}}(a_j^{\tau^i} \mid s_\phi, k). \tag{5.72}$$

If the solution to the optimization problem is accurate enough, then states that are clustered into a single abstract state generally have a similar policy in the MDPs used for training.

Although it is possible to jointly solve this optimization problem, for simplicity let us assume $\pi_{\tilde{\phi}}$ is fixed and provided to the learner. We then parameterize the abstraction function ϕ by a vector θ representing the weights of a neural network, $\tilde{\phi}(\cdot \mid s; \theta)$. Further, we use softmax activation to ensure that $\tilde{\phi}$ outputs a probability distribution.

A good setting of θ can be found by performing stochastic gradient ascent on the objective above, as is standard when optimizing neural networks [191]:

$$\theta \leftarrow \theta + \alpha \nabla_\theta \sum_{k=1}^{K} \sum_{i=1}^{M} \sum_{j=1}^{T(\tau^i)} \log \sum_{s_\phi \in \mathcal{S}_\phi} \tilde{\phi}(s_\phi \mid s_j^{\tau^i}; \theta) \pi_{\tilde{\phi}}(a_j^{\tau^i} \mid s_\phi, k). \tag{5.73}$$

As in the experiments earlier in the chapter, we here use the Adam optimizer [162].

ANALYSIS. Ideally, as per the abstraction desiderata, we would like the abstractions learned by our procedure to support efficient discovery of good policies. When the state space is continuous, this is particularly challenging, as the agent is constantly encountering ground states never seen during training. In light of these difficulties, we next prove that a $\tilde{\phi}$ learned according to our procedure on some finite data set of n experiences can still ensure a form of bounded expected value loss, assuming the state distribution is unchanged.

Concretely, let us suppose that the probability distribution used to generate states during both training and evaluation is some fixed ρ. Then, it is natural to consider how well the learned state abstraction will support learning under the friendly assumption that future states will *also* be sampled according to ρ. In particular, I next present a generalization error bound on the dissimilarity between π^* and $\pi_{\tilde{\phi}}$ for the learned $\tilde{\phi}$, in expectation under the sampling distribution ρ. This error bound can be decomposed into three components: 1) the training error (in training $\tilde{\phi}$), 2) the Rademacher Complexity [36] of Φ, and 3) the size of the data set used to train $\tilde{\phi}$, n. More formally, the result is as follows.

Theorem 5.3. *For any error probability $\delta \in (0, 1)$, n the size of the training data set, Δ the training error, and for some fixed distribution on states ρ using during training, the following holds with probability $1 - \delta$,*

$$\mathbb{E}_{s\sim\rho}\left[\left\|\left(\pi^*(\cdot \mid s) - \pi_{\tilde{\phi}}(\cdot \mid s)\right)\right\|_1\right] \le \frac{\Delta}{2} + 2\sqrt{2}Rad(\Phi) + \sqrt{\frac{2\ln\frac{1}{\delta}}{n}}. \tag{5.74}$$

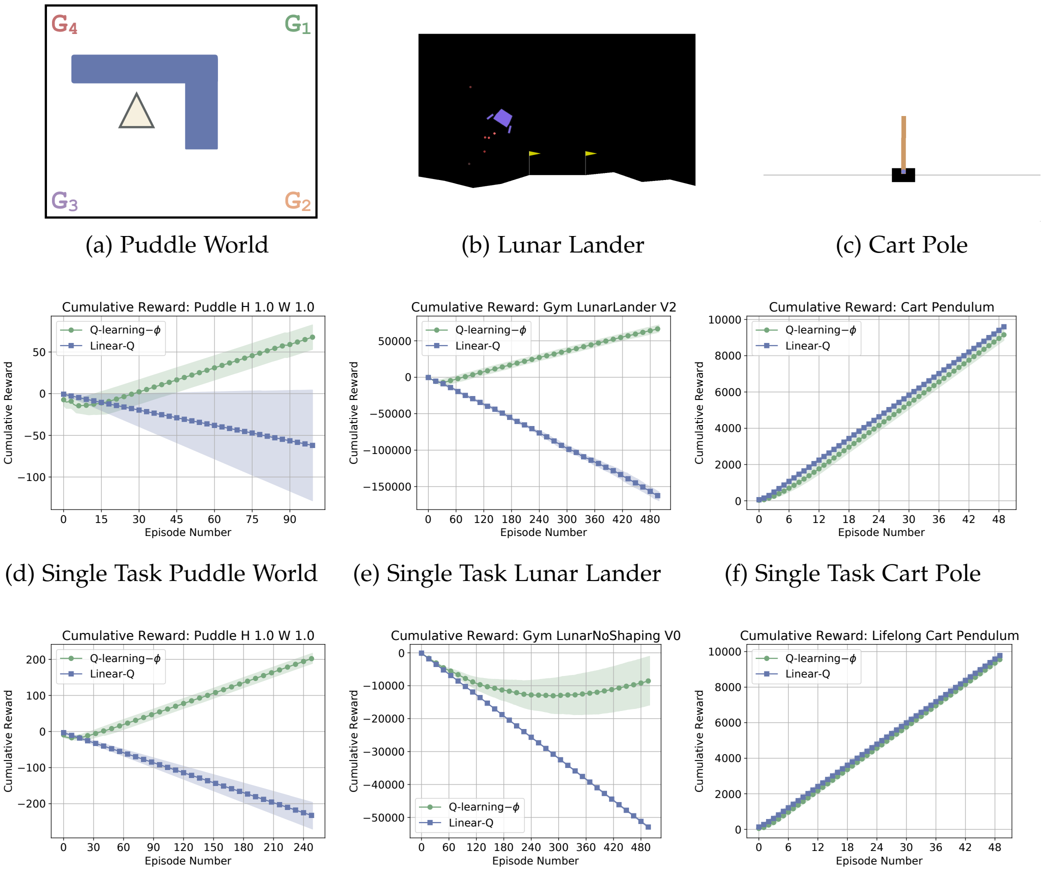
The proof was first introduced by Asadi et al. [24]—see Theorem 1. The main power of this bound comes from the application of Rademacher Complexity [36], which measures the richness of the function family Φ. This result indicates the effect of the capacity of the model family (smooth neural networks, in our case), and of the sample size (n) on the overall gap in quality between the abstract policy and the optimal policy. This result is powerful as it tells us that states sampled from the distribution ρ, even those not seen during training, are likely to be well accommodated by the learned state abstraction.

EXPERIMENTS. I now summarize empirical results from two sets of experiments that investigate the utility of the proposed approach.

1. Single Task: We collect an initial data set $\mathcal{D}_{\text{train}}$ of size n to be used to train the state abstraction $\tilde{\phi}$ based on MDP M. Then, we evaluate the performance of tabular Q-learning on M given this state abstraction. These experiments provide an initial sanity check as to whether the state abstraction can facilitate learning at all.

2. Multi-Task: We next consider a collection of MDPs $\{M_1, \ldots, M_k\}$. We collected an initial data set $\mathcal{D}_{\text{train}}$ of (s, a) tuples from a (strict) subset of MDPs in the collection. We used $\mathcal{D}_{\text{train}}$ to construct a state abstraction ϕ, which we then gave to Q-learning to learn on one or many of the remaining MDPs. Critically, we evaluate Q-learning on MDPs *not* seen during the training of ϕ.

For each of the two experiment types, we evaluate in three different domains, Puddle World, Lunar Lander, and Cart Pole. Open source implementations of Lunar Lander and Cart Pole are available by Brockman et al. [56]. We contrast the learning efficiency



(a) Puddle World

(b) Lunar Lander

(c) Cart Pole

(d) Single Task Puddle World

(e) Single Task Lunar Lander

(f) Single Task Cart Pole

(g) Puddle World Transfer

(h) Lunar Lander Transfer

(i) Cart Pole Transfer

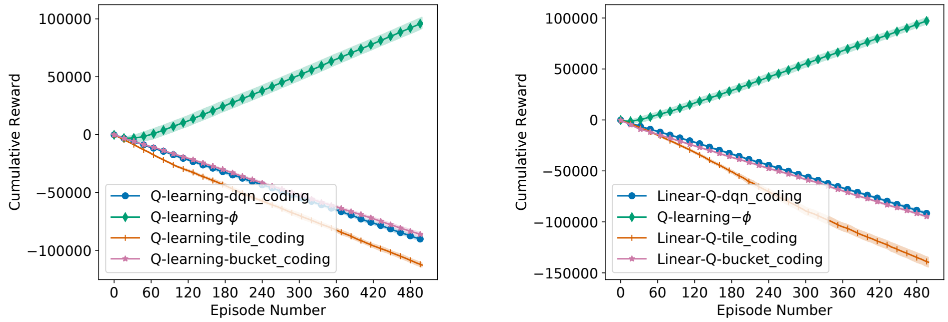
Figure 5.11: Learning curves for the single task experiments (top) and the transfer experiments (bottom).

of tabular Q-learning paired with the learned state abstraction (green) with Q-learning using a linear function approximator (blue) [234, 177]. The features used by the linear approximator are those provided by the standard implementation of each domain. In Puddle World, the state is a pair of real numbers denoting spatial coordinates; in Cart Pole, the state is constituted by four real numbers indicating the location, velocity, angle, and angular velocity of the pole; and in Lunar Lander, the state is eight real numbers indicating things like the position and angle of the lander. Full parameter settings and other experimental details are available in our code.[5.2]

[5.2] https://github.com/david-abel/continuous_state_sa

Figure 5.11 presents results for both experiments, with the single task results in the middle row and the transfer results in the bottom row. Each figure presents the mean cumulative reward per episode with 95% confidence intervals, averaged over 25 instances. In Puddle World, we see that the learned state abstraction is capable of consistently supporting extremely sample efficient learning in both the single task and transfer case—by around episode 30 in both cases, tabular Q-learning reliably converges to a policy that effectively takes the agent directly to the goal while avoiding the puddle. The same is true of Lunar Lander, only more samples are required in the transfer case. In contrast, for the given sample budgets, the approach using the linear function approximator is unable to improve its policy at all. Lastly, in Cart Pole, we see both approaches are able to find near-optimal policies in relatively few samples.

In a final experiment, we contrast the learning performance of tabular Q-learning using several different state-discretization methods on the single task variant of the Lunar Lander domain. First, we use our same approach, shown in green. Second, we contrast tile coding (orange) [307], a naive form of discretization we call bucket coding (pink), and an approach that uses state features learned by a Deep Q-Network [238] trained on the same task (blue). Results are presented in Figure 5.12. The data are quite clear: on average, the state abstraction learned by our approach is sufficient for enabling extremely sample efficient learning on Lunar Lander. In contrast, none of the other state representations



(a) Tabular Q

(b) Linear Q

Figure 5.12: A comparison of different state discretization methods in Lunar Lander.

support the learning of good behavioral policies. For more on these ideas, experiments, and the algorithm, see original work by Asadi et al. [24].

To summarize, this chapter draws on the tools of information theory to cast the process of state abstraction as a form of *compression*. I introduced an algorithmic framework that can efficiently produce state abstractions that trade off between compression and value preservation. Through a variety of visuals, empirical study, and analysis, I have demonstrated the power of this approach to discover good state abstractions for RL.

This brings Part 2 to a close. I next shift focus to action abstraction.

Part 3

ACTION ABSTRACTION

6

FINDING OPTIONS THAT MINIMIZE PLANNING TIME

This chapter is based on "Finding Options that Minimize Planning Time" [144] led by Yuu Jinnai, joint with D. Ellis Hershkowitz, Michael L. Littman, and George Konidaris.

Action abstraction defines the process of forming high level behaviors such as "go to the bridge", in place of "rotate right leg so many degrees". Such a mechanism is deeply connected to many other important practices of agency, including the discovery and manipulation of useful subgoals, efficient long-horizon planning, and credit assignment. The primary formalism I adopt for capturing action abstraction is the *options* framework [311]. Several other names describe roughly the same process, including skills, temporal abstraction, and macro-actions. I treat options as sufficiently general to capture all of these, but of course there are subtleties to each particular type. For further background on action abstraction and options, see Section 2.3.

As with state abstraction, my objective in this part of the dissertation is to bring formal clarity to the discovery of *good* action abstractions (those that satisfy the abstraction desiderata—see Section 2.4).

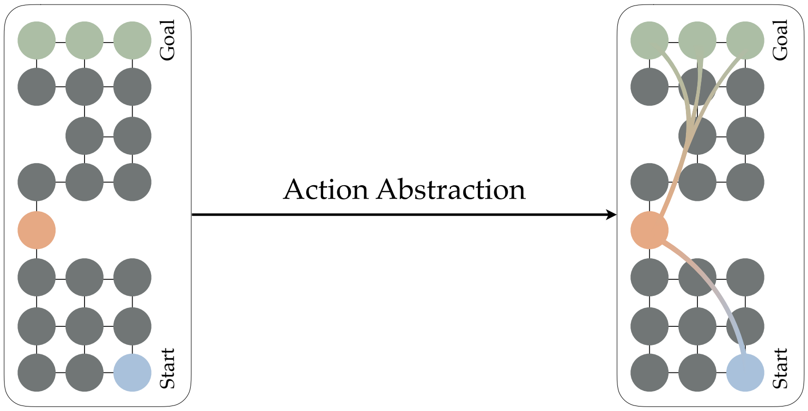


Figure 6.1: Action abstraction.

In this chapter, I first study one notion of "good action abstraction". I present analysis of the problem of finding options that make the process of *planning* as efficient as possible. To make this problem concrete I will make several simplifying assumptions that allow for appropriate analysis. In particular, I will ground the speed of planning in terms of how many iterations of VI (Algorithm 2.3) are required to return an accurate value function for the whole state space. I prove that this problem is NP-hard and hard to approximate well. Fortunately, these hardness results also come along with two approximation algorithms that each nearly match the lower bound of approximation hardness under friendly assumptions. One of the algorithms is based on an approximation of set cover by Chvatal [69], and the other is based on the procedure by Archer [18]. In simple experiments, the options found by these approximation algorithms are nearly competitive (in terms of acceleration of VI) with those options found by solving the NP-hard problem.

Why might we care about this problem? Well, again, there are many things abstract actions can do to enhance a learning agent's capabilities. Intuitively, carrying out long-horizon simulations of the right kind can be immensely useful, so long as the simulations are sufficiently well informed. Imagine boiling a pot of water. By setting a particular degree of heat beneath the flame, it is easy to predict that in some number of minutes that the water will boil with extremely high probability. It is not necessary to know precisely how long it will take, or exactly how much water will evaporate before you turn the heat down. To be useful, it is simply sufficient to know that the heat will lead to water boiling in a reasonable time frame. It is this activity that carries a great deal of promise, and for which the options formalism is well suited to study. Hence, a general negative result illustrating that it is difficult to find the right options suggests that there is more nuance to the problem of discovering good options than simply optimizing relative to a given task— I liken this result to a version of the No Free Lunch theorem [346], according to which no learning algorithm can dominate all others on all problems. In a sense, the hardness results presented here suggest that no option discovery algorithm can efficiently find the right options on all problems, but perhaps on a well chosen subset, such a task is easier. To summarize, the results of this chapter are useful for understanding the limitations of option discovery, evaluating option discovery methods, and for guiding future option discovery algorithms. I return to this discussion later in the chapter.

6.1 FORMALIZING THE PROBLEM

I now formalize what it means to find the set of options that is optimal for planning. I will then use this formalism to establish hardness results for computing options that help with planning, both in the worst and approximate cases. The main positive result is the existence of an approximation algorithm with a principled theoretical foundation.

To ground this study, I will restrict attention to a particular notion of planning in several ways. First, I only study the acceleration of VI, rather than the full scope of planning algorithms. Indeed, I take VI to be both sufficiently general and canonical to capture the rough structure of many planning algorithms. Second, I ignore any increase to the branching factor. This is a big component, as adding options will help reduce the number of iterations of VI, but will necessarily increase the number of actions evaluated at each state. Hence, the proposed model only captures a part, but not the whole picture, of planning acceleration. Finally, I concentrate only on finite MDPs. While these restrictions limit the scope of the result, it is important to establish this the computational difficulty of this problem in a restricted setting before considering the more general cases.

Precisely, I will show that the problem of finding options that minimize the number of iterations required by VI,

1. is $2^{\log^{1-\epsilon} n}$-hard to approximate for any $\epsilon > 0$ unless NP $\subseteq$ DTIME($n^{\text{poly}\log n}$),[6.1] where n is the input size;
2. is $\Omega(\log n)$-hard to approximate even for deterministic MDPs unless PTIME = NP;
3. has a $O(n)$-approximation algorithm;
4. has a $O(\log n)$-approximation algorithm for deterministic MDPs.

In Section 6.4, I present A-MOMI, a polynomial-time approximation algorithm that has $O(n)$ suboptimality in general and $O(\log n)$ suboptimality for deterministic MDPs. Note that the expression $2^{\log^{1-\epsilon} n}$ is only slightly smaller than n: if $\epsilon = 0$ then $\Omega(2^{\log n}) = \Omega(n)$. Thus, A-MOMI is close to the best possible approximation factor. In addition, I will consider the complementary problem of finding a set of k options that minimize the number of VI iterations until convergence. I show that this problem is also NP-hard, even for a deterministic MDP. After establishing these complexity results, I highlight a brief empirical study comparing the performance of two heuristic approaches for option discovery: betweenness options [291] and eigenoptions [211], with those options discovered by the new approximation algorithm.

6.2 Options and Value Iteration

I here study the *value-planning problem*, defined as follows.

Definition 6.1. *The **value-planning problem** is defined as follows:* `given` *an MDP* $M = (\mathcal{S}, \mathcal{A}, R, T, \gamma, \rho_0)$ *and an* $\epsilon \in \mathbb{R}_{\geq 0}$, `return` *a value function,* V_t *such that* $|V_t(s) - V^*(s)| < \epsilon$ *for all* $s \in \mathcal{S}$.

[6.1] This is a standard complexity assumption: see, for example, Dinitz et al. [91]

As discussed in Section 2.3, options have a well defined transition and reward model for each state named the multi-time model [264]:

$$T_\gamma(s' \mid s, o) = \sum_{k=0}^{\infty} \gamma^k \beta_o(s') p(s', k \mid s, o), \tag{6.1}$$

$$R_\gamma(s, o) = \mathop{\mathbb{E}}_{k, s_{1\ldots k}} \left[r_1 + \gamma r_2 \ldots + \gamma^{k-1} r_k \,\middle|\, s, o \right]. \tag{6.2}$$

To run VI with options, it is natural to substitute the MTM for the standard reward and transition function and apply the same exact operations. The algorithm then computes a sequence of functions $V_0, V_1, \ldots, V_t$ using the Bellman Equation on the MTM:

$$V_{i+1}(s) = \max_{o \in A \cup \Omega(s)} \left(R_\gamma(s, o) + \sum_{s' \in S} T_\gamma(s' \mid s, o) V_i(s') \right). \tag{6.3}$$

Throughout this chapter, I will assume that the model of each option is given to the agent and ignore the computational cost for computing the model for the options. This is yet another assumption that will help simplify the analysis, but will also restrict the scope of the result—indeed, understanding the total difficulty of option discovery (model computation and all), is of deep importance. In Chapter 7, I develop an alternative model to the MTM that is simpler to estimate while retaining desirable properties.

Here, I study the problem of choosing a subset of options $\mathcal{O}'$ from a given set $\mathcal{O}$ to add to $\mathcal{A}$ that minimizes the number of iterations required for VI to converge.[6.2]

Definition 6.2. *The* ***number of iterations $L(\mathcal{O})$*** *of VI using the joint action set $\mathcal{A} \cup \mathcal{O}$, with $\mathcal{O}$ a non-empty set of options, is the smallest b at which $|V_b(s) - V^*(s)| < \epsilon$ for all $s \in \mathcal{S}$.*

POINT OPTIONS. Due to the generality of the options framework, a single option can in fact encode several completely unrelated sets of different behaviors. For example, consider the nine-state MDP pictured in Figure 6.2. In this MDP, I include the initiation,

[6.2] To ensure that $|V^*(s) - V_i(s)| < \epsilon$ for each $s \in \mathcal{S}$, run VI until $|V_{i+1}(s) - V_i(s)| < \epsilon(1-\gamma)/2\gamma$ for each $s \in \mathcal{S}$ [343].

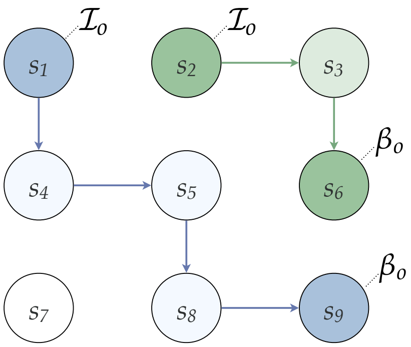


Figure 6.2: A single option can encode multiple unrelated behaviors.

policy, and termination of a single option. The option initiations in s_1 and s_2, and terminates in s_9 and s_6. However, the policy executed from s_1 and s_2 ultimately produce entirely independent trajectories. Consequently, this single option defines two separate behaviors. For this reason, it can be difficult to reason about the impact of adding a single option in the traditional sense—it might be the case that one option in fact defines arbitrarily many separate behaviors. In fact, as the MDP grows larger, a combinatorial number of behaviors can emerge from a single option. It can thus be difficult to address the question: which *single* option helps planning the most? Thus, for the purposes of the analysis, we choose to focus attention on a special subclass of options that only allow for a single continuous stream of behavior:

Definition 6.3. *A **point option** is any option o whose initiation set $\mathcal{I}_o$ and termination condition β_o correspond to one state each:*

$$|\{s \in \mathcal{S} : \mathcal{I}_o(s) = 1\}| = 1, \tag{6.4}$$

$$|\{s \in \mathcal{S} : \beta_o(s) > 0\}| = |\{s \in \mathcal{S} : \beta_o(s) = 1\}| = 1. \tag{6.5}$$

I let $\mathcal{O}_p$ denote the set containing all point options. Note that trivially $\mathcal{O}_p \subset \mathcal{O}_{\text{all}}$. Additionally observe two key properties of point options: 1) an arbitrary collection of options in a finite MDP can be represented as a collection of point options, and 2) a point option is simply a regular option paired with a particular kind of state abstraction that groups together all states in the initiation and termination sets (assuming deterministic termination). For these reasons, point options are surprisingly general.

6.3 COMPLEXITY RESULTS

The main results of this chapter focus on two computational problems, first introduced by Jinnai et al. [144];

1. MinOptionMaxIter (MOMI): Given a set of options $\mathcal{O}$, which subset $\mathcal{O}' \subseteq \mathcal{O}$ allows VI to converge in at most ℓ iterations?
2. MinIterMaxOption (MIMO): Given a set of options $\mathcal{O}$, which subset of k or fewer options $\mathcal{O}'_k \subseteq \mathcal{O}$ will minimizes the number of iterations of VI to convergence?

More formally, the two problems are defined as follows.

Definition 6.4. *The **MinOptionMaxIter** problem is defined as follows:* `given` *an MDP M, a non-negative real-value ϵ, and an integer ℓ,* `return` *$\mathcal{O}$ that minimizes $|\mathcal{O}|$ subject to $\mathcal{O} \subseteq \mathcal{O}_p$ and $L(\mathcal{O}) \leq \ell$.*

MinIterMaxOption (MIMO).

Definition 6.5. *The **MinIterMaxOption** problem is as follows:* `given` *an MDP M, a non-negative real-value ϵ, and an integer k,* `return` *$\mathcal{O}$ that minimizes $L(\mathcal{O})$, subject to $\mathcal{O} \subseteq \mathcal{O}_p$ and $|\mathcal{O}| \leq k$.*

I now present the main result of the chapter, which states that both MOMI and MIMO are NP-hard.

Theorem 6.1. *MOMI and MIMO are NP-hard.*

This result was first proven by Jinnai et al. [144]—see Theorem 1. For proofs of all results introduced in this chapter, see original work by Jinnai et al. [144].

Generalizations of MOMI and MIMO

It is natural to consider whether the hardness results of Theorem 6.1 extend to more general settings. Let us now turn to extensions of these problems that offer significant coverage of settings relevant to finding the optimal options for accelerating VI.

First, consider the case where the given options are *not* point options, but rather may be an arbitrary subset of $\mathcal{O}_{\text{all}}$. When the given set of options is in fact $\mathcal{O}_{\text{all}}$, MOMI is solved trivially, since the optimal option for accelerating VI is one that initiates everywhere and executes π^*. However, in practice, the set $\mathcal{O}_{\text{all}}$ is likely to be inaccessible. Instead, it is often preferable to focus on classes of options that can be constructed with a restricted computational or sample budget. To capture this variant of the problem, let us now introduce a generalization of MOMI:

Definition 6.6. *The* $\textbf{MOMI}_{\textbf{\textit{gen}}}$ *problem is defined as follows:* `given` *an MDP M, a non-negative real-value ϵ, $\mathcal{O}' \subseteq \mathcal{O}_{all}$, and an integer ℓ,* `return` *$\mathcal{O}$ minimizing $|\mathcal{O}|$ subject to $L(\mathcal{O}) \leq \ell$ and $\mathcal{O} \subseteq \mathcal{O}'$.*

In this way, $\mathcal{O}'$ can denote options that satisfy other criteria, such as those that can be constructed or estimated given some resource budget, or obtain of other desirable properties. As is expected, this problem is again NP-hard, since both MOMI$_{gen}$ and MIMO$_{gen}$ are supersets of MOMI and MIMO respectively.

Theorem 6.2. *MOMI$_{gen}$ and MIMO$_{gen}$ are NP-hard.*

Another relevant relaxation of MIMO and MOMI is to move to the multitask or lifelong setting discussed in Chapter 4—how does this problem change when our goal is to identify options that accelerate planning on multiple MDPs? More concretely, given a distribution over MDPs D, we would like to compute the smallest set of options $\mathcal{O}'_D$ that minimize the *expected* number of iterations to solve $M \sim D$. I refer to this problem as MOMI$_{multi}$, defined as follows.

Definition 6.7. *The* ***MOMI***$_{\boldsymbol{multi}}$ *defines the following computational problem:* `given` *a probability distribution over MDPs D, $\mathcal{O}' \subseteq \mathcal{O}_{all}$, a non-negative real-value ϵ, and an integer ℓ,* `return` *$\mathcal{O}$ that minimizes $|\mathcal{O}|$ such that $\mathbb{E}_{M \sim D}[L_M(\mathcal{O})] \leq \ell$ and $\mathcal{O} \subseteq \mathcal{O}'$.*

As expected, the same extension can be applied to MIMO, too.

Theorem 6.3. *MOMI$_{multi}$ and MIMO$_{multi}$ are NP-hard.*

The proof follows from the fact that MOMI$_{multi}$ is a superset of MOMI$_{gen}$ and MIMO$_{multi}$ is a superset of MIMO$_{gen}$.

In light of the computational difficulty of both problems, the appropriate approach is to find a suitable approximation algorithms. However, even approximately solving MOMI is hard. More precisely:

Theorem 6.4.

1. *MOMI is $\Omega(\log n)$ hard to approximate even for deterministic MDPs unless $P = NP$.*
2. *MOMI$_{gen}$ is $2^{\log^{1-\epsilon} n}$-hard to approximate for any $\epsilon > 0$ even for deterministic MDP unless $NP \subseteq DTIME(n^{poly \log n})$.*
3. *MOMI is $2^{\log^{1-\epsilon} n}$-hard to approximate for any $\epsilon > 0$ unless $NP \subseteq DTIME(n^{poly \log n})$.*

Note that an $O(n)$-approximation is achievable by the trivial algorithm that returns a set of all candidate options. Thus, Theorem 6.4 roughly states that there is no polynomial time approximation algorithms other than the trivial algorithm for MOMI.

In the next section we show that an $O(\log n)$-approximation is achievable if the MDP is deterministic, and the agent is given the set containing all point options. Thus, together, these two results give a formal separation between the hardness of abstraction in MDPs with and without stochasticity.

In summary, the problem of computing optimal behavioral abstractions for accelerating VI is computationally intractable.

6.4 APPROXIMATION ALGORITHMS

I now provide polynomial-time approximation algorithms, A-MIMO and A-MOMI, to solve MOMI and MIMO respectively. Both algorithms have bounded suboptimality that is slightly worse than a constant factor for deterministic MDPs.

The analysis requires several assumptions. First, there is exactly one absorbing state $s_g \in \mathcal{S}$ with $T(s_g \mid a, s_g) = 1$ and $R(s_g, a) = 0$. Second, that every optimal policy eventually reaches s_g with probability 1. Third, there is no cycle with a positive reward involved in the optimal policy's trajectory. That is, $V_+^\pi(s) := \mathbb{E}[\sum_{t=0}^{\infty} \max\{0, R(s,a)\}] < \infty$ for all policies π. Note that we can convert a problem with multiple goals to a problem with a single goal by adding a new absorbing state s_g to the MDP and adding a transition from each of the original goals to s_g.

Unfortunately, these algorithms are computationally more involved than solving the MDP itself through standard methods, and are thus unlikely to be practical. Instead, they are useful for analyzing and evaluating options discovered by heuristic algorithms. If the option set found by an option discovery method outperforms the option set found by one the following approximation algorithms (in planning performance), then it is strong evidence that the option set found by the heuristic is close to the optimal option set (for that MDP). The approximation algorithms are guaranteed to have bounded suboptimality if the MDP is deterministic, so any heuristic method that provably exceeds our algorithm's performance will also guarantee bounded suboptimality. Further, these algorithms may be a useful foundation to help guide future option discovery methods.

APPROXIMATION ALGORITHM: A-MOMI. I now describe a polynomial-time approximation algorithm, A-MOMI, that uses set cover to solve MOMI. The overview of the procedure is as follows.

1. Compute an asymmetric distance function $d_\epsilon(s, s') : \mathcal{S} \times \mathcal{S} \rightarrow \mathbb{N}$ representing the number of iterations for a state s to reach its ϵ-optimal value if we add a point option from a state s' to a goal state s_g.

2. For every state s_i, compute a set of states X_{s_i} within $\ell - 1$ distance of reaching s_i. The set X_{s_i} represents the states that converge within ℓ steps if a point option is added from s_i to s_g.

3. Let $\mathcal{X}$ be a set of X_{s_i} for every $s_i \in \mathcal{S} \setminus X_g^+$, where X_g^+ is a set of states that converges within ℓ without any options.

4. Solve the set-cover optimization problem to find a set of subsets that covers the entire state space using the approximation algorithm by Chvatal [69]. This process corresponds to finding a minimum set of subsets $\{X_{s_i}\}$ that makes every state in $\mathcal{S}$ converge within ℓ steps.

5. Generate a set of point options with initiation states set to one of the center states in the solution of the set-cover, and termination states set to the goal.

The distance function $d_\epsilon : \mathcal{S} \times \mathcal{S} \rightarrow \mathbb{N}$, is defined as follows.

Definition 6.8. *The asymmetric distance* $\boldsymbol{d_\epsilon(s_i, s_j)}$ *is one minus the number of iterations for* s_i *to reach* ϵ*-optimal if a point option is added from* s_j *to* s_g*.*

More formally, let $d'_\epsilon(s_i)$ denote the number of iterations needed for the value of state s_i to satisfy $|V(s_i) - V^*(s_i)| < \epsilon$, and let $d'_\epsilon(s_i, s_j)$ be an upper bound of the number of iterations needed for the value of s_i to satisfy $|V(s_i) - V^*(s_i)| < \epsilon$, if the value of s_j is initialized such that $|V(s_j) - V^*(s_j)| < \epsilon$. Let $d_\epsilon(s_i, s_j) := \min(d'_\epsilon(s_i) - 1, d'_\epsilon(s_i, s_j))$. For simplicity, I use d as shorthand for d_ϵ.

Note that we need to solve the MDP once to compute d. The quantity $d(s, s')$ can be computed once the MDP is solved without any options and have stored all value functions V_i for $i = 1, \ldots, b$ until convergence as a function of V_1: $V_i(s) = f(V_1(s_0), V_1(s_1), \ldots)$. If a point option is added from s' to s_g, then $V_1(s') = V^*(s')$. Thus, $d(s, s')$ is the smallest i such that $V_i(s)$ is ϵ-optimal if we replace $V_1(s')$ with $V^*(s')$ when computing $V_i(s)$ as a function of V_1. With these pieces in play, we can now state the properties of A-MOMI.

Theorem 6.5. A-MOMI *has the following properties:*

1. A-MOMI *runs in polynomial time.*
2. *It guarantees that the MDP is solved within ℓ iterations using the option set acquired by* A-MOMI $\mathcal{O}$.
3. *If the MDP is deterministic, the option set is at most $O(\log n)$ times larger than the smallest option set possible to solve the MDP within ℓ iterations.*

Note that the approximation bound for a deterministic MDP will inherit any improvements to the approximation algorithm for set cover. Set cover is known to be NP-hard to approximate up to a factor of $(1 - o(1)) \log n$ [92], thus there may be an improvement on the approximation ratio for the set cover problem, which will also improve the approximation ratio of A-MOMI.

APPROXIMATION ALGORITHM: A-MIMO. The outline of the approximation algorithm for MIMO (A-MIMO) is as follows.

1. Compute $d_\epsilon(s, s') : \mathcal{S} \times \mathcal{S} \to \mathbb{N}$ for each pair of states.
2. Using this distance function, solve an asymmetric k-center problem, which finds a set of center states that minimizes the maximum number of iterations for every state to converge.
3. Generate point options with initiation states set to the center states in the solution of the asymmetric k-center problem and termination conditions to the goal.

As in A-MOMI, we first compute the distance function, which is the most computationally demanding part of the algorithm. Then, we use d to solve the asymmetric k-center problem [256] on $(\mathcal{U}, d, k)$ to get a set of centers, which we use as initiation states for point options. The asymmetric k-center problem is a generalization of the metric k-center problem where the function d obeys the triangle inequality, but is not necessarily symmetric:

Definition 6.9. *Th* ***asymmetric k-center problem*** *is defined as follows:* `given` *a set of elements $\mathcal{U}$, a function $d : \mathcal{U} \times \mathcal{U} \to \mathbb{N}$, and an integer k,* `return` *$\mathcal{C}$ that minimizes $P(\mathcal{C}) = \max_{s \in \mathcal{U}} \min_{c \in \mathcal{C}} d(s, c)$ subject to $|\mathcal{C}| \leq k$.*

We solve this problem using a polynomial-time approximation algorithm proposed by Archer [18]. The algorithm has a suboptimality bound of $O(\log^* k)$[6.3] where $k < |\mathcal{U}|$. It is known that the problem cannot be solved within a factor of $\log^* |\mathcal{U}| - \theta(1)$ unless P=NP [68]. As the procedure by Archer [18] often finds a set of options smaller than k, we generate the rest of the options by greedily adding $\log k$ options at once. Finally, we generate a set of point options with initiation-states set to one of the centers and the termination state set to the goal state of the MDP. That is, for every $c \in \mathcal{C}$, we generate a point option starting from c to the goal state s_g.

Theorem 6.6. A-MIMO *has the following properties:*

1. A-MIMO *runs in polynomial time.*
2. *If the MDP is deterministic, it has a bounded suboptimality of $O(\log^* k)$.*
3. *The number of iterations to solve the MDP using the option set acquired is upper bounded by $P(\mathcal{C})$.*

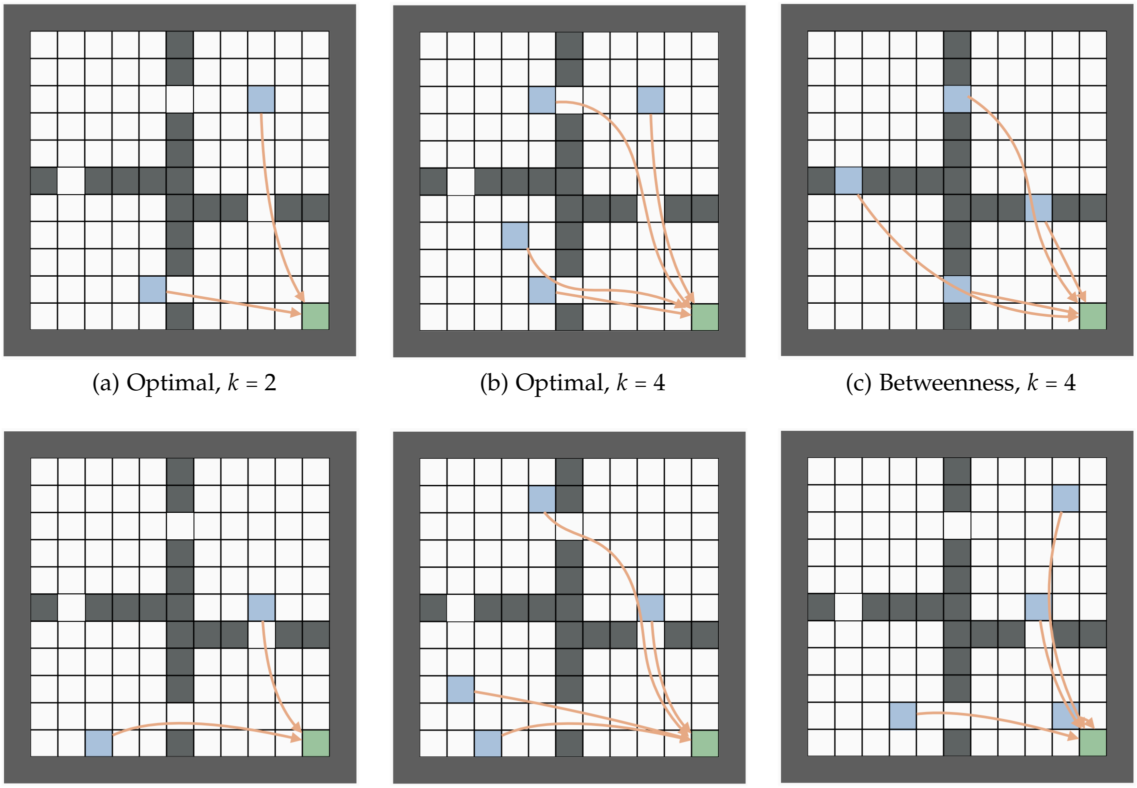
With the primary analysis established, I now turn to an empirical study of these algorithms and the options they construct.

[6.3] $\log^*$ is the number of times the logarithm function must be iteratively applied before the result is less than or equal to 1.

6.5 EXPERIMENTS

I next turn to an empirical study that examines the performance of VI using options generated by the approximation algorithms on simple grid worlds. The first domain is the same Four Rooms grid world studied in previous chapters, and the second is a 9×9 grid world with no walls. In both domains, the agent's goal is to reach the top right corner.

Visualizations. First, we visualize a variety of option types, including the optimal point options, those found by our approximation algorithms, and several option types proposed in the literature. To generate these visuals, we compute the optimal set of point options



(a) Optimal, $k = 2$ (b) Optimal, $k = 4$ (c) Betweenness, $k = 4$

(d) Approximation, $k = 2$ (e) Approximation, $k = 4$ (f) Eigenoptions, $k = 4$

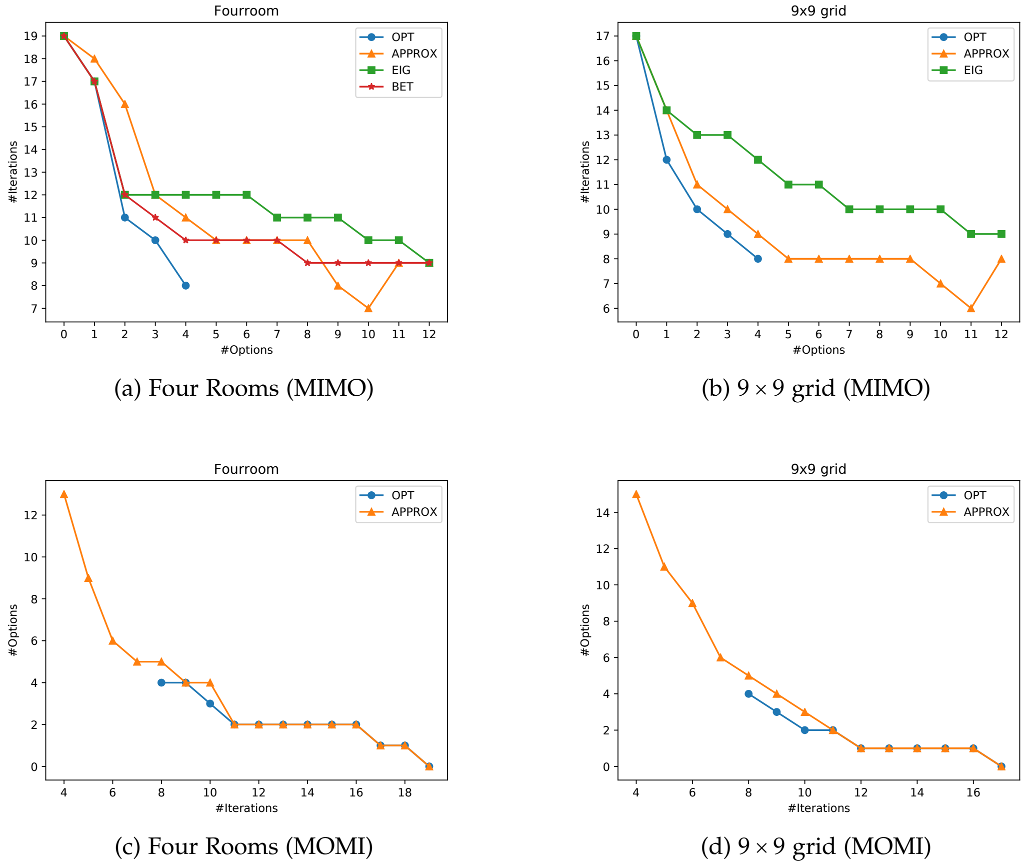
Figure 6.3: Qualitative comparison of the optimal point options with options generated by the approximation algorithm A-MIMO.

by enumerating every possible set of point options and picking the best. We also contrast these options with eigenoptions [212] and options based on betwenness by Şimşek and Barto [291]. Figure 6.3 shows the optimal and bounded suboptimal set of options computed by A-MIMO for $k = 2$ and $k = 4$.

Figure 6.3c shows the four bottleneck states with highest shortest-path betweenness centrality in the state-transition graph. Observe that the optimal options are quite close to the bottleneck states, suggesting that bottleneck states are also useful for planning as a heuristic to find useful subgoals. Figure 6.3f shows the set of subgoals discovered by graph Laplacian analysis following the method of Machado et al. [211]. Both eigenoptions and betweenness options are designed for use in RL, rather than planning, but there is still meaningful qualitative difference in the options discovered. Indeed, one potential reason to acquire good options to plan is for their eventual use in model-based RL.

While such visuals can only highlight qualitative differences in the different methods, it is still apparent that the approximation algorithm and the optimal algorithm find reasonably similar sets of options. That is, contrasting Figure 6.3a with Figure 6.3d, we see that the options generated are in fact relatively similar—both initiate in the bottom left and top left rooms respectively, and are around 10 steps away from the goal. The same is true when $k = 4$: note that in Figure 6.3b and Figure 6.3a there is roughly one option in each room in both cases.

Quantitative Evaluation. Next, we turn to a quantitative evaluation to directly contrast the impact the discovered options have on the speed of VI. Specifically, we run VI using the set of options generated by A-MIMO and A-MOMI and compare their performance to the optimal set of options found by solving the full NP-hard problem. Figure 6.4a and Figure 6.4b present the number of iterations on the Four Rooms and 9×9 grid using a set of options of size k. The experimental results suggest that the approximation algorithm tends to find a set of options slightly worse in performance than the optimal ones. For betweenness options and eigenoptions, we evaluated every subset of options among the four and present results for the best subset found. Because betweenness options are



(a) Four Rooms (MIMO)

(b) 9 × 9 grid (MIMO)

(c) Four Rooms (MOMI)

(d) 9 × 9 grid (MOMI)

Figure 6.4: Quantitative evaluation comparing the planning speed up resulting from the options computed by solving MIMO and MOMI in various ways.

placed close to the optimal options, the performance is close to optimal especially when the number of options are small.

In addition, we used A-MOMI to find a minimum option set to solve the MDP within the given number of iterations. Figure 6.4c and Figure 6.4d show the number of options generated by A-MOMI compared to the minimum number of options. As the data indicate, the optimal approach and A-MOMI find options of similar quality, suggesting that in these simple problems, the approximation algorithm is as effective as solvine thee full NP-hard problem.

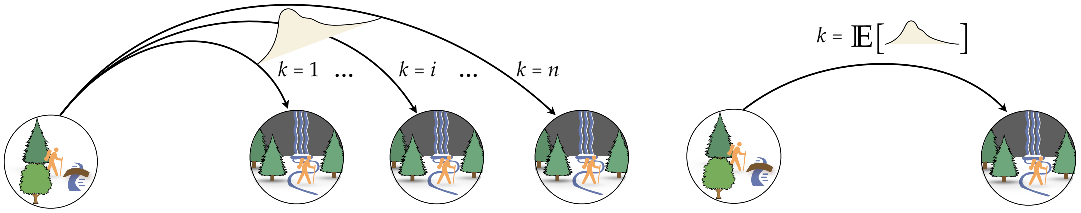
In this chapter, I address a fundamental question concerning the use of action abstractions that help accelerate planning in MDPs. This led to two problem formulations for finding options: 1) minimize the size of option set given a maximum number of iterations (MOMI) and 2) minimize the number of iterations given a maximum size of option set (MIMO). The main results prove that these two problems are both computationally intractable under several assumptions—we here suppose the option models are given, the branching factor is ignored, and that VI is sufficiently general to capture planning algorithms. These assumptions do limit the scope of these results. Fortunately, each problem also permits a polynomial-time algorithm for MDPs with bounded reward and goal states, with bounded optimality for deterministic MDPs. Experimental data support the usefulness of the approximation algorithms, though it is important to be mindful of their computational costs.

7

THE EXPECTED-LENGTH MODEL OF OPTIONS

This chapter is based on "The Expected-Length Model of Options" [10] jointly led by John Winder, also with Marie desJardins and Michael L. Littman.

Making accurate long horizon predictions about the effects of an action can improve an agent's ability to make meaningful decisions. For instance, the hiker in the forest from Chapter 1 is sure to rely on the ability to predict which high level behaviors will lead them to the bridge, the tent, or the waterfall. With such predictive power, agents can take into account the long-term outcomes of an action, and use this information to make informed plans that account for contingencies, uncertainty, and ultimately help determine which actions will maximize value.



(a) Multi-Time Model

(b) Expected-Length Model

Figure 7.1: An intuitive illustration of the MTM (left) and ELM (right).

However, learning models that are suitable for use in making long horizon predictions is challenging. Even ε-accurate one-step models are known to lead to an exponential increase in the error of n-step predictions as a function of the horizon [157, 54], though recent approaches show it is possible to diminish this error through smoothness assumptions about the environment [23]. Moreover, composing an accurate one-step model into an n-step model is known to give rise to predictions of states dissimilar to those seen during training of the model, leading to poor generalization [314]. By encoding only relevant long horizon sequences of behavior, options offer one promising approach for supporting the discovery of accurate long term models. How to obtain an option model tractably, however, remains an open question. To this end, this chapter studies the problem of efficiently computing option models from experience.

I first discuss the sense in which the traditional multi-time model (MTM) of options [263, 264], is highly parameterized, and thus difficult to compute or learn under reasonable constraints. Intuitively, the density modeled by the MTM tracks the outcome of a given option over all possible time steps (Figure 7.1a), which can be impractical to compute even in small domains. In light of this difficulty, I motivate the construction of an alternate model that I call the expected-length model (ELM). The main idea behind the ELM, and indeed, this chapter, is that it is not necessary to model the full joint distribution of possible outcomes of an option, as in the MTM. Instead, it is sufficient to estimate 1) how long, on average, an option takes to run, and 2) a categorical distribution over states where the option terminates. The ELM is formed by combining these two pieces of information (Figure 7.1b) .

I then prove that in goal-based MDPs, the differences in value functions induced by the MTM and the ELM is bounded as a function of the horizon and other relevant properties of the option and environment. I then conduct an empirical study contrasting the performance of the ELM with the MTM in a variety of MDPs. The findings from these experiments suggest that in the right kinds of problems, the ELM is a suitable replacement for the MTM.

7.1 THE EXPECTED-LENGTH MODEL

I now introduce the new option model and motivate its properties. The ELM is defined as follows.

Definition 7.1. *The **expected-length model** (ELM) for a given option o in state s supposes that the distribution of time steps taken by the option can be well approximated by its expected value,* μ_k*:*

$$T_{\mu_k}(s' \mid s, o) := \gamma^{\mu_k} p(s' \mid s, o), \tag{7.1}$$

$$R_{\mu_k}(s, o, s') := \gamma^{\mu_k} \mathbb{E}\left[r_1 + r_2 \ldots + r_{\mu_k} \mid s, o\right], \tag{7.2}$$

where $p(s' \mid s, o)$ *denotes the probability of terminating in* s'*, given that the option was executed in s.*

Modeling only the expected number of time steps throws away information—it ignores, essentially, the particulars of how executing the option can play out. Consider an agent in the usual Four Rooms domain with an option for moving from the top-left room to the top-right one. Suppose the primitive actions are stochastic, with a small probability of moving the agent in the wrong direction. Due to the non-zero probability of slipping, the option may sometimes take five, ten, or even more steps to reach the top-right room. Instead of modeling the full distribution of the number of time steps taken, ELM averages over these quantities (represented by μ_k), and models the transition as taking place over this expected number of time steps. I provide additional intuition for ELM in Section 7.2 by working through a concrete example.

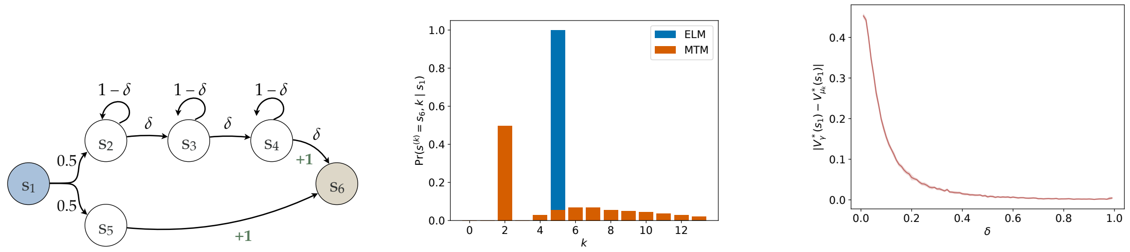
The main result of this chapter demonstrates that this process of distillation is acceptable and desirable, leading to simpler models and often improving the rate at which models are learned. Specifically, I prove that, under mild assumptions, ELM induces similar value functions to MTM, where the bound depends on primarily on the amount of stochasticity in the MDP (and the option's trajectory). From experimental evidence, I conclude that ELM option models can perform competitively to MTM.

7.2 A SIMPLE EXAMPLE

Let us first develop intuition behind ELM through an example, concentrating on the transition model.

Example 7.1. *Consider the six-state MDP in Figure 7.2a, chosen to accentuate the differences in ELM and MTM. Suppose an option initiates in s_1 (shown in blue), and terminates in s_6 (shown in tan). For simplicity, suppose $\beta_o(s) = \mathbb{1}\{s = s_6\}$. The option policy is depicted by the arrows—when the option executes its policy in s_1, it lands in s_2 with probability $\frac{1}{2}$ and s_5 with probability $\frac{1}{2}$. In s_2, when the option executes its policy, the agent stays in s_2 with probability $1 - \delta$, and transitions to s_3 with probability δ (and so on for s_3 and s_4). Conversely, in s_5, the option transitions to s_6 with probability 1.*

Consider the process of estimating the option model at s_6: $T_\gamma(s_6 \mid s_1, o)$ under the MTM. To construct a proper estimate, the MTM must estimate the probability of termination in each state over all possible time steps to determine $\mathbb{P}(s^{(1)} = s_6 \mid s_1, o), \mathbb{P}(s^{(2)} = s_6 \mid s_1, o), \ldots$. This computation involves estimation over arbitrarily many time steps; in some cases, like this one, we might find a closed form based on convergence of the geometric series, but agents cannot always intuit this fact from limited data. In contrast, the ELM models this distribution according to μ_k, the average number of time steps.



(a) (b) Model Diff. $\delta = 0.4$ (c) Value Diff. vs. δ

Figure 7.2: An illustration of the difference between ELM and MTM.

Given the true MDP transition function T, I run n rollouts of the option to termination. Supposing each rollout reports (s, o, r, s', k), with r the cumulative reward received and k the number of time steps taken, it is natural to estimate μ_k with the maximum likelihood estimator (MLE) $\hat{\mu}_k = \frac{1}{n} \sum_{i=1}^{n} k_i$. Additionally, we can estimate $p(s' \mid s, o)$, the probability that o terminates in s', by modeling it as a categorical distribution with $\ell = |\mathcal{S}|$ parameters. Then, it is feasible to estimate each ℓ_i with an MLE.

To summarize:

- The ELM estimates μ_k and $p(s' \mid s, o)$, for each s' of relevance, by using an MLE based on data collected from rollouts of the option.

- The MTM must estimate the probability of terminating in each state, at each time step. It is unclear how to capture this infinite set of probabilities of value economically.

Figure 7.2b shows their differences in the quantity $\mathbb{P}(s^{(k)} = s_6 \mid s_1, o)$, for each time step k. The MTM (in orange) distributes the transition probability across many lengths k. Approximately half of the time, s_6 is reached in two steps via the bottom route through s_5; the rest of the probability mass is spread across higher values, reflecting longer paths (via s_2). The ELM (in blue) instead assumes the option takes $\mu_k = 5$ steps. For both models, each non-zero bar represents a parameter that needs to be estimated, giving a sense of the difficulty in estimating each distribution.

Next consider the mean value difference under each model averaged over 20 runs, presented with 95% confidence intervals in Figure 7.2c. Observe that this value difference which decreases to nearly 0.15 as δ tends to 1 (with VMAX = 1.0). This trend is predicted by the analysis I conduct in Section 7.3, which suggests that the higher the variance over expected number of time steps, the more the ELM deviates from the MTM.

Difficulty of Finding Option Models

The goal of the ELM is to simplify the MTM to be able to estimate and compute the model of a given option more efficiently.

ESTIMATION. Learning an option's MTM involves estimating a complicated probability distribution. Specifically, the general case requires parameters for the (potentially unbounded) number of time steps taken to reach a given s' conditioned on initiating o in s, for each $s' \in \mathcal{S}$. For such cases, a common assumption to make in analyzing complexity is to model the process out to some finite horizon. One reasonable approximation might involve limiting the sum inside the MTM to the first $\lambda = (1-\gamma)^{-1}$ steps as an artificial horizon, thereby yielding $\lambda|\mathcal{S}|^2$ parameters to estimate. In contrast, ELM requires learning the parameters of a categorical distribution indicating the probability of terminating in each state. With one multinomial for each state, any learning algorithm must estimate $2|\mathcal{S}|^2$ total parameters. Depending on the stochasticity inherent in the environment, option policy, and option-termination condition, estimating this smaller number of parameters is likely to be considerably easier ($\lambda \gg 2$).

COMPUTATION. The MTM requires performing the equivalent computation of a Bellman backup until the option is guaranteed to have terminated *just to compute the option's reward function* (Equation 2.34). Due to the decreasing relevance of future time steps due to γ, one might again only compute out to λ time steps to determine R_γ and T_γ. Thus, computing R_γ is roughly as hard as computing the value function of the option's policy (at least out to λ time steps), requiring computational hardness similar to that of an algorithm like Value Iteration (Algorithm 2.3), which is known to be $O(|\mathcal{S}|^2|\mathcal{A}|)$ per iteration, with a rough convergence rate of $\tilde{O}(\lambda|T|)$ for $|T|$ as a measure of the complexity of the true transition function [322, 209]. Conversely, ELM is well suited to construction via Monte Carlo methods. Consider a single simulated experience $e = (s, o, r, s', t)$, of the initial state, the option, termination state, cumulative reward, and time taken. This experience contains each data point needed to compute the components of option o's ELM (Equation 7.1 and Equation 7.2), all sampled directly from the appropriate distributions. With the ELM, option models can be learned from these simulations, $\mathcal{E}$, with each $e \in \mathcal{E}$ needing only labels of where the option began, where it ended, how much reward it received, and how long it took. It is therefore sufficient to run a number of rollouts proportional to the desired accuracy when using ELM. Relying on such methods for computing the MTM again requires estimating a potentially large number of parameters, which is often untenable.

In considering both estimation and computation, note that these are not conclusive analyses of the computational and statistical difficulty of obtaining each model, but take the insights discussed to serve as sufficient motivation for further exploration of the ELM. For instance, there is some similarity in determining the MTM and TD(λ) when $\lambda = 1$ [305], so such estimation can be feasible (see, for instance, Chapter 4 of Parr [258]).

I now turn to the primary analysis of the chapter, which illustrates the deviation between the MTM and the ELM for each of the transition, reward, and value functions.

7.3 ANALYSIS

The main result of this chapter bounds the value difference between the ELM and MTM in goal-based MDPs. This theorem holds under the following two assumptions.

Assumption 7.1. *All MDPs considered are goal-based. That is, in each MDP, there is a unique goal state s_g such that $\forall_{s \in \mathcal{S}, a \in \mathcal{A}} : R(s, a, s_g) = 1$, where $T(s_g \mid s_g, a) = 1$. All other rewards are zero.*

This assumption is used to bound the difference of the ELM and MTM reward functions, but is not required to bound their transition functions.

The next assumption is useful in the analysis, but conceptually there is no reason it cannot be removed in future.

Assumption 7.2. *For every option, the termination probability is non-zero in every state, bounded below by a fixed constant $\beta_{\min} \in (0, 1]$.*

Indeed, while these assumptions slightly limit the scope of the analysis of the ELM, I take the setting to still be sufficiently interesting to offer insights about learning and using option models. Naturally, the relaxation of each assumption is a sensible direction for future work.

At a high level I now show that the following claims hold under these two assumptions:

1. Lemma 7.1: $||T_\gamma - T_{\mu_k}||_\infty$ is bounded.
2. Lemma 7.2: $||R_\gamma - R_{\mu_k}||_\infty$ is bounded in goal-based MDPs.
3. Theorem 7.1: $||V_\gamma^* - V_{\mu_k}^*||_\infty$ is bounded in goal-based MDPs.

I begin with the two lemmas that show the transition and rewards of the ELM are reasonable approximations of the MTM in goal-based MDPs.

Lemma 7.1. *Under Assumption 7.2, the expected-length transition model is sufficiently close to the transition model of the multi-time model. More formally, for any option $o \in \mathcal{O}$, for some real $\tau > 1$, for $\delta = \frac{\sigma_{k,o}^2}{\tau^2}$, and for any state pair $(s, s') \in \mathcal{S} \times \mathcal{S}$, with probability $1 - \delta$:*

$$|T_\gamma(s' \mid s, o) - T_{\mu_k}(s' \mid s, o)| \leq \gamma^{\mu_{k,o} - \tau}(2\tau + 1)e^{-\beta_{\min}}. \tag{7.3}$$

Proof of Lemma 7.1.

> Let $T_\gamma(s' \mid s, o)$ denote the multi-time model, and let $T_{\mu_k}(s' \mid s, o)$ denote the expected length model.

For a fixed but arbitrary state-option-state triple (s, o, s'):

$$\begin{aligned}
&|T_{\gamma}(s' \mid s, o) - T_{\mu_k}(s' \mid s, o)| && (7.4)\\
&= |\sum_{t=1}^{\infty} \gamma^t \mathbb{P}(S_t = s' \mid s, o)\beta(s') - \gamma^{\mu_k} \sum_{t=1}^{\infty} \mathbb{P}(S_t = s' \mid s, o)\beta(s')| && (7.5)\\
&= |\sum_{t=1}^{\infty} \gamma^t \mathbb{P}(s_t = s' \mid s, o)\beta(s') - \gamma^{\mu_k} \mathbb{P}(S_t = s' \mid s, o)\beta(s')| && (7.6)\\
&= |\sum_{t=1}^{\infty} (\gamma^t - \gamma^{\mu_k}) \mathbb{P}(S_t = s' \mid s, o)\beta(s')| && (7.7)\\
&= |\sum_{t=1}^{\infty} (\gamma^t - \gamma^{\mu_k}) \mathbb{P}(S_t = s' \mid s, o)\beta(s')| && (7.8)
\end{aligned}$$

Note that $\mathbb{P}(s_t = s' \mid s, o)\beta(s')$ is bounded above:

$$\mathbb{P}(S_t = s' \mid s, o)\beta(s') \leq (1 - \beta_{min})^t, \tag{7.9}$$

since, in order to be in state s_t at time t we have to *not* terminate in each of $s_1, \ldots s_t$. Further, we know that:

$$(1 - x)^t \leq e^{-xt} \tag{7.10}$$

for any $x \in [0, 1]$. Therefore:

$$\mathbb{P}(S_t, = s' \mid s, o)\beta(s') \leq e^{-\beta_{min} t} \tag{7.11}$$

So, rewriting:

$$\begin{aligned}
|T_{\gamma}(s' \mid s, o) - T_{\mu_k}(s' \mid s, o)| &= |\sum_{t=1}^{\infty} (\gamma^t - \gamma^{\mu_k}) \mathbb{P}(s_t = s', \beta(s') \mid s, o)| && (7.12)\\
&\leq |\sum_{t=1}^{\infty} (\gamma^t - \gamma^{\mu_k}) e^{-\beta_{min} t}|. && (7.13)
\end{aligned}$$

Thus:

$$|T_\gamma(s' \mid s, o) - T_{\mu_k}(s' \mid s, o)| \leq |\sum_{t=1}^{\infty}(\gamma^t - \gamma^{\mu_k})e^{-\beta_{min}t}| \tag{7.14}$$

Let K denote the random variable indicating the number of time steps taken by the option. Now, note that by Chebyshev's inequality, we know that for any $\tau > 1$:

$$\mathbb{P}\{|K - \mu_k| \geq \tau\} \leq \frac{\sigma^2}{\tau^2}. \tag{7.15}$$

Thus, letting $\delta = \frac{\sigma^2}{\tau^2}$, we find that:

$$\mathbb{P}\{|K - \mu_k| \leq \tau\} \geq 1 - \frac{\sigma^2}{\tau^2} = 1 - \delta. \tag{7.16}$$

Thus, with probability $1 - \delta$:

$$\begin{aligned}
|T_\gamma(s' \mid s, o) - T_{\mu_k}(s' \mid s, o)| &\leq \left| \sum_{t=\mu_k-\tau}^{\mu_k+\tau} (\gamma^t - \gamma^{\mu_k})e^{-\beta_{min}t} \right| && (7.17)\\
&\leq \sum_{t=\mu_k-\tau}^{\mu_k+\tau} \left|(\gamma^t - \gamma^{\mu_k})\right| e^{-\beta_{min}t} && (7.18)\\
&\leq \sum_{t=\mu_k-\tau}^{\mu_k+\tau} |\gamma^{\mu_k-\tau}| e^{-\beta_{min}t} && (7.19)\\
&= \gamma^{\mu_k-\tau} \sum_{t=\mu_k-\tau}^{\mu_k+\tau} e^{-\beta_{min}t} && (7.20)\\
&\leq \gamma^{\mu_k-\tau}(2\tau+1)e^{-\beta_{min}} && (7.21)
\end{aligned}$$

Therefore, for $\delta = \frac{\sigma^2}{\tau^2}$:

$$\mathbb{P}\{|T_\gamma(s' \mid s, o) - T_{\mu_k}(s' \mid s, o)| \leq \gamma^{\mu_k-\tau}(2\tau+1)e^{-\beta_{min}}\} \geq 1 - \delta. \tag{7.22}$$

□

Lemma 7.2. *Under Assumption 7.1 and Assumption 7.2, ELM's reward model is similar to MTM's reward model. More formally, for a given option o, for* $\delta = \frac{\sigma^2_{k,o}}{\tau^2}$*, for some* $\tau > 1$*, for any state s:*

$$|R_\gamma(s,o) - R_{\mu_k}(s,o)| = |T_\gamma(s_g \mid s,o) - T_{\mu_k}(s_g \mid s,o)|. \tag{7.23}$$

Thus, by Lemma 7.1, with probability $1 - \delta$*:*

$$|R_\gamma(s,o) - R_{\mu_k}(s,o)| \leq \gamma^{\mu_{k,o}-\tau}(2\tau+1)e^{\beta_{\min}}. \tag{7.24}$$

Proof of Lemma 7.2.

In the goal-based MDP considered all rewards are either 0 or 1 (Assumption 7.1). Thus, if a given option *cannot* reach the goal state, the two reward models are identical, since all accumulated rewards by the option will be zero:

$$|R_\gamma(s,o) - R_{\mu_k}(s,o)| = 0. \tag{7.25}$$

Conversely, if the option *can* reach the goal state, then the expected reward of the option is just the probability, under the relevant transition model (T_γ or T_{μ_k}) of reaching the goal. Therefore, more generally:

$$R_\gamma(s,o) = T_\gamma(s_g \mid s,o), \tag{7.26}$$

$$R_{\mu_k}(s,o) = T_{\mu_k}(s_g, s,o). \tag{7.27}$$

Consequently, by definition:

$$|R_\gamma(s,o) - R_{\mu_k}(s,o)| = |T_\gamma(s_g \mid s,o) - T_{\mu_k}(s_g \mid s,o)| \tag{7.28}$$

Thus, we conclude by applying Lemma 7.1, for $\delta = \frac{\sigma^2}{\tau^2}$, for any s and o:

$$\mathbb{P}\left\{|R_\gamma(s,o) - R_{\mu_k}(s,o)| \leq \gamma^{\mu_k - \tau}(2\tau + 1)e^{\beta_{min}}\right\} \geq 1 - \delta. \tag{7.29}$$

□

Notably, Lemma 7.1 does not depend on Assumption 7.1—it applies to any finite MDP. Hence, it is likely that the ELM reward function is similar to the MTM in a more general class of MDPs than goal-based problems, but I leave such a direction open for future work. With these lemmas in place, I now present the main result of the chapter.

Theorem 7.1. *In goal-based MDPs, the value of any policy over options under ELM is bounded relative to the value of the policy under the multi-time model, with high probability.*

More formally, under Assumption 7.1 and Assumption 7.2, for any policy over options π_o, some real valued $\tau > 1$, $\varepsilon = \gamma^{\mu_{k,o} - \tau}(2\tau + 1)e^{-\beta_{min}}$, $\delta = \frac{\sigma^2}{\tau^2}$, for any state $s \in \mathcal{S}$, with probability $1 - \delta$:

$$|V_\gamma^{\pi_o}(s) - V_{\mu_k}^{\pi_o}(s)| \leq \frac{\varepsilon(1 - \gamma^{\mu_k}) + \gamma^{\mu_k}\frac{\varepsilon}{2}\text{RMax}}{(1 - \gamma^{\mu_k})(1 - \gamma^{\mu_k} + \frac{\varepsilon}{2}\gamma^{\mu_k})}.$$

Proof of Theorem 7.1.

Let

$$\varepsilon = \gamma^{\mu_k - \tau}(2\tau + 1)e^{-\beta_{min}}, \tag{7.30}$$

and again let $\delta = \frac{\sigma^2}{\tau^2}$. By Lemma 7.1 and Lemma 7.2, we know that the reward and transition models are bounded, each with probability $1 - \delta$:

$$|R_\gamma(s,o) - R_{\mu_k}(s,o)| \leq \varepsilon, \tag{7.31}$$

$$|T_\gamma(s' \mid s,o) - T_{\mu_k}(s' \mid s,o)| \leq \varepsilon. \tag{7.32}$$

Then, let

$$V_{\gamma,\varepsilon}^{\pi_\gamma}(s) = R_\gamma(s,o) + \gamma^{\mu_k} \sum_{s' \in \mathcal{S}} \left(\mathbb{P}(s' \mid s,o) + \varepsilon \right) V_{\gamma,\varepsilon}^{\mu_k}(s'). \tag{7.33}$$

Note that, by the transition model bound above:

$$||V_\gamma^{\pi_\gamma} - V_{\mu_k}^{\pi_\gamma}||_\infty \le ||V_{\gamma,\varepsilon}^{\pi_\gamma} - V_{\mu_k}^{\pi_\gamma}||_\infty \tag{7.34}$$

Then, by Lemma 4 by Strehl and Littman [299], we upper bound the right hand side of Equation 7.34 with probability $1-\delta$, for any option o, any policy π, for any state s:

$$|Q_{\gamma,\varepsilon}^{\pi}(s,o) - Q_{\mu_k}^{\pi}(s,o)| \le \frac{(1-\gamma^{\mu_k})\varepsilon + \gamma^{\mu_k}\frac{\varepsilon}{2}\text{RMAX}}{(1-\gamma^{\mu_k})(1-\gamma^{\mu_k}+\frac{\varepsilon}{2}\gamma^{\mu_k})}. \tag{7.35}$$

By combining Equation 7.34 and Equation 7.35, we conclude the proof. □

Thus, in goal-based MDPs, the value of the two models is bounded. The dominant terms in the bound are τ and $\gamma^{\mu_k - \tau}$, which roughly capture the variance over the number of time steps taken by the option and the length of the option's execution. When the option's execution is nearly deterministic, τ is close to 1, and the bound collapses to $3\gamma^{\mu_k}$. Therefore, the bound is tightest when 1) the option or MDP is not very stochastic, and 2) the option executes for a long period of time. Further, the bound is quite loose; the proof of Lemma 7.1 uses Chebyshev's inequality, which does not sharply characterize concentration of measure, and the proof relies on at least one other loose approximation. Hence, in practice, it is likely that the two models will be closer; the experiments I next provide further support for the closeness of the two models in a variety of traditional MDPs.

Finally, for clarity, note that the typical convergence guarantees of the Bellman Operator are preserved under the ELM. The property follows naturally from the main result of Littman and Szepesvári [208], since the ELM is still a well-formed transition model, and $\gamma^{\mu_k} \in [0,1)$ for any $\gamma \in [0,1)$:

Remark 7.1. *The Bellman Operator using the ELM (in place of the MTM) converges to the fixed point* $V^*_{\mu_k}$.

To summarize, in goal-based MDPs, the ELM gives rise to value functions that do not deviate too dramatically from the MTM. The degree of deviation in these value functions scales with the stochasticity inherent to the underlying MDP and option, and inversely with the expected-length of the option. I now discuss experiments investigating the utility of the ELM in RL.

7.4 EXPERIMENTS

I next describe the findings from simulated experiments that offer further support for the hypothesis that the ELM is suitable replacement for the MTM in specific MDPs.

METHODOLOGY. Each experiment is framed as a hierarchical model-based RL problem. That is, the agent reasons with a collection of primitive actions *and* options. All option models are initially unknown, and thus the learning agent estimates each options' reward and transition model from past experience. Throughout, the baseline RL algorithm used is R-Max [54]. As discussed in Section 2.1, R-Max treats unknown (s, a) pairs (or in this case, (s, o) pairs), as providing maximum reward until they become "known" by being executed some m number of times. It is here that the MTM and the ELM differ in application: a transition under the MTM requires adding and updating as many parameters as needed across all k possible time steps, while a transition under the ELM needs only update its running average, μ_k. This is in part due to the number of parameters involved in R-Max, and it is worth noting that a tighter concentration inequality than Hoeffding's (used in R-Max to determine m) may yield similar performance. The policy for each option is computed by running VI (Algorithm 2.3) over the approximate option models.

Each experiment consists of 30 independent trials and is conduced as follows. Every trial, I first sample a new reward function R_i from a prespecified distribution over tasks,

D. Each reward function induces a goal-based problem that provides a large positive reward at goal states, a negative reward at any transition into a failure state, and zero otherwise. The presence of negative rewards deviates slightly from Assumption 7.1, but not in a way that prevents straightforward application of the ELM. Indeed, expanding Lemma 7.2 to more general classes of reward functions is an important open question.

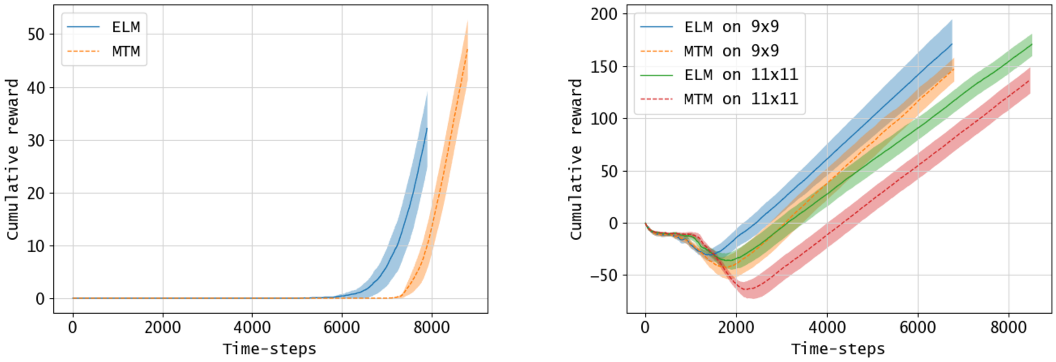
Each trial consists of 300 episodes, terminating at either a goal state, a failure state, or upon reaching a pre-determined maximum number of steps. The hierarchies used are based on options or MAXQ task hierarchies from existing literature [88]. I set $m = 5$ for the confidence parameter in R-Max. Across all MDPs, $\gamma = 0.99$, and all transitions are stochastic with probability $4/5$ of an action succeeding, otherwise transitioning with probability $1/5$ to a different adjacent state (as if another action had been selected).

Experiments are conducted in the following domains: the standard Four Rooms domain seen in previous chapters; Bridge Room, a grid world with a large central room containing pits (failure states) spanned by a bridge with two longer safe corridors on either side; the Taxi domain studied in the experiments from Chapter 3, for which tasks are defined by hierarchical options composed of other options; and, the discrete Playroom domain [295, 180], also using a hierarchy of options.

The Bridge Room domain is a grid world with a large central room contains a bridge of traversable cells that are flanked by failure states. The agent starts on one side of the bridge, with the goal state across the bridge. Two corridors on either side of the central room offer safe but longer pathways from the start state to the bridge. The agent is only given options for moving to the doorways between rooms. The bridge is short but crossing it is dangerous due to the stochasticity inherent in the environment. The optimal policy, then, is to move through either corridor around the bridge room.

The discrete Playroom domain [295, 180] defines a complex sequential decision problem. The agent has three effectors: 1) an eye, 2) a hand, and 3) a marker. Each marker is moved separately. The environment contains music and lights (both start off) and several objects that can be interacted with if both the hand and eye are over them. There is a switch that turns the lights on or off, a green button that turns music on, a red button that turns music off, a ball that can be thrown towards the marker, a bell that rings when hit by the ball, and a monkey that cries only when the lights are off, the music is on, and the bell rings; the goal is to make the monkey cry. Following Konidaris et al. [180], the agent plans over the interact primitive action and options for moving each effector to each object.

RESULTS. I contrast discounted cumulative reward (performance) and time steps (sample complexity) achieved by the ELM compared to the MTM in each of the above domains. Figure 7.3, Figure 7.5, and Figure 7.6 present performance curves with 95% confidence intervals. Overall, the data suggest that the ELM and the MTM attain the same asymptotic performance across each domain, reflecting the fact that they both eventually converge to policies with similar values for each task. Further, the results suggest that the ELM often requires fewer absolute samples to achieve the same quality behavior. This fact is reflected by the learning curve of the ELM terminating earlier than the MTM when plotted over time steps in Figure 7.3a, given that both approaches are run for a consistent number of episodes. This result suggests that policies formed using ELM reach the goal earlier in learning, since the agent more quickly finds a good policy. That is, since the



(a) Four Rooms.

(b) Bridge Room.

Figure 7.3: Learning with options in grid worlds.

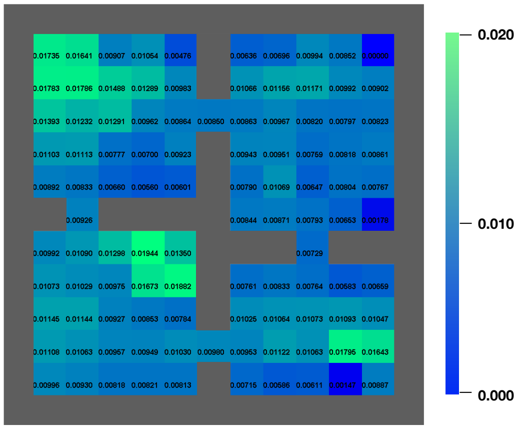
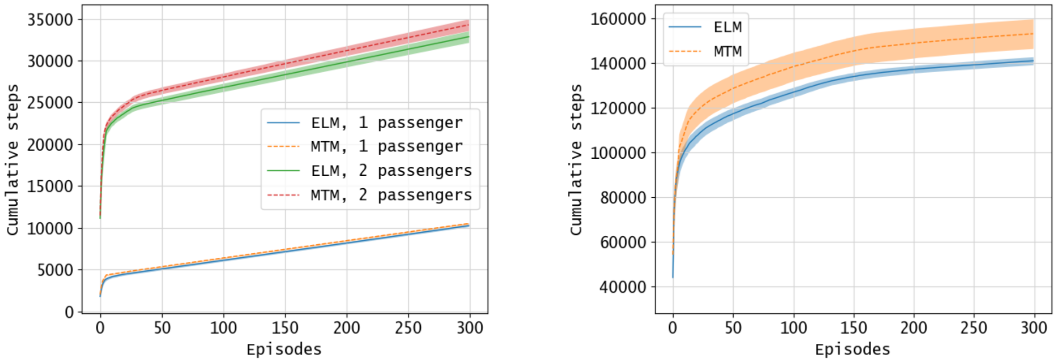


Figure 7.4: The difference in value between ELM and MTM in the Four Rooms task.

goal state is terminal, R-Max using the ELM tends to find the goals more quickly, and thus experiences shorter episodes than the variant using the MTM.

Figure 7.4 displays the difference of the value functions learned under these models in the Four Rooms MDP. In this problem, $\max_s V^*(s) = 1/(1-0.99) = 1/.01 = 100$. The largest gap between V^*_γ and $V^*_{\mu_k}$ is just around 0.02, confirming their similarity in this MDP. Most importantly, despite the difference in the value functions, the policies generated from both are identical; both the MTM and ELM are able to support the discovery of a high-value policy. This is further evidence that the models learned under the ELM are nearly-correct, but per the learning curves in Figure 7.3a, can often be acquired sooner.

Figure 7.3b presents results on two variants of the Bridge Room domain. The inflection points in the learning curves reflect the average point in learning when option models are considered known by R-Max. Observe that the ELM consistently tends to converge earlier in learning, reflecting its ability to quickly estimate the ELM, and thus more quickly make use of the available options to plan. In the 9×9 variant of Bridge Room, however, the results are not statistically significant. For this smaller domain, the bridge is short enough that the approach using the ELM may get lucky and cross the bridge safely several times. If this were to occur, the agent will learn to expect higher reward from the bridge option, negatively impacting the ELM's overall performance until it eventually learns the impact of occasionally falling into a pit. In other words, the high-level behavior of choosing to



(a) Taxi, one and two passenger variants.

(b) Taxi, three passengers.

Figure 7.5: Learning with options in Taxi.

cross the bridge, rather than take the long route around it, tends to yield high variance outcomes over reward. Hence, the confidence interval of the ELM on 9×9 in Figure 7.3b widens as the ELM is less consistent across trials; this domain was used precisely to exhibit this potential downside of the ELM. As suggested by the analysis, domains with high stochasticity are likely to prove problematic for the ELM.

For the Taxi domain, I consider the cumulative number of samples as task complexity increases from one to three passengers. In each case, observe that both approaches are able to learn models in relatively few episodes. In the case of one and two passengers (Figure 7.5a), the approaches achieve similar performance, and the benefit of the ELM over the MTM is statistically significant but minimal. For the largest Taxi task involving three passengers (Figure 7.5b), the results are similar but have lower variance.

Figure 7.6 presents results, again measuring the cumulative steps taken in the discrete Playroom domain. Here, the patterns observed in the other examples recur, though the two approaches diverge later than in the Taxi experiments. This behavior is due to the immense state-action space that must be learned for the effector-moving options. That is, even as the option models are being learned, ELM's practical effect is apparent—favoring expected length leads to the generation of overall shorter plans.

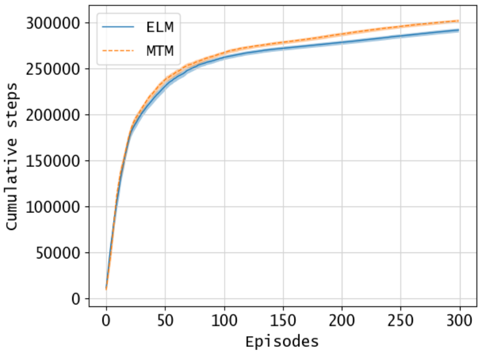


Figure 7.6: Learning with options in Playroom.

Let us take stock. In this chapter, I proposed a simpler option model, the ELM, to replace the standard MTM. The analysis and experiments presented in this chapter showcase the ELM's potential for retaining a reasonable approximation of the MTM while removing the overhead in its construction. The main theorem (Theorem 7.1) bounds the value difference of MTM and ELM in goal-based problems, and the experimental findings corroborate the claim that the ELM can be a suitable replacement for the MTM for RL, especially when the environment is not very stochastic.

8

DISCOVERING OPTIONS FOR EXPLORATION

This chapter is based on "Discovering Options for Exploration by Minimizing Cover Time" [145] led by Yuu Jinnai, also in collaboration with Jee Won Park and George Konidaris.

One of the central challenges of RL is the *explore-exploit* dilemma, discussed briefly in Chapter 2. At a high level, the dilemma highlights the fact that RL agents acting in an unknown world must simultaneously discover *new* things about their surroundings while using what is already known to make good decisions. Such a trade off is especially challenging in environments where the vast majority of rewards are zero with only occasional signal that differentiates good and bad decisions.

Let us return once again to the hiker camping in the forest from Chapter 1. The last few nights, they unfortunately developed a bit of back pain from sleeping inside the tent in a thin sleeping bag. However, the hiker also brought a hammock and a (now broken, sadly) inflatable air mattress. The sun is beginning to set and our hiker is deliberating over possible sleeping configurations. They are presented with a variety of choices: they could move their tent to a new location with softer or flatter ground, rotate their sleeping bag, find a suitable spot to hang the hammock, or try to fix the air mattress. Critically, the hiker does not know in advance which of these activities will help them reliably get a good night of rest and let their back recover. Testing each activity requires time and energy as well, as it is not trivial to hang a hammock or fix an air mattress. Additionally, there is uncertainty over the desirability of each outcome, and the only way to get true signal about the impact of each decision is to actually experiment with a particular sleeping configuration for a period of time. How is the hiker to proceed?

This is precisely an instance of the explore-exploit dilemma: the hiker could exploit what they know now and continue sleeping in their current spot where they have consistently been safe, warm, and comfortable except for the recent back pain. Or, they can explore new activities, facing the uncertainty inherent in each of the new sleeping modes. Depending on the hiker's willingness to experiment, or the potential severity of the outcomes, different strategies might make sense.

Implicit in the hiker's situation is that they are already aware of the other relevant sleeping strategies even if they have not actually tried them before. This is a key source of the utility of action abstraction in facing down the explore-exploit dilemma. If the agent did not have the capacity to consider the "set-up-and-sleep-in-hammock" option, but rather faced down the exponential policy space formed by every permutation of primitive actions, then the problem is effectively hopeless (should I sleep in my sleeping bag, or execute some choice of actions $a_i, a_j, a_k, a_\ell, \ldots, \forall_{i,j,k,\ell,\ldots}$?).

In this way, options have the potential to dramatically alter the exploration problem. Thus, options that are able to accelerate exploration are directly in line with the second desiderata (efficient decision making). Long-horizon actions can enable agents to focus on a particular objective conditioned on a consistent intent, giving rise to directed exploration strategies that prune away irrelevant action sequences.

In light of these intuitions, this chapter studies the problem of discovering options that can aid in exploration. I describe a new polynomial time algorithm first introduced by

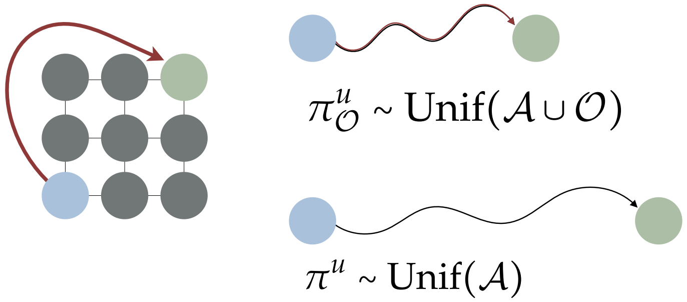

$\pi_{\mathcal{O}}^{u} \sim \mathrm{Unif}(\mathcal{A} \cup \mathcal{O})$
$\pi^{u} \sim \mathrm{Unif}(\mathcal{A})$

Figure 8.1: An example illustrating the main idea behind covering options: the expected length of the random walk between relevant states can be reduced by well chosen options.

Jinnai et al. [145] that improves exploration in finite, sparse reward MDPs by constructing *covering options* that minimize the expected number of steps to reach a (previously unknown) rewarding state without task-dependent reward information. An intuitive illustration of this idea is pictured in Figure 8.1. I suggest that this algorithm is in line with both the first and second abstraction desiderata, since the options do not require unreasonable resource budgets, and are aimed at improving the sample efficiency of RL. Concretely, the proposed algorithm finds a set of options that reduce the expected *cover time* [57] of a random walk over the combined action space, $\mathcal{A} \cup \mathcal{O}$.

Computationally, this problem is equivalent to finding a set of edges to add to the MDP's transition graph that minimize the expected cover time, which is known to be a hard combinatorial optimization problem [52, 53]. In light of this difficulty, the algorithm instead seeks to minimize an upper bound of the expected cover time given as a function of the algebraic connectivity of the graph Laplacian [104, 57, 67] using the heuristic method by Ghosh and Boyd [117]. I study the practical utility of the proposed option discovery algorithm in six finite goal-based MDPs, finding that covering options can indeed accelerate exploration.

The proposed option discovery algorithm makes the assumption that the behavior of an RL agent at the beginning of its learning process may be modeled as a random walk induced by a fixed stationary policy. Without any other knowledge about a given problem, this is sensible: a reasonable default exploration policy is to execute actions uniformly at random (at least until more data is collected). Additionally, this random walk will help establish a simple form of *worst-case* analysis. Surely other more nuanced exploration methods will be faster than a fixed policy. Under this assumption, we build toward an upper bound on the expected cover time of a random walk in the MDP.

As in Chapter 6, this chapter studies *point options* that initiate and terminate in exactly one (possibly different) state each. For more on point options, see Chapter 6, and specifically Definition 6.3. Adding a point option to an MDP corresponds to inserting a single edge into the graph induced by the MDP's transition dynamics. Throughout the chapter, I will refer to the state s with $\beta_o(s) = 1$ as the subgoal state.

8.1 COVER TIME

In this section, I motivate the use of cover time as a mechanism for studying exploration in RL. The cover time is the time required for a random walk to visit all the vertices in a graph [57]. To be precise in grounding this concept, let us first introduce several additional definitions.

First, assume we are given a discrete Markov chain $\{X_t\}$ with state space V denoting the vertices of graph G. The hitting time H_{ij}, where $i, j \in V$, is defined as

$$H_{ij} := \inf \left\{ t \in \mathbb{N} : X_t = j \text{ and } X_0 = i \right\}. \tag{8.1}$$

In other words, H_{ij} is the greatest lower bound on the number of time step t required to reach state j after starting at state i. The cover time starting from state i is then defined as

$$C_i := \max_{j \in V} H_{ij}, \tag{8.2}$$

and the expectation of cover time, $\mathbb{E}[C(G)]$, is the expected cover time of trajectories induced by the random walk, maximized over the start state [57]. Thus, the expected cover time bounds how likely a random walk will lead to a rewarding state, formalized in the following result.

Theorem 8.1. *Given an MDP M that encodes a goal-based MDP with goal g, where a non-positive reward $r_c < 0$ is given for entering non-goal states and $\gamma = 1$. Let W be a random walk transition matrix, $W(s,s') = \sum_{a \in A} \pi(s)T(s' \mid s,a)$ then:*

$$\forall_g : V_g^{\pi}(s) \geq r_c \mathbb{E}[C(G)]. \tag{8.3}$$

where $\mathbb{E}[C(G)]$ is the expected cover time of a transition matrix W.

The proof was first introduced by Jinnai et al. [145]—see Theorem 1.

Intuitively, this result suggests that a smaller expected cover time may translate to more efficient exploration. More formally, let P be a random walk induced by a fixed policy π in an MDP with start state distribution ρ_0. Broder and Karlin [57] prove that the expected cover time $\mathbb{E}[C(G)]$ of P is bounded by a function of the second largest eigenvalue of the random walk matrix $\lambda_{k-1}(P)$ as follows.

$$\mathbb{E}[C(G)] \leq \frac{n^2 \ln n}{1 - \lambda_{k-1}(P)}(1 + o(1)), \tag{8.4}$$

where $n = |V|$ and k is the number of eigenvalues. The normalized graph Laplacian of an unweighted undirected graph is defined as:

$$\mathcal{L} := I - T^{-1/2} A T^{-1/2}, \tag{8.5}$$

where I is an identity matrix [67]. The random walk matrix can be written in terms of the Laplacian:

$$P = T^{-1}A = T^{-1/2}(I - \mathcal{L})T^{1/2}. \tag{8.6}$$

Note that since P and $I - \mathcal{L}$ are *similar* matrices, they have the same eigenvalues and eigenvectors. Therefore, $\lambda_{k-1}(P) = 1 - \lambda_2(\mathcal{L})$, where $\lambda_2(\mathcal{L})$ is the second smallest eigenvalue of $\mathcal{L}$. By Equation 8.4,

$$\mathbb{E}[C(G)] \leq \frac{n^2 \ln n}{\lambda_2}(1 + o(1)). \tag{8.7}$$

Hence, the larger the $\lambda_2(\mathcal{L})$ is, the smaller the upper bound on the expected cover time.

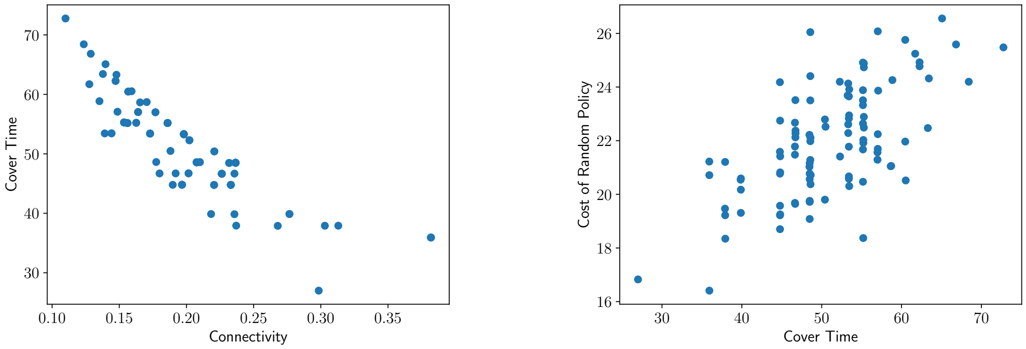
The second smallest eigenvalue of $\mathcal{L}$ is known as the *algebraic connectivity* of the graph, with its corresponding eigenvector referred to as the *Fiedler vector* [104]. There are several operations that can be applied to the graph to increase the algebraic connectivity. For instance, if we add well chosen nodes to the graph, the connectivity changes, but not necessarily in a way that improves the cover time. Alternatively, we might reroute edges in the graph. However, in general it is undesirable to reroute edges as this changes the space of representable policies and may destroy an agent's ability to represent a high value policy. Finally, we might add entirely new edges to the graph—in RL, this is effect of adding options to the primitive action space. Therefore, adding edges is a reliable way to reduce the cover time without potentially sacrificing optimality. Naturally, if more information about the problem is available (such as which primitive actions may be pruned on a per-state basis), other operations may be considered as well to lower cover time.

8.1.1 *Cover Time Experiments*

In the previous section, I illustrated how the algebraic connectivity of an MDP's transition graph relates to the expected cover time. I now describe a simple experiment that further examines this relationship.

The experiment consists of two steps. First, generate 100 random connected graphs with 10 nodes with edge density fixed to 0.3.[8.1] Second, approximate the expected cover

[8.1] The graph generation process proceeds as follows. First, start with a single node. Pick one node from the existing graph and add an edge to connect to a new node. Follow this procedure for the number of nodes



(a) (Connectivity) ~ (cover time)

(b) (cover time) ~ (cost of the policy)

Figure 8.2: The relationship between (a) algebraic connectivity (λ_2) and cover time on randomly generated graphs, and (b) the cover time of a random walk vs. the cost of random policy.

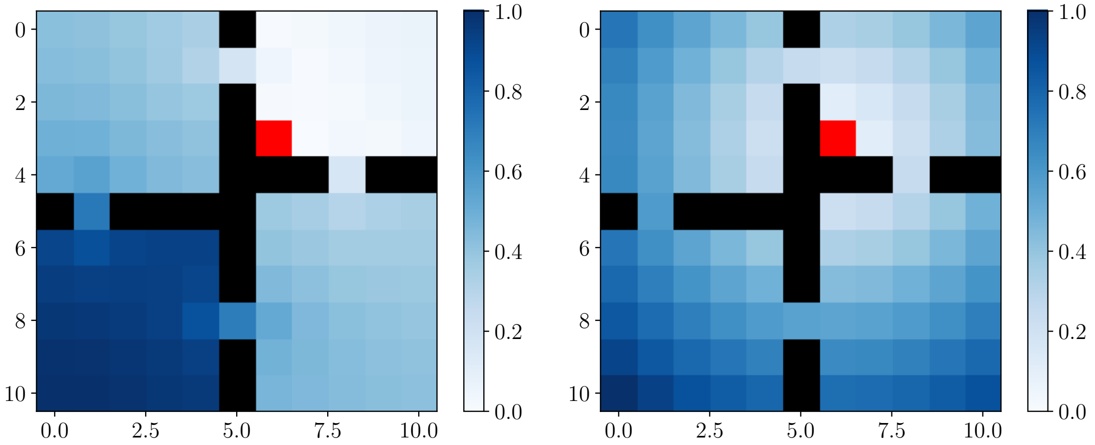
time of a random walk on a random graph by sampling 10,000 trajectories induced by the random walk and computing their average cover time.

Figure 8.2a shows the relationship of the algebraic connectivity and the expected cover time of the random walk induced by the uniform random policy. The takeaway from these results is that the random walk tends to have smaller expected cover time when the underlying state-transition graph has larger algebraic connectivity. Conversely, Figure 8.2b shows the expected cost of a random policy from the initial state to reach the goal state. Here, observe that the cost of a random policy is correlated to the cover time, as expected.

8.2 COVERING OPTIONS

As discussed, computing the precise set of edges that minimize expected cover time in a graph is thought to be NP-Hard [13]. Even a good solution is hard to find due to Braess's paradox [52, 53] which states that the expected cover time does not monotonically decrease as edges are added to the graph.

$n-1$, generating a random tree of size n. Then, pick an edge uniformly randomly from E^c until the edge density reaches the threshold.



(a) Fiedler vector

(b) Euclidean distance

Figure 8.3: The distance between the red state and all other states, measured via Fiedler vector (left) and Euclidean distance (right).

Therefore, the expected cover time is often minimized indirectly by maximizing algebraic connectivity [104, 67]. In particular, the expected cover time is upper bounded by quantity related to the algebraic connectivity (Equation 8.7), and by maximizing this quantity, the bound can be minimized [57]. Choosing the set of edges that maximize the algebraic connectivity of a given graph is known to be NP-hard [241]. However, Ghosh and Boyd [117] developed an approximation procedure for this problem that proceeds as follows.

1. Compute the second smallest eigenvalue and its corresponding eigenvector (the Fiedler vector) of the Laplacian of the state transition graph G.

2. Let v_i and v_j be the states with the largest and smallest values in the Fiedler vector respectively. Generate two point options, the first with $\mathcal{I}_o = \{v_i\}$, $\beta_{o_1} = \mathbb{1}\{v_j\}$, and the second with $\mathcal{I}_{o_1} = \{v_j\}$ and $\beta_{o_2} = \mathbb{1}\{v_i\}$. Each option policy is the optimal path from the initial state to the termination state.

3. Set $G \leftarrow G \cup \{(v_i, v_j)\}$ and repeat the process until the number of options reaches k.

Intuitively, the algebraic connectivity represents how well the graph is connected. The Fiedler vector is an embedding of a graph to a line (that is, a real number) where nodes connected by an edge tend to be placed close to one another (for example, see Figure 8.3). A pair of nodes with the maximum and minimum value in the Fiedler vector are the most distant nodes in this embedding space. Thus, our proposed method greedily connects the two most distant nodes in the embedding, thereby greedily maximizing the algebraic connectivity to a first order approximation [117].

The result is a set of options that is guaranteed to minimize the lower bound on the expected cover time, as summarized by the following result.

Theorem 8.2. *Assume that a random walk induced by a policy π is a uniform random walk:*

$$W(u,v) := \begin{cases} 1/d_u & \textit{if } u \textit{ and } v \textit{ are adjacent,} \\ 0 & \textit{otherwise,} \end{cases} \tag{8.8}$$

where d_u is the degree of the node u. Adding two options by the algorithm improves the upper bound of the cover time if the multiplicity of the second smallest eigenvalue is one:

$$\mathbb{E}[C(G')] \leq \frac{n^2 \ln n}{\lambda_2(\mathcal{L}) + F}(1 + o(1)), \tag{8.9}$$

where $\mathbb{E}[C(G')]$ is the expected cover time of the augmented graph, $F = \frac{(v_i - v_j)^2}{6/(\lambda_3 - \lambda_2) + 3/2}$, and v_i, v_j are the maximum and minimum values of the Fiedler vector.

The proof was first introduced by Jinnai et al. [145]—see Theorem 2.

The state transition graph G may either be given to the agent as input or learned during interaction with the MDP. Throughout, we assume that the graph is undirected and strongly connected, so every state is reachable from every other state. As in the work by Machado et al. [211], the proposed algorithm can be generalized to MDP's with rich state spaces using an incidence matrix instead of an adjacency matrix; such a generalization was recently carried out by Jinnai et al. [146] to great effect.

To summarize the approach, the expected cover time of a random walk is a useful measure for approximating the exploration difficulty of goal-based MDPs. Under this approximation, we design an option discovery algorithm that decreases the cover time by choosing options that connect states that are most distant according to the Fiedler vector. I now discuss findings from an empirical study first carried out by Jinnai et al. [145] that examines the utility of covering options.

8.3 EXPERIMENTS

We conduct experiments with six finite MDPs, many of which should be familiar from previous chapters. These domains include the 9×9 grid from Chapter 6, the standard Four Rooms grid world, Parr's maze [259], Taxi from Chapter 3, the classic disc puzzle Towers of Hanoi, and a discrete driving domain called Race Track. Towers of Hanoi is a classic game consisting of three pegs that can hold different size discs (sorted in decreasing order of disc diameter) on any of the pegs. The goal is to move all discs from a single initial peg to a goal peg while keeping the constraint that each smaller disc is placed above a larger one (or is the only disc on the peg). In Race Track, the agent must reach the finish line by driving a car without hitting the track's boundary. The agent may change the car's vertical and lateral velocity by +1, -1, or 0 in each time step. If the car hits the track boundary, it is moved back to the starting position.

In each MDP, we compare the performance of covering options, eigenoptions [211], and betweenness options [291]. These methods were chosen for similar reasons discussed in Chapter 6: they are closely related option discovery methods that do not require reward information. More detail about eigenoptions and betweenness options is provided in subsection 2.3.1.

To make the comparison more direct, we experiment with a point option variant of the eigenoption method, though notably this was not the intend structure for eigenoptions. For the k-eigenvectors that correspond to the smallest k eigenvalues, we generate a point option from a state with the highest (or lowest) value to a state the lowest (or

	Four Rooms		9×9 grid	
	λ_2	$\mathbb{E}[C(G')]$	λ_2	$\mathbb{E}[C(G')]$
Covering options	**0.065**	**672.0**	**0.24**	**258.6**
Eigenoptions	0.054	695.9	0.19	261.5
No options	0.023	1094.8	0.12	460.5

Table 8.1: Comparison of the algebraic connectivity and the expected cover time of covering options and eigenoptions.

highest) value in the eigenvector. The point option constructed in this way minimizes the eigenvalue of each corresponding eigenvector.

First, we present a simple quantitative evaluation measuring the impact covering options and eigenoptions have on the algebraic connectivity (λ_2) and expected cover time ($\mathbb{E}[C(G')]$) in each of the Four Rooms and 9×9 grid worlds. Results are presented in Table 8.1. In both domains the covering options achieve larger algebraic connectivity and smaller expected cover time than eigenoptions as desired, providing initial confirmation that covering options perform as expected.

8.3.1 *Visuals*

We next present a series of visualizations that highlight important qualitative properties of covering options. Figure 8.4 visualizes the eight covering options and eigenoptions found in Four Rooms and the 9×9 grid world. Note that each algorithm may output many

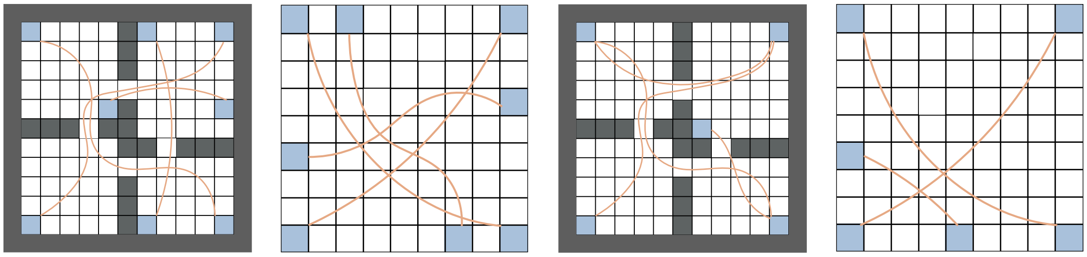
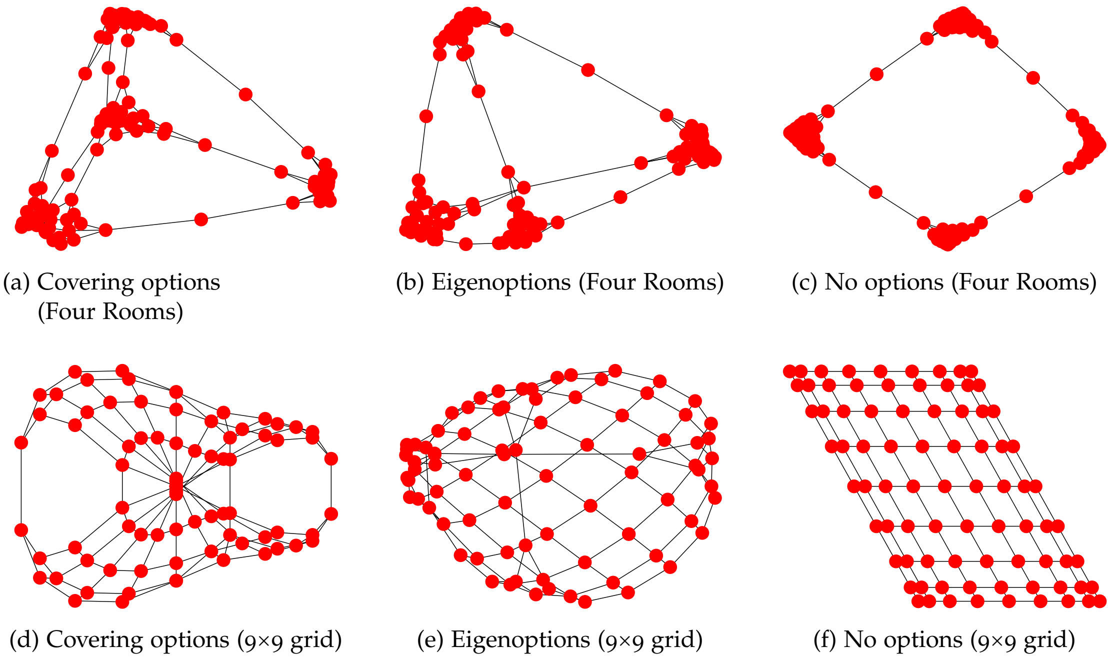

(a) Covering options (b) Covering options (c) Eigenoptions (d) Eigenoptions

Figure 8.4: Visualization of covering options and eigenoptions in Four Rooms and the 9×9 grid world.

different sets of options, and we here choose to visualize just one set. Observe that in each MDP, both option types tend to connect options that are far apart in the underlying MDP. In Four Rooms, for instance, eigenoptions and covering options tend to connect the states in the opposite corners together. In the 9×9 grid world, this quality is also present. The options found by both approaches tend to connect states that are far apart, suggesting that they each increase the algebraic connectivity of the MDP's transition graph.

Next, we further highlight the qualitative impact different options have on these two grid worlds. Figure 8.5 presents the spectral graph drawing [181] of the state-transition graph augmented with each option type. The spectral graph drawing is a technique that is used to visualize the graph topology using eigenvectors of the graph Laplacian. Each node n in the state-space graph is placed at $(v_2(n), v_3(n))$ in the (x, y)-coordinate, where v_i is the i-th smallest eigenvector of the graph Laplacian. These visuals provide further qualitative support for the hypothesis that the option generation methods are successfully connecting distant states.


(a) Covering options (Four Rooms)

(b) Eigenoptions (Four Rooms)

(c) No options (Four Rooms)

(d) Covering options (9×9 grid)

(e) Eigenoptions (9×9 grid)

(f) No options (9×9 grid)

Figure 8.5: Spectral graph drawing of the state-transition graph.

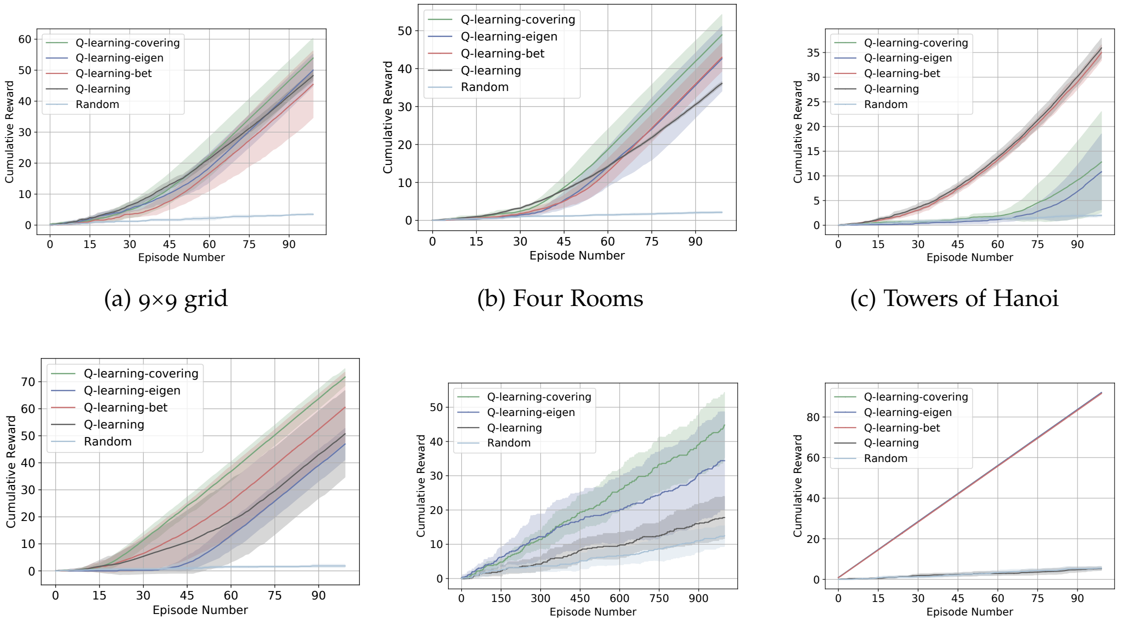
8.3.2 *RL with Options*

We next present findings from two sets of experiments examining how the options found by our algorithm can impact RL.

OPTIONS COMPUTED OFFLINE. In the first variant, we suppose that the transition graph is given to the option discovery method offline, and the computed options are given to an RL algorithm before learning begins. In each experiment we test with Q-learning as the underlying RL algorithm paired with different option types, where $\alpha = 0.1, \gamma = 0.95$. Each approach is run for 100 episodes with 100 steps per episode for the 9×9 grid, and 500 steps per episode in Four Rooms, Hanoi, and Taxi.

As one caveat, following the methods of Machado et al. [211], we evaluate our method using a sample-based approach for option discovery in both Race Track and Parr's maze. That is, instead of giving the agent access to the whole adjacency matrix, the agent instead samples 100 trajectories of a uniform random policy in the MDP, and uses this data to form an incidence matrix. We sampled each trajectory for 1000 steps for Parr's maze and 100 steps for the Race Track domain, and use these data to generate an incidence matrix to inform option generation. As the agent has no prior knowledge on states not present in the incidence matrix, the agent terminates the option if it reaches a state outside of the incidence matrix.

Figure 8.6 presents the mean cumulative reward per averaged over five runs in each MDP. In some cases, the options neither accelerate nor deteriorate the learning—for instance, in both the 9×9 grid and Four Rooms, each of covering, eigen, and betweenness options all perform comparably to regular Q-learning. In Four Rooms, the quality of the policy learned by each of the variants including options does seem to be slightly higher on average than that of Q-learning, though it is not a statistically significant improvement. Conversely, in Race Track, we see covering options and betweenness options dramatically



(a) 9×9 grid

(b) Four Rooms

(c) Towers of Hanoi

(d) Taxi

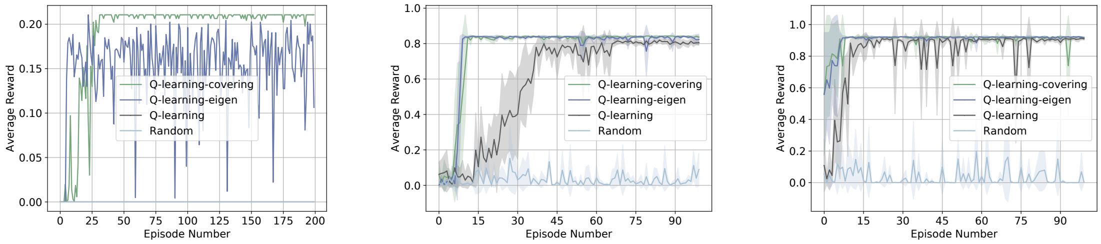
(e) Parr's Maze

(f) Race Track

Figure 8.6: Comparison of RL performance with different option generation methods.

outperform all other methods, while in Towers of Hanoi, covering options and eigenoptions negatively impact learning. In summary, the data suggest that each option type can both help or hurt learning depending on the context, but options tend to help more often than hurt on the studied domains.

OPTIONS COMPUTED ONLINE. Lastly, we study the case where options are discovered online during learning. Each agent generates four options to add to their option set every 10,000 step for Parr's maze and 500 steps for the Towers of Hanoi and Taxi, until $|\mathcal{O}| = 32$. Each agent is given 100 episodes consisting of 10,000 steps each in Parr's maze and 100 steps each for Hanoi and Taxi. The policy of each option is computed by forming the greedy policy relative to a Q function learned by running Q-learning ($\alpha = 0.1, \gamma = 0.95$) on the sampled data to convergence. Here, I give an intrinsic reward of 1 to the agent when it reaches the subgoal state and ignore the rewards from the environment.



(a) Parr's maze

(b) Towers of Hanoi

(c) Taxi

Figure 8.7: Comparison of online option generation methods.

Results are presented in Figure 8.7, indicating the average reward per episode. Observe that across all three domains, Q-learning paired with covering options is able to reliably find a good policy, giving support to the claim that covering options can in fact accelerate exploration. In Parr's maze, for instance, a goal-base problem with a long horizon before any goal is obtained, the approach with covering options is able to find the goal a non-negligible fraction of episodes after around 25 episodes, whereas an agent with primitive actions is unable to find the goal throughout all of learning. Further observe that covering options and eigenoptions tend to perform similarly, suggesting that they can each be useful for accelerating exploration in RL.

8.3.3 *Concluding Remarks*

In this chapter, I illustrated the sense in which appropriate action abstractions can accelerate exploration in RL. In the previous two chapters, I concentrated on finding options that make planning efficient (Chapter 6), and motivated an new alternative to the standard option models, (Chapter 7). Collectively, the results established in this part of the dissertation offer support for the great potential of action abstraction to accelerate and improve RL, and provide concrete paths to action abstraction that can satisfy the desiderata.

I now turn to the next and final part of the dissertation in which I study good combinations of state and action abstraction.

Part 4

STATE-ACTION ABSTRACTION

9

VALUE PRESERVING STATE-ACTION ABSTRACTIONS

This chapter is based on "Value Preserving State-Action Abstractions" [11] with Nathan Umbanhowar, Khimya Khetarpal, Dilip Arumugam, Doina Precup, Michael L. Littman.

In light of the separate benefits of state and action abstraction (see Part 2 and Part 3), it has long been of interest as to how to appropriately combine the two methods. To this end, the focus of this chapter is on the following question.

Which combinations of state abstractions and options preserve
representation of near-optimal policies?

The main result of this chapter summarize new analysis addressing this question, providing a concrete step toward state-action abstractions that can satisfy all three desiderata. Specifically, I here introduce combinations of state abstractions (ϕ) and options ($\mathcal{O}$) that are guaranteed to preserve representation of near-optimal policies in finite MDPs. These combinations, and the analysis thereof, resemble the classes of approximate state abstraction studied in Chapter 3—the main theorem of the chapter (Theorem 9.1) highlights the general relationship between approximate knowledge used in forming these abstractions and the quality of the best policy representable in the abstract. I will then extend this result to the case of *hierarchical* abstractions, providing a general scheme for characterizing value-preserving hierarchies under mild assumptions.

To perform this analysis, I first define ϕ-relative options, a general formalism for analyzing the value loss of a state abstraction paired with a set of options. I then prove four sufficient conditions, along with one separate necessary condition, for ϕ-relative options to preserve near-optimal behavior in any finite MDP. I further prove that ϕ-relative options can be composed to induce a hierarchy that preserves near-optimal behavior under appropriate assumptions about the hierarchy's construction. I suggest these results can support the development of principled methods that learn and make use of value-preserving abstractions.

9.1 ANALYSIS: STATE-ACTION ABSTRACTIONS

I incorporate state and action abstraction into RL as pictured in Figure 9.1. When the environment transitions to a new state s, the agent processes s via ϕ yielding the abstract state, s_ϕ. Then, the agent chooses an option from among those that initiate in s_ϕ and follows the chosen option's policy until termination, where this process repeats. In this

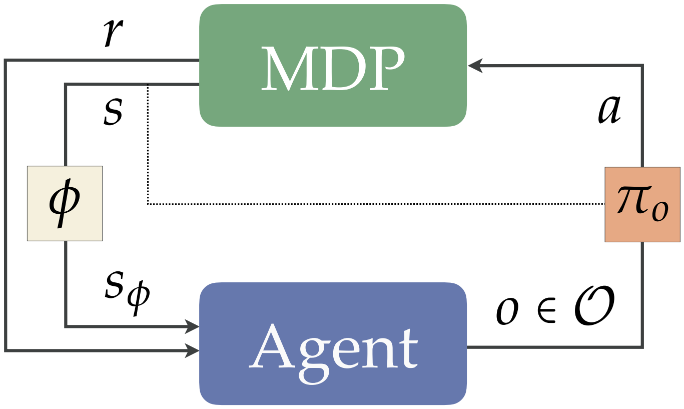


Figure 9.1: State and action abstraction in RL.

way, an RL agent can reason in terms of abstract state and action alone, without knowing the true state or action space.

To analyze the value loss of these joint abstractions, I first introduce ϕ-relative options, a simple means of combining state abstractions with options.

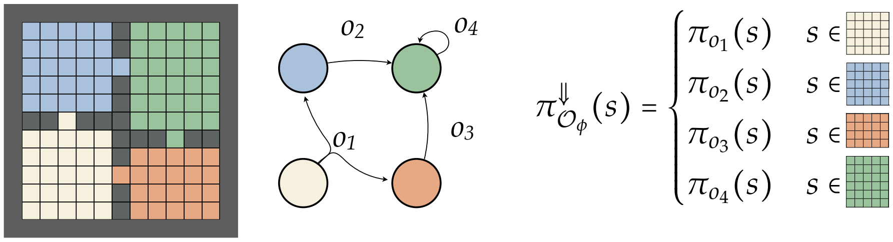
Definition 9.1. *For a given ϕ, an option o is said to be **ϕ-relative** if and only if there is some $s_\phi \in \mathcal{S}_\phi$ such that, for all $s \in \mathcal{S}$:*

$$\mathcal{I}_o = \{s' \in s_\phi\}, \quad \beta_o(s) = \mathbb{1}\{s \notin s_\phi\}, \quad \pi_o \in \Pi_{s_\phi}, \tag{9.1}$$

where $\Pi_{s_\phi} = \left\{\pi : \{s' \in s_\phi\} \rightarrow \Delta(\mathcal{A})\right\}$ is the set of all ground policies defined over ground states in s_ϕ, and $s \in s_\phi$ is shorthand for $s \in \{s' \in \mathcal{S} : \phi(s') = s_\phi\}$.

Intuitively, these options initiate in exactly one abstract state and terminate when the option policy leaves the abstract state. I henceforth denote $(\phi, \mathcal{O}_\phi)$ as a state abstraction paired with a set of ϕ-relative options, and denote $\mathcal{O}_\phi$ as any non-empty set that 1) contains only ϕ-relative options, and 2) contains at least one option that initiates in each $s_\phi \in \mathcal{S}_\phi$.

EXAMPLE. Let us again consider the classical Four Rooms domain. Suppose that the state abstraction ϕ turns each room into an abstract state. Then any ϕ-relative option in this domain is one that initiates anywhere in one of the rooms and terminates as soon as the agent leaves that room, as pictured in Figure 9.2a. The only degree of flexibility in grounding a set of ϕ-relative options for the given ϕ, then, is which *policies* are associated with each option, and how many options are available in each abstract state. If, for instance, the optimal policy π^* were chosen for an option in the top right room, but the uniform random policy were available everywhere else, how might that impact the overall suboptimality of the policies induced by the abstraction? I now build toward the



(a) Assignment of options to each s_ϕ via $\pi_{\mathcal{O}_\phi}$.

(b) Construction of $\pi^{\Downarrow}_{\mathcal{O}_\phi}$.

Figure 9.2: Grounding policy $\pi_{\mathcal{O}_\phi}$ to $\pi^{\Downarrow}_{\mathcal{O}_\phi}$.

main result of the chapter (Theorem 9.1) that clarifies the precise conditions under which ε-optimal policies are representable under different $(\phi, \mathcal{O}_\phi)$ pairs.

As discussed in Chapter 2, the value loss of an abstraction captures the gap in ground value between the best ground policy and best abstract policy. While value loss has a straightforward definition for state abstraction, it is not so clear for action abstraction. To analyze the value loss of state-action abstraction pairs $(\phi, \mathcal{O}_\phi)$, I first show that any such pair gives rise to an abstract policy over $\mathcal{S}_\phi$ and $\mathcal{O}_\phi$ that induces a unique policy in the original MDP (over the entire state space). Critically, this property does not hold for arbitrary options due to their semi-Markovian nature.

Remark 9.1. *Every deterministic policy defined over abstract states and ϕ-relative options, $\pi_{\mathcal{O}_\phi} : \mathcal{S}_\phi \to \mathcal{O}_\phi$, induces a unique Markov policy in the ground MDP, $\pi^{\Downarrow}_{\mathcal{O}_\phi} : \mathcal{S} \to \Delta(\mathcal{A})$. We let $\Pi_{\mathcal{O}_\phi}$ denote the set of abstract policies representable by the pair $(\phi, \mathcal{O}_\phi)$, and $\Pi^{\Downarrow}_{\mathcal{O}_\phi}$ denote the corresponding set of policies in the original MDP.*

Proof of Remark 9.1.

Consider an arbitrary deterministic policy $\pi_{\mathcal{O}_\phi}$. By definition, this policy assigns one option to each abstract state. Let $\mathcal{O}_\pi$ denote the set of options this policy assigns.

By construction of ϕ-relative options, for every ground state $s \in \mathcal{S}$ there is one unique option $o_{\phi(s)} \in \mathcal{O}_\pi$ that can be executed in s.

Therefore, we construct a policy $\pi^{\Downarrow}_{\mathcal{O}_\phi}$ as the combination of option policies in $\mathcal{O}_\pi$. Specifically, letting $\pi_{o_{\phi(s)}}$ denote the option policy of the option in $\mathcal{O}$ that is assigned to $\phi(s)$:

$$\pi^{\Downarrow}_{\mathcal{O}_\phi}(s) = \pi_{o_{\phi(s)}}(s) \tag{9.2}$$

□

This remark gives us a means of translating a policy over ϕ-relative options into a policy over the original state and action space, $\mathcal{S}$ and $\mathcal{A}$. Consequently, it is possible to extend the notion of value loss studied in previous chapters to apply to a set of options paired with a state abstraction: every $(\phi, \mathcal{O}_\phi)$ pair yields a set of policies in the original MDP, $\Pi^{\Downarrow}_{\mathcal{O}_\phi}$. The value loss of $(\phi, \mathcal{O}_\phi)$ as then the value loss of the best policy in this set.

Definition 9.2. *The **value loss** of $(\phi, \mathcal{O}_\phi)$ is the smallest degree of suboptimality achievable:*

$$L(\phi, \mathcal{O}_\phi) := \min_{\pi_{\mathcal{O}_\phi} \in \Pi_{\mathcal{O}_\phi}} \left\| V^* - V^{\pi^{\Downarrow}_{\mathcal{O}_\phi}} \right\|_\infty . \tag{9.3}$$

Note that this notion of value loss is *not* well defined for options in general, since they induce a semi-MDP: there is no well-formed *ground* value function of a policy over options, but rather, a semi-Markov value function. As a simple illustration, consider a ground state s_g, two options o_1 and o_2 (either of which could be executing in s_g), and a policy $\pi_{\phi,o}$ over abstract states and options. It could be that o_1 or o_2 is currently executing when s_g is entered or that either option has just terminated, requiring $\pi_{\phi,o}$ to select a new option. Each of these three cases induces a distinct value $V^{\pi_{\phi,o}}(s_g)$ which is then difficult to distill into a single ground value function. This is a key reason to restrict attention to

ϕ-relative options, each of which retains structure that couples with the corresponding state abstraction ϕ to yield value functions in the ground MDP.

9.1.1 *Four Classes of Value Preserving State-Action Abstractions*

I now show how different classes of ϕ-relative options can represent near-optimal policies. We define an option class by a predicate $\lambda : \mathcal{O}_\phi \mapsto \{0, 1\}$, and say that a set of ϕ-relative options $\mathcal{O}_\phi$ belongs to the class $\mathcal{O}_{\phi,\lambda}$ if and only if $\lambda(\mathcal{O}_\phi) = 1$.

I begin by summarizing the four new ϕ-relative option classes, drawing inspiration from other forms of abstraction [86, 273, 203, 142, 4, 247] discussed in more detail in subsection 2.2.1 and subsection 2.3.1. For each class, I will refer to the *optimal* option in s_ϕ, $o^*_{s_\phi}$, as the ϕ-relative option that initiates in s_ϕ and executes π^* until termination. These classes were chosen as they closely parallel existing properties studied in the literature. The four classes are as follows:

1. *Similar Q^* Functions:* In each s_ϕ, there is at least one option o that has similar Q^* to $o^*_{s_\phi}$.

2. *Similar Models:* In each s_ϕ, there is at least one option o that has a similar multi-time model [264] to $o^*_{s_\phi}$.

3. *Similar k-Step Distributions:* In each s_ϕ, there is at least one option o that has a similar k-step termination state distribution to $o^*_{s_\phi}$, based off the hierarchical construction introduced by Nachum et al. [247]. Loss bounds will only apply to goal-based MDPs.

4. *Approximate MDP Homomorphisms:* Any deterministic $\pi_{\mathcal{O}_\phi}$ can encode an MDP homomorphism. The MDP homomorphism option class is defined by a guarantee on the quality of the resulting homomorphism.

I now present each class in full technical detail. As stated, the first two classes guarantee ε closeness of values and models respectively. More concretely:

SIMILAR Q^*-FUNCTIONS $(\mathcal{O}_{\phi,Q^*_\varepsilon})$. The ε-similar Q^* predicate defines an option class where:

$$\lambda(\mathcal{O}_\phi) \equiv \forall_{s_\phi \in \mathcal{S}_\phi} \exists_{o \in \mathcal{O}_\phi} : \max_{s \in s_\phi} |Q^*_{s_\phi}(s, o^*_{s_\phi}) - Q^*_{s_\phi}(s, o)| \le \varepsilon_Q, \tag{9.4}$$

where

$$Q^*_{s_\phi}(s,o) := R(s, \pi_o(s)) + \gamma \sum_{s' \in \mathcal{S}} T(s' \mid s, \pi_o(s)) \left(\mathbb{1}(s' \in s_\phi) Q^*_{s_\phi}(s', o) + \mathbb{1}(s' \notin s_\phi) V^*(s') \right). \tag{9.5}$$

This Q-function describes the expected return of starting in state s, executing a ϕ-relative option o until leaving $\phi(s)$, then following the optimal policy thereafter. More generally, this class of $\phi, \mathcal{O}_\phi$ pairs captures all cases where each abstract state has at least one option that is useful. Note that the identity state abstraction paired with the degenerate set of options that exactly encodes the execution of each primitive action will necessarily be an instance of this class.

SIMILAR MODELS $(\mathcal{O}_{\phi,M_\varepsilon})$. The ε-similar T and R predicate defines an option class where:

$$\lambda(\mathcal{O}_\phi) \equiv \forall_{s_\phi \in \mathcal{S}_\phi} \exists_{o \in \mathcal{O}_\phi} : \left\| T^{s'}_{s,o^*_{s_\phi}} - T^{s'}_{s,o} \right\|_\infty \le \varepsilon_T \text{ and } \left\| R_{s,o^*_{s_\phi}} - R_{s,o} \right\|_\infty \le \varepsilon_R, \tag{9.6}$$

where $R_{s,o}$ and $T^{s'}_{s,o}$ are shorthand for the reward model and multi-time model of Sutton et al. [311]. Roughly, this class states that there is at least one option in each abstract state that behaves similarly to the optimal option in that abstract state, $o^*_{s_\phi}$, throughout its execution in the abstract state.

I next derive two classes of ϕ-relative options based on abstraction formalisms from existing literature. The first is based on the hierarchical construction introduced by Nachum et al. [247], while the second shows that ϕ-relative options can describe an MDP homomorphism [273].

SIMILAR K-STEP DISTRIBUTIONS ($\mathcal{O}_{\phi,\tau}$). Let $p(s',k \mid s,o)$ denote the probability of option o terminating in s' after k steps, given that it initiated in s. We define this class by the following predicate:

$$\lambda(\mathcal{O}_\phi) \equiv \forall_{s_\phi \in \mathcal{S}_\phi} \exists_{o \in \mathcal{O}_\phi} \forall_{k \in \mathbb{N}} : \max_{s \in s_\phi, s' \in \mathcal{S}} |p(s',k \mid s, o^*_{s_\phi}) - p(s',k \mid s,o)| \leq \tau. \tag{9.7}$$

Intuitively, this class of $\mathcal{O}_\phi$ states that in each abstract state, there exists an option that can induce sufficiently similar k-step state distributions to executing the optimal option in that abstract state, $o^*_{s_\phi}$.

APPROXIMATE MDP HOMOMORPHISMS ($\mathcal{O}_{\phi,H}$). As discussed in Chapter 2, MDP homomorphisms define mappings from one MDP to another in a way that preserves desirable properties [270]. The main idea behind these mappings, as with state and action abstraction, is to identify symmetries in the underlying environmental MDP that can be expressed through a simpler model than the original MDP. An approximate MDP homomorphism extends this notion of equivalence to similarity, thereby allowing greater opportunity to compress [273]. To define this class of ϕ-relative options, we first define the one-step abstract transition and reward functions for a ϕ-relative option o. That is, for $w : \mathcal{S} \to [0,1]$ any valid weighting function such that $\sum_{s \in s_\phi} w(s) = 1$:

$$T_\phi(s'_\phi \mid s_\phi, o) = \sum_{s \in s_\phi} w(s) \sum_{s' \in s'_\phi} T(s' \mid s, \pi_o(s)), \tag{9.8}$$

$$R_\phi(s_\phi, o) = \sum_{s \in s_\phi} w(s) R(s, \pi_o(s)). \tag{9.9}$$

Next, we introduce the quantities K_p and K_r of Ravindran and Barto [273]:

$$K_p = \max_{s \in \mathcal{S}, a \in \mathcal{A}} \sum_{s_\phi \in \mathcal{S}_\phi} \left| \sum_{s' \in s_\phi} T(s' \mid s, a) - T_\phi(s_\phi \mid \phi(s), \pi_{\mathcal{O}_\phi}(\phi(s))) \right|, \tag{9.10}$$

$$K_r = \max_{s \in \mathcal{S}, a \in \mathcal{A}} |R(s,a) - R_\phi(\phi(s), \pi_{\mathcal{O}_\phi}(\phi(s)))|. \tag{9.11}$$

These capture the maximum discrepancy between the model of the ground MDP and the model of the induced abstract MDP defined according to $(\phi, \mathcal{O}_\phi)$. Using these quantities, the class of ϕ-relative options is defined as follows.

$$\lambda(\mathcal{O}_\phi) \equiv \forall\ \pi_{\mathcal{O}_\phi} \in \Pi_{\mathcal{O}_\phi} : K_p \leq \varepsilon_p \text{ and } K_r \leq \varepsilon_r. \tag{9.12}$$

These four classes will constitute four sufficient conditions for $(\phi, \mathcal{O}_\phi)$ pairs to yield bounded value loss.

9.1.2 *Main Result*

The main result of this chapter establishes the bounded value loss of pairs $(\phi, \mathcal{O}_\phi)$ where $\mathcal{O}_\phi$ belongs to any of these four classes, and the size of the bound depends on the degree of approximation (ε_Q; ε_R, ε_T; τ; and ε_r, ε_p).

Theorem 9.1. *(Main Result) For any ϕ, the four introduced classes of ϕ-relative options satisfy:*

$$L(\phi, \mathcal{O}_{\phi,Q_\varepsilon^*}) \leq \frac{\varepsilon_Q}{1-\gamma}, \tag{9.13}$$

$$L(\phi, \mathcal{O}_{\phi,M_\varepsilon}) \leq \frac{\varepsilon_R + |\mathcal{S}|\varepsilon_T \text{RM}\textsc{ax}}{(1-\gamma)^2}, \tag{9.14}$$

$$L(\phi, \mathcal{O}_{\phi,\tau}) \leq \frac{\tau\gamma|\mathcal{S}|}{(1-\gamma)^2}, \tag{9.15}$$

$$L(\phi, \mathcal{O}_{\phi,H}) \leq \frac{2}{1-\gamma}\left(\varepsilon_r + \frac{\gamma \text{RM}\textsc{ax}}{1-\gamma}\frac{\varepsilon_p}{2}\right), \tag{9.16}$$

where the $L(\phi, \mathcal{O}_{\phi,\tau})$ bound holds in goal-based MDPs and the other three hold in any finite MDP.

Proof of Theorem 9.1.

> We prove this claim using four separate proofs, each targeting one class.
>
> *Proof.* $(L(\phi, \mathcal{O}_{\phi,Q_\varepsilon^*}) \leq \frac{\varepsilon_Q}{1-\gamma})$

Consider $L(\phi, \mathcal{O}_{\phi,Q_\varepsilon^*}) = \min_{\pi^{\Downarrow}_{\mathcal{O}_\phi} \in \Pi^{\Downarrow}_{\mathcal{O}_\phi}} \max_{s \in \mathcal{S}} |V^*(s) - V^{\pi^{\Downarrow}_{\mathcal{O}_\phi}}(s)|$. Since $V^*(s) \geq V^\pi(s)$ for all π, we henceforth drop the absolute value for convenience.

To proceed, we recall that $o^*_{s_\phi}$ is the ϕ-relative option that executes π^* in every state and terminates when it leaves the abstract state s_ϕ:

$$\begin{aligned} o^*_{s_\phi} := \forall_{s \in \mathcal{S}} : (\mathcal{I}(s) &\equiv \phi(s) = s_\phi, \\ \beta(s) &\equiv \phi(s) \neq s_\phi, \\ \pi(s) &= \pi^*(s)). \end{aligned} \tag{9.17}$$

Note that since $o^*_{s_\phi}$ always chooses actions according to π^*, that $Q^*_{s_\phi}(s, o^*_{s_\phi}) = V^*(s)$ (where $Q^*_{s_\phi}$ is defined according to Equation 9.5).

Then, by the Q_ε^* predicate, we can construct a policy over abstract states and options $\mu_{\mathcal{O}_\phi} \in \Pi_{\mathcal{O}_\phi}$ with the following property:

$$\forall_{s_\phi \in \mathcal{S}_\phi, s \in s_\phi} : Q^*_{s_\phi}(s, o^*_{s_\phi}) - Q^*_{s_\phi}(s, \mu_{\mathcal{O}_\phi}(s_\phi)) \leq \varepsilon_Q. \tag{9.18}$$

Note that $\mu_{\mathcal{O}_\phi}(s_\phi)$ outputs an option. As in Equation 9.18, we henceforth denote $s_\phi = \phi(s)$ and correspondingly $s'_\phi = \phi(s')$.

Then it must be the case that

$$L(\phi, \mathcal{O}_{\phi,Q_\varepsilon^*}) \leq \max_{s \in \mathcal{S}} V^*(s) - V^{\mu^{\Downarrow}_{\mathcal{O}_\phi}}(s). \tag{9.19}$$

Let $Q_t^*(s,o)$ denote the expected discounted reward of executing option o, then executing t options under $\mu_{\mathcal{O}_\phi}$, then following the optimal policy thereafter. Note that

$$\lim_{t\to\infty} Q_t^*(s, \mu_{\mathcal{O}_\phi}(s_\phi)) = V^{\mu_{\mathcal{O}_\phi}^{\Downarrow}}(s), \tag{9.20}$$

because $Q_t^*(s, \mu_{\mathcal{O}_\phi}(s_\phi))$ is the expected discounted reward of executing $t+1$ options under $\mu_{\mathcal{O}_\phi}$, then following the optimal policy thereafter.

We next show by induction on t that

$$\max_{s\in\mathcal{S}} V^*(s) - V^{\mu_{\mathcal{O}_\phi}^{\Downarrow}}(s) = \max_{s\in\mathcal{S}} \lim_{t\to\infty} V^*(s) - Q_t^*(s, \mu_{\mathcal{O}_\phi}(s_\phi)) \le \frac{\varepsilon_Q}{1-\gamma}. \tag{9.21}$$

In particular, we wish to show that

$$\forall_{t\in\mathbb{N}} : \max_{s\in\mathcal{S}} V^*(s) - Q_t^*(s, \mu_{\mathcal{O}_\phi}(s_\phi)) \le \sum_{i=0}^{t} \varepsilon_Q \gamma^i. \tag{9.22}$$

(Base Case)

When $t = 0$, for all $s \in \mathcal{S}$,

$$Q_0^*(s, \mu_{\mathcal{O}_\phi}(s_\phi)) = Q_{s_\phi}^*(s, \mu_{\mathcal{O}_\phi}(s_\phi)), \tag{9.23}$$

because both quantities represent the expected discounted reward of executing the option $\mu_{\mathcal{O}_\phi}(s_\phi)$ then following the optimal policy thereafter. It follows that

$$\begin{aligned}
\max_{s\in\mathcal{S}} V^*(s) - Q_0^*(s, \mu_{\mathcal{O}_\phi}(s_\phi)) &= \max_{s\in\mathcal{S}} V^*(s) - Q_{s_\phi}^*(s, \mu_{\mathcal{O}_\phi}(s_\phi)), && (9.24)\\
&= \max_{s\in\mathcal{S}} Q_{s_\phi}^*(s, o_{s_\phi}^*) - Q_{s_\phi}^*(s, \mu_{\mathcal{O}_\phi}(s_\phi)), && (9.25)\\
&\le \varepsilon_Q, && (9.26)\\
&= \sum_{i=0}^{0} \varepsilon_Q \gamma^0, && (9.27)
\end{aligned}$$

where the inequality holds by definition of $\mu_{\mathcal{O}_\phi}$.

(Inductive Case)

We assume as the inductive hypothesis that

$$\max_{s \in \mathcal{S}} V^*(s) - Q_k^*(s, \mu_{\mathcal{O}_\phi}(s_\phi)) \leq \sum_{i=0}^{k} \varepsilon_Q \gamma^i, \tag{9.28}$$

and want to show that

$$\max_{s \in \mathcal{S}} V^*(s) - Q_{k+1}^*(s, \mu_{\mathcal{O}_\phi}(s_\phi)) \leq \sum_{i=0}^{k+1} \varepsilon_Q \gamma^i. \tag{9.29}$$

To begin, fix $s \in \mathcal{S}$ and consider

$$\begin{aligned} & V^*(s) - Q_{k+1}^*(s, \mu_{\mathcal{O}_\phi}(s_\phi)) && (9.30) \\ &= V^*(s) - \left(R_o(s, \mu_{\mathcal{O}_\phi}(s_\phi)) + \sum_{s' \in \mathcal{S}} T_o(s' \mid s, \mu_{\mathcal{O}_\phi}(s_\phi)) Q_k^*(s', \mu_{\mathcal{O}_\phi}(s'_\phi)) \right) && (9.31) \\ &= V^*(s) - R_o(s, \mu_{\mathcal{O}_\phi}(s_\phi)) - \sum_{s' \in \mathcal{S}} T_o(s' \mid s, \mu_{\mathcal{O}_\phi}(s_\phi)) Q_k^*(s', \mu_{\mathcal{O}_\phi}(s'_\phi)) && (9.32) \end{aligned}$$

where R_o and T_o indicate the reward and multi-time models.

Now, subtract and add $\sum_{s' \in \mathcal{S}} T_o(s'|s, \mu_{\mathcal{O}_\phi}(s_\phi)) V^*(s')$:

$$\begin{aligned} = \quad & V^*(s) - R_o(s, \mu_{\mathcal{O}_\phi}(s_\phi)) - \sum_{s' \in \mathcal{S}} T_o(s' \mid s, \mu_{\mathcal{O}_\phi}(s_\phi)) V^*(s') && (9.33) \\ & + \sum_{s' \in \mathcal{S}} T_o(s' \mid s, \mu_{\mathcal{O}_\phi}(s_\phi)) V^*(s') - \sum_{s' \in \mathcal{S}} T_o(s' \mid s, \mu_{\mathcal{O}_\phi}(s_\phi)) Q_k^*(s', \mu_{\mathcal{O}_\phi}(s'_\phi)) \\ = \quad & V^*(s) - Q_{s_\phi}^*(s, \mu_{\mathcal{O}_\phi}(s_\phi)) && (9.34) \\ & + \sum_{s' \in \mathcal{S}} T_o(s' \mid s, \mu_{\mathcal{O}_\phi}(s_\phi)) \left[V^*(s') - Q_k^*(s', \mu_{\mathcal{O}_\phi}(s'_\phi)) \right] && (9.35) \\ = \quad & Q_{s_\phi}^*(s, o_{s_\phi}^*) - Q_{s_\phi}^*(s, \mu_{\mathcal{O}_\phi}(s_\phi)) && (9.36) \\ & + \sum_{s' \in \mathcal{S}} T_o(s' \mid s, \mu_{\mathcal{O}_\phi}(s_\phi)) \left[V^*(s') - Q_k^*(s', \mu_{\mathcal{O}_\phi}(s'_\phi)) \right] && (9.37) \\ \leq \quad & \varepsilon_Q + \sum_{s' \in \mathcal{S}} T_o(s' \mid s, \mu_{\mathcal{O}_\phi}(s_\phi)) \left[V^*(s') - Q_k^*(s', \mu_{\mathcal{O}_\phi}(s'_\phi)) \right], && (9.38) \end{aligned}$$

by definition of $\mu_{\mathcal{O}_\phi}$. Continuing, we have that:

$$= \quad \varepsilon_Q + \sum_{s' \in \mathcal{S}} \sum_{n=1}^{\infty} p(s', n \mid s, \mu_{\mathcal{O}_\phi}(s_\phi)) \gamma^n \left[V^*(s') - Q^*_k(s', \mu_{\mathcal{O}_\phi}(s'_\phi)) \right] \tag{9.39}$$

$$\leq \quad \varepsilon_Q + \sum_{s' \in \mathcal{S}} \sum_{n=1}^{\infty} p(s', n \mid s, \mu_{\mathcal{O}_\phi}(s_\phi)) \gamma^n \sum_{i=0}^{k} \varepsilon_Q \gamma^i, \tag{9.40}$$

by the inductive hypothesis. Then:

$$= \quad \varepsilon_Q + \gamma \sum_{s' \in \mathcal{S}} \sum_{n=0}^{\infty} p(s', n+1 \mid s, \mu_{\mathcal{O}_\phi}(s_\phi)) \gamma^n \sum_{i=0}^{k} \varepsilon_Q \gamma^i \tag{9.41}$$

$$= \quad \varepsilon_Q + \gamma \sum_{i=0}^{k} \varepsilon_Q \gamma^i \sum_{s' \in \mathcal{S}} \sum_{n=0}^{\infty} p(s', n+1 \mid s, \mu_{\mathcal{O}_\phi}(s_\phi)) \gamma^n \tag{9.42}$$

$$\leq \quad \varepsilon_Q + \gamma \sum_{i=0}^{k} \varepsilon_Q \gamma^i \cdot 1 \tag{9.43}$$

$$= \quad \sum_{i=0}^{k+1} \varepsilon_Q \gamma^i, \tag{9.44}$$

since $p(s', n+1 \mid s, \mu_{\mathcal{O}_\phi}(s_\phi))$ is a probability distribution and γ is less than 1.

All together, we've shown that $V^*(s) - Q^*_{k+1}(s, \mu_{\mathcal{O}_\phi}(s_\phi)) \leq \sum_{i=0}^{k+1} \varepsilon_Q \gamma^i$ for all $s \in \mathcal{S}$, which implies that

$$\max_{s \in \mathcal{S}} V^*(s) - Q^*_{k+1}(s, \mu_{\mathcal{O}_\phi}(s_\phi)) \leq \sum_{i=0}^{k+1} \varepsilon_Q \gamma^i, \tag{9.45}$$

as desired.

It follows by induction that

$$\forall_{t \in \mathbb{N}} : \max_{s \in \mathcal{S}} V^*(s) - Q^*_t(s, \mu_{\mathcal{O}_\phi}(s_\phi)) \leq \sum_{i=0}^{t} \varepsilon_Q \gamma^i. \tag{9.46}$$

Therefore,

$$
\begin{aligned}
L(\phi, \mathcal{O}_{\phi,Q^*_\varepsilon}) &\leq \max_{s \in \mathcal{S}} V^*(s) - V^{\mu^{\Downarrow}_{\mathcal{O}_\phi}}(s) && (9.47)\\
&= \max_{s \in \mathcal{S}} \lim_{t \to \infty} V^*(s) - Q^*_t(s, \mu_{\mathcal{O}_\phi}(s_\phi)) && (9.48)\\
&\leq \lim_{t \to \infty} \sum_{i=0}^{t} \varepsilon_Q \gamma^i && (9.49)\\
&= \frac{\varepsilon_Q}{1-\gamma}, && (9.50)
\end{aligned}
$$

which completes the proof. □

Proof. $(L(\phi, \mathcal{O}_{\phi,M_\varepsilon}) \leq \frac{\varepsilon_R + |\mathcal{S}|\varepsilon_T \text{VMax}}{1-\gamma})$

We show that this class is a subclass of the $\mathcal{O}_{\phi,Q^*_\varepsilon}$ class. Therefore, it stands to show that, given our class definition, there exists an option in every abstract state that is near-optimal in Q-value.

Fix $s \in \mathcal{S}$. Let $s_\phi = \phi(s)$. By the M_ε predicate, there exists an option $o \in \mathcal{O}_\phi$ such that

$$
||T^{s'}_{s,o^*_{s_\phi}} - T^{s'}_{s,o}||_\infty \leq \varepsilon_T \text{ and } ||R_{s,o^*_{s_\phi}} - R_{s,o}||_\infty \leq \varepsilon_R. \tag{9.51}
$$

Now, we consider the difference in optimal Q-values between $o^*_{s_\phi}$ and o. We first have that:

$$
\begin{aligned}
Q^*_{s_\phi}(s,o) &= R(s, \pi_o(s)) + \gamma \sum_{s' \in \mathcal{S}} T(s' \mid s, \pi_o(s)) \left(\mathbb{1}(s' \in s_\phi) Q^*_{s_\phi}(s', o) + \mathbb{1}(s' \notin s_\phi) V^*(s') \right)\\
&= R_o(s,o) + \sum_{s' \in \mathcal{S}} T_o(s' \mid s, o) V^*(s'),
\end{aligned}
\tag{9.52}
$$

with R_o and T_o denoting the standard multi-time model of Sutton et al. [311].

By symmetry,

$$Q^*_{s_\phi}(s, o^*_{s_\phi}) = R_o(s, o^*_{s_\phi}) + \sum_{s' \in \mathcal{S}} T_o(s' \mid s, o^*_{s_\phi}) V^*(s'). \tag{9.53}$$

Therefore,

$$\begin{aligned}
& |Q^*_{s_\phi}(s, o^*_{s_\phi}) - Q^*_{s_\phi}(s, o)| && (9.54) \\
\leq \quad & |R_o(s, o^*_{s_\phi}) - R_o(s, o)| + |\sum_{s' \in \mathcal{S}} \Big(T_o(s' \mid s, o^*_{s_\phi}) - T_o(s' \mid s, o) \Big) V^*(s')| && (9.55) \\
\leq \quad & |R_o(s, o^*_{s_\phi}) - R_o(s, o)| + \sum_{s' \in \mathcal{S}} |T_o(s' \mid s, o^*_{s_\phi}) - T_o(s' \mid s, o)||V^*(s')| && (9.56) \\
\leq \quad & \varepsilon_R + |\mathcal{S}| \varepsilon_T \text{VM}\textsc{ax}, && (9.57)
\end{aligned}$$

by the model similarity assumption. We have now shown that any option with near-optimal models has a near-optimal Q-value with $\varepsilon_Q = \varepsilon_R + |\mathcal{S}|\varepsilon_T \text{VM}\textsc{ax}$. Therefore, by the previous result,

$$L(\phi, O_{\phi, M_\varepsilon}) \leq \frac{\varepsilon_R + |\mathcal{S}|\varepsilon_T \text{VM}\textsc{ax}}{1 - \gamma}. \tag{9.58}$$

□

....

Proof. $(L(\phi, \mathcal{O}_{\phi,\tau}) \leq \frac{\tau\gamma|\mathcal{S}|}{(1-\gamma)^2})$

We first state rigorously our definition of a goal-based MDP.

Definition 9.3. *A goal-based MDP is an MDP with some number of goal states, denoted $\mathcal{S}_G \subseteq \mathcal{S}$. The reward function is such that $R(s,a) = 1$ if $s \in \mathcal{S}_G$, $R(s,a) = 0$ otherwise, and* *the episode terminates after receiving a reward in a goal state. Furthermore, we assume that each goal state exists in its own abstract state: $s \neq s_g \Rightarrow \phi(s_g) \neq \phi(s)$, where $s_g \in \mathcal{S}_G, s \in \mathcal{S}$.*

We show that this class is a subclass of the $\mathcal{O}_{\phi,Q^*_\varepsilon}$ class in goal-based MDPs. In particular, it stands to show that given our class definition, there exists an option in every abstract state that is near-optimal in Q-value.

First, note that in the abstract states containing a goal state, any option is optimal since $R(s,a) = 1$ regardless of action. Therefore, we restrict our attention to an arbitrary $s \in \mathcal{S} \setminus \mathcal{S}_G$, fixing $s_\phi = \phi(s)$. Let o be an option available in s_ϕ such that $\max_{s \in s_\phi, s' \in \mathcal{S}} |p(s', k \mid s, o^*_{s_\phi}) - p(s', k \mid s, o)| \leq \tau$, by the option class definition. Then

$$
\begin{aligned}
& Q^*_{s_\phi}(s, o^*_{s_\phi}) - Q^*_{s_\phi}(s,o) && (9.59)\\
&= R_o(s, o^*_{s_\phi}) + \sum_{s' \in \mathcal{S}} T_o(s'|s, o^*_{s_\phi}) V^*(s') - R_o(s,o) - \sum_{s' \in \mathcal{S}} T_o(s'|s,o) V^*(s') && (9.60)\\
&= \sum_{s' \in \mathcal{S}} \left[T_o(s'|s, o^*_{s_\phi}) - T_o(s'|s,o) \right] V^*(s'), && (9.61)
\end{aligned}
$$

where we drop the R_o terms since $s \notin \mathcal{S}_G$, each goal state has its own abstract state, and $R(s,a) = 0$ for $s \notin \mathcal{S}_G$. Continuing, we have that

$$
Q^*_{s_\phi}(s, o^*_{s_\phi}) - Q^*_{s_\phi}(s,o) = \sum_{s' \in \mathcal{S}} \left[\sum_{k=1}^{\infty} |p(s', k \mid s, o^*_{s_\phi}) - p(s', k \mid s, o)| \gamma^k \right] V^*(s), \qquad (9.62)
$$

writing out the multi-time model. This implies that

$$
Q^*_{s_\phi}(s, o^*_{s_\phi}) - Q^*_{s_\phi}(s,o) \leq \sum_{s' \in \mathcal{S}} \frac{\tau \gamma}{1-\gamma} V^*(s). \qquad (9.63)
$$

Now, note that $V^*(s') = \sum_{s_g \in \mathcal{S}_G} \sum_{t=0}^{\infty} p(s_g, t \mid s', \pi^*) \gamma^t$ in a goal-based MDP, where $p(s_g, t \mid s', \pi^*)$ is the probability of being in state s_g after t timesteps, starting from

s' and following π^*. Indeed, this gives that $V^*(s') \le 1$ since $p(s_g, t \mid s', \pi^*)$ is a probability distribution and γ is less than one. Therefore,

$$Q^*_{s_\phi}(s, o^*_{s_\phi}) - Q^*_{s_\phi}(s, o) \le \frac{\tau\gamma|\mathcal{S}|}{1-\gamma}. \tag{9.64}$$

We have shown that there exists an option, o, in any abstract state that is near-optimal in Q-value, with $\varepsilon_Q = \frac{\tau\gamma|\mathcal{S}|}{1-\gamma}$. Therefore, by the $\mathcal{O}_{\phi,Q^*_\varepsilon}$ bound,

$$L(\phi, \mathcal{O}_{\phi,\tau}) \le \frac{\tau\gamma|\mathcal{S}|}{(1-\gamma)^2}, \tag{9.65}$$

as desired. □

Proof. $\left(L(\phi, \mathcal{O}_{\phi,H}) \le \frac{2}{1-\gamma}\left(\varepsilon_r + \frac{\gamma \text{RMax}}{1-\gamma}\frac{\varepsilon_p}{2}\right)\right)$

We prove this result by illustrating the connection between our formalisms and the work of Ravindran and Barto [273]. To do so, we first restate their definition of an approximate homomorphism.

Definition 9.4. *An **approximate MDP homomorphism** [273] $h : M \mapsto M'$ from an MDP $M = (\mathcal{S}, \mathcal{A}, \Psi, T, R, \gamma)$ to an MDP $M' = (\mathcal{S}', \mathcal{A}', \Psi', T', R', \gamma)$ is a surjection from Ψ to Ψ', defined by a tuple of surjections $\langle f, \{g_s : s \in S\}\rangle$, with $h((s,a)) = (f(s), g_s(a))$, where $f : S \to S'$ and $g_s : \mathcal{A}_s \to \mathcal{A}'_{f(s)}$ for $s \in S$, such that for all s, s' in $\mathcal{S}$ and $a \in \mathcal{A}_s$:*

$$T'\big(f\big(s'\big) \mid f(s), g_s(a)\big) = \sum_{(q,b)\in[(s,a)]_h} w_{qb} \sum_{s''\in[s']_f} T\big(s'' \mid q, b\big) \tag{9.66}$$

$$R'\left(f(s), g_s(a)\right) = \sum_{(q,b)\in[(s,a)]_h} w_{qb}\, R(q,b), \tag{9.67}$$

where $[(s,a)]_h$ denotes the preimage of $h((s,a))$, $[s']_f$ denotes the preimage of $f(s')$, and $\sum_{(q,b)\in[(s,a)]_h} w_{qb} = 1$. Furthermore, Ψ and Ψ' denote the sets of admissible state-action pairs in the ground and abstract MDP respectively. Based on Ψ and Ψ', A_s denotes the set of actions available in state s of the ground MDP, and $A'_{f(s)}$ denotes the set of abstract actions available in state $f(s)$ of the abstract MDP.

We now illustrate how our definitions of ϕ, R_ϕ, T_ϕ with respect to a given $\pi_{\mathcal{O}_\phi}$ induce an approximate homomorphism. First, note that our ϕ precisely corresponds to their definition of f, a state abstraction. Then, fix $s_\phi \in \mathcal{S}_\phi$, and let $A'_{s_\phi} = \{\pi_{\mathcal{O}_\phi}(s_\phi)\}$ with $g_s(a) = \pi_{\mathcal{O}_\phi}(s_\phi)\forall_{s\in s_\phi}\ \forall_{a\in A}$.

We now consider our definitions of T_ϕ and R_ϕ:

$$T_\phi(s'_\phi \mid s_\phi, o) = \sum_{s\in s_\phi} w(s) \sum_{s'\in s'_\phi} T(s' \mid s, \pi_o(s)) \qquad R_\phi(s_\phi, o) = \sum_{s\in s_\phi} w(s) R(s, \pi_o(s)), \tag{9.68}$$

We note that these are precisely an instance of P' and R' as defined above, with $w_{qb} = 0$ whenever $b \neq \pi_o(q)$. We write $w(s)$ to denote this choice of weighting function, which depends only on the action prescribed by π_o. We select this choice of weighting function (as opposed to a weighting dependent on all available actions) in order to faithfully represent the 1-step behavior of executing an option in the abstract MDP.

By these connections, a deterministic policy $\pi_{\mathcal{O}_\phi}$ over ϕ-relative options coupled with our choice of weighting function defines an approximate homomorphism. We further adapt their definitions of K_p and K_r to our notational setting, which describe the maximum discrepancy in models between the ground and abstract MDPs.

$$K_p = \max_{s\in\mathcal{S}, a\in\mathcal{A}} \sum_{s_\phi\in\mathcal{S}_\phi} \left| \sum_{s'\in s_\phi} T(s' \mid s, a) - T_\phi(s_\phi \mid \phi(s), \pi_{\mathcal{O}_\phi}(\phi(s)))\right|, \tag{9.69}$$

$$K_r = \max_{s \in \mathcal{S}, a \in \mathcal{A}} |R(s,a) - R_\phi(\phi(s), \pi_{\mathcal{O}_\phi}(\phi(s)))|. \tag{9.70}$$

The main theorem of Ravindran and Barto [273] guarantees that the value loss of the optimal policy in the abstract MDP $\mathcal{M}'$ is upper-bounded by

$$\frac{2}{1-\gamma}\left(K_r + \frac{\gamma}{1-\gamma}\delta_{r'}\frac{K_p}{2}\right),$$

where $\delta_{r'}$ is upper-bounded by RMax. Let $\mu_{\mathcal{O}_\phi} \in \Pi_{\mathcal{O}_\phi}$ denote the optimal policy in the abstract MDP. By our option class definition, all abstract policies $\pi_{\mathcal{O}_\phi} \in \Pi_{\mathcal{O}_\phi}$ induce homomorphisms with bounded K_p, K_r. Thus, $\mu_{\mathcal{O}_\phi}$ has bounded K_p, K_r. Then:

$$\begin{aligned} L(\phi, \mathcal{O}_{\phi,H}) &= \min_{\pi_{\mathcal{O}_\phi} \in \Pi_{\mathcal{O}_\phi}} \left|\left| V^* - V^{\pi^{\Downarrow}_{\mathcal{O}_\phi}} \right|\right|_\infty, && (9.71) \\ &\leq \left|\left| V^* - V^{\mu^{\Downarrow}_{\mathcal{O}_\phi}} \right|\right|_\infty, && (9.72) \\ &\leq \frac{2}{1-\gamma}\left(K_r + \frac{\gamma}{1-\gamma}\delta_{r'}\frac{K_p}{2}\right) && (9.73) \\ &\leq \frac{2}{1-\gamma}\left(\varepsilon_r + \frac{\gamma \text{RMax}}{1-\gamma}\frac{\varepsilon_p}{2}\right), && (9.74) \end{aligned}$$

as desired. □

Having proven the value loss for each $\mathcal{O}_\phi$ class, the result follows. □

Observe that when the approximation parameters are zero, many of the bounds collapse to 0 as well. This illustrates the trade off made between the amount of knowledge used to construct the abstractions and the degree of optimality ensured, as was the case with approximate state abstractions in Chapter 3. In a sense, this result is the spiritual successor to Theorem 3.1, extended to options. Further note that the value loss of the state abstraction does not appear in any of the above bounds—indeed, ϕ will *implicitly* affect the value loss as a function of the diameter of each abstract state. Finally, observe that, as with Theorem 3.1, each of the above classes expresses a *sufficient* condition needed for a pair $(\phi, \mathcal{O}_\phi)$ to preserve value.

It is useful, however, to identify not just sufficient conditions, but also necessary. To this end, we next establish one necessary condition of all (globally) value preserving $(\phi, \mathcal{O}_\phi)$ classes.

Theorem 9.2. *For any $(\phi, \mathcal{O}_\phi)$ pair with $L(\phi, \mathcal{O}_\phi) \leq \eta$, there exists at least one option per abstract state that is η-optimal in Q-value. Precisely, if $L(\phi, \mathcal{O}_\phi) \leq \eta$, then:*

$$\forall_{s_\phi \in \mathcal{S}_\phi} \forall_{s \in s_\phi} \exists_{o \in \mathcal{O}_\phi} : Q^*_{s_\phi}(s, o^*_{s_\phi}) - Q^*_{s_\phi}(s, o) \leq \eta. \tag{9.75}$$

Proof of Theorem 9.2.

Let $\mu_{\mathcal{O}_\phi} = \arg\min_{\pi_{\mathcal{O}_\phi} \in \Pi_{\mathcal{O}_\phi}} \left|\left| V^* - V^{\pi^{\Downarrow}_{\mathcal{O}_\phi}} \right|\right|_\infty$.

Suppose, for a contradiction, that there exists an abstract state s_ϕ for which there is no η-optimal option in $\mathcal{O}_\phi$. Then it must be the case that

$$Q^*_{s_\phi}(s, o^*) - Q^*_{s_\phi}(s, \mu_{\mathcal{O}_\phi}(s)) > \eta \tag{9.76}$$

for some $s \in s_\phi$.

By, $Q^*_{s_\phi}(s, o^*) = V^*(s)$, this implies that

$$V^*(s) - Q^*_{s_\phi}(s, \mu_{\mathcal{O}_\phi}(s)) > \eta. \tag{9.77}$$

Then, note that $Q^*_{s_\phi}(s, \mu_{\mathcal{O}_\phi}(s)) \geq V^{\mu^{\Downarrow}_{\mathcal{O}_\phi}}(s)$ because $Q^*_{s_\phi}$ describes the expected return of executing option $\mu_{\mathcal{O}_\phi}(s)$, then switching to optimal behavior, whereas $V^{\mu^{\Downarrow}_{\mathcal{O}_\phi}}$

describes the expected return of executing $\mu_{\mathcal{O}_\phi}(s)$ then continuing to execute options according to $\mu_{\mathcal{O}_\phi}$.

Noticing that $V^*(s) \geq Q^*_{s_\phi}(s, \mu_{\mathcal{O}_\phi}(s)) \geq V^{\mu^{\Downarrow}_{\mathcal{O}_\phi}}(s)$, we have that

$$V^*(s) - V^{\mu^{\Downarrow}_{\mathcal{O}_\phi}}(s) > \eta. \tag{9.78}$$

This implies that

$$\begin{aligned} L(\phi, \mathcal{O}_\phi) &= \min_{\pi_{\mathcal{O}_\phi} \in \Pi_{\mathcal{O}_\phi}} \left\| V^* - V^{\pi^{\Downarrow}_{\mathcal{O}_\phi}} \right\|_\infty & (9.79) \\ &= V^*(s) - V^{\mu^{\Downarrow}_{\mathcal{O}_\phi}}(s) & (9.80) \\ &> \eta, & (9.81) \end{aligned}$$

which contradicts the premise. Therefore, it must be true that

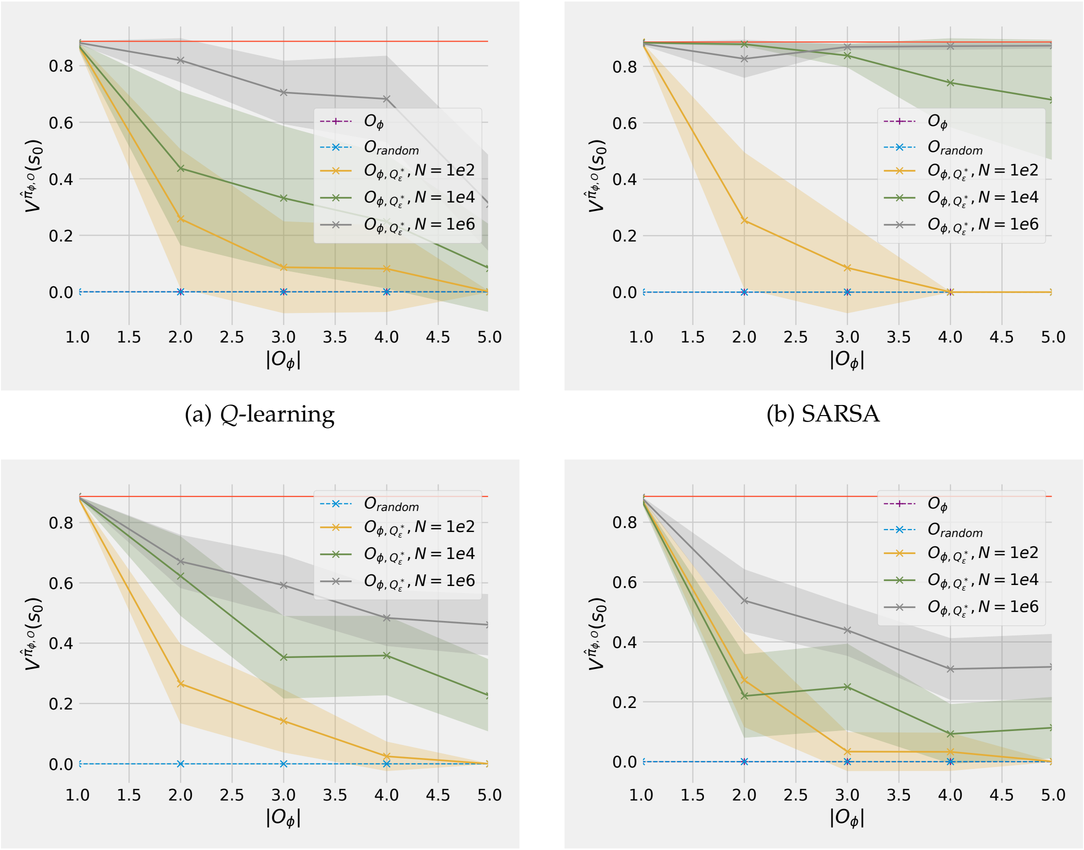
$$\forall_{s_\phi \in \mathcal{S}_\phi} \forall_{s \in s_\phi} \exists_{o \in \mathcal{O}_\phi} : Q^*_{s_\phi}(s, o^*) - Q^*_{s_\phi}(s, o) \leq \eta. \tag{9.82}$$

□

This theorem tells us that for any agent acting using these join state-action abstractions, if there exists an abstract state for which there is not an η-optimal option, then the agent cannot represent a globally near-optimal policy.

9.1.3 *Experiment*

I next conduct a simple experiment to test whether value preserving options enable simple RL algorithms to find near-optimal policies. The experiment illustrates an important property of one of the introduced $\phi, \mathcal{O}_\phi$ classes, and is organized as follows. First, I construct a $\phi, \mathcal{O}_\phi$ pair belonging to the $\mathcal{O}_{\phi, Q^*_\varepsilon}$ class using dynamic programming. I give this pair to one of four different RL algorithms: Q-learning [337], SARSA [276], Double Q-learning [325], and R-Max [54]. For each algorithm, I vary the number of interactions it is allowed to have with the environment, N, ranging from $N = 100$ to $N = 1,000,000$. As



(a) Q-learning

(b) SARSA

(c) Double Q

(d) R-Max

Figure 9.3: Empirical evidence that the $\phi, \mathcal{O}_\phi$ pairs from Theorem 9.1 preserve value.

expected, the environment is the Four Rooms MDP with a single goal location in the top right and start location in the bottom left. The state abstraction ϕ maps each state into one of four abstract states, denoting each of the four rooms. I vary both the number of options added per abstract state ($|\mathcal{O}_\phi|$) and the sample budget given to each algorithm (N), and present the value of the policy discovered by the final episode for each setting of $|\mathcal{O}_\phi|$ and N. The code is publicly available for extension and reproduction. [9.1]

Results are presented in Figure 9.3. First, note that with only one option per abstract state, all four algorithms can trivially find a near-optimal policy, even with a small sample budget. This is predicted by Theorem 9.1: the included options preserve value, and so

[9.1] https://github.com/david-abel/vpsa_aistats2020 any assignment of options to abstract states will yield a near-optimal policy. In contrast, if randomly chosen options are used instead (shown in blue, labeled as O_{random}), the learning algorithm fails to find a good policy even with a high sample budget ($N = 1e6$ was used). Second, we find that as the number of options increases, the added branching factor causes each algorithm to find a lower value policy with the same number of samples. However, by Theorem 9.1 we know each set of options preserves value; as the sample budget increases we see that the value of the discovered policy tends toward optimal in each algorithm. For SARSA, for instance, there is a dramatic difference between the lowest setting of N and the higher two settings. In short, the $\phi, \mathcal{O}_\phi$ pairs defined by Theorem 9.1 do in fact preserve value, but will also affect the sample budget required to find a good policy, with the exact extent changing depending on the RL algorithm. I foresee the combination of value preserving abstractions with those that lower learning complexity (see recent work by Brunskill and Li [58], Fruit et al. [112]) as a key direction for future work.

I further visualize the learned value function of Q-learning with and without ϕ-relative options after the same sample budget, depicted in Figure 9.4. Notably, since ϕ-relative options update entire blocks of states, we see large regions of the state space with the same learned value function. Conversely, Q-learning only tends to explore (and estimate

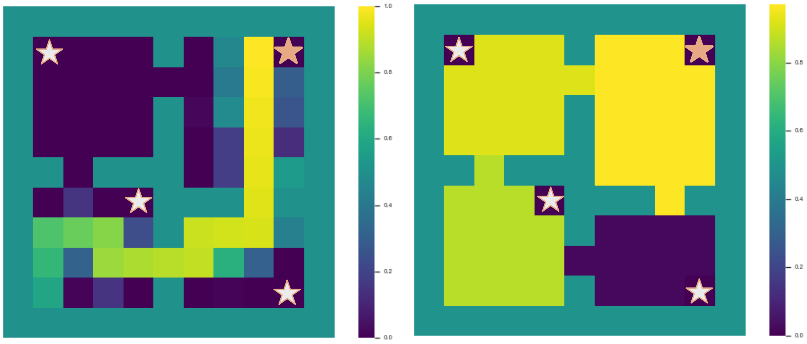


Figure 9.4: Comparison of the learned value function with regular Q-learning (left) and Q-learning with $\phi, \mathcal{O}_\phi$.

the values of) a narrow region of the state space. The visual highlights this important qualitative difference between learning with and without action abstractions.

9.2 HIERARCHICAL ABSTRACTION

I now present an extension of Theorem 9.1 that applies to hierarchies consisting of $(\phi, \mathcal{O}_\phi)$ pairs. I prove that the value loss compounds linearly if we are to construct a hierarchy using algorithms that generate a well-behaved ϕ and $\mathcal{O}_\phi$. To do so, we require two definitions and additional notation. We first define a *hierarchy* as n sets of $(\phi, \mathcal{O}_\phi)$ pairs, as pictured in Figure 9.5.

Definition 9.5. *A* ***depth n hierarchy****, denoted H_n, is a list of n state abstractions, $\phi^{(n)}$, and a list of n sets of ϕ-relative options, $\mathcal{O}_\phi^{(n)}$. The components $(\mathcal{I}, \beta, \pi)$ of each of the i-th set of options, $\mathcal{O}_{\phi,i}$ are defined over the $(i-1)$-th abstract state space,*

$$\mathcal{S}_{\phi,i-1} = \{\phi_{i-1}(\phi_{i-2}(\ldots\phi_1(s)\ldots)) \mid s \in \mathcal{S}\}. \tag{9.83}$$

I next introduce additional notation to refer to values, states, options, and policies at each level of the hierarchy. Let $\pi_n : \mathcal{S}_{\phi,n} \to \mathcal{O}_{\phi,n}$ denote the level n policy encoded by the hierarchy, with Π_n the space of all policies encoded in this way. I let s_i denote shorthand

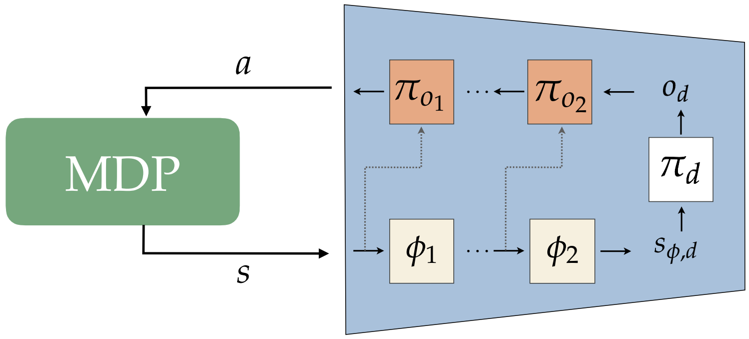


Figure 9.5: The construction of a hierarchy from $(\phi, \mathcal{O}_\phi)$ pairs.

for $s_{\phi,i} := \phi^i(s) = \phi_i(\ldots\phi_1(s)\ldots)$, with s a state in the ground MDP. I further denote V_i as the i-th level's value function, defined as follows for some ground state s.

$$V_i^{\pi}(s) = \max_{o \in \mathcal{O}_i} \left(R_i\left(s_i, o\right) + \sum_{s' \in \mathcal{S}_i} T_i\left(s' \mid s_i, o\right) V_i^{\pi}(s') \right), \tag{9.84}$$

where,

$$R_i(s_i, o) := \sum_{s_{i-1} \in s_i} w_i(s_{i-1}) R_{s_{i-1},o}, \tag{9.85}$$

$$T_i(s_i' \mid s_i, o) := \sum_{s_{i-1} \in s_i} \sum_{s_{i-1}' \in \mathcal{S}_{i-1}} w_i(s_{i-1}) T_{s_{i-1},o}^{s_{i-1}'}. \tag{9.86}$$

Again, $R_{s,o}$ and $T_{s,o}^{s'}$ are defined according to the multi-time model [311], $s_i \in \mathcal{S}_{\phi,i}$ is a level i state resulting from $\phi^i(s)$, and w_i is an aggregation weighting function for level i. Note that V_0 is the ground value function, which we refer to as V for simplicity. The full list of notation for this section is presented in Table 9.1.

9.2.1 *Hierarchy Analysis*

I now extend Theorem 9.1 to hierarchies of fixed but arbitrary depth, building on two key observations. First, any policy π_n represented at the top level of a hierarchy H_n also has a unique Markov policy in the ground MDP, which we denote $\pi_n^{\Downarrow}$ (in contrast to $\pi_n^{\downarrow}$, which moves the level n policy to level $n-1$). I summarize this fact in the following remark:

Remark 9.2. *Every deterministic policy π_i defined by the i-th level of a hierarchy, H_n, induces a unique policy in the ground MDP, which we denote $\pi_i^{\Downarrow}$.*

To be precise, note that $\pi_i^{\downarrow}$ specifies the level i policy π_i mapped into level π_{i-1}, whereas $\pi_i^{\Downarrow}$ refers to the policy at π_i mapped into π_0. The process of forming this ground policy $\pi_n^{\Downarrow}$ from a policy at the top level of the hierarchy π_i, is pictured in Figure 9.6.

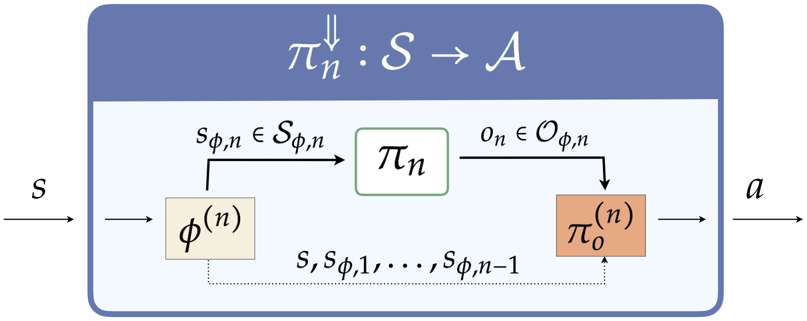

$\pi_n^{\Downarrow} : \mathcal{S} \to \mathcal{A}$
s
$s_{\phi,n} \in \mathcal{S}_{\phi,n}$
π_n
$o_n \in \mathcal{O}_{\phi,n}$
a
$\phi^{(n)}$
$\pi_o^{(n)}$
$s, s_{\phi,1}, \ldots, s_{\phi,n-1}$

Figure 9.6: The process of grounding a hierarchical policy.

ϕ	A state abstraction function.
$\mathcal{O}_\phi$	A set of ϕ-relative options.
$L(\phi, \mathcal{O}_\phi)$	The value loss of the $\phi, \mathcal{O}_\phi$ pair.
$\pi_{\mathcal{O}_\phi}$	A policy that maps each abstract state to an option.
$\pi^{\Downarrow}_{\mathcal{O}_\phi}$	A policy over $\mathcal{S}$ and $\mathcal{A}$, induced by $\pi_{\mathcal{O}_\phi}$.
H_n	A hierarchy of depth n, denoting the pair of lists $(\phi^{(n)}, \mathcal{O}_\phi^{(n)})$.
$\phi^{(n)}$	A list of n state abstractions, where $\phi_i : \mathcal{S}_{\phi,i-1} \to \mathcal{S}_{\phi,i}$.
ϕ_i	The i-th state abstraction in a list $\phi^{(n)}$.
ϕ^i	The result of applying the first i state abstractions to s, $\phi_i(\ldots\phi_1(s)\ldots)$.
$\mathcal{S}_{\phi,i}$	The i-th abstract state space, with $\mathcal{S}_{\phi,0}$ the ground state space.
s_i	A state belonging to $\mathcal{S}_{\phi,i}$
V_i^π	Value of level i under policy π, defined according to R_i and T_i.
$\mathcal{O}_{\phi,i}$	Options at level i, with each component defined over states in $\mathcal{S}_{\phi,i-1}$.
R_i	A reward function over level i states and options.
T_i	A transition function over level i states and options.
π_i	The policy over level i of the hierarchy such that $\pi_i : \mathcal{S}_i \to \mathcal{O}_{\phi,i}$.
$\pi_i^{\downarrow}$	A policy over $\mathcal{S}_{\phi,i-1}$ and $\mathcal{O}_{\phi,i-1}$, induced by π_i.
$\pi_i^{\Downarrow}$	A policy over $\mathcal{S}$ and $\mathcal{A}$, induced by π_i.

Table 9.1: Hierarchical abstraction notation.

The second key insight is that the value loss of $(\phi, \mathcal{O}_\phi)$ pairs applies in a straightforward way to hierarchies, H_n.

Definition 9.6. *The **value loss** of a depth n hierarchy H_n is the smallest degree of suboptimality across all policies representable at the top level of the hierarchy:*

$$L(H_n) := \min_{\pi_n \in \Pi_n} \left\| V^* - V^{\pi_n^{\Downarrow}} \right\|_\infty . \tag{9.87}$$

This quantity denotes how suboptimal the best hierarchical policy is in the ground MDP. Therefore, the guarantee we present expresses a condition on *global* optimality rather than *recursive* or *hierarchical* optimality [88].

I next show that there exist value-preserving hierarchies by bounding the above quantity for well constructed hierarchies. To prove this result, we require two assumptions.

Assumption 9.1. *The value function is consistent throughout the hierarchy. That is, for every level of the hierarchy $i \in \mathbb{N}_{\leq n-1}$, for any policy π_i over states $\mathcal{S}_{\phi,i}$ and options $\mathcal{O}_{\phi,i}$, there is a small $\kappa \in \mathbb{R}_{\geq 0}$ such that:*

$$\max_{s \in \mathcal{S}} \left| V_{i-1}^{\pi_i^{\downarrow}} \left(\phi^{i-1}(s) \right) - V_i^{\pi_i} \left(\phi^i(s) \right) \right| \leq \kappa \tag{9.88}$$

Assumption 9.2. *Subsequent levels of the hierarchy can represent policies similar in value to the best policy at the previous level. That is, for every $i \in \mathbb{N}_{\leq n-1}$, letting $\pi_i^\diamond = \arg\min_{\pi_i \in \Pi_i} || V_0^* - V_0^{\pi_i^{\Downarrow}} ||_\infty$, there is a small $\ell \in \mathbb{R}_{\geq 0}$ such that:*

$$\min_{\pi_{i+1}^{\downarrow} \in \Pi_{i+1}^{\downarrow}} \left\| V_i^{\pi_i^\diamond} - V_i^{\pi_{i+1}^{\downarrow}} \right\|_\infty \leq \ell. \tag{9.89}$$

It is likely that both assumptions are true given the right choice of state abstractions, options, and methods of constructing abstract MDPs. As some motivating evidence, a claim closely related to Assumption 9.1 was proven in Chapter 3 as Equation 3.14, and Assumption 9.2 is of similar structure to Theorem 9.1. These two assumptions (along with Theorem 9.1) give rise to hierarchies that can represent near-optimal behavior.

Theorem 9.3. *Consider two algorithms: 1) $\mathscr{A}_\phi$: given an MDP M, outputs a ϕ, and 2) $\mathscr{A}_{\mathcal{O}_\phi}$: given M and a ϕ, outputs a set of options $\mathcal{O}_\phi$ such that there are constants κ and ℓ for which Assumption 9.1 and Assumption 9.2 are satisfied. Then, by repeated application of $\mathscr{A}_\phi$ and $\mathscr{A}_{\mathcal{O}_\phi}$, we can construct a hierarchy of depth n such that*

$$L(H_n) \leq n(\kappa + \ell). \tag{9.90}$$

Proof of Theorem 9.3.

We present the proof of the bound for a two level hierarchy, but the same strategy generalizes to n levels via induction.

Let ℓ be the known upper bound for $L(\phi, \mathcal{O})$, obtained by any of the $(\phi, \mathcal{O}_\phi)$ pairs from Theorem 9.1.

By definition of $L(\phi, \mathcal{O})$:

$$\min_{\pi_1 \in \Pi_1} ||V_0^* - V_0^{\pi_1^\downarrow}||_\infty \leq \ell. \tag{9.91}$$

By Assumption 9.1:

$$\forall_{\pi_1 \in \Pi_1} : ||V_0^{\pi_1^\downarrow} - V_1^{\pi_1}||_\infty \leq \kappa. \tag{9.92}$$

Letting $\pi_1^\diamond = \arg\min_{\pi_1 \in \Pi_1} ||V_0^* - V_0^{\pi_1^\downarrow}||_\infty$, by Assumption 9.2:

$$\min_{\pi_2^\downarrow \in \Pi_2^\downarrow} ||V_1^{\pi_1^\diamond} - V_1^{\pi_2^\downarrow}||_\infty \leq \ell. \tag{9.93}$$

By Assumption 9.1:

$$\forall_{\pi_2^\downarrow \in \Pi_2^\downarrow} : ||V_1^{\pi_2^\downarrow} - V_0^{\pi_2^\Downarrow}||_\infty \leq \kappa. \tag{9.94}$$

Therefore, by the triangle inequality:

$$\min_{\pi_2 \in \Pi_2} ||V_0^* - V_0^{\pi_2^\Downarrow}||_\infty \leq 2(\kappa + \ell). \qquad \square$$

This theorem offers a clear path for extending the guarantees of ϕ-relative options beyond the typical two-timescale setup observed in recent work [30, 247] to fully realize the benefits of (multi-level) hierarchical abstraction. Moreover, both Assumption 9.1 and Assumption 9.2 are sufficient—together with ϕ-relative options that satisfy Theorem 9.1—to construct a hierarchy with low value loss. One conclusion to draw is that algorithms for leveraging hierarchies may want to explicitly search for structures that satisfy our assumptions: 1) value function smoothness up and down the hierarchy, and 2) policy richness at each level of the hierarchy.

I have here proven which state-action abstractions are guaranteed to preserve representation of near-optimal policies. To do so, I introduced ϕ-relative options, a simple but expressive formalism for combining state abstractions with options. Under this formalism, I proposed four classes of ϕ-relative options with bounded value loss. Lastly, I proved that under mild conditions, pairs of state-action abstractions can be recursively combined to induce hierarchies that also possess near-optimality guarantees.

I take these results, along with those established in Part 2 and Part 3, to serve as a concrete path toward principled abstraction discovery and use in RL.

10

CONCLUSION

The thesis of this dissertation is that insights from computational complexity, decision-theoretic planning, and information theory can shape principled abstraction discovery algorithms that empower RL agents. I defended this thesis on two fronts: 1) state abstraction (Part 2), and 2) action abstraction (Part 3), with a final note on the tightly woven connections between these two (Part 4). In this final chapter, I offer broader outlooks on abstraction and its role in both AI and RL.

10.1 WHY ABSTRACTION?

In Chapter 1, I suggested that the process of abstraction is critical to the success of any adaptive sequential decision making agent. Naturally, such a claim is speculative, as the space of all possible agents is vast. It is not yet clear which properties precisely separate effective agents from the ineffective ones. I now revisit this point with the results established in the dissertation in hand. I set out to present the strongest argument about the potential for abstraction to contribute to solutions to the RL problem, though naturally much of the present discussion remains speculative.

I take there to be three senses in which understanding the role of abstraction in RL is useful:

1. Abstraction is sufficient (and perhaps, necessary) for grounding simulated states and actions to observation and behavior.

2. If the space of relevant worlds can be characterized by simple underlying laws, then abstraction may be viewed (charitably, perhaps) as the process of recovering this simplicity from an agent-centric experience of the world.

3. Even if the best RL agents do not explicitly abstract, there is likely to be *implicit* abstraction taking place in their computation. Furthermore, understanding the implicit mechanisms that support effective agency is still of deep scientific importance.

I now expand on each point in more detail.

First, abstraction is *sufficient* for grounding simulated states and actions to the observation and behavior space defined by the world. As highlighted throughout the dissertation and in prior work, these internal representations can be immensely useful for unleashing the power *and* reliability of computation onto sequential decision making problems. Konidaris [170] presents a compelling case for the necessity of abstraction from this perspective, too, arguing that "a necessary but understudied requirement for general intelligence is the ability to form task-specific abstract representations" (2019, p. 1). Like the views presented in this dissertation, Konidaris goes on to suggest that RL is an appropriate paradigm to formalize and investigate abstraction in the context of agency. Abstraction is at least one vehicle for carrying out simulated decision making in fictional, but grounded, state-action space. Using this capacity, agents are empowered to consider past counterfactuals or inform present decision from alternative courses of future behavior. Such practices appear critical to effective agency.

Second, abstraction can also be viewed as the recovery of simple underlying world laws from agent-centric experience. To expand on this point, let us make two assumptions. First, the space of worlds of interest are those with exceedingly simple descriptions that give rise to complex phenomena. Second, that agents of interest are resource-bounded, as has been articulated by many [286, 287, 118, 278, 201, 115]. Which kinds of resource-bounded agents will be most successful in such a space of worlds? An agent using a simple model can support more valuable computation per time step; it is likely that those models that recover more of the underlying laws will be more capable of making informed predictions about their surroundings. For more on this point, I expand on this argument in detail elsewhere [2].

Third, let us suppose that the most reliably successful RL agents are those that do not make explicit use of abstraction. I suggest that some form of abstraction is likely to be taking place *implicitly* within such agents.[10.1] That is, the more an agent can specialize to a particular distribution of worlds, the more effective they can be at learning and solving tasks in those worlds. I speculate that it is likely that the process of abstraction is necessarily tied to this process of specialization. This reasoning is supported in part by the bias-complexity trade off discussed throughout the dissertation; as an abstraction becomes more aggressive, the space of representable entities becomes less rich, thereby making many problems critical to learning and decision making easier, but potentially compromising the quality of the best entities representable. In this sense, I conjecture that the most effective learning agents will carry out abstraction in some form, whether implicitly or explicitly.

One might worry that without a clear protocol for testing for the implicit use of abstraction, this claim is unfalsifiable. However, one straightforward mechanism for testing whether an agent abstracts implicitly is to again turn to information theory. As has been argued elsewhere (see work by Legg and Hutter [193] and Dowe et al. [95] and references therein) *compression* may be tied to fundamental aspects of intelligent systems. Thus, let us suppose that some form of compression-based test will suffice for determining whether abstraction is implicitly used by an agent. Then, I claim, if it is feasible to determine which *kinds* of compression are essential to effective agency, and which are not. In this sense, abstraction remains critical.

[10.1] This conjecture is supported in part by point 2: simple underlying worlds yield simple explanatory models.

There are many other reasons to take abstraction-based approaches to RL as promising beyond those discussed thus far. With abstractions, there is a more direct path toward shared knowledge among a community of agents, simple communication between the agents, or mechanisms that test for the reliability, failure modes, safety, and robustness of an agent. As hinted at in Chapter 1, a hiker that first learns walking into a tree is painful is sure to share this finding with their community. As such discoveries become bigger and more significant, abstraction is essential to effectively convey knowledge across broad populations of agents. In addition to communication, the role of abstraction in relation to these other properties is also deserving of attention. Indeed, there are many fundamental questions left to address.

10.2 THE ROAD AHEAD

There are many remaining steps to realize the potential of abstraction in RL. The primary contributions of this thesis target finite MDPs, the class of state aggregation functions $\phi : \mathcal{S} \rightarrow \mathcal{S}_\phi$, and action abstractions $\omega : \mathcal{A} \mapsto \mathcal{O}$ where the set $\mathcal{O}$ is assumed to operate on these finite state spaces. The analysis and algorithms describe methods for finding and using abstractions that are guaranteed to retain desirable properties, with empirical support illustrating the potential for these abstractions to accelerate learning and planning. However, these results are restricted in several ways.

First, the kinds of functions studied are themselves relatively weak. The act of state aggregation or discretization can only go so far to simplify state spaces. This formalism fundamentally lacks the capacity to express the kinds of powerful relations or descriptions that appear in common language. The most natural expansion of the results presented in Part 2 will go beyond aggregation functions to richer function families that can define objects, relations, predicates, and their kin. Establishing the same degree of understanding about these more powerful function families is essential to a comprehensive understanding of how agents come to act in complex environments. The same can be said of options.

Second, the primary focus of this dissertation is on finite MDPs. As discussed briefly in Chapter 2, there are many schemes for defining the space of relevant worlds, with finite MDPs being one suitable choice. Much of the analysis and the vast majority of the experimental study in this dissertation takes place in the context of simple grid worlds and their kin, with only a few exceptions. Thus, a second critical direction is to expand the primary analysis beyond finite state-action spaces. Some preliminary directions toward this goal were summarized in Section 5.4, but there is more to be done.

Third, this dissertation focuses on the learning problem facing a single agent. In reality, of course, many agents of relevance occupy a community. These agents learn, act, explore, and plan based on the beliefs and behaviors of other agents. How might the abstraction desiderata change if two agents or more are learning cooperatively in the same world? It is likely, for instance, that agents ought to specialize their abstractions while retaining enough overlap to allow for communication. Understanding abstraction when multiple agents are present is a key direction for further work.

10.3 CONCLUDING REMARKS

In conclusion, I take understanding abstraction in RL to be of fundamental importance to a holistic science of AI. The formalisms and analysis of this dissertation build on a long line of research to provide new clarity on how to discover and use good abstractions in RL. There is still much to be done, but the road ahead is an exciting one.

BIBLIOGRAPHY

[1] Pieter Abbeel and Andrew Y. Ng. Apprenticeship Learning via Inverse Reinforcement Learning. In *Proceedings of the International Conference on Machine learning*, 2004.

[2] David Abel. Concepts in bounded rationality: Perspectives from reinforcement learning. Master's thesis, Brown University, 2019.

[3] David Abel. A theory of state abstraction for reinforcement learning. In *Proceedings of the Doctoral Consortium of the AAAI Conference on Artificial Intelligence*, 2019.

[4] David Abel, D. Ellis Hershkowitz, and Michael L. Littman. Near optimal behavior via approximate state abstraction. In *Proceedings of the International Conference on Machine Learning*, 2016.

[5] David Abel, Dilip Arumugam, Lucas Lehnert, and Michael L. Littman. Toward good abstractions for lifelong learning. In *NeurIPS Workshop on Hierarchical Reinforcement Learning*, 2017.

[6] David Abel, Dilip Arumugam, Lucas Lehnert, and Michael L. Littman. State abstractions for lifelong reinforcement learning. In *Proceedings of the International Conference on Machine Learning*, 2018.

[7] David Abel, Yuu Jinnai, Yue Guo, George Konidaris, and Michael L. Littman. Policy and value transfer in lifelong reinforcement learning. In *Proceedings of the International Conference on Machine Learning*, 2018.

[8] David Abel, Edward C. Willams, Stephen Brawner, Emily Reif, and Michael L. Littman. Bandit-based solar panel control. In *Proceedings of the Conference on Innovative Applications of Artificial Intelligence*, 2018.

[9] David Abel, Dilip Arumugam, Kavosh Asadi, Yuu Jinnai, Michael L. Littman, and Lawson L.S. Wong. State abstraction as compression in apprenticeship learning. In *Proceedings of the AAAI Conference on Artificial Intelligence*, 2019.

[10] David Abel, John Winder, Marie desJardins, and Michael L. Littman. The expected-length model of options. In *Proceedings of the International Joint Conference on Artificial Intelligence*, 2019.

[11] David Abel, Nathan Umbanhowar, Khimya Khetarpal, Dilip Arumugam, Doina Precup, and Michael L. Littman. Value preserving state-action abstractions. In *Proceedings of the International Conference on Artificial Intelligence and Statistics*, 2020.

[12] Riad Akrour, Filipe Veiga, Jan Peters, and Gerhard Neumann. Regularizing reinforcement learning with state abstraction. In *Proceedings of the International Conference on Intelligent Robots and Systems*. IEEE, 2018.

[13] David Aldous and James Fill. Reversible Markov chains and random walks on graphs. Technical report, University of California, Berkeley, 1995.

[14] Shun-Ichi Amari. Natural gradient works efficiently in learning. *Neural computation*, 10(2):251–276, 1998.

[15] Ankit Anand, Aditya Grover, Mausam, and Parag Singla. ASAP-UCT: abstraction of state-action pairs in UCT. In *Proceedings of the International Joint Conference on Artificial Intelligence*, 2015.

[16] Ankit Anand, Ritesh Noothigattu, Mausam, and Parag Singla. OGA-UCT: on-the-go abstractions in UCT. In *Proceedings of the International Conference on Automated Planning and Scheduling*, 2016.

[17] David Andre and Stuart Russell. State abstraction for programmable reinforcement learning agents. In *Proceedings of the AAAI Conference on Artificial Intelligence*, 2002.

[18] Aaron Archer. Two $O(\log^* k)$-approximation algorithms for the asymmetric k-center problem. In *Proceedings of the International Conference on Integer Programming and Combinatorial Optimization*, 2001.

[19] Brenna D. Argall, Sonia Chernova, Manuela Veloso, and Brett Browning. A survey of robot learning from demonstration. *Robotics and autonomous systems*, 57(5):469–483, 2009.

[20] Suguru Arimoto. An algorithm for computing the capacity of arbitrary discrete memoryless channels. *IEEE Transactions on Information Theory*, 18(1):14–20, 1972.

[21] Kavosh Asadi. Strengths, weaknesses, and combinations of model-based and model-free reinforcement learning. Master's thesis, University of Alberta, 2015.

[22] Kavosh Asadi and Michael L. Littman. An alternative softmax operator for reinforcement learning. In *Proceedings of the International Conference on Machine Learning*, 2017.

[23] Kavosh Asadi, Dipendra Misra, and Michael L. Littman. Lipschitz continuity in model-based reinforcement learning. In *Proceedings of the International Conference on Machine Learning*, 2018.

[24] Kavosh Asadi, David Abel, and Michael L. Littman. Learning state abstractions for transfer in continuous control. *arXiv preprint arXiv:2002.05518*, 2020.

[25] John Asmuth, Lihong Li, Michael L. Littman, Ali Nouri, and David Wingate. A Bayesian sampling approach to exploration in reinforcement learning. In *Proceedings of the Conference on Uncertainty in Artificial Intelligence*, 2009.

[26] Christopher G. Atkeson and Stefan Schaal. Robot learning from demonstration. In *Proceedings of the International Conference on Machine Learning*, 1997.

[27] Fred Attneave. Some informational aspects of visual perception. *Psychological Review*, 3(61), 1954.

[28] Peter Auer and Ronald Ortner. Logarithmic online regret bounds for undiscounted reinforcement learning. In *Advances in Neural Information Processing Systems*, 2007.

[29] Sven Axsäter. State aggregation in dynamic programming — an application to scheduling of independent jobs on parallel processors. *Operations Research Letters*, 2 (4):171–176, 1983.

[30] Pierre-Luc Bacon, Jean Harb, and Doina Precup. The option-critic architecture. In *Proceedings of the AAAI Conference on Artificial Intelligence*, 2017.

[31] Akhil Bagaria and George Konidaris. Option discovery using deep skill chaining. In *Proceedings of the International Conference on Learning Representations*, 2020.

[32] Aijun Bai and Stuart Russell. Efficient reinforcement learning with hierarchies of machines by leveraging internal transitions. In *Proceedings of the International Joint Conference on Artificial Intelligence*, 2017.

[33] Horace B. Barlow. Possible principles underlying the transformation of sensory messages. *Sensory communication*, 1:217–234, 1961.

[34] André Barreto, Will Dabney, Rémi Munos, Jonathan J. Hunt, Tom Schaul, Hado van Hasselt, and David Silver. Successor features for transfer in reinforcement learning. In *Advances in Neural Information Processing Systems*, 2017.

[35] André Barreto, Diana Borsa, Shaobo Hou, Gheorghe Comanici, Eser Aygün, Philippe Hamel, Daniel Toyama, Jonathan Hunt, Shibl Mourad, David Silver, and Doina Precup. The option keyboard: Combining skills in reinforcement learning. In *Advances in Neural Information Processing Systems*, 2019.

[36] Peter L. Bartlett and Shahar Mendelson. Rademacher and gaussian complexities: Risk bounds and structural results. *Journal of Machine Learning Research*, 3(Nov), 2002.

[37] Andrew G. Barto and Sridhar Mahadevan. Recent advances in hierarchical reinforcement learning. *Discrete event dynamic systems*, 13(1-2):41–77, 2003.

[38] Hannah M. Bayer and Paul W. Glimcher. Midbrain dopamine neurons encode a quantitative reward prediction error signal. *Neuron*, 47(1):129–141, 2005.

[39] Marc G. Bellemare, Yavar Naddaf, Joel Veness, and Michael Bowling. The arcade learning environment: An evaluation platform for general agents. *Journal of Artificial Intelligence Research*, 47:253–279, 2013.

[40] Marc G. Bellemare, Will Dabney, and Rémi Munos. A distributional perspective on reinforcement learning. In *Proceedings of the International Conference on Machine Learning*, 2017.

[41] Richard Bellman. Dynamic programming and Lagrange multipliers. *Proceedings of the National Academy of Sciences*, 42(10):767–769, 1956.

[42] Richard Bellman. A Markovian decision process. *Journal of Mathematics and Mechanics*, pages 679–684, 1957.

[43] Toby Berger. *Rate distortion theory: A mathematical basis for data compression*. Prentice-Hall, 1971.

[44] Dimitri P. Bertsekas. Feature-based aggregation and deep reinforcement learning: A survey and some new implementations. *IEEE/CAA Journal of Automatica Sinica*, 6 (1):1–31, 2018.

[45] Dimitri P. Bertsekas and David A. Castanon. Adaptive aggregation methods for infinite horizon dynamic programming. *IEEE Transactions on Automatic Control*, 34: 589–598, 1989.

[46] Dimitri P. Bertsekas and John N. Tsitsiklis. *Neuro-dynamic programming*, volume 5. Athena Scientific Belmont, MA, 1996.

[47] Richard Blahut. Computation of channel capacity and rate-distortion functions. *IEEE transactions on Information Theory*, 18(4):460–473, 1972.

[48] Matthew Botvinick and Ari Weinstein. Model-based hierarchical reinforcement learning and human action control. *Philosophical Transactions of the Royal Society B: Biological Sciences*, 369(1655), 2014.

[49] Matthew Botvinick, Ari Weinstein, Alec Solway, and Andrew G. Barto. Reinforcement learning, efficient coding, and the statistics of natural tasks. *Current Opinion in Behavioral Sciences*, 5:71–77, 2015.

[50] Matthew M. Botvinick, Yael Niv, and Andew G. Barto. Hierarchically organized behavior and its neural foundations: A reinforcement learning perspective. *Cognition*, 113(3):262–280, 2009.

[51] George E.P. Box. Science and statistics. *Journal of the American Statistical Association*, 71(356):791–799, 1976.

[52] Dietrich Braess. Über ein paradoxon aus der verkehrsplanung. *Unternehmensforschung*, 12(1):258–268, 1968.

[53] Dietrich Braess, Anna Nagurney, and Tina Wakolbinger. On a paradox of traffic planning. *Transportation science*, 39(4):446–450, 2005.

[54] Ronen I. Brafman and Moshe Tennenholtz. R-MAX: A general polynomial time algorithm for near-optimal reinforcement learning. *JMLR*, 3(Oct):213–231, 2002.

[55] Pierre Brémaud. *Markov chains: Gibbs fields, Monte Carlo simulation, and queues*, volume 31. Springer Science & Business Media, 2013.

[56] Greg Brockman, Vicki Cheung, Ludwig Pettersson, Jonas Schneider, John Schulman, Jie Tang, and Wojciech Zaremba. OpenAI gym, 2016.

[57] Andrei Z. Broder and Anna R. Karlin. Bounds on the cover time. *Journal of Theoretical Probability*, 2(1):101–120, 1989.

[58] Emma Brunskill and Lihong Li. PAC-inspired option discovery in lifelong reinforcement learning. In *Proceedings of the International Conference on Machine Learning*, 2014.

[59] Emma Brunskill and Lihong Li. The online coupon-collector problem and its application to lifelong reinforcement learning. *arXiv preprint arXiv:1506.03379*, 2015.

[60] Robert R. Burridge, Alfred A. Rizzi, and Daniel E. Koditschek. Sequential composition of dynamically dexterous robot behaviors. *The International Journal of Robotics Research*, 18(6):534–555, 1999.

[61] Tom Bylander. Complexity results for planning. In *Proceedings of the International Joint Conference on Artificial Intelligence*, volume 10, 1991.

[62] Tom Bylander. The computational complexity of propositional STRIPS planning. *Artificial Intelligence*, 69(1-2):165–204, 1994.

[63] Feng Cao and Soumya Ray. Bayesian hierarchical reinforcement learning. In *Advances in Neural information Processing Systems*, 2012.

[64] Pablo Samuel Castro. Scalable methods for computing state similarity in deterministic Markov decision processes. In *Proceedings of the AAAI Conference on Artificial Intelligence*, 2020.

[65] Pablo Samuel Castro and Doina Precup. Automatic construction of temporally extended actions for MDPs using bisimulation metrics. In *Proceedings of the European Workshop on Reinforcement Learning*, 2011.

[66] David Chapman and Leslie Pack Kaelbling. Input generalization in delayed reinforcement learning: An algorithm and performance comparisons. In *Proceedings of the International Joint Conference on Artificial Intelligence*, 1991.

[67] Fan R.K. Chung. *Spectral graph theory*. American Mathematical Society, 1996.

[68] Julia Chuzhoy, Sudipto Guha, Eran Halperin, Sanjeev Khanna, Guy Kortsarz, Robert Krauthgamer, and Joseph Seffi Naor. Asymmetric k-center is $\log^* n$-hard to approximate. *Journal of the ACM*, 52(4):538–551, 2005.

[69] Vasek Chvatal. A greedy heuristic for the set-covering problem. *Mathematics of operations research*, 4(3):233–235, 1979.

[70] Kamil Ciosek and David Silver. Value iteration with options and state aggregation. *arXiv:1501.03959*, 2015.

[71] Luis C. Cobo, Peng Zang, Charles L. Isbell, and Andrea L. Thomaz. Automatic state abstraction from demonstration. In *Proceedings of the International Joint Conference on Artificial Intelligence*, 2011.

[72] Luis C. Cobo, Charles L. Isbell, and Andrea L. Thomaz. Automatic task decomposition and state abstraction from demonstration. In *Proceedings of the International Conference on Autonomous Agents and Multiagent Systems*, 2012.

[73] David A. Cohn, Zoubin Ghahramani, and Michael I. Jordan. Active learning with statistical models. *Journal of Artificial Intelligence Research*, 4:129–145, 1996.

[74] Rémi Coulom. Efficient selectivity and backup operators in Monte-Carlo tree search. In *Proceedings of the International Conference on Computers and Games*, 2006.

[75] Thomas M. Cover and Joy A. Thomas. *Elements of information theory*. John Wiley & Sons, 2012.

[76] Ken Currie and Austin Tate. O-plan: the open planning architecture. *Artificial intelligence*, 52(1):49–86, 1991.

[77] Bruno Castro da Silva, George Konidaris, and Andrew G. Barto. Learning parameterized skills. In *Proceedings of the International Conference on Machine Learning*, 2012.

[78] Bruno Castro da Silva, George Konidaris, and Andrew G. Barto. Active learning of parameterized skills. In *Proceedings of the International Conference on Machine Learning*, 2014.

[79] Will Dabney, Georg Ostrovski, David Silver, and Rémi Munos. Implicit quantile networks for distributional reinforcement learning. In *Proceedings of the International Conference on Machine Learning*, 2018.

[80] Will Dabney, Mark Rowland, Marc G. Bellemare, and Rémi Munos. Distributional reinforcement learning with quantile regression. In *Proceedings of the AAAI Conference on Artificial Intelligence*, 2018.

[81] Will Dabney, Zeb Kurth-Nelson, Naoshige Uchida, Clara Kwon Starkweather, Demis Hassabis, Rémi Munos, and Matthew Botvinick. A distributional code for value in dopamine-based reinforcement learning. *Nature*, pages 1–5, 2020.

[82] Christoph Dann, Tor Lattimore, and Emma Brunskill. Unifying PAC and regret: Uniform PAC bounds for episodic reinforcement learning. In *Advances in Neural Information Processing Systems*, 2017.

[83] Peter Dayan. Improving generalization for temporal difference learning: The successor representation. *Neural Computation*, 5(4):613–624, 1993.

[84] Peter Dayan and Bernard W. Balleine. Reward, motivation, and reinforcement learning. *Neuron*, 36(2):285–298, 2002.

[85] Peter Dayan and Geoffrey E. Hinton. Feudal reinforcement learning. In *Advances in Neural Information Processing Systems*, 1993.

[86] Thomas Dean and Robert Givan. Model minimization in Markov decision processes. In *Proceedings of the AAAI Conference on Artificial Intelligence*, 1997.

[87] Richard Dearden and Craig Boutilier. Abstraction and approximate decision-theoretic planning. *Artificial Intelligence*, 89(1):219–283, 1997.

[88] Thomas G. Dietterich. Hierarchical reinforcement learning with the MAXQ value function decomposition. *Journal of Artificial Intelligence Research*, 2000.

[89] Thomas G. Dietterich. State abstraction in MAXQ hierarchical reinforcement learning. In *Advances in Neural Information Processing Systems*, 2000.

[90] Bruce L. Digney. Learning hierarchical control structures for multiple tasks and changing environments. In *Proceedings of the International Conference on Simulation of Adaptive Behavior*, 1998.

[91] Michael Dinitz, Guy Kortsarz, and Ran Raz. Label cover instances with large girth and the hardness of approximating basic *k*-spanner. In *International Colloquium on Automata, Languages, and Programming*. Springer, 2012.

[92] Irit Dinur and David Steurer. Analytical approach to parallel repetition. In *Proceedings of the ACM Symposium on Theory of Computing*, 2014.

[93] Carlos Diuk, Andre Cohen, and Michael L. Littman. An object-oriented representation for efficient reinforcement learning. In *Proceedings of the International Conference on Machine Learning*, 2008.

[94] Carlos Diuk, Lihong Li, and Bethany R. Leffler. The adaptive *k*-meteorologists problem and its application to structure learning and feature selection in reinforcement learning. In *Proceedings of the International Conference on Machine Learning*, 2009.

[95] David L. Dowe, José Hernández-Orallo, and Paramjit K. Das. Compression and intelligence: social environments and communication. In *Proceedings of the International Conference on Artificial General Intelligence*, 2011.

[96] Simon S. Du, Akshay Krishnamurthy, Nan Jiang, Alekh Agarwal, Miroslav Dudík, and John Langford. Provably efficient RL with rich observations via latent state decoding. In *Proceedings of the International Conference on Machine Learning*, 2019.

[97] Maria Eckstein and Anne Collins. Evidence for hierarchically-structured reinforcement learning in humans. In *Proceedings of Annual Conference of the Cognitive Science Society*, 2018.

[98] Kutluhan Erol, James Hendler, and Dana S. Nau. HTN planning: Complexity and expressivity. In *Proceedings of the AAAI Conference on Artificial Intelligence*, 1994.

[99] Eyal Even-Dar and Yishay Mansour. Approximate equivalence of Markov decision processes. In *Learning Theory and Kernel Machines*. Springer, 2003.

[100] Benjamin Eysenbach, Abhishek Gupta, Julian Ibarz, and Sergey Levine. Diversity is all you need: Learning skills without a reward function. In *Proceedings of the International Conference on Representation Learning*, 2019.

[101] Zhengzhu Feng, Richard Dearden, Nicolas Meuleau, and Richard Washington. Dynamic programming for structured continuous Markov decision problems. In *Proceedings of the Conference on Uncertainty in Artificial Intelligence*, 2004.

[102] Norm Ferns, Prakash Panangaden, and Doina Precup. Metrics for finite Markov decision processes. In *Proceedings of the Conference on Uncertainty in Artificial Intelligence*, 2004.

[103] Norm Ferns, Pablo Samuel Castro, Doina Precup, and Prakash Panangaden. Methods for computing state similarity in Markov decision processes. In *Proceedings of the Conference on Uncertainty in Artificial Intelligence*, 2006.

[104] Miroslav Fiedler. Algebraic connectivity of graphs. *Czechoslovak Mathematical Journal*, 23(2):298–305, 1973.

[105] Richard E. Fikes and Nils J. Nilsson. STRIPS: A new approach to the application of theorem proving to problem solving. *Artificial intelligence*, 2(3-4):189–208, 1971.

[106] Carlos Florensa, Yan Duan, and Pieter Abbeel. Stochastic neural networks for hierarchical reinforcement learning. In *Proceedings of the International Conference on Learning Representations*, 2017.

[107] Bennett L. Fox. Discretizing dynamic programs. *Journal of Optimization Theory and Applications*, 11(3):228–234, 1973.

[108] Vincent François-Lavet, Yoshua Bengio, Doina Precup, and Joelle Pineau. Combined reinforcement learning via abstract representations. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 33, 2019.

[109] Linton C. Freeman. A set of measures of centrality based on betweenness. *Sociometry*, pages 35–41, 1977.

[110] Jerome H. Friedman, Jon Louis Bentley, and Raphael Ari Finkel. An algorithm for finding best matches in logarithmic expected time. *ACM Transactions on Mathematical Software*, 3(3):209–226, 1977.

[111] Ronan Fruit and Alessandro Lazaric. Exploration–exploitation in MDPs with options. In *Proceedings of the International Conference on Artificial Intelligence and Statistics*, 2017.

[112] Ronan Fruit, Matteo Pirotta, Alessandro Lazaric, and Emma Brunskill. Regret minimization in MDPs with options without prior knowledge. In *Advances in Neural Information Processing Systems*, 2017.

[113] Natalia H. Gardiol and Leslie Pack Kaelbling. Envelope-based planning in relational MDPs. In *Advances in Neural Information Processing Systems*, 2004.

[114] Carles Gelada, Saurabh Kumar, Jacob Buckman, Ofir Nachum, and Marc G. Bellemare. DeepMDP: Learning continuous latent space models for representation learning. In *Proceedings of the International Conference on Machine Learning*, 2019.

[115] Samuel J. Gershman, Eric J. Horvitz, and Joshua Tenenbaum. Computational rationality: A converging paradigm for intelligence in brains, minds, and machines. *Science*, 349(6245):273–278, 2015.

[116] Mohammad Ghavamzadeh, Sridhar Mahadevan, and Rajbala Makar. Hierarchical multi-agent reinforcement learning. *Autonomous Agents and Multi-Agent Systems*, 13 (2):197–229, 2006.

[117] Arpita Ghosh and Stephen Boyd. Growing well-connected graphs. In *Decision and Control, 2006 45th IEEE Conference on*, pages 6605–6611. IEEE, 2006.

[118] Gerd Gigerenzer and Daniel G. Goldstein. Reasoning the fast and frugal way: models of bounded rationality. *Psychological review*, 103(4):650, 1996.

[119] Fausto Giunchiglia and Toby Walsh. A theory of abstraction. *Artificial intelligence*, 57(2-3):323–389, 1992.

[120] Robert Givan, Sonia Leach, and Thomas Dean. Bounded parameter Markov decision processes. In *European Conference on Planning*, 1997.

[121] Nakul Gopalan, Marie desJardins, Michael L. Littman, James MacGlashan, Shawn Squire, Stefanie Tellex, John Winder, and Lawson L.S. Wong. Planning with abstract Markov decision processes. In *Proceedings of the International Conference on Automated Planning and Scheduling*, 2017.

[122] Geoffrey J. Gordon. Chattering in SARSA(λ). Technical report, Carnegie Mellon University Learning Lab, 1996.

[123] Carlos Guestrin, Daphne Koller, and Ronald Parr. Max-norm projections for factored MDPs. In *Proceedings of the International Joint Conference on Artificial Intelligence*, 2001.

[124] Jean Harb, Pierre-Luc Bacon, Martin Klissarov, and Doina Precup. When waiting is not an option: Learning options with a deliberation cost. In *Proceedings of the AAAI Conference on Artificial Intelligence*, 2018.

[125] Juris Hartmanis and Richard E. Stearns. *Algebraic structure theory of sequential machines*. Prentice-Hall, Inc., 1966.

[126] Anna Harutyunyan, Peter Vrancx, Pierre-Luc Bacon, Doina Precup, and Ann Nowé. Learning with options that terminate off-policy. In *Proceedings of the AAAI Conference on Artificial Intelligence*, 2018.

[127] Anna Harutyunyan, Will Dabney, Thomas Mesnard, Mohammad Gheshlaghi Azar, Bilal Piot, Nicolas Heess, Hado van Hasselt, Gregory Wayne, Satinder Singh, and Doina Precup. Hindsight credit assignment. In *Advances in Neural Information Processing Systems*, 2019.

[128] Milos Hauskrecht, Nicolas Meuleau, Leslie Pack Kaelbling, Thomas Dean, and Craig Boutilier. Hierarchical solution of Markov decision processes using macro-actions. In *Proceedings of the Conference on Uncertainty in Artificial Intelligence*, 1998.

[129] Bernhard Hengst. Discovering hierarchy in reinforcement learning with HEXQ. *Proceedings of the International Conference on Machine Learning*, 2002.

[130] Matteo Hessel, Joseph Modayil, Hado van Hasselt, Tom Schaul, Georg Ostrovski, Will Dabney, Dan Horgan, Bilal Piot, Mohammad Gheshlaghi Azar, and David Silver. Rainbow: Combining improvements in deep reinforcement learning. In *Proceedings of the AAAI Conference on Artificial Intelligence*, 2018.

[131] Mark K. Ho, David Abel, Thomas L. Griffiths, and Michael L. Littman. The value of abstraction. *Current Opinion in Behavioral Sciences*, 2019.

[132] Mark K. Ho, David Abel, Jonathan D. Cohen, Michael L. Littman, and Thomas L. Griffiths. The efficiency of human cognition reflects planned use of information processing. In *Proceedings of the AAAI Conference on Artificial Intelligence*, 2020.

[133] Jesse Hostetler, Alan Fern, and Thomas G. Dietterich. State Aggregation in Monte Carlo Tree Search. In *Proceedings of the AAAI Conference on Artificial Intelligence*, 2014.

[134] Marcus Hutter. *Universal artificial intelligence: Sequential decisions based on algorithmic probability*. Springer, 2005.

[135] Marcus Hutter. Extreme state aggregation beyond MDPs. In *Proceedings of the International Conference on Algorithmic Learning Theory*, 2014.

[136] Marcus Hutter. Extreme state aggregation beyond Markov decision processes. *Theoretical Computer Science*, 650:73–91, 2016.

[137] Glenn A. Iba. A heuristic approach to the discovery of macro-operators. *Machine Learning*, 3(4):285–317, 1989.

[138] David Isele, Mohammad Rostami, and Eric Eaton. Using task features for zero-shot knowledge transfer in lifelong learning. In *Proceedings of the International Joint Conference on Artificial Intelligence*, 2016.

[139] Steven James, George Konidaris, and Benjamin Rosman. An analysis of Monte Carlo tree search. In *Proceedings of the AAAI Conference on Artificial Intelligence*, 2017.

[140] Steven James, Benjamin Rosman, and George Konidaris. Learning to plan with portable symbols. In *ICML Workshop on Planning and Learning*, 2018.

[141] Nan Jiang, Satinder Singh, and Richard Lewis. Improving UCT planning via approximate homomorphisms. In *Proceedings of the International Conference on Autonomous Agents and Multi-Agent Systems*, 2014.

[142] Nan Jiang, Alex Kulesza, and Satinder Singh. Abstraction selection in model-based reinforcement learning. In *Proceedings of the International Conference on Machine Learning*, 2015.

[143] Nan Jiang, Alex Kulesza, Satinder Singh, and Richard Lewis. The dependence of effective planning horizon on model accuracy. In *Proceedings of the International Conference on Autonomous Agents and Multiagent Systems*, 2015.

[144] Yuu Jinnai, David Abel, D. Ellis Hershkowitz, Michael L. Littman, and George Konidaris. Finding options that minimize planning time. In *Proceedings of the International Conference on Machine Learning*, 2019.

[145] Yuu Jinnai, Jee Won Park, David Abel, and George Konidaris. Discovering options for exploration by minimizing cover time. In *Proceedings of the International Conference on Machine Learning*, 2019.

[146] Yuu Jinnai, Jee Won Park, Marlos C. Machado, and George Konidaris. Exploration in reinforcement learning with deep covering options. In *Proceedings of the International Conference on Learning Representations*, 2020.

[147] Nicholas K. Jong and Peter Stone. State abstraction discovery from irrelevant state variables. In *Proceedings of the International Joint Conference on Artificial Intelligence*, 2005.

[148] Nicholas K. Jong and Peter Stone. Hierarchical model-based reinforcement learning: R-MAX + MAXQ. In *Proceedings of the International Conference on Machine Learning*, 2008.

[149] Nicholas K. Jong, Todd Hester, and Peter Stone. The utility of temporal abstraction in reinforcement learning. In *Proceedings of the International Conference on Autonomous Agents and Multiagent Systems*, 2008.

[150] Rico Jonschkowski and Oliver Brock. Learning state representations with robotic priors. *Autonomous Robots*, 39(3):407–428, 2015.

[151] Anders Jonsson and Andrew G. Barto. Automated state abstraction for options using the U-tree algorithm. In *Advances in Neural Information Processing Systems*, 2001.

[152] Leslie Pack Kaelbling. Hierarchical reinforcement learning: Preliminary results. In *Proceedings of the International Conference on Machine Learning*, 1993.

[153] Leslie Pack Kaelbling, Michael L. Littman, and Andrew W. Moore. Reinforcement learning: A survey. *Journal of Artificial Intelligence Research*, pages 237–285, 1996.

[154] Leslie Pack Kaelbling, Michael L. Littman, and Anthony R. Cassandra. Planning and acting in partially observable stochastic domains. *Artificial intelligence*, 101(1-2): 99–134, 1998.

[155] Sham Kakade. *On the sample complexity of reinforcement learning*. PhD thesis, University of London, 2003.

[156] Maximilian Karl, Maximilian Soelch, Justin Bayer, and Patrick van der Smagt. Deep variational Bayes filters: Unsupervised learning of state space models from raw data. In *Proceedings of the International Conference on Learning Representations*, 2017.

[157] Michael Kearns and Satinder Singh. Near-optimal reinforcement learning in polynomial time. *Machine learning*, 49(2-3):209–232, 2002.

[158] Charles Kemp and Joshua Tenenbaum. Structured statistical models of inductive reasoning. *Psychological review*, 116(1):20, 2009.

[159] Kristian Kersting, Martijn Van Otterlo, and Luc De Raedt. Bellman goes relational. In *Proceedings of the International Conference on Machine learning*, 2004.

[160] Khimya Khetarpal, Martin Klissarov, Maxime Chevalier-Boisvert, Pierre-Luc Bacon, and Doina Precup. Options of interest: Temporal abstraction with interest functions. In *Proceedings of the AAAI Conference on Artificial Intelligence*, 2020.

[161] Hyunjik Kim and Andriy Mnih. Disentangling by factorising. In *Proceedings of the International Conference on Machine Learning*, 2018.

[162] Diederik P. Kingma and Jimmy Ba. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*, 2014.

[163] Diederik P. Kingma and Max Welling. Auto-encoding variational Bayes. In *Proceedings of the International Conference on Learning Representations*, 2014.

[164] Jon Klein. BREVE: a 3D environment for the simulation of decentralized systems and artificial life. In *Proceedings of the International Conference on Artificial life*, 2003.

[165] Craig A. Knoblock. Automatically generating abstractions for planning. *Artificial Intelligence*, 68(2):243–302, 1994.

[166] Levente Kocsis and Csaba Szepesvári. Bandit based monte-carlo planning. In *In Proceedings of the European Conference on Machine Learning*, 2006.

[167] Daphne Koller and Ronald Parr. Computing factored value functions for policies in structured MDPs. In *Proceedings of the International Joint Conference on Artificial Intelligence*, 1999.

[168] George Konidaris. *Autonomous Robot Skill Acquisition*. PhD thesis, University of Massachusetts Amherst, 2011.

[169] George Konidaris. Constructing abstraction hierarchies using a skill-symbol loop. In *Proceedings of the International Joint Conference on Artificial Intelligence*, 2016.

[170] George Konidaris. On the necessity of abstraction. *Current opinion in behavioral sciences*, 29:1–7, 2019.

[171] George Konidaris and Andrew G. Barto. Autonomous shaping: Knowledge transfer in reinforcement learning. In *Proceedings of the International Conference on Machine Learning*, 2006.

[172] George Konidaris and Andrew G. Barto. Building portable options: Skill transfer in reinforcement learning. In *Proceedings of the International Joint Conference on Artificial Intelligence*, 2007.

[173] George Konidaris and Andrew G. Barto. Efficient skill learning using abstraction selection. In *Proceedings of the International Joint Conference on Artificial Intelligence*, 2009.

[174] George Konidaris and Andrew G. Barto. Skill discovery in continuous reinforcement learning domains using skill chaining. In *Advances in Neural Information Processing Systems*, 2009.

[175] George Konidaris, Scott Kuindersma, Roderic Grupen, and Andrew G. Barto. Constructing skill trees for reinforcement learning agents from demonstration trajectories. In *Advances in Neural Information Processing Systems*, 2010.

[176] George Konidaris, Scott Kuindersma, Roderic Grupen, and Andrew G. Barto. Autonomous skill acquisition on a mobile manipulator. In *Proceedings of the AAAI Conference on Artificial Intelligence*, 2011.

[177] George Konidaris, Sarah Osentoski, and Philip S. Thomas. Value function approximation in reinforcement learning using the Fourier basis. In *Proceedings of the AAAI Conference on Artificial Intelligence*, 2011.

[178] George Konidaris, Leslie Pack Kaelbling, and Tomás Lozano-Pérez. Constructing symbolic representations for high-level planning. In *Proceedings of the AAAI Conference on Artificial Intelligence*, 2014.

[179] George Konidaris, Leslie Pack Kaelbling, and Tomás Lozano-Pérez. Symbol acquisition for probabilistic high-level planning. In *Proceedings of the International Joint Conference on Artificial Intelligence*, 2015.

[180] George Konidaris, Leslie Pack Kaelbling, and Tomás Lozano-Pérez. From skills to symbols: Learning symbolic representations for abstract high-level planning. *Journal of Artificial Intelligence Research*, 2018.

[181] Yehuda Koren. On spectral graph drawing. In *International Computing and Combinatorics Conference*, pages 496–508. Springer, 2003.

[182] Richard E. Korf. Learning to solve problems by searching for macro-operators. Technical report, Carnegie Mellon University, 1983.

[183] Richard E. Korf. Macro-operators: A weak method for learning. *Artificial intelligence*, 26(1):35–77, 1985.

[184] Mark Kroon and Shimon Whiteson. Automatic feature selection for model-based reinforcement learning in factored MDPs. In *Proceedings of the International Conference on Machine Learning and Applications*, 2009.

[185] Ben J.A. Krose and Joris W.M. Van Dam. Adaptive state space quantisation for reinforcement learning of collision-free navigation. In *Proceedings of the International Conference on Intelligent Robots and Systems*, 1992.

[186] Tejas D. Kulkarni, Karthik Narasimhan, Ardavan Saeedi, and Josh Tenenbaum. Hierarchical deep reinforcement learning: Integrating temporal abstraction and intrinsic motivation. In *Advances in Neural Information Processing Systems*, 2016.

[187] Michail G. Lagoudakis and Ronald Parr. Least-squares policy iteration. *Journal of Machine Learning research*, 4(Dec):1107–1149, 2003.

[188] Kim G. Larsen and Arne Skou. Bisimulation through probabilistic testing. *Information and computation*, 94(1):1–28, 1991.

[189] Tor Lattimore and Csaba Szepesvári. Bandit algorithms. *preprint*, 2018.

[190] Hoang M. Le, Nan Jiang, Alekh Agarwal, Miroslav Dudík, Yisong Yue, and Hal Daumé III. Hierarchical imitation and reinforcement learning. In *Proceedings of the International Conference on Machine Learning*, 2018.

[191] Yann LeCun, Yoshua Bengio, and Geoffrey Hinton. Deep learning. *Nature*, 521 (7553):436, 2015.

[192] Ivan S.K. Lee and Henry Y.K. Lau. Adaptive state space partitioning for reinforcement learning. *Engineering Applications of Artificial Intelligence*, 17(6):577–588, 2004.

[193] Shane Legg and Marcus Hutter. Universal intelligence: A definition of machine intelligence. *Minds and machines*, 17(4):391–444, 2007.

[194] Lucas Lehnert and Michael L Littman. Transfer with model features in reinforcement learning. *arXiv preprint arXiv:1807.01736*, 2018.

[195] Lucas Lehnert and Michael L Littman. Successor features support model-based and model-free reinforcement learning. *arXiv preprint arXiv:1901.11437*, 2019.

[196] Lucas Lehnert, Romain Laroche, and Harm van Seijen. On value function representation of long horizon problems. In *Proceedings of the AAAI Conference on Artificial Intelligence*, 2018.

[197] Jan Leike. *Nonparametric General Reinforcement Learning*. PhD thesis, Australian National University, 2016.

[198] Timothée Lesort, Natalia Díaz-Rodríguez, Jean-Franois Goudou, and David Filliat. State representation learning for control: An overview. *Neural Networks*, 108:379–392, 2018.

[199] Sergey Levine, Chelsea Finn, Trevor Darrell, and Pieter Abbeel. End-to-end training of deep visuomotor policies. *Journal of Machine Learning Research*, 17(1):1334–1373, 2016.

[200] Andrew Levy, George Konidaris, Robert Platt, and Kate Saenko. Learning multi-level hierarchies with hindsight. In *Proceedings of the International Conference on Learning Representations*, 2019.

[201] Richard L. Lewis, Andrew Howes, and Satinder Singh. Computational rationality: Linking mechanism and behavior through bounded utility maximization. *Topics in cognitive science*, 6(2):279–311, 2014.

[202] Lihong Li. *A unifying framework for computational reinforcement learning theory*. PhD thesis, Rutgers University, 2009.

[203] Lihong Li, Thomas J. Walsh, and Michael L. Littman. Towards a unified theory of state abstraction for MDPs. In *Proceedings of the International Symposium on Artificial Intelligence and Mathematics*, 2006.

[204] Lihong Li, Michael L. Littman, Thomas J. Walsh, and Alexander L. Strehl. Knows what it knows: A framework for self-aware learning. *Machine Learning*, 82(3):399–443, 2011.

[205] Zhuoru Li, Akshay Narayan, and Tze-Yun Leong. An efficient approach to model-based hierarchical reinforcement learning. In *Proceedings of the AAAI Conference on Artificial Intelligence*, 2017.

[206] Yitao Liang, Marlos C. Machado, Erik Talvitie, and Michael Bowling. State of the art control of atari games using shallow reinforcement learning. In *Proceedings of the International Conference on Autonomous Agents and Multiagent Systems*, 2016.

[207] Long-Ji Lin. Self-improving reactive agents based on reinforcement learning, planning and teaching. *Machine learning*, 8(3-4):293–321, 1992.

[208] Michael L. Littman and Csaba Szepesvári. A generalized reinforcement-learning model: Convergence and applications. In *Proceedings of the International Conference on Machine Learning*, 1996.

[209] Michael L. Littman, Thomas L. Dean, and Leslie Pack Kaelbling. On the complexity of solving Markov decision problems. In *Proceedings of the International Joint Conference on Artificial Intelligence*, 1995.

[210] James MacGlashan. *Multi-source option-based policy transfer*. PhD thesis, University of Maryland, Baltimore County, 2013.

[211] Marios C. Machado, Marc G. Bellemare, and Michael Bowling. A Laplacian framework for option discovery in reinforcement learning. In *Proceedings of the International Conference on Machine Learning*, 2017.

[212] Marlos C. Machado, Clemens Rosenbaum, Xiaoxiao Guo, Miao Liu, Gerald Tesauro, and Murray Campbell. Eigenoption discovery through the deep successor representation. In *Proceedings of the International Conference on Learning Representations*, 2017.

[213] Marlos C. Machado, Clemens Rosenbaum, Xiaoxiao Guo, Miao Liu, Gerald Tesauro, and Murray Campbell. Eigenoption Discovery through the Deep Successor Representation. In *Proceedings of the International Conference on Learning Representations*, 2018.

[214] Sridhar Mahadevan and Mauro Maggioni. Value function approximation with diffusion wavelets and Laplacian eigenfunctions. In *Advances in Neural Information Processing Systems*, 2006.

[215] Sridhar Mahadevan and Mauro Maggioni. Proto-value functions: A Laplacian framework for learning representation and control in Markov decision processes. *Journal of Machine Learning Research*, 8(Oct):2169–2231, 2007.

[216] Odalric-Ambrym Maillard, Phuong Nguyen, Ronald Ortner, and Daniil Ryabko. Optimal regret bounds for selecting the state representation in reinforcement learning. In *Proceedings of the International Conference on Machine Learning*, 2013.

[217] Sultan Javed Majeed and Marcus Hutter. Performance guarantees for homomorphisms beyond Markov decision processes. In *Proceedings of the AAAI Conference on Artificial Intelligence*, 2019.

[218] Sultan Javed Majeed and Marcus Hutter. Performance guarantees for homomorphisms beyond Markov decision processes. In *Proceedings of the AAAI Conference on Artificial Intelligence*, 2019.

[219] Travis Mandel, Yun-En Liu, Emma Brunskill, and Zoran Popovic. Efficient Bayesian clustering for reinforcement learning. In *Proceedings of the International Joint Conference on Artificial Intelligence*, 2016.

[220] Daniel J. Mankowitz, Timothy A. Mann, and Shie Mannor. Time-regularized interrupting options. In *Proceedings of the International Conference on Machine Learning*, 2014.

[221] Daniel J. Mankowitz, Timothy A. Mann, and Shie Mannor. Adaptive skills adaptive partitions (asap). In *Advances in Neural Information Processing Systems*, 2016.

[222] Timothy A. Mann and Shie Mannor. Scaling up approximate value iteration with options: Better policies with fewer iterations. In *Proceedings of the International Conference on Machine Learning*, 2014.

[223] Timothy A. Mann, Shie Mannor, and Doina Precup. Approximate value iteration with temporally extended actions. *Journal of Artificial Intelligence Research*, 2015.

[224] Shie Mannor, Ishai Menache, Amit Hoze, and Uri Klein. Dynamic abstraction in reinforcement learning via clustering. In *Proceedings of the International Conference on Machine Learning*, 2004.

[225] Andrew McCallum. *Reinforcement learning with selective perception and hidden state*. PhD thesis, University of Rochester, 1995.

[226] John McCarthy, Marvin L. Minsky, Nathaniel Rochester, and Claude E. Shannon. A proposal for the Dartmouth summer research project on Artificial Intelligence, August 31, 1955. *AI magazine*, 27(4):12–12, 2006.

[227] Amy McGovern. acQuire-macros: An algorithm for automatically learning macro-actions. In *NeurIPS Workshop on Abstraction and Hierarchy in Reinforcement Learning*, 1998.

[228] Amy McGovern and Andrew G. Barto. Automatic discovery of subgoals in reinforcement learning using diverse density. In *Proceedings of the International Conference on Machine Learning*, 2001.

[229] Amy McGovern, Richard S. Sutton, and Andrew H. Fagg. Roles of macro-actions in accelerating reinforcement learning. In *Proceedings of the Grace Hopper Celebration of Women in Computing*, 1997.

[230] Neville Mehta, Mike Wynkoop, Soumya Ray, Prasad Tadepalli, and Thomas G. Dietterich. Automatic induction of MAXQ hierarchies. In *NeurIPS Workshop on Hierarchical Organization of Behavior*, 2007.

[231] Neville Mehta, Sriraam Natarajan, Prasad Tadepalli, and Alan Fern. Transfer in variable-reward hierarchical reinforcement learning. *Machine Learning*, 73(3):289, 2008.

[232] Neville Mehta, Soumya Ray, Prasad Tadepalli, and Thomas G. Dietterich. Automatic discovery and transfer of MAXQ hierarchies. In *Proceedings of the International Conference on Machine Learning*, 2008.

[233] Neville Mehta, Soumya Ray, Prasad Tadepalli, and Thomas G. Dietterich. Automatic Discovery and Transfer of Task Hierarchies in Reinforcement Learning. *AI Magazine*, 32(1):35, 2011.

[234] Francisco S. Melo, Sean P. Meyn, and M. Isabel Ribeiro. An analysis of reinforcement learning with function approximation. In *Proceedings of the International Conference on Machine Learning*, 2008.

[235] Ishai Menache, Shie Mannor, and Nahum Shimkin. Q-cut - dynamic discovery of sub-goals in reinforcement learning. In *Proceedings of the European Conference on Machine Learning*, 2002.

[236] Jacob Menashe and Peter Stone. State abstraction synthesis for discrete models of continuous domains. In *Proceedings of the AAAI Spring Symposium Series*, 2018.

[237] Dipendra Misra, Mikael Henaff, Akshay Krishnamurthy, and John Langford. Kinematic state abstraction and provably efficient rich-observation reinforcement learning. *arXiv preprint arXiv:1911.05815*, 2019.

[238] Volodymyr Mnih, Koray Kavukcuoglu, David Silver, Andrei A. Rusu, Joel Veness, Marc G. Bellemare, Alex Graves, Martin Riedmiller, Andreas K. Fidjeland, Georg Ostrovski, Stig Petersen, Charles Beattie, Amir Sadik, Ioannis Antonoglou, Helen King, Dharshan Kumaran, Daan Wierstra, Shane Legg, and Demis Hassabis. Human-level control through deep reinforcement learning. *Nature*, 518(7540):529, 2015.

[239] Volodymyr Mnih, Adrià Puigdomènech Badia, Mehdi Mirza, Alex Graves, Timothy P. Lillicrap, Tim Harley, David Silver, and Koray Kavukcuoglu. Asynchronous Methods for Deep Reinforcement Learning. In *Proceedings of the International Conference on Machine Learning*, 2016.

[240] Andrew W. Moore. The parti-game algorithm for variable resolution reinforcement learning in multidimensional state-spaces. In *Advances in Neural Information Processing Systems*, 1994.

[241] Damon Mosk-Aoyama. Maximum algebraic connectivity augmentation is NP-hard. *Operations Research Letters*, 36(6):677–679, 2008.

[242] Jonathan Mugan and Benjamin Kuipers. Towards the application of reinforcement learning to undirected developmental learning. In *Proceedings of the International Conference on Epigenetic Robots*, 2008.

[243] Jonathan Mugan and Benjamin Kuipers. Autonomously learning an action hierarchy using a learned qualitative state representation. In *Proceedings of the International Joint Conference on Artificial Intelligence*, 2009.

[244] Jonathan Mugan and Benjamin Kuipers. Autonomous learning of high-level states and actions in continuous environments. *IEEE Transactions on Autonomous Mental Development*, 4(1):70–86, 2011.

[245] Brendan Mumey and Tomáš Gedeon. Optimal mutual information quantization is NP-complete. In *Neural Information Coding*, 2003.

[246] Ofir Nachum, Shixiang Shane Gu, Honglak Lee, and Sergey Levine. Data-efficient hierarchical reinforcement learning. In *Advances in Neural Information Processing Systems*, 2018.

[247] Ofir Nachum, Shixiang Gu, Honglak Lee, and Sergey Levine. Near-optimal representation learning for hierarchical reinforcement learning. In *Proceedings of the International Conference on Learning Representations*, 2019.

[248] Allen Newell, John Clark Shaw, and Herbert A. Simon. Empirical explorations of the logic theory machine: a case study in heuristic. In *Papers presented at the Western Joint Computer Conference: Techniques for Reliability*.

[249] Andrew Y. Ng and Stuart Russell. Algorithms for inverse reinforcement learning. In *Proceedings of the International Conference on Machine Learning*, 2000.

[250] Sam Nicol and Iadine Chadès. Which states matter? an application of an intelligent discretization method to solve a continuous pomdp in conservation biology. *PloS one*, 7(2), 2012.

[251] Maillard Odalric-Ambrym, Phuong Nguyen, Ronald Ortner, and Daniil Ryabko. Optimal regret bounds for selecting the state representation in reinforcement learning. In *Proceedings of the International Conference on Machine Learning*, 2013.

[252] Junhyuk Oh, Satinder Singh, and Honglak Lee. Value prediction network. In *Advances in Neural Information Processing Systems*, 2017.

[253] Ronald Ortner, Odalric-Ambrym Maillard, and Daniil Ryabko. Selecting near-optimal approximate state representations in reinforcement learning. In *Proceedings of the International Conference on Algorithmic Learning Theory*, 2014.

[254] Ronald Ortner, Matteo Pirotta, Alessandro Lazaric, Ronan Fruit, and Odalric-Ambrym Maillard. Regret bounds for learning state representations in reinforcement learning. In *Advances in Neural Information Processing Systems*, 2019.

[255] Ian Osband, Daniel Russo, and Benjamin Van Roy. (more) efficient reinforcement learning via posterior sampling. In *Advances in Neural Information Processing Systems*, 2013.

[256] Rina Panigrahy and Sundar Vishwanathan. An $O(\log^* n)$ approximation algorithm for the asymmetric p-center problem. *Journal of Algorithms*, 27(2):259–268, 1998.

[257] Christos H. Papadimitriou and John N. Tsitsiklis. The complexity of Markov decision processes. *Mathematics of Operations Research*, 12(3):441–450, 1987.

[258] Ronald Parr. *Hierarchical Control and Learning for Markov Decision Processes*. PhD thesis, University of California, Berkeley, 1998.

[259] Ronald Parr and Stuart Russell. Reinforcement learning with hierarchies of machines. In *Advances in Neural Information Processing Systems*, 1998.

[260] Ronald Parr, Lihong Li, Gavin Taylor, Christopher Painter-Wakefield, and Michael L. Littman. An analysis of linear models, linear value-function approximation, and feature selection for reinforcement learning. In *Proceedings of the International Conference on Machine learning*. ACM, 2008.

[261] Marc Pickett and Andrew G. Barto. PolicyBlocks: An algorithm for creating useful macro-actions in reinforcement learning. In *Proceedings of the International Conference on Machine Learning*, 2002.

[262] Doina Precup. *Temporal abstraction in reinforcement learning*. PhD thesis, University of Massachusetts Amherst, 2001.

[263] Doina Precup and Richard S. Sutton. Multi-time models for reinforcement learning. In *ICML Workshop on Modelling in Reinforcement Learning*, 1997.

[264] Doina Precup and Richard S. Sutton. Multi-time models for temporally abstract planning. In *Advances in Neural Information Processing Systems*, 1998.

[265] Jefferson Provost, Benjamin Kuipers, and Risto Miikkulainen. Developing navigation behavior through self-organizing distinctive-state abstraction. *Connection Science*, 18(2):159–172, 2006.

[266] Martin L. Puterman. *Markov decision processes: discrete stochastic dynamic programming*. John Wiley & Sons, 2014.

[267] Lorna C. Quandt, Yune Sang Lee, and Anjan Chatterjee. Neural bases of action abstraction. *Biological psychology*, 129:314, 2017.

[268] Pravesh Ranchod, Benjamin Rosman, and George Konidaris. Nonparametric Bayesian reward segmentation for skill discovery using inverse reinforcement learning. In *Proceedings of the Conference on Intelligent Robots and Systems*, 2015.

[269] Balaraman Ravindran. *SMDP homomorphisms: An algebraic approach to abstraction in semi Markov decision processes*. PhD thesis, University of Massachusetts Amherst, 2003.

[270] Balaraman Ravindran and Andrew G. Barto. Model minimization in hierarchical reinforcement learning. In *Proceedings of the International Symposium on Abstraction, Reformulation, and Approximation*, 2002.

[271] Balaraman Ravindran and Andrew G. Barto. Relativized options: Choosing the right transformation. In *Proceedings of the International Conference on Machine Learning*, 2003.

[272] Balaraman Ravindran and Andrew G. Barto. SMDP homomorphisms: An algebraic approach to abstraction in semi-Markov decision processes. In *Proceedings of the International Joint Conference on Artificial Intelligence*, 2003.

[273] Balaraman Ravindran and Andrew G. Barto. Approximate homomorphisms: A framework for non-exact minimization in Markov decision processes. In *Proceedings of the International Conference on Knowledge Based Computer Systems*, 2004.

[274] Andrew Reibman, Roger Smith, and Kishor Trivedi. Markov and Markov reward model transient analysis: An overview of numerical approaches. *European Journal of Operational Research*, 40(2):257–267, 1989.

[275] Matthew Riemer, Miao Liu, and Gerald Tesauro. Learning abstract options. In *Advances in Neural Information Processing Systems*, 2018.

[276] Gavin A. Rummery and Mahesan Niranjan. *On-line Q-learning using connectionist systems*, volume 37. University of Cambridge, Department of Engineering Cambridge, UK, 1994.

[277] Stuart Russell and Peter Norvig. *Artificial Intelligence: A Modern Approach*. Prentice Hall, 2009.

[278] Stuart Russell and Devika Subramanian. Provably Bounded-Optimal Agents. *Journal of Artificial Intelligence Research*, 2:575–609, 1995. ISSN 1076-9757.

[279] Earl D. Sacerdoti. Planning in a hierarchy of abstraction spaces. *Artificial Intelligence*, 5(2):115 – 135, 1974.

[280] Jürgen Schmidhuber. Reinforcement learning in Markovian and non-Markovian environments. In *Advances in Neural Information Processing Systems*, 1991.

[281] Wolfram Schultz, Peter Dayan, and P. Read Montague. A neural substrate of prediction and reward. *Science*, 275(5306):1593–1599, 1997.

[282] Gideon Schwarz. Estimating the dimension of a model. *The Annals of Statistics*, 6(2): 461–464, 1978.

[283] Shai Shalev-Shwartz and Shai Ben-David. *Understanding Machine Learning: From Theory to Algorithms*. Cambridge University Press, 2014.

[284] Claude E. Shannon. A mathematical theory of communication. *The Bell System Technical Journal*, 27:379–423, 1948.

[285] David Silver and Kamil Ciosek. Compositional planning using optimal option models. In *Proceedings of the International Conference on Machine Learning*, 2012.

[286] Herbert A. Simon. *Models of man; social and rational.* Wiley, 1957.

[287] Herbert A. Simon. Theories of bounded rationality. *Decision and organization*, 1(1): 161–176, 1972.

[288] Chris R. Sims. Rate–distortion theory and human perception. *Cognition*, 152:181–198, 2016.

[289] Chris R. Sims. Efficient coding explains the universal law of generalization in human perception. *Science*, 360(6389):652–656, 2018.

[290] Özgür Şimşek and Andrew G. Barto. Using relative novelty to identify useful temporal abstractions in reinforcement learning. In *Proceedings of the International Conference on Machine Learning*. ACM, 2004.

[291] Özgür Şimşek and Andrew G. Barto. Skill characterization based on betweenness. In *Advances in Neural Information Processing Systems*, 2009.

[292] Özgür Şimşek, Alicia P. Wolfe, and Andrew G. Barto. Identifying useful subgoals in reinforcement learning by local graph partitioning. In *Proceedings of the International Conference on Machine Learning*, 2005.

[293] Satinder Singh. Scaling reinforcement learning algorithms by learning variable temporal resolution models. In *Proceedings of the International Machine Learning Conference*, 1992.

[294] Satinder Singh, Tommi Jaakkola, and Michael I. Jordan. Reinforcement learning with soft state aggregation. In *Advances in Neural Information Processing Systems*, 1995.

[295] Satinder Singh, Andrew G. Barto, and Nuttapong Chentanez. Intrinsically motivated reinforcement learning. In *Advances in Neural Information Processing Systems*, 2005.

[296] Matthijs Snel and Shimon Whiteson. Multi-task reinforcement learning: Shaping and feature selection. In *Proceedings of the European Workshop on Reinforcement Learning*, 2011.

[297] Alec Solway, Carlos Diuk, Natalia Córdova, Debbie Yee, Andrew G. Barto, Yael Niv, and Matthew Botvinick. Optimal behavioral hierarchy. *PLoS computational biology*, 10(8):e1003779, 2014.

[298] Martin Stolle and Doina Precup. Learning options in reinforcement learning. In *Proceedings of the International Symposium on Abstraction, Reformulation, and Approximation*, 2002.

[299] Alexander L. Strehl and Michael L. Littman. A theoretical analysis of model-based interval estimation. In *Proceedings of the International Conference on Machine Learning*, 2005.

[300] Alexander L. Strehl, Lihong Li, Eric Wiewiora, John Langford, and Michael L. Littman. PAC model-free reinforcement learning. In *Proceedings of the International Conference on Machine Learning*, 2006.

[301] Alexander L. Strehl, Lihong Li, and Michael L. Littman. Reinforcement learning in finite MDPs: PAC analysis. *Journal of Machine Learning Research*, 10:2413–2444, 2009.

[302] DJ Strouse and David J. Schwab. The deterministic information bottleneck. *Neural Computation*, 29(6):1611–1630, 2017.

[303] Wen Sun, Nan Jiang, Akshay Krishnamurthy, Alekh Agarwal, and John Langford. Model-based RL in contextual decision processes: PAC bounds and exponential improvements over model-free approaches. In *Proceedings of the Conference on Learning Theory*, 2019.

[304] Tobias Sutter, David Sutter, Peyman Mohajerin Esfahani, and John Lygeros. Efficient approximation of channel capacities. *IEEE Transactions on Information Theory*, 61: 1649–1666, 2015.

[305] Richard S. Sutton. Learning to predict by the methods of temporal differences. *Machine Learning*, 3(1):9–44, 1988.

[306] Richard S. Sutton. Introduction: The challenge of reinforcement learning. In *Reinforcement Learning*, pages 1–3. Springer, 1992.

[307] Richard S. Sutton. Generalization in reinforcement learning: Successful examples using sparse coarse coding. In *Advances in Neural Information Processing Systems*, 1996.

[308] Richard S. Sutton. The reward hypothesis, 2004. URL http://incompleteideas.net/rlai.cs.ualberta.ca/RLAI/rewardhypothesis.html.

[309] Richard S. Sutton and Andrew G. Barto. *Reinforcement Learning: An Introduction*. MIT Press, 1998.

[310] Richard S. Sutton and Andrew G. Barto. *Reinforcement Learning: An Introduction*. MIT press, 2018.

[311] Richard S. Sutton, Doina Precup, and Satinder Singh. Between MDPs and semi-MDPs: A framework for temporal abstraction in reinforcement learning. *Artificial Intelligence*, 1999.

[312] Richard S. Sutton, David A. McAllester, Satinder Singh, and Yishay Mansour. Policy gradient methods for reinforcement learning with function approximation. In *Advances in Neural Information Processing Systems*, 2000.

[313] Adrien Ali Taïga, Aaron Courville, and Marc G. Bellemare. Approximate exploration through state abstraction. *arXiv preprint arXiv:1808.09819*, 2018.

[314] Erik Talvitie. Self-correcting models for model-based reinforcement learning. In *Proceedings of the AAAI Conference on Artificial Intelligence*, 2017.

[315] Jonathan Taylor, Doina Precup, and Prakash Panagaden. Bounding performance loss in approximate MDP homomorphisms. In *Advances in Neural Information Processing Systems*, 2008.

[316] Sebastian Thrun. Is learning the n-th thing any easier than learning the first? In *Advances in Neural Information Processing Systems*, 1996.

[317] Sebastian Thrun and Anton Schwartz. Finding structure in reinforcement learning. In *Advances in Neural Information Processing Systems*, 1995.

[318] Naftali Tishby, Fernando C. Pereira, and William Bialek. The information bottleneck method. In *Proceedings of the Allerton Conference on Communication, Control, and Computing*, 1999.

[319] Saket Tiwari and Philip S. Thomas. Natural option critic. In *Proceedings of the AAAI Conference on Artificial Intelligence*, 2019.

[320] Anurag Ajay Josh Tenenbaum Leslie Pack Kaelbling Tom Silver, Rohan Chitnis. Learning skill hierarchies from predicate descriptions and self-supervision. In *AAAI Workshop on Generalization in Planning*, 2020.

[321] Nicholay Topin, Nicholas Haltmeyer, Shawn Squire, John Winder, Marie desJardins, and James MacGlashan. Portable option discovery for automated learning transfer in object-oriented Markov decision processes. In *Proceedings of the International Joint Conference on Artificial Intelligence*, 2015.

[322] Paul Tseng. Solving H-horizon, stationary Markov decision problems in time proportional to $\log(H)$. *Operations Research Letters*, 9(5):287–297, 1990.

[323] William T.B. Uther and Manuela M. Veloso. Tree based discretization for continuous state space reinforcement learning. In *Proceedings of the AAAI Conference on Artificial Intelligence*, 1998.

[324] Leslie G. Valiant. A theory of the learnable. *Communications of the ACM*, 27(11): 1134–1142, 1984.

[325] Hado van Hasselt. Double Q-learning. In *Advances in Neural Information Processing Systems*, 2010.

[326] Benjamin Van Niekerk, Steven James, Adam Earle, and Benjamin Rosman. Composing value functions in reinforcement learning. In *Proceedings of the International Conference on Machine Learning*, 2019.

[327] Benjamin Van Roy. Performance loss bounds for approximate value iteration with state aggregation. *Mathematics of Operations Research*, 31(2):234–244, 2006.

[328] Harm van Seijen and Richard S. Sutton. A deeper look at planning as learning from replay. In *Proceedings of the International Conference on Machine learning*, 2015.

[329] Harm van Seijen, Shimon Whiteson, and Leon Kester. Efficient abstraction selection in reinforcement learning. *Computational Intelligence*, 30(4):657–699, 2014.

[330] Vladimir N. Vapnik and Aleksei Y. Chervonenkis. On the uniform convergence of relative frequencies of events to their probabilities. *Theory of Probability & Its Applications*, 16(2):264–280, 1971.

[331] Joel Veness, Kee Siong Ng, Marcus Hutter, William Uther, and David Silver. A Monte-Carlo AIXI approximation. *Journal of Artificial Intelligence Research*, 40(1):95–142, 2011.

[332] Alexander Sasha Vezhnevets, Simon Osindero, Tom Schaul, Nicolas Heess, Max Jaderberg, David Silver, and Koray Kavukcuoglu. Feudal networks for hierarchical reinforcement learning. In *Proceedings of the International Conference on Machine Learning*, 2017.

[333] Thomas J. Walsh, Lihong Li, and Michael L. Littman. Transferring state abstractions between MDPs. In *ICML Workshop on Structural Knowledge Transfer for Machine Learning*, 2006.

[334] Thomas J. Walsh, István Szita, Carlos Diuk, and Michael L. Littman. Exploring compact reinforcement-learning representations with linear regression. In *Proceedings of the Conference on Uncertainty in Artificial Intelligence*, 2009.

[335] Thomas J. Walsh, Sergiu Goschin, and Michael L. Littman. Integrating sample-based planning and model-based reinforcement learning. In *Proceedings of the AAAI Conference on Artificial Intelligence*, 2010.

[336] Thomas J. Walsh, Daniel K. Hewlett, and Clayton T. Morrison. Blending autonomous exploration and apprenticeship learning. In *Advances in Neural Information Processing Systems*, 2011.

[337] Christopher J.C.H. Watkins and Peter Dayan. q-learning. *Machine learning*, 8(3-4): 279–292, 1992.

[338] Denise M. Werchan, Anne G.E. Collins, Michael J. Frank, and Dima Amso. 8-month-old infants spontaneously learn and generalize hierarchical rules. *Psychological science*, 26(6):805–815, 2015.

[339] Shimon Whiteson, Matthew E. Taylor, and Peter Stone. Adaptive tile coding for value function approximation. Technical report, University of Texas at Austin, 2007.

[340] Ward Whitt. Approximations of dynamic programs, i. *Mathematics of Operations Research*, 3(3):231–243, 1978.

[341] Ward Whitt. Approximations of dynamic programs, ii. *Mathematics of Operations Research*, 4(2):179–185, 1979.

[342] Marco Wiering and Jürgen Schmidhuber. HQ-learning. *Adaptive Behavior*, 6(2):219–246, 1997.

[343] Ronald J. Williams and Leemon C. Baird. Tight performance bounds on greedy policies based on imperfect value functions. Technical report, College of Computer Science, Northeastern University, 1993.

[344] Aaron Wilson, Alan Fern, Soumya Ray, and Prasad Tadepalli. Multi-task reinforcement learning: a hierarchical Bayesian approach. In *Proceedings of the International Conference on Machine learning*, 2007.

[345] John Winder, Stephanie Milani, Matthew Landen, Erebus Oh, Shane Parr, Shawn Squire, Marie desJardins, and Cynthia Matuszek. Planning with abstract learned models while learning transferable subtasks. In *Proceedings of the AAAI Conference on Artificial Intelligence*, 2020.

[346] David H. Wolpert. The lack of a priori distinctions between learning algorithms. *Neural computation*, 8(7), 1996.

[347] Shangtong Zhang and Shimon Whiteson. DAC: The double actor-critic architecture for learning options. In *Advances in Neural Information Processing Systems*, 2019.