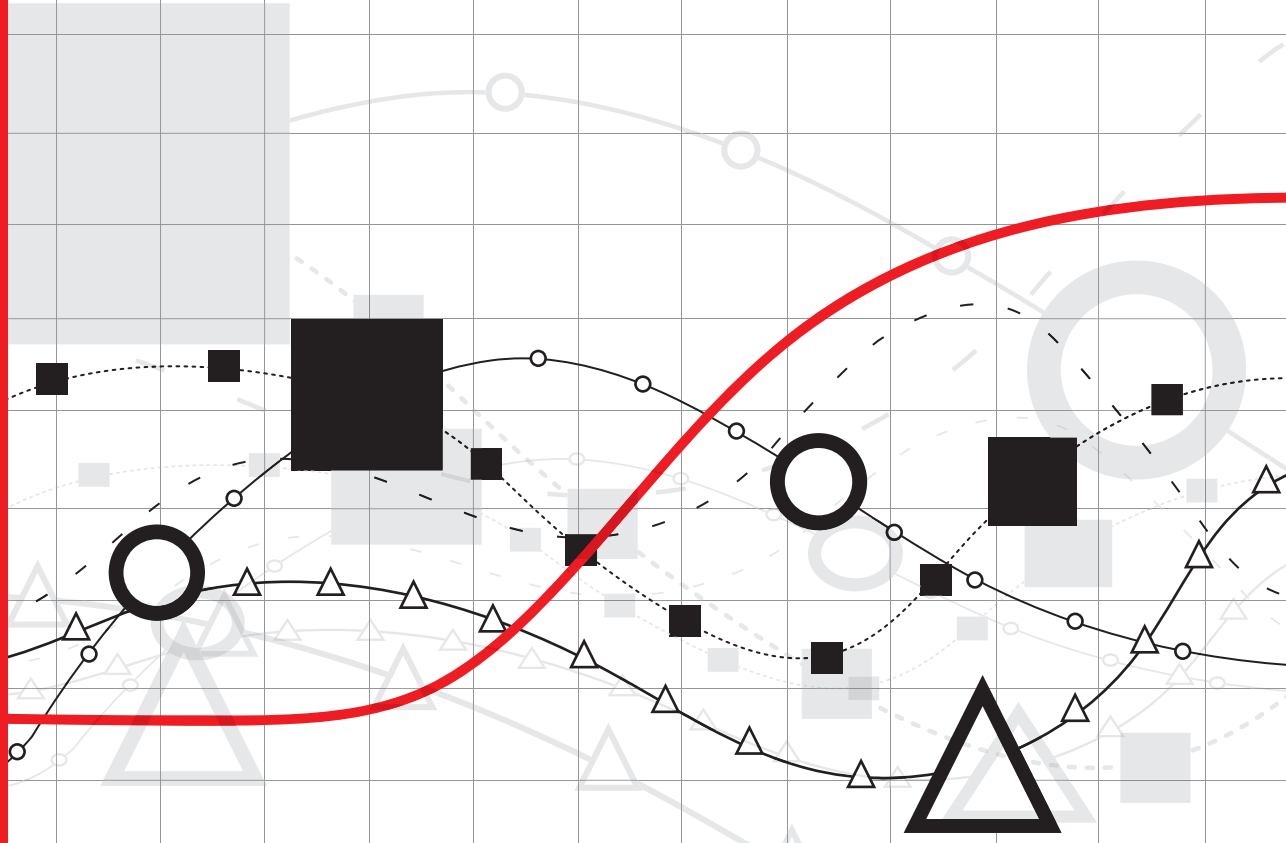
Robust and Reliable Decision-Making Under Uncertainty



Marnix Suilen

Institute for Computing and Information Sciences

RADBOUD UNIVERSITY PRESS

Radboud Dissertation Series

Robust and Reliable Decision-Making Under Uncertainty

Marnix Suilen

Robust and Reliable Decision-Making Under Uncertainty
Marnix Robert Suilen

Radboud Dissertation Series
ISSN: 2950-2772 (Online); 2950-2780 (Print)

Published by RADBOUD UNIVERSITY PRESS
Postbus 9100, 6500 HA Nijmegen, The Netherlands
www.radbouduniversitypress.nl

Cover: Proefschrift AIO | Guus Gijben
Printing: DPN Rikken/Pumbo

ISBN: 9789465150970
DOI: 10.54195/9789465150970
Free download at: https://doi.org/10.54195/9789465150970

RADBOUD
UNIVERSITY
PRESS

Robust and Reliable Decision-Making Under Uncertainty

Proefschrift ter verkrijging van de graad van doctor
aan de Radboud Universiteit Nijmegen
op gezag van de rector magnificus prof. dr. J.M. Sanders,
volgens besluit van het college voor promoties
in het openbaar te verdedigen op

maandag 29 september 2025
om 14:30 uur precies

door

Marnix Robert Suilen

geboren op 2 februari 1996
te Roermond

Promotoren:
Prof. dr. N.H. Jansen (Ruhr-Universität Bochum, Duitsland)
Prof. dr. F.W. Vaandrager

Manuscriptcommissie:
Prof. dr. M.I.A. Stoelinga
Prof. dr. N. Hawes (University of Oxford, Verenigd Koninkrijk)
Prof. dr. ir. J.P. Katoen (RWTH Aachen, Duitsland)
Prof. dr. J. Křetínský (Masarykova Univerzita, Tsjechië)
Prof. dr. M.T.J. Spaan (Technische Universiteit Delft)

Contents

Summary

Sequential decision-making is a fundamental problem encountered in many application areas such as robotics, finance, and healthcare. Since many of these decision-making problems are based on data, they are inherently affected by uncertainty that arises from insufficient, incomplete, or incorrect data. With the rise of *artificial intelligence* (AI), the reliance on data to solve decision-making problems is greater than ever, as also evidenced by the success of *reinforcement learning* (RL). As a consequence, uncertainty in sequential decision-making is unavoidable. This uncertainty must be accounted for to ensure solutions to these decision-making problems are *robust* and *reliable* against incomplete information, misspecification, or adversarial perturbation while maintaining performance.

Sequential decision-making under uncertainty is usually formalized through *Markov decision processes* (MDPs), and various extensions such as *robust MDPs*, that account for *model uncertainty*, or *partially observable MDPs* (POMDPs), which account for *state uncertainty*. This thesis presents several contributions to increase the robustness and reliability of solutions to sequential decision-making problems formulated through these models.

A Tutorial on Robust Markov Decision Processes. Our first contribution is a short tutorial on robust MDPs. Starting with an introduction to the general model, we discuss its semantics and structural assumptions such as *rectangularity*. We then explain how standard dynamic programming for MDPs can be extended to *robust dynamic programming*. Finally, we discuss some commonly used instances of robust MDPs, such as interval MDPs (IMDPs), L_1-MDPs, and multi-environment MDPs (MEMDPs). This tutorial is meant for readers from the formal methods and AI communities who are familiar with MDPs.

Finite-Memory Policies for Robust POMDPs. For our second contribution, we present two novel algorithms to compute finite-memory policies in *robust POMDPs*. Robust POMDPs formalize decision-making problems that exhibit both partial observability (*i.e.*, state uncertainty) as well as imprecise probabilities (*i.e.*, model uncertainty). Policies for robust POMDPs must be robust against both to ensure acceptable performance. To that end, we encode the problem of finding an optimal robust finite-memory policy as a semi-infinite, nonlinear optimization problem. Methods from the robust optimization literature are used to derive two algorithms that make this optimization finite and iteratively convexify or linearize it into a convex optimization problem ready to be solved.

Robust Anytime Learning of Markov Decision Processes. For our third contribution, we consider a robust RL setting where the underlying environment may

change over time. Our goal is to learn a robust MDP that is adaptable to such changes. To that end, we present a novel model-based RL algorithm that employs *linearly updating intervals*, a Bayesian approach to updating probability intervals in the presence of new, possibly inconsistent, data. Together with a sliding window approach that discards old data and integration with robust dynamic programming, we can learn robust policies that are conservative yet adaptable when the underlying environment changes.

Extending the Scope of Reliable Offline RL. Finally, we present two contributions to the offline RL problem of *safe policy improvement* (SPI). SPI concerns the computation of a new policy that outperforms a behavior policy with a reliability guarantee from a fixed dataset collected by the behavior policy. Together, our two contributions extend the scope of SPI algorithms. For our first contribution, we extend existing SPI methods to work in partially observable environments and with finite-memory policies. For our second contribution to SPI, we introduce a novel approach that significantly reduces the amount of data required to establish the same improvement guarantee within the SPI algorithms, thus making the most out of the available data.

All these contributions add to the robustness and reliability of sequential decision-making problems in their own way and highlight the many facets involved. Yet, they all share the same throughline: robustness and reliability can be ensured using model-based approaches that explicitly account for uncertainty.

Samenvatting

Sequentiële besluitvorming is een fundamenteel probleem dat zich voordoet in veel toepassingsgebieden, zoals robotica, financiën en gezondheidszorg. Omdat veel van deze besluitvormingsproblemen gebaseerd zijn op data, worden ze inherent beïnvloed door onzekerheid die voortkomt uit onvoldoende, onvolledige of onjuiste data. Met de opkomst van *kunstmatige intelligentie* (*artificial intelligence*; AI) is de afhankelijkheid van data om besluitvormingsproblemen op te lossen groter dan ooit, zoals ook blijkt uit het succes van *reinforcement learning* (*versterkend leren*; RL). Als gevolg hiervan is onzekerheid in sequentiële besluitvorming onvermijdelijk. Met deze onzekerheid moet rekening worden gehouden om ervoor te zorgen dat oplossingen voor deze besluitvormingsproblemen *robuust* en *betrouwbaar* zijn, ondanks onvolledige informatie, misspecificatie of vijandige verstoringen, en tegelijkertijd de prestaties behouden.

Sequentiële besluitvorming onder onzekerheid wordt meestal geformaliseerd via *Markov decision processes* (MDPs) en diverse uitbreidingen, zoals *robust MDPs*, die rekening houden met *modelonzekerheid*, of *partially observable MDPs* (POMDPs), die rekening houden met *toestandsonzekerheid*. Dit proefschrift presenteert verschillende bijdragen om de robuustheid en betrouwbaarheid oplossingen voor sequentiële besluitvormingsproblemen die via deze modellen worden geformuleerd te vergroten.

A Tutorial on Robust Markov Decision Processes. Onze eerste bijdrage is een korte tutorial over robust MDPs. Beginnend met een inleiding tot het algemene model, bespreken we de semantiek en structurele aannames, zoals *rectangularity*. Vervolgens leggen we uit hoe standaard *dynamic programming* technieken voor MDPs kunnen worden uitgebreid naar *robust dynamic programming*. Tot slot bespreken we enkele veelgebruikte voorbeelden van robuste MDPs, zoals interval MDPs (IMDPs), L_1-MDPs en multi-environment MDPs (MEMDPs). Deze tutorial is bedoeld voor lezers uit de formele methoden- en AI-gemeenschap die bekend zijn met MDPs.

Finite-Memory Policies for Robust POMDPs. Voor onze tweede bijdrage presenteren we twee nieuwe algoritmen voor het berekenen van *finite-memory polcies* in *robust POMDPs*. Robust POMDPs formaliseren besluitvormingsproblemen die zowel gedeeltelijke waarneembaarheid zijn (*i.e.*, met toestandsonzekerheid) als waarschijnlijkheidsintervallen (*i.e.*, modelonzekerheid) vertonen. Policies voor robust POMDPs moeten robuust zijn tegen beide vormen van onzekerheid om acceptabele prestaties te garanderen. Daartoe coderen we het probleem van het vinden van een optimale robuuste finite-memory policy als een semi-oneindig, niet-lineair optimalisatieprobleem. Methoden uit de literatuur over robuuste optimalisatie worden

gebruikt om twee algoritmen af te leiden die dit optimalisatieprobleem eindig maken en deze iteratief convex of lineair maken tot een convex optimalisatieprobleem dat klaar is om opgelost te worden.

Robust Anytime Learning of Markov Decision Processes. Voor onze derde bijdrage beschouwen we een robuust RL probleem waarin de onderliggende omgeving naar verloop van tijd kan veranderen. Ons doel is om een robust MDP te leren die aanpasbaar is aan dergelijke veranderingen. We presenteren een nieuw modelgebaseerd RL-algoritme dat gebruikmaakt van *lineair updatende intervallen*, een Bayesiaanse benadering voor het updaten van waarschijnlijkheidsintervallen in de aanwezigheid van nieuwe, mogelijk inconsistente, data. In combinatie met een *sliding window*-benadering, die oude data weggooit, en integratie met robust dynamic programming, kunnen we robuuste policies leren die conservatief maar aanpasbaar zijn wanneer de onderliggende omgeving verandert.

Extending the Scope of Reliable Offline RL. Tot slot presenteren we twee bijdragen aan het offline RL-probleem van *safe policy improvement* (SPI). SPI betreft de berekening van een nieuwe policy die beter presteert dan een *behavior policy* met een betrouwbaarheidsgarantie op basis van een vaste dataset die door het behavior policy is verzameld. Samen breiden onze twee bijdragen de toepasbaarheid van SPI-algoritmen uit. Voor onze eerste bijdrage breiden we bestaande SPI-methoden uit om te werken in gedeeltelijk waarneembare omgevingen en finite-memory policies. Voor onze tweede bijdrage aan SPI introduceren we een nieuwe aanpak die de hoeveelheid data die nodig is om dezelfde verbeteringsgarantie binnen de SPI-algoritmen vast te stellen aanzienlijk vermindert, waardoor de beschikbare data optimaal wordt benut.

Al deze bijdragen dragen op hun eigen manier bij aan de robuustheid en betrouwbaarheid van sequentiële besluitvormingsproblemen en benadrukken de vele facetten die hierbij betrokken zijn. Toch delen ze allemaal dezelfde rode draad: robuustheid en betrouwbaarheid kunnen worden gegarandeerd met behulp van modelgebaseerde methodes die expliciet rekening houden met onzekerheid.

Acknowledgments

This thesis would not exist without many great people.

Years ago, when I needed a topic for my bachelor's thesis, I came across Nils. A then newly hired assistant professor who had topics available about things called "Markov decision processes". I had no clue what these were back then, and little did I know I would end up writing a Ph.D. thesis in which the abbreviation "MDP" is mentioned over one thousand times. Nils, thank you for introducing me to the wonderful world of MDPs and giving me the opportunity to pursue a Ph.D. with you – a career path I had not envisioned for myself at the time until you brought it up. You gave me the freedom to pursue whatever direction I found interesting, but also made sure I would produce some tangible results every now and then. You also taught me that polishing a paper until the last minute before a deadline is always worthwhile, just as being ambitious and submitting our work to highly competitive venues is always worth the shot. The papers on which this thesis is based are a testament to everything I learned from you.

Besides Nils, I was lucky enough to have a second advisor. Frits, even though we never did any research together, you always had some good advice on how to 'sell' results in a presentation whenever needed.

This thesis would not exist in its current form, and I would not be writing these acknowledgments, without the members of the manuscript committee: Nick Hawes, Joost-Pieter Katoen, Jan Křetínský, Matthijs Spaan, and Mariëlle Stoelinga. I am grateful you took the time to read and review this thesis and provided me with some valuable feedback.

None of the work presented in this thesis, nor the other papers published during my Ph.D., would have been possible without so many great collaborators: Thom Badings, Christel Baier, Eline Bovy, Murat Cubuktepe, Clemens Dubslaff, Arnd Hartmanns, Nils Jansen, Sebastian Junges, Ahmadreza Marandi, David Parker, Thiago D. Simão, Ufuk Topcu, Marck van der Vegt, and Patrick Wienhöft. I am happy to have worked with all of you and look forward to our continued collaborations. There are plenty of interesting problems left to explore!

What made my Ph.D. journey particularly enjoyable was being surrounded by such great colleagues. Christoph, Dennis, Eline, Maris, Merlijn, Thiago, Thom, Wietze; thanks to you, there was never a dull moment in Nils' paper factory, where hard work could always be paused in an instant for a brief conversation on the most random topics. Many thanks as well to everyone else in the Department of Software Science for the fun conversations at the coffee meetings, lunches, and borrels.

With Nils moving to Bochum, the AI-FM family expanded with Joshua, Jule, Marcel, Markel, Maxi, Miriam, Öznur, and Verena. It was a pleasure to visit you in Bochum several times, and I look forward to many more trips in the future.

Outside of academia, there are some good friends whom I have known since high school. Bas, Bob, Jules, Niek, and Ruben (a.k.a. de Hutsmutsers), I could always count on you guys to cheer me on when a paper got accepted or rejected, even though I am pretty sure none of you have any idea what my research is about. I would also like to thank Patrick, Perry, and Robin for our weekly Sunday evening conversations, which have been a cornerstone these past years.

To my father, Frits, and brother, Floris, thanks for your unconditional support. I will never forget how the first comment you made when showing you the first paper I wrote was that the letter 'A' was upside down (yes, it is meant to be written as ∀).

To my mother, Nelly, I would have loved for you to see me grow through this entire process. Unfortunately, that opportunity was not given to us, which will always sadden me. I dedicate this thesis to you.

Finally, Iris, thank you for your support, pushing me to actually finish this thesis, listening to me try to explain abstract nonsense to you, and everything else that is *almost-surely* better said in person.

Marnix Suilen
May 21, 2025
Nijmegen, The Netherlands

1

Introduction

From determining the best move to make in a board or video game (Mnih et al., 2015; Silver et al., 2017), to more serious applications such as robot navigation (Klingspor et al., 1997; Thrun et al., 2005), decision support systems (Kochenderfer, 2015), and in areas like finance (Bahrammirzaee, 2010) or healthcare (D'aeth et al., 2023; Jiang et al., 2017), *sequential decision-making* problems are everywhere.

These problems are inherently subject to *uncertainty* that arises from many different sources. For instance, the roll of a die may be a core component of a game, introducing uncertainty about the outcome of a certain play. In robotics, sensor noise or imprecise actuators may introduce uncertainty about where the robot is located. In systems that rely on human interaction, such as decision support systems, variations in the human operator's response time introduce uncertainty on how quickly the system's advice is carried out. Finally, in healthcare, a plethora of underlying reasons influence a patient's recovery and thus introduce uncertainty on the effects of a treatment plan. These are only a few examples of how uncertainty inherently affects sequential decision-making problems.

All such decision-making problems rely on an exact formulation of the outcome of a decision. These formulations may be based on (historical) data or expert opinions. With the rapid development and deployment of artificial intelligence (AI; Russell and Norvig, 2020), decision-making problems have become intertwined with data, as most notably evidenced by the rise of *reinforcement learning* (RL; Sutton and Barto, 1998), and their solutions deployed in the real world. As a consequence, uncertainty in sequential decision-making is unavoidable, raising a whole range of safety concerns that need to be accounted for (Dalrymple et al., 2024). The central question is how to properly account for this uncertainty, in all its facets, such that solutions to these decision-making problems are *robust* and *reliable*.

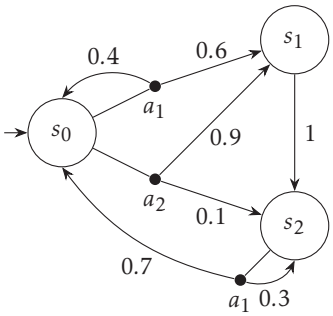


Figure 1.1: An example MDP model. The states are $\{s_0, s_1, s_2\}$, the actions are $\{a_1, a_2\}$, and the transition probabilities are given by the graph. For instance, the probability of arriving in s_1 when taking action a_1 from s_0 is 0.6.

1.1 Decision-Making Under Uncertainty

Given their wide range of applications, it is only natural that sequential decision-making under uncertainty is studied across multiple scientific disciplines, such as operations research, formal methods, and AI. While each of these communities has a different emphasis on the type of decision-making problems they are interested in, and thus the precise formalization used, they find common ground in the basic mathematical model often used for these problems: *Markov decision processes*.

Markov decision processes (MDPs; Puterman, 1994) formalize *sequential decision-making under uncertainty* problems in which the decision-maker, often called the *agent*, aims to make optimal decisions for some objective. More specifically, the agent operates in an *environment*. The *states* describe the current configuration of the environment. At each state, the agent is presented with one or more choices called *actions*. After choosing an action, the agent receives a *reward*, and the environment configuration transitions to some new state drawn from a *probability distribution* over all states. Figure 1.1 contains an example of an MDP.

MDPs only specify an environment in which an agent operates. An *objective* specifies what the decision-making problem should optimize for. Common objectives include maximizing the *expected discounted cumulative reward*, the *probability of reaching a certain target state*, or the *expected cumulative reward for reaching a certain target state*. A *policy* is a structured representation of when the agent should choose what action in a given MDP. The obtained expected reward or reachability probability under that policy is called the *value* or the *performance* of that policy. The solution to the decision-making problem modeled as an MDP is an optimal policy and its performance. When the MDP model is completely known, that is, all states, actions, transition probabilities, and rewards are given, the problem of finding an optimal policy for an objective is also known as a *planning* problem.

Planning in MDPs is well-studied and, for many objectives, computationally efficient to solve. Specifically, MDPs with any of the objectives mentioned admit optimal policies that are *memoryless* and *deterministic*. Optimal policies for these

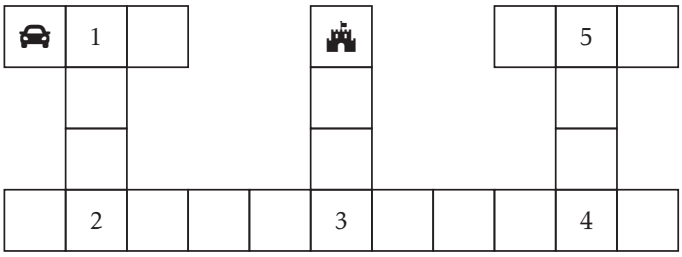


Figure 1.2: An example grid navigation task, inspired by the Cheese Maze example by McCallum (1993). The car represents the agent, and the castle is the target. The intersections in this problem are numbered.

objectives in MDPs can be computed in polynomial time (Baier and Katoen, 2008; Puterman, 1994). More intricate objectives may be expressed using temporal logics, such as (probabilistic) linear temporal logic (LTL; Pnueli, 1977) or computation tree logic ((P)CTL; Clarke and Emerson, 1981; Hansson and Jonsson, 1994), but at the cost of increased computational complexity. While for PCTL, planning in MDPs is still polynomial (in the size of the MDP and logical formula), it is double exponential time complete for PCTL*, which includes LTL, and policies are in general no longer memoryless (Bianco and de Alfaro, 1995).

Solving MDPs, *i.e.*, computing an optimal policy and its performance for an objective, is *the* classic dynamic programming problem (Bellman, 1957). Notable instances of dynamic programming algorithms for MDPs are *value iteration* and *policy iteration*. Additionally, MDPs are straightforwardly encoded into linear optimization problems, providing a polynomial time solution method for many objectives such as the ones aforementioned (see, *e.g.*, Baier and Katoen, 2008).

We illustrate some of the basic concepts of MDPs through Example 1.

Example 1. Consider Figure 1.2, which shows a grid on which the car, the agent, attempts to reach the castle. Such a navigation problem can naturally be modeled as an MDP with a reachability objective. Each cell is a state, the actions are the four cardinal directions the car can move in, and the transition function is given by the probability of reaching one cell from another.

Suppose it is raining and the roads are slippery. With some probability, say 10%, the car slips and moves two cells ahead in the chosen direction instead of one. The agent is now at risk of overshooting some of the intersections. For instance, moving east from the starting position now has a 10% chance of ending up in the cell right of intersection 1. A policy that eventually reaches the castle with probability one simply needs to assign which direction the agent should move for each cell, *i.e.*, it is indeed memoryless deterministic.

Depending on the slip probabilities, getting the car onto an intersection may take a few attempts. Still, eventually, it will succeed as the probability of successively overshooting an intersection in this scenario approaches zero. Thus, *eventually* the agent will always arrive at the castle.

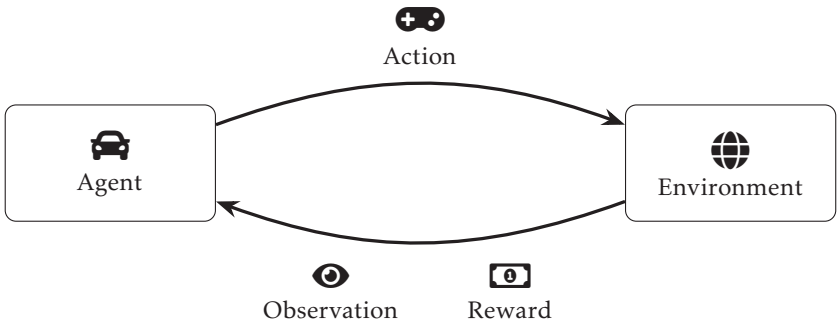


Figure 1.3: Schematic overview of the reinforcement learning loop.

1.1.1 Reinforcement Learning

Reinforcement learning (RL) is a paradigm to solve sequential decision-making problems through exploration in a trial-and-error fashion (Sutton and Barto, 1998). In the planning setting, the agent could exploit knowledge of the environment's transition dynamics to compute a policy, but in the RL setting, the agent does not have this knowledge. Therefore, the agent has to *explore* the environment to collect data. By repeatedly observing a state, taking an action, and observing a successor state and obtaining a reward, as illustrated in Figure 1.3, the agent may, over time, infer what actions are good and what actions are bad. By doing so, it learns a policy, and with enough exploration even converges to an optimal policy.

Consequently, the greatest success stories of RL are found in areas where such trial-and-error behavior poses no harm. Most notably, RL has been used to achieve super-human performance in playing video and board games such as Atari 2600 classics (Mnih et al., 2015) or Go (Silver et al., 2017).

> *Example* 2. Consider the navigation task of Figure 1.2 again. In an RL setting, the agent does not know the transition dynamic of the environment. That is, our agent does not know which cells are adjacent and with what probability they may end up in another cell when moving in a certain direction. From (randomly) trying actions and observing the cell they move to, the agent may learn the transition dynamics and eventually reach the castle.

1.2 Sources of Uncertainty in Decision-Making

We now take a closer look at two of the key sources of uncertainty in sequential decision-making: *state uncertainty* and *stochastic uncertainty*.

1.2.1 State Uncertainty Through Partial Observability

A key assumption MDPs rely on is that the states of the environment in which the agent operates are *observable*. These states provide all necessary information

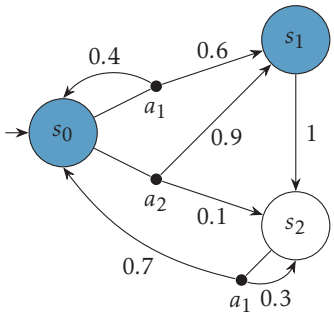


Figure 1.4: An example POMDP model, adapted from the MDP in Figure 1.1. The states are again $\{s_0, s_1, s_2\}$, the actions are again $\{a_1, a_2\}$, and the transition probabilities once more are given by the graph. The observations are represented by the color $\{blue, white\}$, and the observation function assigns the observations *blue* to states s_0 and s_1 and *white* to state s_3. Upon taking action a_1, the agent arrives in state s_0 with probability 0.4 and in state s_1 with probability 0.6, but contrary to MDPs, the agent cannot observe the state in a POMDP and instead only sees the state's observation. In this case, both states have observation *blue*, and hence, the agent cannot identify in which state they arrived.

on which the agent can base their decisions, and the agent can observe all this information. Partially observable Markov decision processes (POMDPs; Åström, 1965; Kaelbling et al., 1998) extend MDPs with an observation function that provides partial information on the states through observations based on the action the agent chose and the successor state they arrived in. POMDPs are a very general framework that allows the modeling of many real-world applications, such as healthcare (Hauskrecht and Fraser, 2000), conservation of endangered species (Chadès et al., 2011, 2012), or aircraft collision avoidance (Kochenderfer, 2015). An example POMDP is illustrated in Figure 1.4. We continue Example 1 by adding *state uncertainty* through partial observability.

Example 3. Consider Figure 1.2 again. Now, suppose the agent is not aware of its position but can only observe in which directions it can move from one cell, as illustrated in Figure 1.5. This is a form of partial observability, and the agent cannot (immediately) distinguish several cells anymore. For example, the intersections 1 and 5, or 2, 3, and 4 will now be identical to the agent, as will the vertical and horizontal roads.

This partial observability poses a problem for our agent. In the cells between intersections 1 and 2, the agent needs to move south, but between 3 and the castle, they need to move north. So, the problem that arises is how to distinguish these cells to make the right decisions.

The solution is straightforward: *memory*. Clearly, if the agent remembers where they were a few steps before, that information could be used to *infer* where they are now. For example, knowing the agent starts in the top left, seeing an intersection of type 1 or 5 implies the agent is at 1. Then, the agent

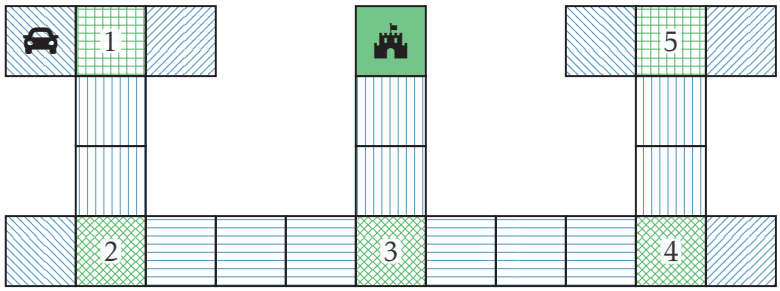


Figure 1.5: The grid navigation task from Figure 1.2, adapted with observations. The patterns and colors show which cells the agent cannot be distinguished from each other. For instance, all cells where the agent can only move vertically are indicated by blue vertical lines.

> should move south until observing an intersection of type 2, 3, or 4. Knowing we came from 1, this has to be intersection 2. From here, we move east until we see an intersection again, which could be either 3 or 4 due to the stochastic transition dynamics. From there, we move north until we reach the castle or see an intersection, which has to be 5. In the latter case, we repeat the above reasoning but are now mirrored.

As illustrated in Example 3, partial observability requires the agent to use memory. While the idea of using memory seems straightforward, it only presents us with new problems: how much memory do we need, and what do we need to remember? In general, optimal policies for POMDPs with infinite horizons require infinite memory, rendering this problem undecidable (Madani et al., 2003). Nonetheless, several successful approaches to approximate such infinite memory policies have been proposed, most notably point-based value iteration techniques (Kurniawati et al., 2008; Pineau et al., 2003; Spaan and Vlassis, 2005).

Finite memory can also make good approximations of the optimal policy (Bonet, 2002). Policies with finite memory can be concisely represented by automata known as *finite-state controllers* (FSCs; Meuleau et al., 1999), and computed via policy iteration (Poupart and Boutilier, 2003) or convex optimization techniques (Amato et al., 2010; Junges et al., 2018). When small enough, FSCs can also be considered more explainable than arbitrary finite-memory or belief-based policies (Bork et al., 2024; Dujardin et al., 2017)

1.2.2 Stochastic Uncertainty: Where Do Probabilities Come From?

In the standard planning setting, it is simply assumed that the transition probabilities of the MDP exist and are known. This assumption is, however, rather strong in many applications. Probabilities may be estimated from data or derived from subjective expert opinions, which naturally carries the risk of making inaccurate estimates, especially when data is limited or flawed or different experts have disagreeing opinions.

Literature distinguishes two types of uncertainty: *aleatoric* and *epistemic* un-

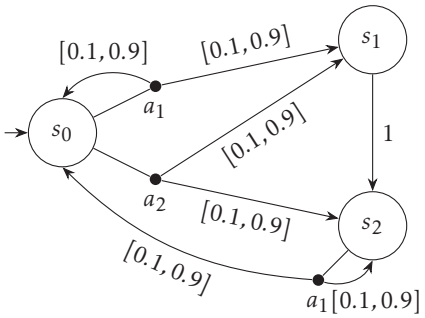


Figure 1.6: An example RMDP adapted from the MDP in Figure 1.1. The uncertainty set at each state-action pair of this RMDP is given by all valid probability distributions within the intervals. Hence, all distributions of the form $(p, 1-p)$ with $p \in [0.1, 0.9]$.

certainty (Hüllermeier and Waegeman, 2021; Soize, 2017). Aleatoric uncertainty is inherent to the system or environment being modeled, such as in board games where a die is rolled or a networking protocol that uses coin flips to decide which party goes first. While we know the probability distribution that is being drawn from precisely, the *outcome* of the draw will always remain uncertain. Hence, the stochastic uncertainty of (PO)MDPs in the transition (and observation) function is inherently aleatoric.

On the other hand, epistemic uncertainty stems from a lack of knowledge and may be reduced by incorporating further data. Such a lack of knowledge may, for instance, come from insufficient data to reliably estimate a probability distribution.

Consider a straightforward estimation technique such as maximum likelihood estimation (MLE). MLE neglects the amount of data used when computing the estimate. For example, suppose we estimate the probability of observing *heads* after a coin flip. If we flip the coin twice and observe heads once and tails once, the MLE derives a probability of $1/2$ for heads. If we flip the coin a thousand times, and observe heads precisely five hundred times, the MLE probability would be the same, *i.e.*, $500/1000 = 1/2$. Yet, intuitively, we have more confidence in the latter estimate than in the former, as it is based on more data. We can formalize this intuition by adding a confidence interval around the MLE estimate, for instance, through *probably approximately correct* (PAC) learning (Valiant, 1984).

Such confidence intervals should ideally be taken into account by the model describing a decision-making problem. *Robust* Markov decision processes (RMDPs; Iyengar, 2005; Nilim and Ghaoui, 2005; Wiesemann et al., 2013) allow for precisely that. RMDPs extend standard MDPs with an *uncertainty set* that describes sets of probability distributions for each state-action pair.

Intuitively, we may think of RMDPs as a set of MDPs that differ only in their transition function. The agent now needs to make decisions that are *robust* against all possible transition functions in this uncertainty set. Consequently, we obtain a robustness guarantee on the resulting policy and its performance. Regardless of which environment the resulting policy is deployed in, as long as that environment

is contained within the uncertainty set of the RMDP, the performance is guaranteed to be at least that of the policy in the worst-case environment.

Computing robust policies and their performance in RMDPs is more involved than in MDPs, and depends on the *structure* of the uncertainty set. An uncertainty set that satisfies an independence condition where it may be partitioned into independent uncertainty sets per state-action pair is known as (s,a)-*rectangular* (Iyengar, 2005). Uncertainty sets that partition into independent sets per state (and thus with possible dependencies between different actions at a state) are called s-rectangular (Wiesemann et al., 2013). When the uncertainty set is (s,a)-rectangular, dynamic programming methods for MDPs, such as value and policy iteration, can be extended to RMDPs in a straightforward manner (Iyengar, 2005; Nilim and Ghaoui, 2005). For s-rectangular RMDPs, solution methods are more tailor-made to specific types of uncertainty set (Ho et al., 2018, 2021; Wiesemann et al., 2013).

RMDPs thus provide a strong formalization for robust decision-making under uncertainty, provided that the uncertainty set is constructed in an appropriate way. Besides planning problems where the probabilities may be inaccurate, RMDPs also find a natural application in RL, which we shall discuss later in Section 1.4.1. .

Unfortunately, RMDPs are a relatively young formalization and, thus, to the best of our knowledge, are not included in any standard textbooks in operations research, formal methods, or AI. Furthermore, only one recent survey is available (Ou and Bi, 2024) that discusses the state-of-the-art in RMDP literature and clearly targets an audience already familiar with RMDPs and robust dynamic programming. Given that RMDPs are a natural extension of MDPs with many applications across communities, the first research question treated in this thesis arises.

> **Research question 1**
>
> How can we make RMDP literature accessible to a wider audience with basic familiarity with standard MDPs and dynamic programming?

1.3 Robust Planning Under Partial Observability

We discussed planning in MDPs, RMDPs, and POMDPs above. What about *robust POMDPs*? Indeed, as one would expect, robust POMDPs combine the model uncertainty of RMDPs and the state uncertainty of POMDPs. Planning in robust POMDPs, however, is still largely understudied. Osogami (2015) extends the value iteration approaches for POMDPs to robust POMDPs, while Burns and Brock (2007) propose a sampling-based method which is thus strictly speaking not robust.

Although the use of finite-memory policies through FSCs in POMDPs has been extensively studied, they have not been considered for robust POMDPs yet. Therefore, a natural direction to investigate is whether existing methods to compute FSCs for POMDPs can be made robust or whether a more dedicated approach is needed for robust POMDPs. This leads us to the second research question of this thesis.

Research question 2

How to efficiently plan with finite-state controllers in robust POMDPs?

1.4 Robust and Reliable Reinforcement Learning

As introduced in Section 1.1.1, RL has shown great promise in solving complicated decision-making problems where a model of the environment is not readily available. Yet, RL faces several challenges when dealing with real-world problems. Dulac-Arnold et al. (2021) identify nine of such challenges, out of which the following two concern the robustness and reliability of RL.

i. Interacting with environments that are partially observable or non-stationary.

ii. Training offline on fixed datasets of past interactions of a behavior policy.

The first challenge concerns robustness against two sources of uncertainty: partial observability and changing environments. Both present the agent with a lack of information. Under partial observability, the observations are generally insufficient to determine transition probabilities, as different states may have the same observations and are thus lumped together in the data. A similar problem appears when the underlying environment is non-stationary and changes. Even when these environments are fully observable, data from one environment may get mixed up with data from another environment, again lumping together data from two different probability distributions.

The second challenge also concerns robustness but of a different kind. Offline datasets are inherently finite and, very likely, too small to provide sufficient data to reliably estimate the probability distributions throughout the environment. Thus, offline RL needs to be robust against the epistemic uncertainty stemming from a lack of data in an attempt to provide reliable results.

We now consider each of these two challenges and the specific research questions addressed in this thesis that arise from them in more detail.

1.4.1 Robustness in Reinforcement Learning

Robust RL is an umbrella term for any RL method that deals with robustness to uncertainty, disturbances, or (structural) changes in the environment (Moos et al., 2022). RMDPs form the backbone of (model-based) robust RL as they naturally capture uncertainty about the transition probabilities. The worst-case policy of an RMDP is robust against any uncertainty captured by the uncertainty set. Hence, a fundamental problem is how to construct uncertainty sets that are conservative enough to ensure robustness.

Probably approximately correct (PAC) learning is commonly used to construct such uncertainty sets. An interval around a probability estimated from data is PAC if, with high confidence, the actual probability lies within the interval. Such an interval can be computed via concentration inequalities such as Hoeffding's inequality (Hoeffding, 1963). Similarly, one can construct a set of probability distributions

around an estimated distribution that is PAC using Weissman's bound (Weissman et al., 2003). PAC learning techniques have been used extensively in the formal methods community in *statistical model checking* (Agha and Palmskog, 2018; Ashok et al., 2019), and in the AI community in RL (Jaksch et al., 2010; Strehl and Littman, 2008; Strehl et al., 2006, 2009).

A fundamental drawback of PAC learning methods is, however, that they require samples to be *independent and identically distributed* (i.i.d.). To ensure the i.i.d. requirement, it is natural to assume the underlying environment the samples are drawn from is stationary. When the environment is not stationary, sliding window approaches (*i.e.*, forgetting old data after some time) are employed (Cheung et al., 2020; Gajane et al., 2018; McCallum, 1995). These methods are, however, typically used in combination with *optimism*, *i.e.*, best-case policies to ensure efficient exploration, instead of robustness. Altogether, these observations lead us to the next research question of this thesis.

> **Research question 3**
>
> How to make model-based RL more robust against changing environments?

1.4.2 Reliable Offline Reinforcement Learning

We now turn to the second challenge we listed for RL. The standard RL paradigm assumes some form of sampling access to collect data about the environment. In many applications, such access cannot be provided or granted. In real-world applications such as robotics or healthcare, direct interaction can be impractical or dangerous (Levine et al., 2020). Other examples include predictive maintenance (Andriotis and Papakonstantinou, 2021), where data is collected under the current maintenance plan, which may be impractical to change for to sole purpose of additional data collection, or conservation of endangered species (Chadès et al., 2012), and management of invasive species (Chadès et al., 2011), both instances of decision-making problems where interventions in the environment are costly as they may upset a delicate balance and hence, online exploration is out of the question. Furthermore, alternatives such as simulators or digital twins may not be available or insufficiently capture the nuances of the real-world application for reliable learning (Ramakrishnan et al., 2020; Zhao et al., 2020).

Offline RL, also known as batch RL (Lange et al., 2012), mitigates these concerns by allowing the agent only access to a fixed dataset of past interactions generated by a so-called *behavior policy*. Offline RL algorithms compute a new policy without further interactions with the environment (Levine et al., 2020). Methods that can reliably improve the performance of a policy are key in (offline) RL.

Safe policy improvement (SPI) is a specific offline RL problem that concerns the computation of a new policy that outperforms the behavior policy with a reliability guarantee. This reliability guarantee comes in a PAC-style that states that the improved policy produced by an SPI algorithm outperforms the behavior policy with high probability and up to some *admissible performance loss*. To provide this reliability guarantee, most SPI methods assume the environment and dataset are

fully observable, *i.e.*, modeled by an MDP (Laroche et al., 2019; Petrik et al., 2016).

The restriction to fully observable environments poses a serious limitation on the applicability of SPI, as most real-world problems are partially observable due to, for instance, noisy sensors (Kochenderfer, 2015). As such, having SPI methods work with partially observable datasets obtained from POMDPs would significantly expand their applicability. So far, SPI for POMDPs was only studied for memoryless policies (Thomas et al., 2015; Yeager et al., 2022). However, POMDP policies often require a notion of memory and are then typically represented by finite-state controllers. As such, the first key challenge to employing SPI in practical settings is to develop an SPI algorithm that can work with finite-memory policies and partially observable datasets. We summarize this first challenge for SPI in the following research question.

> **Research question 4**
>
> How to do safe policy improvement on finite-memory policies with partially observable datasets?

The reliability guarantees provided by SPI algorithms depend on the size of the dataset and usually adhere to a conservative bound on the minimal amount of samples required. Since this bound often turns out to be too large for practical applications of SPI, it is instead turned into a hyperparameter (see, *e.g.*, Laroche et al., 2019). As the offline nature of SPI prevents further data collection, the second key challenge to employing SPI in practical settings is to exploit the dataset as efficiently as possible and thus compute improved policies from smaller datasets. Previous work shows that exploiting underlying structures in the environment through a factorized state space (Simão and Spaan, 2019a) and structure learning methods (Simão and Spaan, 2019b). We summarize this second challenge for SPI in the following research question.

> **Research question 5**
>
> How to exploit the dataset as efficiently as possible in safe policy improvement?

1.5 Contributions and Structure of the Thesis

The contributions of this thesis aim to provide technical solutions to the research questions posed above. All our methods are inherently *model-based*. Model-based reasoning allows us, for instance, to exactly evaluate the performance of a policy, derive formal guarantees on, *e.g.*, the correctness of a learned model or the improvement of a learned policy.

We now introduce each of the contributions presented in this thesis and its structure. In Chapter 2, we start with a general background. In particular, we cover MDPs and their semantics, dynamic programming, and the extension to POMDPs. The chapters that follow present the technical contributions of this thesis.

1.5.1 A Tutorial on Robust Markov Decision Processes

Chapter 3 is dedicated to robust MDPs (RMDPs). The chapter is structured as a short survey, starting with a tutorial in which we introduce the general model of RMDPs, discuss their semantics and structural assumptions such as *rectangularity*, and show how dynamic programming can be extended to *robust dynamic programming*. We then discuss some commonly used instances of RMDPs, such as interval MDPs (IMDPs), L_1-MDPs, and multi-environment MDPs (MEMDPs). Applications of RMDPs, especially in the context of reinforcement learning, are discussed in subsequent chapters.

To the best of our knowledge, there is only one survey on RMDPs available (Ou and Bi, 2024), which targets an audience already familiar with most concepts and notations used in the RMDP literature. In contrast, our overview is meant to be accessible to readers with a basic understanding of MDPs and dynamic programming (as introduced in Chapter 2). As such, Chapter 3 constitutes the first contribution of this thesis towards Research question 1.

> **Contribution 1**
>
> We present a tutorial on RMDPs and robust dynamic programming.

Chapter origins. Chapters 2 and 3 are partly based on joint work with Thom Badings, Eline M. Bovy, David Parker, and Nils Jansen that was published in (Suilen et al., 2024a). The author was the main person responsible for the contents used in these two chapters.

1.5.2 Finite-Memory Policies for Robust POMDPs

In Chapter 4, we study planning in robust POMDPs. We develop two novel algorithms based on convex optimizations for planning in (s,a)-rectangular robust POMDPs with interval uncertainty. In contrast to existing literature, we do not consider belief-based policies but finite-memory policies represented by FSCs. Both algorithms start with a robust nonlinear optimization problem that is then iteratively either convexified into a robust convex qudratically constrained quadratic program (QCQP), or linearized into a robust linear program (LP). We then apply standard techniques from robust optimization to these robust convex optimization problems to derive finite convex optimization problems. These iterative algorithms improve over previous solutions until a local optimum is attained, at which point the robust FSC and its performance can be extracted. In our experimental evaluation, we compare both algorithms and highlight their scalability. Thus, our second contribution towards Research question 2 can be formulated as follows.

> **Contribution 2**
>
> We present novel algorithms based on convex optimization to compute finite-memory policies for robust POMDPs.

Chapter origins. This chapter is based on two papers. The first was published at IJCAI 2020 (Suilen et al., 2020) and is joint work with Murat Cubuktepe, Nils Jansen, and Ufuk Topcu. The second paper was published at AAAI 2021 (Cubuktepe et al., 2021) and is joint work with Murat Cubuktepe, Nils Jansen, Sebastian Junges, Ahmadreza Marandi, and Ufuk Topcu. The author was responsible for co-developing the theory, implementation, empirical evaluation, and writing both papers. For this chapter, the contents of the two papers have been merged into a continuous story and include some minor extensions to the theory and a completely revised empirical evaluation.

1.5.3 Robust Anytime Learning of Markov Decision Processes

In Chapter 5, we consider a robust RL setting where the underlying environment may change over time. Our goal is to learn an RMDP that is adaptable to such changes. To that end, we use *linearly updating intervals* (Walter and Augustin, 2009), a Bayesian approach to updating probability intervals in the presence of new, possibly inconsistent, data. We integrate these linearly updating intervals into a robust RL loop that learns an IMDP. Together with a sliding window approach that discards old data and integrates with robust dynamic programming, we can learn robust policies that are conservative and adaptable when the underlying environment changes. We show the applicability of our approach in an experimental evaluation that also compares to other learning methods. We also perform an ablation that shows the need for both the linearly updating intervals and the sliding window to deliver the robustness and adaptability of our approach. Hence, our third contribution is towards Research question 3.

> **Contribution 3**
>
> We present a new approach to robust reinforcement learning in MDPs where the underlying environment may change over time.

Chapter origins. This chapter is based on joint work with Thiago D. Simão, David Parker, and Nils Jansen that was published at NeurIPS 2022 (Suilen et al., 2022). The author was the main person responsible for developing the theory, implementation and empirical evaluation, and writing the paper. For this chapter, parts of the original paper have been rewritten and extended with additional material.

1.5.4 Extending the Scope of Offline RL

Chapter 6 presents our final two contributions to the offline reinforcement learning problem of safe policy improvement. Together, these two contributions extend the scope of SPI algorithms.

For our first contribution to SPI, we extend SPI algorithms, such as *safe policy improvement with baseline bootstrapping* (SPIBB; Laroche et al., 2019), to partially observable environments. The dataset now consists of observations, actions, and rewards stemming from a POMDP that we assume to be k-Markovian (Brafman

and Giacomo, 2024; Kaelbling et al., 1996). We show that under this assumption, the improvement guarantee of SPIBB still holds. Our experimental evaluation also shows that this approach is relevant and promising in environments that do not satisfy the assumption. In summary, this contribution is towards Research question 4 and can be formulated as follows.

> **Contribution 4**
>
> We extend SPIBB to compute finite-memory policies from datasets collected in partially observable environments.

In our second contribution to the SPI problem, we introduce a novel approach that significantly reduces the amount of data required to establish the same improvement guarantee within the SPIBB algorithm, thus making the most out of the available data. We devise a new transformation for the underlying MDP model and the dataset collected from it that limits its branching factor. This transformation expands the state space of the underlying MDP but allows us to exploit the fixed branching factor when computing the improvement guarantee. This last technical contribution of this thesis is thus towards Research question 5.

> **Contribution 5**
>
> We present new techniques that provide stronger improvement guarantees given the same amount of data in SPI algorithms such as SPIBB.

Chapter origins. This chapter is based on two papers. The first was published at AAAI 2023 (Simão et al., 2023) and is joint work with Thiago D. Simão and Nils Jansen. The author contributed to the theory, implementation and empirical evaluation, and the writing. The second paper was published at IJCAI 2023 (Wienhöft et al., 2023) and is joint work with Patrick Wienhöft, Clemens Dubslaff, Thiago D. Simão, Christel Baier, and Nils Jansen. The author contributed to the implementation and empirical evaluation, the writing, and parts of the theory. For this chapter, the contents of the two papers have been merged into a continuous story, and some minor extensions of the theory have been included.

1.5.5 Other Peer-Reviewed Publications

We briefly summarize the other peer-reviewed publications the author contributed to that are not included in this thesis.

Balancing Wind and Batteries: Towards Predictive Verification of Smart Grids (Badings et al., 2021). This paper studies balancing demand and wind power availability in a smart grid with batteries. Classical day-ahead planning for this problem relies on weather forecasts to predict future wind power. As the actual wind conditions often deviate from forecasts, short-term flexibility in storage and generation is used to fill potential gaps. Where previous approaches rely on sampling, our method employs more rigorous probabilistic verification techniques

by formalizing the problem as a continuous-space Markov decision process with discrete controls. To mitigate state space explosion, we exploit specific structural properties of the model to implement an iterative exploration method that reuses pre-computed values as wind data is updated.

Decision-Making Under Uncertainty: Beyond Probabilities (Badings et al., 2023c). This position paper provides an overview of the state-of-the-art in decision-making under uncertainty, with a specific focus on aleatoric and epistemic uncertainty. The paper presents a short review of Markov decision processes (MDPs) and extensions to account for partial observability (POMDPs), adversarial behavior, and models that exhibit uncertainty in a more robust interpretation, *i.e.*, RMDPs. We discuss several solution techniques for discrete and continuous models, ranging from formal verification over control-based abstractions to reinforcement learning. Finally, we list and discuss several key challenges that arise when dealing with rich types of uncertainty in a model-based fashion.

Imprecise Probabilities Meet Partial Observability: Game Semantics for Robust POMDPs (Bovy et al., 2024). This paper studies robust POMDPs (RPOMDPs) and their semantics. In particular, we establish that the notions of static and dynamic uncertainty semantics of robust MDPs do not coincide on RPOMDPs and are the two extremes of an entire spectrum. We define an explicit transformation from RPOMDP to *partially observable stochastic game* (POSG) and show that for different uncertainty semantics, the transformation yields semantically different POSGs. Consequently, the uncertainty semantics do not coincide, and different semantics may have different optimal policies and values.

A PSPACE Algorithm for Almost-Sure Rabin Objectives in Multi-Environment MDP (Suilen et al., 2024b). This paper presents an algorithm to decide almost-sure Rabin objectives in multi-environment Markov decision processes (MEMDPs). We show that in MEMDPs, belief-support-based strategies are sufficient for these objectives, in contrast to general POMDPs. We use this observation to develop a recursive algorithm that operates in PSPACE to decide whether there exists a policy that can satisfy the almost-sure Rabin objective in the MEMDP. This paper received the *best paper award* at CONCUR 2024.

2

Decision-Making Under Uncertainty: Foundations

In this chapter, we provide the necessary definitions and background knowledge for the results presented in this dissertation. We introduce Markov decision processes (MDPs), the objectives we consider in this thesis, and how dynamic programming is used to find optimal policies for these objectives in MDPs. Next, we introduce partially observable MDPs (POMDPs). Robust MDPs and robust dynamic programming are given a dedicated treatment in Chapter 3.

Basic notations. For a set X, we denote its cardinality by $|X|$ and X^* denotes the set of all (in)finite sequences over X. The concatenation of two sequences $\omega, \omega' \in X^*$ is written as $\omega : \omega'$. A discrete probability distribution over a finite set X is a function $\mu \colon X \to [0,1]$ such that $\sum_{x \in X} = 1$. The set of all discrete probability distributions over X is denoted by $\mathcal{D}(X)$. A distribution $\mu \in \mathcal{D}(X)$ is *Dirac* when there exists precisely one element $x \in X$ with $\mu(x) = 1$. We write $\mathbb{N}$ and $\mathbb{R}$ for the standard sets of natural and real numbers, and $\mathbb{R}_{\geq 0}$ for the non-negative reals. Partial functions are denoted $f \colon X \rightharpoonup Y$, and we write $\perp$ to denote *undefined*. We write $[m : n]$ for the set of natural numbers $\{m, \ldots, n\} \subset \mathbb{N}$, and $\mathbb{1}[x{=}x']$ for the indicator function, returning 1 if $x = x'$ and 0 otherwise.

2.1 Markov Decision Processes

We define Markov decision processes (MDPs) and their semantics.

Definition 2.1 (MDP)**.** A Markov decision process (MDP) is a tuple of the form $\langle S, s_\iota, A, P, R \rangle$, where S is a finite set of states with $s_\iota \in S$ the initial state, A is a finite set of actions, $P \colon S \times A \rightharpoonup \mathcal{D}(S)$ is the probabilistic transition function, and $R \colon S \times A \rightharpoonup \mathbb{R}_{\geq 0}$ is the (non-negative) reward function.

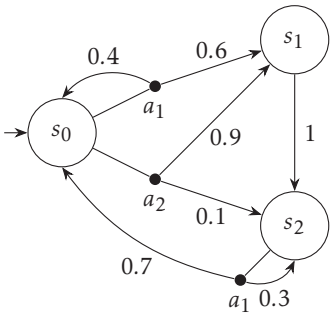


Figure 2.1: An example of an MDP, as also shown in Chapter 1. The states are $\{s_0, s_1, s_2\}$, the actions are $\{a_1, a_2\}$, and the transition probabilities are given by the graph. In state s_2 only action a_1 is enabled.

We use partial functions for the transition and reward functions to allow for *enabled actions*. An action is enabled if $P(s,a)$ is defined. We write $A(s) \subseteq A$ for the set of enabled actions at state s. An example MDP is shown in Figure 2.1. We require that the transition and reward function are consistent with each other, that is, $P(s,a) = \bot \iff R(s,a) = \bot$. For convenience, we write $P(s'|s,a)$ for the probability $P(s,a)(s')$. This notation extends to other functions.

For each state-action pair, we define the set of successor states as $Post_M(s,a) = \{s' \in S \mid P(s'|s,a) > 0\}$. A path in an MDP is an (in)finite sequence of successive states and actions: $\omega = \langle s_0, a_0, s_1, \ldots \rangle \in (S \times A)^* \times S$ where $s_0 = s_\iota$ and $\forall i \in \mathbb{N}\colon\ s_{i+1} \in Post_M(s_i, a_i)$. A path is finite if the sequence is finite, $\omega = \langle s_0, a_0, \ldots, s_k \rangle$, for which we write $last(\omega) = s_k$ for the last state. The set of all paths in MDP M is denoted as Paths_M. When clear from the context, we may omit M and simply write $Post(s,a)$ or Paths. The sequence of states in a path $\omega = \langle s_0, a_0, s_1, \ldots \rangle$ is $states(\omega) = \langle s_0, s_1, \ldots \rangle$.

A *discrete-time Markov chain* (DTMC) is an MDP with only one enabled action in each state: $\forall s \in S\colon\ |A(s)| = 1$. For DTMCs, we omit the actions altogether from the tuple and write $\langle S, s_\iota, P, R \rangle$, where the transition function is equivalent to the total function $P\colon S \to \mathcal{D}(S)$, and the reward function is $R\colon S \to \mathbb{R}$. A *policy* (also called *scheduler* or *strategy*) is a function that maps paths to distributions over actions $\pi\colon \mathsf{Paths} \to \mathcal{D}(A)$. Such policies are called *history-based* and *randomized*. The set of all policies is denoted by Π.

A policy is a *finite-memory* policy if it only requires finite paths. More precisely, finite-memory policies can be encoded by a (finite) automaton known as a *finite-state controller* (FSC).

Definition 2.2 (FSC)**.** A *finite state controller* (FSC) for an MDP $M = \langle S, s_\iota, A, P, R \rangle$ is a tuple $\langle \mathcal{N}, n_\iota, \alpha, \eta \rangle$, where $\mathcal{N}$ is a finite set of *memory nodes*, $n_\iota \in \mathcal{N}$ is the initial node, $\alpha\colon \mathcal{N} \times S \to \mathcal{D}(A)$ is the *action mapping*, and $\eta\colon \mathcal{N} \times S \times A \to \mathcal{D}(\mathcal{N})$ is the memory update function.

A policy is *deterministic* if it only maps to Dirac distributions over actions and *memoryless* (also called *stationary*) if it only considers finite paths of length one,

i.e., it can be represented by an FSC with only a single memory node. Stationary deterministic policies are written as $\pi\colon S \to A$.

Given a policy π for an MDP M, the action choices in M are resolved, resulting in a DTMC. For arbitrary policies $\pi\colon \mathsf{Paths} \to \mathcal{D}(A)$, we construct the following infinite state DTMC.

Definition 2.3 (Induced DTMC)**.** Let $M = \langle S, s_\iota, A, P, R\rangle$ be an MDP and $\pi\colon \mathsf{Paths} \to \mathcal{D}(A)$ a policy. The induced DTMC is defined as $M_\pi = \langle S^*, s_\iota, P_\pi, R_\pi\rangle$, where S^* is the (infinite) set of states, s_ι is the initial state, and the transition and reward functions are defined as

$$P_\pi(\mathit{states}(\omega), \mathit{states}(\omega) : s') = \sum_{a\in A} \pi(a\,|\,\omega)\cdot P(s'\,|\,\mathit{last}(\omega), a),$$
$$R_\pi(\mathit{states}(\omega)) = \sum_{a\in A} \pi(a\,|\,\omega)\cdot R(\mathit{last}(\omega), a),$$

where $\omega \in \mathsf{Paths}$ and $\mathit{states}(\omega) : s'$ denotes concatenation of $\mathit{states}(\omega)$ with s'.

The induced DTMC M_π has a unique probability measure $\mathbb{P}_{M_\pi}$ by the standard cylinder set construction (Baier and Katoen, 2008; Fijalkow et al., 2023).

When the policy π is finite-memory and represented by an FSC, we can simplify the definition of the induced DTMC to the following product construction.

Definition 2.4 (Induced DTMC of an FSC)**.** Let $M = \langle S, s_\iota, A, P, R\rangle$ be an MDP and $\mathcal{F} = \langle \mathcal{N}, n_\iota, \alpha, \eta\rangle$ be an FSC representing the policy π. The induced DTMC is defined as $M_\mathcal{F} = \langle S \times \mathcal{N}, \langle s_\iota, n_\iota\rangle, P_\mathcal{F}, R_\mathcal{F}\rangle$, where the states are given by the product of MDP states S and FSC nodes $\mathcal{N}$, the initial state is $\langle s_\iota, n_\iota\rangle$, and the transition and reward functions are defined as

$$P_\mathcal{F}(\langle s', n'\rangle \,|\, \langle s, n\rangle) = \sum_{a\in A} \alpha(a\,|\,n, s)\cdot P(s'\,|\,s, a)\cdot \eta(n'\,|\,n, s, a),$$
$$R_\mathcal{F}(\langle s, n\rangle) = \sum_{a\in A} \alpha(a\,|\,n, s)\cdot R(s, a).$$

Throughout this thesis, we will assume that finite-memory policies are implicitly encoded as FSCs, and we write M_π for the induced DTMC $M_\mathcal{F}$ when the FSC $\mathcal{F}$ encodes policy π.

2.1.1 Objectives

Objectives specify the goal of the decision-making problem. In general, we are interested in computing some *value* V and an associated (optimal) policy π that achieves this value.

Definition 2.5 (Objectives)**.** We consider four objectives in this thesis: *reachability*, *discounted reward*, and *cumulative reward* (or cost) until reaching a target, which we shall refer to as *reach-reward* and *stochastic shortest path*, respectively. The semantics of these objectives follow those of (standard) temporal logic over paths in MDPs (see, *e.g.*, Baier and Katoen, 2008; Kwiatkowska et al., 2007).

Reachability. The *reachability objective* is to optimize the probability of reaching a target set $T \subseteq S$:

$$\mathbf{P}_{Max}(\Diamond T) = \max_{\pi \in \Pi} \mathbb{P}_\pi [\omega \in \mathsf{Paths}_M \mid \omega \models \Diamond T].$$

Discounted Reward. The *discounted reward* objective is to optimize the discounted sum of expected rewards for some discount factor $\gamma \in (0,1)$:

$$\mathbf{R}_{Max}(\gamma) = \max_{\pi \in \Pi} \mathbb{E}_\pi \left[\sum_{t=0}^{\infty} \gamma^t R(s_t, a_t) \mid s_0 = s_\iota \right],$$

where s_t and a_t are the state and action at step t along a path $\omega \in \mathsf{Paths}_M(s_\iota)$.

Reach-Reward. Let $\omega \in \mathsf{Paths}_M$, the cumulative reward along ω is defined as

$$r(\Diamond T)(\omega) = \begin{cases} \infty & \forall t \in \mathbb{N} \colon s_t \notin T, \\ \sum_{t=0}^{\min\{t \mid s_t \in T\}-1} R(s_t, a_t) & \text{otherwise.} \end{cases}$$

The *reach-reward* objective is to optimize the cumulative reward until reaching a target set $T \subseteq S$:

$$\mathbf{R}_{Max}(\Diamond T) = \max_{\pi \in \Pi} \mathbb{E}_\pi [r(\Diamond T)].$$

Stochastic Shortest Path. Let $r(\Diamond T)(\omega)$ again denote the cumulative reward along a path ω. The stochastic shortest path objective is the minimizing dual of reach-reward and defined as

$$\mathbf{R}_{Min}(\Diamond T) = \min_{\pi \in \Pi} \mathbb{E}_\pi [r(\Diamond T)].$$

The set of these objectives is denoted by

$$\Phi = \{\mathbf{P}_{Max}(\Diamond T), \mathbf{R}_{Max}(\Diamond T), \mathbf{R}_{Min}(\Diamond T), \mathbf{R}_{Max}(\gamma)\}.$$

Naturally, the optimization direction of the reachability and discounted reward objectives may also be reversed to instead *minimize*. Besides optimizing, one may also consider the satisfaction of some threshold. Instead of maximizing or minimizing, we simply seek a policy that ensures that the probability or reward surpasses some threshold. Standard approaches optimize the objective and then check the found optimum against that threshold (Forejt et al., 2011). For MDPs, it is well-known that for all of the objectives $\varphi \in \Phi$, memoryless deterministic policies are sufficient to be optimal (Forejt et al., 2011; Puterman, 1994).

2.2 Classical Dynamic Programming

We now review *dynamic programming*, the technique at the core of solving MDPs via an algorithm called *value iteration*.

2.2.1 Value Iteration

We define state and state-action value functions $V\colon S \to \mathbb{R}$ and $Q\colon S \times A \to \mathbb{R}$, respectively. Dynamic programming updates these value functions iteratively until the least fixed point is reached, at which point the value functions represent the best possible (*i.e.*, optimal) value achievable for the given objective.

We now present value iteration for the reach-reward objective $\mathbf{R}_{Max}(\Diamond T)$ and discuss the adjustments necessary for the other objectives $\varphi \in \Phi$ afterward.

We preprocess the set of states S based on graph properties via the following standard procedure (Baier and Katoen, 2008) and PRISM-semantics for reward objectives (Forejt et al., 2011; Kwiatkowska et al., 2007). Let $T \subseteq S$ be the target set of our objective, let $S^{\infty} \subseteq S$ be the set of states for which there exists a policy that does not reach T almost-surely, and denote the remaining states by $S^{?} = S \setminus (T \cup S^{\infty})$. For all $a \in A$ and target states $s \in T$, let $Q(s,a) = 0$. Similarly, for all $a \in A$ and $s \in S^{\infty}$, let $Q(s,a) = \infty$. For all other states $s \in S^{?}$ and $a \in A$, we initialize the state-action values as $Q(s,a) = 0$. We iteratively update the state and state-action values for all $(s,a) \in S^{?} \times A$ by:

$$V^{(n)}(s) = \max_{a \in A} Q^{(n)}(s,a), \quad Q^{(n+1)}(s,a) = R(s,a) + \sum_{s' \in S} P(s' \mid s,a) V^{(n)}(s').$$

This process is also known as *value iteration*. When doing value iteration, we do not need to keep track of the state-action values Q explicitly but instead can directly compute the state-values V by setting $V(s) = 0$ for all $s \in T$, for all $s \in S^{\infty}$, $V(s) = \infty$, and for all $s \in S^{?}$ we iteratively compute:

$$V^{(n+1)}(s) = \max_{a \in A} \left\{ R(s,a) + \sum_{s' \in S} P(s' \mid s,a) V^{(n)}(s') \right\}, \tag{2.1}$$

The optimal value function V^{*} is the unique least fixed point of the *Bellman equation* in Equation 2.1.

For many objectives in MDPs, such as reach-reward maximization, optimal policies are stationary and deterministic, *i.e.*, of type $\pi\colon S \to A$ (Puterman, 1994). An optimal stationary deterministic policy π^{*} that achieves value V^{*} can be extracted by performing the following one-step dynamic programming procedure:

$$\pi^{*}(s) = \operatorname*{argmax}_{a \in A} \left\{ R(s,a) + \sum_{s' \in S} P(s' \mid s,a) V^{*}(s') \right\}.$$

2.2.2 Policy Evaluation

Policy evaluation is the process of computing the value of an MDP for a given policy. In the formal methods community, this process is also known as *verifying* or *model checking* the induced Markov chain from Definition 2.3 (Baier and Katoen, 2008).

Definition 2.6 (Policy evaluation)**.** The value of a memoryless policy $\pi\colon S \to \mathcal{D}(A)$ is computed by the following Bellman equation:

$$V_\pi^{(n+1)}(s) = \sum_{a \in A} \pi(a|s) \cdot \left(R(s,a) + \sum_{s' \in S} P(s'|s,a) V_\pi^{(n)}(s') \right).$$

Alternatively, we may explicitly construct the induced DTMC $\langle S, s_\iota, P_\pi, R_\pi \rangle$ from Definition 2.3, whose set of states coincides with that of the MDP (and is thus finite) as the policy π is memoryless.

The *performance* or *expected return* of the policy π is defined as its value in the initial state $\rho = V_\pi^*(s_\iota)$. Sometimes, we evaluate multiple (different) policies on an MDP or evaluate a single policy on multiple MDPs. To that end, we may incorporate these elements into the notation and write $\rho(\pi, M, \varphi)$ for the performance of the policy π on the MDP M for objective φ, defined as:

$$\rho(\pi, M, \varphi) = V_{\pi,M,\varphi}(s_\iota).$$

When clear from the context, we may omit any of the symbols for the policy, MDP, or objective.

2.2.3 Policy Iteration

As an alternative to value iteration, MDPs can also be solved through *policy iteration*. Policy iteration consists of two alternating steps: the previously discussed policy evaluation and *policy improvement*.

After evaluating the current policy π and determining its value function V_π^*, the *policy improvement* step looks for a new policy π' that outperforms the current policy as follows. First, compute the state-action values under π as

$$Q_\pi(s,a) = R(s,a) + \sum_{s'} P(s'|s,a) V_\pi^*(s'), \quad \forall s \in S, a \in A(s).$$

The new policy π' is extracted as $\pi'(s) = \operatorname{argmax}_{a \in A} Q_\pi(s,a)$ for all $s \in S$ and has a value at least as good as the previous policy, *i.e.*, $V_{\pi'}^* \geq V_\pi^*$. This process starts with any initial policy and terminates as soon as the policy does not change anymore: $\pi' = \pi$, after which π is guaranteed to be optimal.

Variations and Modifications Towards Other Objectives

Several variations to value iteration have been introduced to resolve issues with accuracy and convergence. Most notably, there are *bounded value iteration* (Brázdil et al., 2014; Haddad and Monmege, 2014), *interval iteration* (Baier et al., 2017) *optimistic value iteration* (Hartmanns and Kaminski, 2020) and *sound value iteration* (Quatmann and Katoen, 2018). An extensive experimental evaluation comparing several methods for solving MDPs can be found in (Hartmanns et al., 2023).

Many other objectives, such as reachability and discounted reward, can be solved by straightforward modifications to the Bellman equation. For maximizing

the reachability probability of a target set $T \subseteq S$, the reward function is removed and the preprocessing step is changed to set $Q(s,a) = 1$ for all $(s,a) \in T \times A$, and $Q(s,a) = 0$ for all $(s,a) \in S^{\infty} \times A$. For discounted reward, the preprocessing is removed altogether, and all state-action pairs are initialized with $Q(s,a) = 0$. The Bellman equation from Equation 2.1 is modified for both cases, respectively:

$$Q^{(n+1)}(s,a) = \sum_{s' \in S} P(s' \mid s,a) V^{(n)}(s'), \qquad \text{(reachability)}$$

$$Q^{(n+1)}(s,a) = R(s,a) + \gamma \sum_{s' \in S} P(s' \mid s,a) V^{(n)}(s'). \qquad \text{(discounted reward)}$$

These modifications can also be directly applied to the state-value function V from Equation 2.1. For stochastic shortest path, we simply need to replace the maximization in the equations for reach-reward by minimization.

2.2.4 Linear Programming

As an alternative to dynamic programming, many objectives for MDPs can also be naturally encoded as a linear optimization problem (LP). These LPs can be solved in polynomial time for many objectives, including the ones we consider (Baier and Katoen, 2008).

We now present the LP formulation for the reach-reward objective, and discuss the adjustments necessary afterward. The LP formulation uses the same preprocessing step as value iteration. Let $\{v_s \in \mathbb{R} \mid s \in S^?\}$ be a set of variables for the state values. The LP for *maximizing* reach-reward is then given by

$$\begin{aligned} &\text{Minimize} && v_{s_I} \\ &\text{Subject to} && \\ &\forall s \in T: && v_s = 0, \\ &\forall s \in S^?, a \in A(s): && v_s \geq R(s,a) + \sum_{s' \in S} P(s' \mid s,a) \cdot v_{s'}. \end{aligned}$$

Indeed, the LP optimization direction is opposite of the objective, *i.e.*, maximizing objectives are encoded by a minimizing LP. After solving the LP, the variable assignments give the state values: $\forall s \in S^?: \ V^*(s) = v_s$.

The LP for the stochastic shortest path objective is the opposite, *i.e.*, the LP becomes a maximization problem, and the state value variables v_s are now constrained from above:

$$\begin{aligned} &\text{Maximize} && v_{s_I} \\ &\text{Subject to} && \\ &\forall s \in T: && v_s = 0, \\ &\forall s \in S^?, a \in A(s): && v_s \leq R(s,a) + \sum_{s' \in S} P(s' \mid s,a) \cdot v_{s'}. \end{aligned}$$

For reachability objectives we omit the reward function from the encoding, while for discounted reward we omit the preprocessing step and remove the target set T.

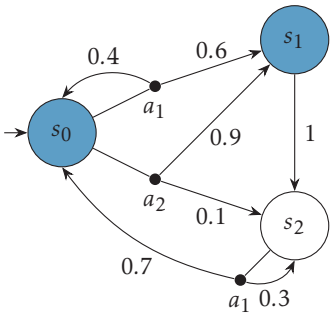


Figure 2.2: An example of a POMDP, as also shown in Chapter 1.

2.3 State Uncertainty: Partially Observable MDPs

Next, we introduce *partially observable* MDPs (POMDPs; Åström, 1965; Kaelbling et al., 1998; Smallwood and Sondik, 1973). POMDPs extend MDPs with *state uncertainty*, meaning the agent has no precise information on which state they are currently in. Observations (hopefully) provide useful information to infer the underlying state.

Definition 2.7 (Partially observable Markov decision process)**.** A partially observable Markov decision process (POMDP) is a tuple $\langle S, b_\iota, A, P, R, Z, O\rangle$ where S is a finite set of states with $b_\iota \in \mathcal{D}(S)$ the initial belief, A is a finite set of actions, $P\colon S \times A \rightharpoonup \mathcal{D}(S)$ is the probabilistic transition function, $R\colon S \times A \rightharpoonup \mathbb{R}$ is the reward function, Z is a finite set of observations, and $O\colon S \times A \rightharpoonup \mathcal{D}(Z)$ is the probabilistic observation function.

We again define $A(s) = \{a \in A \mid P(s,a) \neq \bot\}$ to be the set of enabled actions in state s, and assume the partial functions to be consistent with each other, *i.e.*,

$$\forall s \in S, a \in A(s)\colon P(s,a) \neq \bot \wedge R(s,a) \neq \bot \wedge O(s,a) \neq \bot.$$

An example of a POMDP is illustrated in Figure 2.2.

Paths through a POMDP are the same as for MDPs: sequences of successive states and actions. As the agent cannot observe the states in a POMDP, the agent only observes a *history*. A history of a POMDP M is a sequence of observations and actions in $(Z \times A)^* \times Z$. A history $\langle z_0, a_0, z_1, a_1, \ldots\rangle$ is *valid* for M if there exists a path $\langle s_0, a_0, s_1, a_1, \ldots\rangle \in \mathsf{Paths}_M$ that can generate h. That is, for all observations z_i we have $O(z_i \mid s_{i+1}, a_i) > 0$. Indeed, semantically, the observations of a POMDP are given by the successor state the agent arrives at after taking an action. The set of all (in)finite histories in POMDP M is denoted by Hists_M. When clear from the context, we omit M and simply write Hists.

Histories can be compressed into *beliefs*. A belief is a distribution over the states of a POMDP, *i.e.*, $b \in \mathcal{D}(S)$. Beliefs are *sufficient statistics* for the history of the POMDP (Åström, 1965). Given a belief $b \in \mathcal{D}(S)$, an observation $z \in Z$ and an action

$a \in A$, the successor belief $b' \in \mathcal{D}(S)$ is computed via the following *belief update* rule (Kaelbling et al., 1998):

$$b'(s'|b,z,a) = \frac{O(z|s',a) \cdot \sum_{s \in S} P(s'|s,a) \cdot b(s)}{\sum_{s'' \in S} O(z|s'',a) \cdot \sum_{s \in S} P(s''|s,a) \cdot b(s)}.$$

Essentially, the updated belief b' in state s' is computed by the probability of the agent observing z when arriving in state s' after taking action a, multiplied by the probability of arriving in s' given the current belief b. The denominator is a normalization constant to ensure the updated belief b' forms a probability distribution.

For a finite history h concatenated with action a and observation z, *i.e.*, $h : a : z$, the belief $b(s|h : a : z)$ is computed recursively by applying the belief update rule:

$$b'(s'|h : a : z) = b'(s'|b(\cdot|h), a, z).$$

until the history h is empty, denoted $\emptyset$, and where $b(\cdot|\emptyset)$ is the initial belief.

Using histories or beliefs, we can map a POMDP to a fully observable MDP *history MDP* (Silver and Veness, 2010) or *belief MDP* (Kaelbling et al., 1998).

2.3.1 Policies for POMDPs

As the agent can no longer observe the states in a POMDP, they cannot base their decision on the state information either, and thus policies for POMDPs map histories to (distributions over) actions instead of paths. That is, a policy (for a POMDP) is of the form $\pi : \mathsf{Hists} \to \mathcal{D}(A)$. Since beliefs are sufficient statistics for histories, policies may equivalently be defined as $\pi : \mathcal{D}(S) \to \mathcal{D}(A)$. As with policies for MDPs, π is deterministic if it only maps to Dirac distributions and finite-memory if it can be encoded by a finite-state controller.

FSCs for POMDPs are defined analogously to those for MDPs, except that they now use the observations Z instead of the states S in their memory update function and action mapping.

Definition 2.8 (FSC for POMDP)**.** Let $M = \langle S, b_\iota, A, P, R, Z, O \rangle$ be a POMDP. A finite-state controller for M is a tuple $\langle \mathcal{N}, n_\iota, \alpha, \eta \rangle$ where $\mathcal{N}$ is a finite set of memory nodes with $n_\iota \in \mathcal{N}$ the initial node, and $\alpha : \mathcal{N} \times Z \to \mathcal{D}(A)$ and $\eta : \mathcal{N} \times Z \times A \to \mathcal{D}(\mathcal{N})$ are action mapping and the memory update function.

The size of the FSC is the number of memory nodes $|\mathcal{N}|$. A policy is memoryless when it can be encoded by an FSC with a single (trivial) memory node: $|\mathcal{N}| = 1$. An example FSC is illustrated in Figure 2.3.

As with MDPs, we can also evaluate the performance of an FSC in a POMDP by constructing a Markov chain. This definition is analogous to the construction for MDPs (Definition 2.4) except that we need to account for the observations.

Definition 2.9 (Induced DTMC of an FSC on a POMDP)**.** Let $M = \langle S, b_\iota, A, P, R, Z, O \rangle$ be a POMDP and $\mathcal{F} = \langle \mathcal{N}, n_\iota, \alpha, \eta \rangle$ be an FSC representing a finite-memory policy π. The induced DTMC is defined as $M_\mathcal{F} = \langle S \times \mathcal{N}, \mu_\iota, P_\mathcal{F}, R_\mathcal{F} \rangle$, where the states are

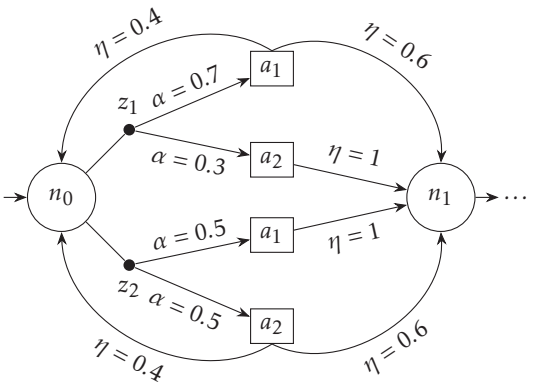


Figure 2.3: A partial example FSC for a POMDP with observations $Z = \{z_1, z_2\}$. The probabilities of the action mapping are indicated by α and the probabilities of the memory update are indicated by η. For instance, starting in memory node n_0 and observing z_1, action a_1 is selected with probability 0.7, while a_2 is drawn with probability 0.3. If a_1 is drawn, the memory node is updated to n_2 with probability 0.6 or remains n_0 with probability 0.4. The action mapping and memory update for node n_1 are omitted.

given by the product of the POMDP states S and FSC memory nodes $\mathcal{N}$, the initial state (distribution) is given by $\mu_\iota(\langle s, n\rangle) = b_0(s) \cdot \mathbb{1}[n = n_0]$, and the transition and reward functions are defined as

$$P_{\mathcal{F}}(\langle s', n'\rangle \mid \langle s, n\rangle) = \sum_{a \in A} \sum_{z \in Z} \alpha(a \mid n, z) \cdot O(z \mid s', a) \cdot P(s' \mid s, a) \cdot \eta(n' \mid n, z, a),$$
$$R_{\mathcal{F}}(\langle s, n\rangle) = \sum_{a \in A} \sum_{z \in Z} \alpha(a \mid n, z) \cdot O(z \mid s', a) \cdot R(s, a).$$

Using the induced DTMC from Definition 2.9, we formally define policy evaluation for a policy represented by an FSC on a POMDP as follows.

Definition 2.10 (FSC policy evaluation)**.** Let M be a POMDP and π a finite-memory policy represented by FSC $\mathcal{F}_\pi$. The performance of π on M for objective $\varphi \in \Phi$ is defined as the value of the Markov chain $M_{\mathcal{F}_\pi}$ weighed by the initial belief

$$\rho(\pi, M, \varphi) = \sum_{s \in S} b_\iota(s) \cdot V_{M_{\mathcal{F}_\pi}, \varphi}(s).$$

FSC policy evaluation encodes the memory nodes into the state space of the POMDP, after which it is reduced to evaluating a memoryless policy. Sometimes, it is convenient to already encode the FSC memory into a POMDP when the action mapping and memory update function are not (yet) defined. Computing an FSC of k memory nodes for a POMDP M can be reduced to computing a memoryless policy on the *k-unfolding* of M (Junges et al., 2018).

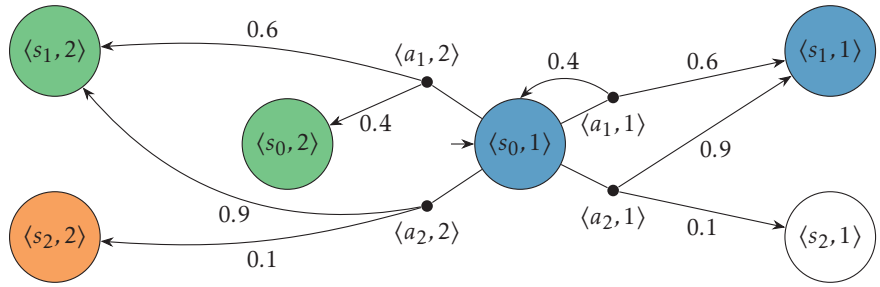


Figure 2.4: Example 2-unfolding of state s_0 of the POMDP from Figure 2.2.

Definition 2.11 (k-unfolding of a POMDP)**.** Let $M = \langle S, b_\iota, A, P, R, Z, O\rangle$ be a POMDP and $\mathcal{N}$ a finite set of k memory nodes. We construct the product POMDP defined by the tuple $M \otimes \mathcal{N} = \langle S \times \mathcal{N}, b_\iota^\otimes, A^\otimes, P^\otimes, R^\otimes, Z^\otimes, O^\otimes\rangle$ as follows. The states are the product $S \times \mathcal{N}$, with initial belief is $b_\iota^\otimes(s,n) = b_\iota(s) \cdot \mathbb{1}[n = n_\iota]$. The actions $A^\otimes = A \times \mathcal{N}$ are tuples of actions and memory nodes. The transition and reward functions are given by

$$P^\otimes(\langle s', n'\rangle \mid \langle s, n\rangle, \langle a, \hat{n}\rangle) = \begin{cases} P(s' \mid s, a) & \text{if } \hat{n} = n', \\ 0 & \text{otherwise,} \end{cases}$$
$$R(\langle s, n\rangle, a) = R(s, a).$$

Finally, the set of observations is $Z^\otimes = Z \times \mathcal{N}$ with the observation function

$$O^\otimes(\langle z, \dot{n}\rangle \mid \langle s, n\rangle, \langle a, \hat{n}\rangle) = \begin{cases} O(z \mid s, a) & \text{if } \dot{n} = \hat{n}, \\ 0 & \text{otherwise.} \end{cases}$$

We illustrate the idea of k-unfolding in the following example.

> *Example* 4 (2-unfolding of a POMDP). In Figure 2.4, we show the 2-unfolding of state s_0 of the POMDP from Figure 2.2. The unfolding for the other states works analogously. States are now pairs of the original states and the memory nodes. Actions are now also pairs of the original actions and the memory nodes, such that the agent's (memoryless) policy in the unfolded POMDP selects both an action as well as a memory node to update to. The set of observations and the observation function are extended such that the agent can observe the current memory node while still adhering to the partial observability of the original state space.

Computing Policies for POMDPs

Where the objectives we consider in this thesis, *i.e.*, reachability, reach-reward, stochastic shortest path, and discounted reward, all admit optimal policies computable in polynomial time in MDPs, this is not the case for these objectives in

POMDPs. Computing optimal policies for infinite and indefinite horizon objectives for POMDPs is, in general, undecidable (Madani et al., 2003). Even when restricting to memoryless policies, finding an optimal policy within that subset is already NP-hard (Vlassis et al., 2012).

A plethora of algorithms to compute policies for POMDPs exist. Most of these methods attempt to approximate the optimal policy or restrict to more tractable subclasses of policies. In particular, many approaches rely on constructing (part of) the belief MDP and performing value iteration (Kaelbling et al., 1998). Notable variants include point-based value iteration (Kurniawati et al., 2008; Pineau et al., 2003; Spaan and Vlassis, 2005) and heuristic search value iteration (Smith and Simmons, 2004). Additionally, there are policy iteration methods (Ji et al., 2007), possibly using FSCs as policy representation (Poupart and Boutilier, 2003) and convex optimization approaches for FSCs (Amato et al., 2010; Junges et al., 2018), as well as Monte-Carlo tree search based methods (Silver and Veness, 2010). Finally, recurrent neural networks have been used to search the space of FSCs (Carr et al., 2021). For a more detailed discussion on many of these approaches, we refer to (Spaan, 2012).

3

A Tutorial on Robust Markov Decision Processes

This chapter presents an overview of the theory of robust Markov decision processes (RMDPs), their semantics and structural assumptions, and how to employ robust dynamic programming to solve RMDPs. The contents of this chapter serve both as a short tutorial, providing a concise introduction to RMDPs and context and references for their applications in AI and formal methods, and as background and preliminary material for the subsequent chapters of this thesis.

3.1 Introduction

Markov decision processes (MDPs) are a fundamental model for tackling decision-making under uncertainty across various areas, such as formal methods (Katoen, 2016), operations research (Puterman, 1994), and artificial intelligence (Sutton and Barto, 1998). Yet, at the core of MDPs is the assumption that the transition probabilities are precisely known, a requirement that is often prohibitive in practice and may lead to suboptimal outcomes (Goyal and Grand-Clément, 2023; Mannor et al., 2007). For example, in data-driven applications in AI such as reinforcement learning (RL) (Sutton and Barto, 1998), transition probabilities are unknown and can only be estimated from data. Furthermore, in formal verification problems, the state-explosion problem often prevents the model from being fully built (Baier and Katoen, 2008; Baier et al., 2019; Clarke et al., 2011). As a remedy, sampling-based approaches that only estimate the transition probabilities, such as statistical model checking (Ashok et al., 2019; Legay et al., 2010), are used. Any sampling-based approach naturally carries the risk of statistical errors and hence, incorrect estimates of the probabilities.

Robust MDPs (RMDPs) overcome this assumption of precise knowledge of the probabilities. An RMDP contains an *uncertainty set* that captures all possible transition functions from which an adversary, typically called *nature*, may choose. Tracing

back to at least interval Markov chains in the formal methods community (Jonsson and Larsen, 1991) and bounded-parameter MDPs in AI (Givan et al., 2000), RMDPs provide a general and flexible framework for modelling MDPs with uncertainty on the transition probabilities (Iyengar, 2005; Nilim and Ghaoui, 2005; Wiesemann et al., 2013). RMDPs provide a rigorous approach to quantifying the impact of data-driven methods for MDPs and, as such, represent an important topic in the intersection of AI and formal methods.

As RMDPs are studied across several research fields, results inevitably become scattered across the communities. While several recent algorithmic developments and applications of RMDPs stem from AI and operations research, see *e.g.*, (Badings et al., 2023b; Goyal and Grand-Clément, 2023; Ho et al., 2021) and many of the other works cited in this thesis, tool support is arguably more mature in the formal methods community. While several of the significant contributions to dynamic programming for RMDPs can be traced back more to the AI than the formal methods community, these algorithms have been implemented in probabilistic model checkers such as PRISM (Kwiatkowska et al., 2011) and Storm (Hensel et al., 2022), which are well-known within formal methods but less so in AI. Because work on RMDPs in formal methods and AI faces many of the same problems, we believe that research in both communities can benefit greatly from each other. In particular, theoretical contributions in one field may improve tool support in the other. Conversely, improved tool support may lead to more advanced applications of RMDPs across both research areas.

This chapter presents a brief tutorial and survey to unify the views on RMDPs from the AI and formal methods communities. While the theory of RMDPs has made significant advances over the years, surveys summarizing these results are, as of yet, sparse. To the best of our knowledge, (Ou and Bi, 2024) is the only other survey on RMDPs available, primarily focusing on summarizing recent technical results. In contrast, we aim to provide an introduction to the theory of robust MDPs and a short review of its connections with other well-known models and applications in the areas of formal methods and AI.

3.2 Robust Markov Decision Processes

In the following, let X be a set of variables. An *uncertainty set* $\mathcal{U}$ is a non-empty set of variable assignments subject to some constraints and is defined as $\mathcal{U} = \{f : X \to \mathbb{R} \mid \text{constraints on } f\}$.

Definition 3.1 (RMDP)**.** A robust Markov decision process (RMDP) is a tuple $\langle S, s_\iota, A, \mathcal{P}, R \rangle$, where the states S, initial state s_ι, actions A and reward function R are defined as for standard MDPs (Definition 2.1), and $\mathcal{P} : \mathcal{U} \to (S \times A \rightharpoonup \mathcal{D}(S))$ is the *uncertain transition function*.

Essentially, the uncertain transition function $\mathcal{P}$ is a set of standard transition functions $P : S \times A \rightharpoonup \mathcal{D}(S)$, and we may thus also write $P \in \mathcal{P}$ for a transition function P that lies inside the uncertain transition function.

While strictly speaking not required, it is convenient to define the set of variables

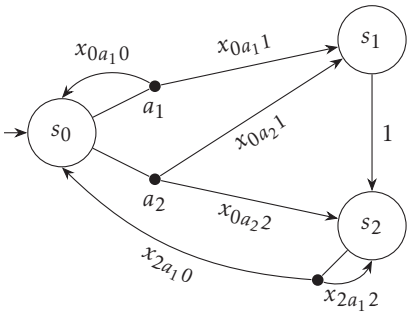


Figure 3.1: An example RMDP.

X to have a unique variable for each possible transition of the RMDP, such that $X = \{x_{sas'} \mid (s,a,s') \in S \times A \times S\}$. The uncertainty set $\mathcal{U}$ is then a set of variable assignments, *i.e.*, functions that map each variable to a real number, subject to constraints. These constraints may, for example, define each variable's allowed range and encode dependencies between different variables. Note that we do not explicitly add a constraint that each state-action pair is assigned a valid probability distribution but leave this implicit in the definition of the uncertain transition function $\mathcal{P}$. Alternatively, one can define RMDPs by having the uncertain transition function assign a function over the variables to each transition, effectively encoding the dependencies there, and having the uncertainty set only define the range of each variable. This construction would, however, require additional adjustments to move most of the discussion that follows (most notably around rectangularity) from the uncertainty set to the uncertain transition function.

Example 5. Figure 3.1 depicts an MDP and an RMDP. Below are three possible uncertainty sets for this RMDP:

$$\begin{aligned}
\mathcal{U}_1 &= \left\{x_{0a_1 1} \in [0.1, 0.9] \wedge x_{0a_2 1} \in [0.1, 0.9] \wedge x_{2a_1 0} \in [0.1, 0.9]\right\}, \\
\mathcal{U}_2 &= \left\{x_{0a_1 1} \in [0.1, 0.4] \wedge x_{0a_2 1} = 2x_{0a_1 1} \wedge x_{2a_1 0} \in [0.1, 0.9]\right\}, \\
\mathcal{U}_3 &= \left\{x_{0a_1 1} \in [0.1, 0.4] \wedge x_{0a_2 1} = 2x_{0a_1 1} \wedge x_{2a_1 0} = x_{0a_1 1}\right\}.
\end{aligned} \tag{3.1}$$

The agent can choose between action a_1 and a_2 in state s_0 and has singleton choices in the other states. An adversary, often called *nature*, chooses variable assignments for $x_{0a_1 0}$, $x_{0a_1 1}$, $x_{0a_2 1}$, $x_{0a_2 2}$, $x_{2a_1 0}$, and $x_{2a_1 2}$. As mentioned above, the restriction that each state-action pair is assigned a valid probability distribution is implied by the definition of the uncertain transition function $\mathcal{P}$. We can therefore focus purely on the choices of $x_{0a_1 0}, x_{0a_2 1}$, and $x_{2a_1 0}$.

The uncertainty sets give restrictions on the possible variable assignments. In uncertainty set $\mathcal{U}_1$, variables $x_{0a_1 0}, x_{0a_2 1}$, and $x_{2a_1 0}$ can each be given any value in the interval $[0.1, 0.9]$. A possible variable assignment in $\mathcal{U}_1$ is $f_1 =$

$\{x_{0a_10} \mapsto 0.3, x_{0a_21} \mapsto 0.1, x_{2a_10} \mapsto 0.8\}$. This variable assignment is not possible in uncertainty sets $\mathcal{U}_2$ and $\mathcal{U}_3$ because of the dependencies between the variables. For example, in uncertainty set $\mathcal{U}_2$, variable x_{0a_21} must be assigned twice the value of x_{0a_10}, whose value must now be in the interval $[0.1, 0.4]$. A possible variable assignment in $\mathcal{U}_2$ is $f_2 = \{x_{0a_10} \mapsto 0.3, x_{0a_21} \mapsto 0.6, x_{2a_10} \mapsto 0.8\}$. We further discuss the effect of dependencies in the uncertainty set in Section 3.3.

We consider the same type of objectives and objectives as for MDPs: *reachability*, *reach-avoid*, *discounted reward*, and *reach-reward*. We focus our detailed discussion of RMDP semantics and robust dynamic programming around reach-reward, and discuss the necessary modifications for the other objectives afterward.

3.3 RMDP Semantics

RMDPs can be seen as a game between the *agent*, who aims to maximize their reward by selecting actions, and an adversarial *nature*, who seeks to minimize the agent's reward by selecting variable assignments from the uncertainty set. Hence, nature simulates the worst-case transition function the agent should be robust against. This game interpretation can be fully formalized into a zero-sum stochastic game (SG), as discussed in Section 3.6.

Intuitively, the game is constructed by adding a new set of states $S \times A$ for nature that consists of tuples of the state-action pairs the agent was in. At each such state-action pair, nature selects a variable assignment from the uncertainty set that determines the transition function $P \in \mathcal{P}$.

The precise rules of the game, and with that the semantics of RMDPs, that determine which variable assignments nature is allowed to choose are controlled by two factors: (1) possible dependencies between nature's choice of the variable assignments between different states or actions, known as (non)-rectangularity; and (2) whether previous choices by nature restrict its future choices, known as the static and dynamic uncertainty semantics. These two factors determine the available policies, *i.e.*, transition functions, for nature, and thus the worst-case transition function that the agent must be robust against. We now discuss both concerns in more detail.

Dependencies between the variables, or lack thereof, immediately follow from the constraints that define the uncertainty set $\mathcal{U}$. *Independence* between states or state-action pairs is commonly referred to as *rectangularity*. Informally, an uncertainty set $\mathcal{U}$ is state-action or (s,a)-rectangular if there are no dependencies between the constraints on the variables at different state-action pairs, and state or s-rectangular if there are no dependencies between constraints on the variables at different states. More formally, using standard notation (Wiesemann et al., 2013):

Definition 3.2 (Rectangularity)**.** The uncertainty set $\mathcal{U}$ is (s,a)-*rectangular* if it can be split into lower dimensional uncertainty sets $\mathcal{U}_{(s,a)}$ that only relate to the variables at the respective state-action pair (s,a), such that their product forms the whole uncertainty set: $\mathcal{U} = \bigtimes_{(s,a)\in S\times A} \mathcal{U}_{(s,a)}$. Similarly, an uncertainty set $\mathcal{U}$ is *s-rectangular*

Uncertainty set & rectangularity		Optimal policy class	Complexity
Convex	(s,a)-rectangular	Memoryless, deterministic	Polynomial
	s-rectangular	Memoryless, randomized	Polynomial
	non-rectangular	History, randomized	NP-hard
Nonconvex	(s,a)-rectangular	Memoryless, deterministic	NP-hard
	s-rectangular	History, randomized	NP-hard
	non-rectangular	History, randomized	NP-hard

Table 3.1: Policy classes that are sufficient for optimality and the computational complexity of policy evaluation for RMDPs with discounted reward objectives with various types of uncertainty sets under static uncertainty semantics, as identified by Wiesemann et al. (2013).

if $\mathcal{U}$ can be split into lower dimensional uncertainty sets $\mathcal{U}_{(s)}$ that only relate to variables at state s, such that $\mathcal{U} = \bigtimes_{s \in S} \mathcal{U}_{(s)}$.

> *Example* 6. We revisit the RMDP in Figure 3.1 and the three possible uncertainty sets in Equation 3.1. The set $\mathcal{U}_1$ is an (s,a)-rectangular uncertainty set, as each variable influences the transition probabilities in only one state-action pair. In other words, there are no dependencies between constraints on the variables at different state-action pairs. In $\mathcal{U}_2$ the transition probabilities for state-action pairs $\langle s_0, a_1 \rangle$ and $\langle s_0, a_2 \rangle$ both depend on variable x_{0a_11}. Therefore, $\mathcal{U}_2$ no longer has independence between actions but is still s-rectangular. The final uncertainty set, $\mathcal{U}_3$, has dependencies between all variables and is, therefore, *non-rectangular*.

The type of rectangularity has, together with whether the uncertainty set is convex or not, direct consequences for the computational complexity of policy evaluation, *i.e.*, computing the performance of a given policy, as well as the type of policy that is required to be optimal for discounted reward objectives under static uncertainty semantics. These results are due to Wiesemann et al. (2013) and presented in Table 3.1.

In s-rectangularity, an additional assumption is made that nature can no longer observe the last action of the agent. This assumption is mentioned explicitly in (Ho et al., 2018, 2021; Wiesemann et al., 2013) but often left implicit. Note that this assumption does influence the optimal policy and value, as demonstrated by Example 7. The question of last-action observability corresponds to the difference between *agent first* and *nature first* semantics in (Bovy et al., 2024).

> *Example* 7 (Last-action observability). Figure 3.2 depicts an RMDP. Below is an s-rectangular uncertainty set:
>
> $$\mathcal{U} = \left\{ x_{0a_11} \in [0.1, 0.9] \wedge x_{0a_11} = x_{0a_22} \right\}.$$

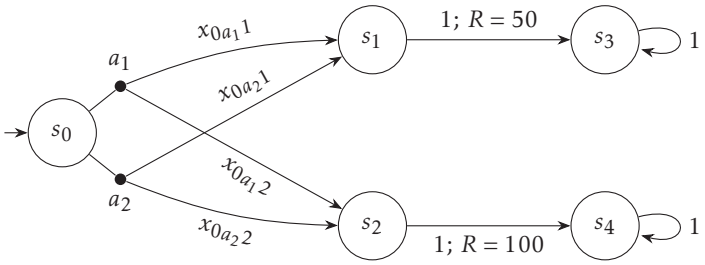


Figure 3.2: An RMDP for Example 7.

Whether or not nature observed the agent's last action determines whether or not nature has to take the dependency between $x_{0a_1 1}$ and $x_{0a_2 2}$ into account. If nature observes the agent's last action, it can achieve an expected reward of 55 by choosing the maximal $x_{0a_1 1} = x_{0a_2 2} = 0.9$ when observing action a_1, and by choosing the minimal $x_{0a_1 1} = x_{0a_2 2} = 0.1$ when observing action a_2. If nature does not have this information, it has to account for both possible agent actions. The best course of action for nature is choosing $x_{0a_1 1} = x_{0a_2 2} = 0.5$, leading to an expected reward of 75 regardless of the agent's choice.

Static and Dynamic Uncertainty Semantics

The second point about RMDP semantics is whether nature's previous choice at a certain state-action pair should restrict its possible future choices. Iyengar (2005) introduced the notions of *static* and *dynamic* uncertainty semantics. Static uncertainty semantics require nature to play a 'once-for-all' policy. If the state-action pair is revisited, nature must use the same variable assignment from the uncertainty set as before. In contrast, under dynamic uncertainty semantics, nature plays memoryless and is free to choose any variable assignment at every step. Simultaneous but independently, Nilim and Ghaoui (2005) introduced these semantics as *time-stationary* and *time-varying* uncertainty models. Note that these notions have only been introduced for (s,a)-rectangular RMDPs and are only of concern in cyclic, infinite horizon models. Interestingly, Iyengar (2005) also shows that the distinction between static and dynamic uncertainty does not matter for reward maximization in (s,a)-rectangular RMDPs. A similar result was established for reachability in interval Markov chains in (Chen et al., 2013). We state the result in general in the following lemma.

Lemma 3.3 (Static and dynamic uncertainty coincide (Iyengar, 2005))**.** *Consider an (s,a)-rectangular RMDP where both agent and nature are restricted to memoryless policies*, i.e., *policies of type $\pi\colon S \to \mathcal{D}(A)$. Let π^{st} be the optimal policy under static uncertainty, and π^{dy} be the optimal policy under dynamic uncertainty semantics. The robust values of these policies coincide*, i.e., $V_{\pi^{st}} = V_{\pi^{dy}}$.

The result from Lemma 3.3 extends to the other objectives we consider under the (common) assumption that nature plays memoryless.

3.3.1 Robust and Optimistic Objectives

Knowing that static and dynamic uncertainty coincide for our objectives in (s,a)-rectangular RMDPs, we can formally define *robust* and *optimistic* objectives for all our objectives for the static uncertainty semantics only. Robust, or *pessimistic*, objectives assume nature plays adversarially. That is, when the agent selects actions to maximize, nature selects transition functions to minimize, and vice versa. *Optimistic* objectives assume nature plays cooperatively, *i.e.*, both players maximize or minimize. The resulting robust and optimistic objectives for an RMDP $\mathcal{M}$ with uncertain transition function $\mathcal{P}$ are defined as follows, where M is the MDP with transition function $P \in \mathcal{P}$.

Definition 3.4 (Robust and optimistic objectives)**.** We extend the objectives for MDPs from Definition 2.5 to the following objectives for RMDPs.

Reachability. The robust *reachability objective* is to optimize the probability of reaching a target set $T \subseteq S$:

$$\mathbf{P}_{MaxMin}(\Diamond T) = \max_{\pi \in \Pi} \inf_{P \in \mathcal{P}} \mathbb{P}_{\pi,P}[\omega \in \mathsf{Paths}_M \mid \omega \models \Diamond T].$$

Discounted Reward. The robust *discounted reward* objective is to optimize the discounted sum of expected rewards for some discount factor $\gamma \in (0,1)$:

$$\mathbf{R}_{MaxMin}(\gamma) = \max_{\pi \in \Pi} \inf_{P \in \mathcal{P}} \mathbb{E}_{\pi,P}\left[\sum_{t=0}^{\infty} \gamma^t R(s_t, a_t) \mid s_0 = s_\iota\right],$$

where s_t and a_t are the state and action at step t along a path $\omega \in \mathsf{Paths}_M(s_\iota)$.

Reach-Reward. Let $\omega \in \mathsf{Paths}_M$, the cumulative reward along ω is defined as

$$r(\Diamond T)(\omega) = \begin{cases} \infty & \forall t \in \mathbb{N}\colon s_t \in T, \\ \sum_{t=0}^{\min\{t \mid s_t \models \in T\}-1} R(s_t, a_t) & \text{otherwise.} \end{cases}$$

The robust *reach-reward* objective is to optimize the cumulative reward until reaching a target set $T \subseteq S$:

$$\mathbf{R}_{MaxMin}(\Diamond T) = \max_{\pi \in \Pi} \inf_{P \in \mathcal{P}} \mathbb{E}_{\pi,P}[r(\Diamond T)].$$

Stochastic Shortest Path. The robust *Stochastic shortest path* objective is to optimize the cumulative reward until reaching a target set $T \subseteq S$:

$$\mathbf{R}_{MinMax}(\Diamond T) = \min_{\pi \in \Pi} \sup_{P \in \mathcal{P}} \mathbb{E}_{\pi,P}[r(\Diamond T)].$$

The set of robust objectives is denoted by

$$\underline{\Phi} = \{\mathbf{P}_{MaxMin}(\Diamond T), \mathbf{R}_{MaxMin}(\Diamond T), \mathbf{R}_{MinMax}(\Diamond T), \mathbf{R}_{MaxMin}(\gamma)\}.$$

Adaptations of these objectives to the optimistic case follow straightforwardly by changing the optimization direction for nature, *i.e.*, the second Max or Min. Consequently, we have the following set of optimistic objectives:

$$\overline{\Phi} = \{\mathbf{P}_{MaxMax}(\Diamond T), \mathbf{R}_{MaxMax}(\Diamond T), \mathbf{R}_{MinMin}(\Diamond T), \mathbf{R}_{MaxMax}(\gamma)\}.$$

3.4 Robust Dynamic Programming

In this section, we discuss how value iteration and policy iteration can be adapted rather straightforwardly for (s,a)-rectangular RMDPs.

Remark 3.5 (Graph preservation). For computational tractability of robust dynamic programming, especially of objectives that rely on preprocessing the underlying graph, such as the reach-reward objective we consider, it is often assumed that the uncertainty set should be *graph preserving*. That is, all variable assignments in the uncertainty set $\mathcal{U}$ imply the same topology for the underlying graphs. Hence, if there exists some $P \in \mathcal{P}$ with $P(s,a,s') = 0$ for some transition, then all other $P' \in \mathcal{P}$ should also have $P'(s,a,s') = 0$.

3.4.1 Robust Value Iteration

Recall Equation 2.1, describing value iteration in a standard MDP. In an RMDP, we do not have access to a precisely defined transition function $P\colon S \times A \to \mathcal{D}(S)$. Instead, we have the uncertain transition function $\mathcal{P}$ that defines a set of such transition functions $P \in \mathcal{P}$.

Robust value iteration adapts value iteration by accounting for the worst-case $P \in \mathcal{P}$ at each iteration. This is achieved by replacing the inner sum $\sum_{s' \in S} P(s' \mid s,a) V^n(s')$ by an *inner minimization problem*:

$$\underline{V}^{(n+1)}(s) = \max_{a \in A} \left\{ R(s,a) + \inf_{P \in \mathcal{P}} \left\{ \sum_{s' \in S} P(s' \mid s,a) \underline{V}^{(n)}(s') \right\} \right\}. \tag{3.2}$$

We write $\underline{V}$ instead of V, which is now the *worst-case* or *pessimistic* value of the RMDP. That is, $\underline{V}$ is a lower bound on the value the agent can possibly achieve. *Best-case* or *optimistic* interpretations also exist, which we discuss later.

Under our assumption that the uncertainty set $\mathcal{U}$ is (s,a)-rectangular, we may replace the global minimization problem $\inf_{P \in \mathcal{P}}$ by a local one:

$$\underline{V}^{(n+1)}(s) = \max_{a \in A} \left\{ R(s,a) + \inf_{P(s,a) \in \mathcal{P}(s,a)} \left\{ \sum_{s' \in S} P(s' \mid s,a) \underline{V}^{(n)}(s') \right\} \right\}. \tag{3.3}$$

If, additionally, the uncertainty set $\mathcal{U}$ is convex, for instance, because all constraints are linear, the inner minimization problem can be solved efficiently via, *e.g.*, convex optimization methods. Hence, robust value iteration extends regular value iteration by solving an additional inner problem at every iteration. In general, the computational tractability of RMDPs primarily relies on whether or not this inner problem is efficiently solvable.

As discussed in Section 3.3 and shown in Table 3.1, memoryless deterministic policies are sufficient for optimality in (s,a)-rectangular RMDPs with convex uncertainty sets. Thus, an optimal *robust policy* $\underline{\pi}^*$ can again be extracted via

$$\underline{\pi}^*(s) = \operatorname*{argmax}_{a \in A} \left\{ R(s,a) + \inf_{P(s,a) \in \mathcal{P}(s,a)} \left\{ \sum_{s' \in S} P(s' \mid s,a) \underline{V}^*(s') \right\} \right\}. \tag{3.4}$$

Optimistic Dynamic Programming

Instead of assuming the worst-case from the uncertainty set, we may also assume the agent and nature play cooperatively. That is, both players attempt to maximize the agent's reward. We instead obtain *optimistic* values $\overline{V}$ that are computed in the same way as the pessimistic values were, except that the inner minimization problem from Equation 3.2 is now replaced by an inner *maximization* problem:

$$\overline{V}^{(n+1)}(s) = \max_{a \in A} \left\{ R(s,a) + \sup_{P \in \mathcal{P}} \left\{ \sum_{s' \in S} P(s,a,s') \overline{V}^{(n)}(s') \right\} \right\}.$$

The optimal *optimistic* policy $\overline{\pi}^*$ is again extracted by one final step of dynamic programming, as in Equation 3.4:

$$\overline{\pi}^*(s) = \operatorname*{argmax}_{a \in A} \left\{ R(s,a) + \sup_{P \in \mathcal{P}} \left\{ \sum_{s' \in S} P(s,a,s') \overline{V}^*(s') \right\} \right\}.$$

3.4.2 Robust Policy Evaluation

We now consider policy evaluation for RMDPs with robust and optimistic objectives.

Definition 3.6 (Robust policy evaluation)**.** The value of a memoryless policy $\pi \colon S \to \mathcal{D}(A)$ is computed by the following robust Bellman equation:

$$\underline{V}_{\pi}^{(n+1)} = \sum_{a \in A} \pi(a|s) \cdot \left(R(s,a) + \inf_{P(s,a) \in \mathcal{P}(s,a)} \left\{ \sum_{s' \in S} P(s'|s,a) \underline{V}_{\pi}^{(n)}(s') \right\} \right).$$

Here, we again use that our uncertainty set is (s,a)-rectangular and convex to ensure an efficiently solvable inner minimization problem.

Robust and Optimistic Performance in RMDPs

Recall that for a policy π, MDP M, and objective φ, the performance of π in M for φ is denoted $\rho(\pi, M, \varphi)$, and defined as the value in the initial state. For RMDPs, the performance of a policy depends on whether we evaluate the policy optimistically or pessimistically, *i.e.*, whether the agent and nature optimize the objective in the same or opposite directions. As such, we denote *robust* or *pessimistic* objectives by $\underline{\varphi}$, and optimistic objectives by $\overline{\varphi}$. The robust performance of a policy π in RMDP $\mathcal{M}$ for objective φ is then defined as $\rho(\pi, \mathcal{M}, \underline{\varphi}) = \underline{V}^*_{\pi,\mathcal{M},\underline{\varphi}}(s_I)$, and the optimistic performance is defined as $\rho(\pi, \mathcal{M}, \overline{\varphi}) = \overline{V}^*_{\pi,\mathcal{M},\overline{\varphi}}(s_I)$. Again, whenever clear from the context, we may omit any of the symbols π, $\mathcal{M}$, or φ.

3.4.3 Robust Policy Iteration

Robust policy iteration (Iyengar, 2005; Kaufman and Schaefer, 2013) extends standard policy iteration in a similar way. We start with some initial memoryless deterministic policy $\pi \colon S \to A$ and perform robust policy evaluation (Definition 3.6). After

convergence, we use the robust state values under the current policy $\underline{V}^*_\pi$ to compute the robust state-action values $\underline{Q}_\pi$:

$$\underline{Q}_\pi(s,a) = R(s,a) + \inf_{P(s,a)\in\mathcal{P}(s,a)} \left\{ \sum_{s'\in S} P(s'|s,a)\underline{V}^*_\pi(s') \right\}.$$

The policy improvement step is performed on these robust state-action values:

$$\pi'(s) = \operatorname*{argmax}_{a\in A} \underline{Q}_\pi(s,a).$$

The process repeats until the policy stabilizes, *i.e.*, $\pi' = \pi$, after which an optimal robust policy $\underline{\pi}^* = \pi'$ has been found.

Methods for s-Rectangular RMDPs

For s-rectangular RMDPs, dynamic programming does not extend so straightforwardly, and a lot of research has focused on finding efficient Bellman operators for various types of uncertainty sets. Most notably, s-rectangular L_1-MDPs (Ho et al., 2018), but also s-rectangular uncertainty sets defined by an L_∞-norm (Behzadian et al., 2021) or ϕ-divergences (Ho et al., 2022). In (Ho et al., 2021), a policy iteration algorithm was introduced. Other methods employ policy gradient techniques (Gadot et al., 2024; Kumar et al., 2023; Wang et al., 2023).

Other Objectives

The RMDP literature primarily focuses on either finite horizon or discounted infinite horizon reward maximization. Adaptation to reachability objectives, such as the reach-reward maximization we consider, is usually straightforward, provided the graph preservation property of Remark 3.5 is met. Temporal logic objectives can be reduced to such reach-reward objectives via a product construction (Wolff et al., 2012). Finally, recent works study average reward (also known as mean payoff) and Blackwell optimality in RMDPs (Chatterjee et al., 2023; Grand-Clément and Petrik, 2023; Grand-Clément et al., 2023). Average reward objectives consider the problem of maximizing the average reward collected in t time steps when $\lim_{t\to\infty}$, and Blackwell optimality balances the standard discounted reward objective by also accounting for long-term reward. A policy is Blackwell optimal if it is optimal for all discount factors sufficiently close to one, *i.e.*, all $\gamma \in [\gamma^*, 1)$ (Puterman, 1994).

Convex Optimization

Given that dynamic programming approaches for MDPs extend with relative ease to RMDPs, especially in the case of (s,a)-rectangular uncertainty sets, a natural question to ask is whether the same goes for convex optimization approaches, and in particular the linear programming (LP) formulation for MDPs. As Iyengar (Iyengar, 2005) already notes, however, that is not the case, and the natural analogue of the LP of MDPs for RMDPs yields, in fact, a nonconvex optimization problem. In contrast,

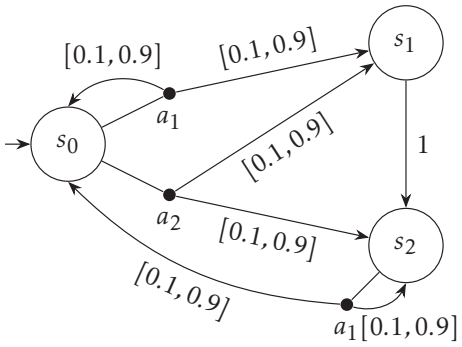


Figure 3.3: An example IMDP.

the optimistic setting does yield tractable LPs via standard dualization techniques, which have been applied to solve PCTL objectives in (s,a)-rectangular RMDPs with convex uncertainty sets (Puggelli et al., 2013).

3.5 Well-Known RMDP Instances

We review common types of RMDPs often used in formal verification and AI, namely interval MDPs, L_1-MDPs, and multi-environment MDPs. For each, we give the more common definition and explain how they fit the RMDP framework.

Interval MDPs

Interval MDPs (IMDPs; Nilim and Ghaoui, 2005), also referred to as bounded-parameter MDPs (Givan et al., 2000) or uncertain MDPs (Wolff et al., 2012), are a special instance of (s,a)-rectangular RMDPs.

Definition 3.7 (IMDP)**.** An interval MDP (IMDP) is a tuple $\langle S, s_\iota, A, \underline{P}, \overline{P}, R\rangle$, where $\underline{P}\colon S \times A \times S \rightharpoonup [0,1]$ and $\overline{P}\colon S \times A \times S \rightharpoonup [0,1]$ are two transition functions that assign lower and upper bounds to each transition, respectively, such that

$$\begin{aligned}
&\forall s, s' \in S, a \in A\colon && \underline{P}(s,a,s') = \bot \iff \overline{P}(s,a,s') = \bot,\\
&\forall s, s' \in S, a \in A(s)\colon && \underline{P}(s,a,s') = 0 \iff \overline{P}(s,a,s') = 0,\\
&\forall s, s' \in S, a \in A(s)\colon && \underline{P}(s,a,s') \leq \overline{P}(s,a,s').
\end{aligned}$$

Our definition of an IMDP requires consistency across enabled actions, as well as that a transition either does not exist (where both $\underline{P}$ and $\overline{P}$ are zero) or is assigned an interval with a non-zero lower bound, thus ensuring graph preservation (Remark 3.5). For IMDPs, however, the statistical model checking literature offers solutions to circumvent this requirement (Ashok et al., 2019; Daca et al., 2016).

Implicitly, IMDPs have a constraint that each state-action pair is required to have a valid probability distribution. An IMDP can also formally be described as an RMDP $(S, s_\iota, A, \mathcal{P}, R)$ where the uncertainty set is of the form $\mathcal{U} = \{f\colon X \to \mathbb{R} \mid$

Algorithm 3.1: Algorithm to solve the IMDP inner problem $\inf_{P \in \mathcal{P}(s,a)}$

1: Sort $S' = \{s'_1, \ldots, s'_m\}$ according to $V^{(n)}$ ascending such that $V^{(n)}(s'_i) \leq V^{(n)}(s'_{i+1})$
2: $\forall s'_i \in S'$: $P(s'_i \mid s,a) \leftarrow 0$
3: $budget = 1 - \sum_{s' \in S'} \underline{P}(s,a,s')$
4: $i \leftarrow 1$
5: **while** $budget - (\overline{P}(s,a,s'_i) - \underline{P}(s,a,s'_i)) \geq 0$ **do**
6: $\quad P(s'_i \mid s,a) \leftarrow \overline{P}(s,a,s'_i)$
7: $\quad budget \leftarrow budget - (\overline{P}(s,a,s'_i) - \underline{P}(s,a,s'_i))$
8: $\quad i \leftarrow i+1$
9: **end while**
10: $P(s'_i \mid s,a) \leftarrow \underline{P}(s,a,s'_i) + budget$
11: $\forall j \in \{i+1, \ldots, m\}$: $P(s,a,s'_j) \leftarrow \underline{P}(s,a,s'_j)$
12: **return** $P(\cdot \mid s,a)$

$\forall (s,a,s') \in S \times A \times S : \; f(x_{sas'}) \in [i,j]_{sas'} \subseteq [0,1] \wedge \forall (s,a) \in S \times A : \; \sum_{s' \in S} f(x_{sas'}) = 1\}$. An example IMDP is depicted in Figure 3.3. Note that this IMDP is precisely the RMDP from Figure 3.1 with uncertainty set $\mathcal{U}_1$, see Example 5.

IMDPs have the nice property that their inner problem can be solved efficiently via a bisection algorithm (Nilim and Ghaoui, 2005), more explicitly given for interval DTMCs in (Katoen et al., 2012) and presented in Algorithm 3.1. This algorithm sorts the successor states $S' = \{s'_1, \ldots, s'_m\}$ of state-action pair (s,a) by the current value $V^{(n)}$ in ascending order. A variable *budget* indicates how much probability mass is still free to assign when we start with assigning the lower bounds to each successor state. Successor states occurring at low indices, *i.e.*, with low values $V^{(n)}$, will be assigned the upper bound of the transition leading to them until the budget runs out. One state will get a remaining *budget* added to its lower bound, which is the first state for which it is no longer possible to replace the lower bound by the upper bound, ensuring the transition probability lies within its interval. The remaining successor states, with high values, will be assigned the lower bounds, as the budget for replacement is now zero. As a result, transition function $P(s,a,\cdot)$ forms a valid probability distribution.

For optimistic dynamic programming, *i.e.*, the case where the inner problem is given by the supremum over the uncertainty set instead of the infimum, we only need to reverse the order in which the states are sorted in line 1 of Algorithm 3.1.

L_1-MDPs

L_1-MDPs (Strehl and Littman, 2008) are another instance of (s,a)-rectangular RMDPs. Whereas IMDPs put an error margin around each individual transition probability, L_1-MDPs put an error margin around each probability distribution at a state-action pair.

Definition 3.8 (L_1-MDP)**.** An L_1-MDP is a tuple $\langle S, s_\iota, A, \tilde{P}, R, d \rangle$, where S, s_ι, A and R are as for standard MDPs, $\tilde{P} \colon S \times A \rightharpoonup \mathcal{D}(S)$ is the centre transition function and

Algorithm 3.2: Algorithm to solve the L_1-MDP inner problem $\inf_{P \in \mathcal{P}(s,a)}$

Sort $S' = \{s'_1, \ldots, s'_m\}$ according to $V^{(n)}$ ascending such that $V^{(n)}(s'_i) \leq V^{(n)}(s'_{i+1})$
$P(s'_1 \mid s,a) \leftarrow \min\{1, \tilde{P}(s'_1 \mid s,a) + d(s,a)/2\}$
$\forall s'_i \neq s'_1 \in S'$: $P(s'_i \mid s,a) \leftarrow \tilde{P}(s'_i \mid s,a)$
$i \leftarrow m$
while $\sum_{j=1}^{m} P(s'_j \mid s,a) > 1$ **do**
    $P(s'_i \mid s,a) \leftarrow \max\{0, 1 - \sum_{j \in \{1,\ldots,m\} \setminus \{i\}} P(s'_j \mid s,a)\}$
    $i \leftarrow i - 1$
end while
return $P(\cdot \mid s,a)$

$d \colon S \times A \to \mathbb{R}_{\geq 0}$ is a distance function assigning a bound to each state-action pair.

An L_1-MDP is an RMDP $(S, s_\iota, A, \mathcal{P}, R)$ where the uncertainty set is given by $\mathcal{U} = \{f \colon X \to \mathbb{R} \mid \forall (s,a) \in S \times A \colon \sum_{s' \in S} |f(x_{sas'}) - \tilde{P}(s' \mid s,a)| \leq d(s,a)\}$, where $d(s,a)$ bounds the L_1-error between the reference distribution $\tilde{P}(s,a)$ and all other distributions.

Similar to IMDPs, the inner optimization problem for L_1-MDPs can be solved efficiently, again by ordering the successor states along their current value and assigning low-ranking states the most possible probability mass and high-ranking states the least probability mass. Specifically, the state with the lowest value s'_s gets probability mass $d(s,a)/2$ added to its estimate $\tilde{P}(s,a,s'_1)$, and the remaining states from high to low get probability mass subtracted up to a total of $d(s,a)/2$, ensuring a valid probability distribution. This algorithm is the dual of the algorithm for computing the optimistic inner problem $\sup_{P \in \mathcal{P}(s,a)}$ of Strehl and Littman (2008) and explicitly given in Algorithm 3.2.

Multi-Environment MDPs

Multi-environment MDPs (MEMDPs; Raskin and Sankur, 2014) model *discrete* uncertainty. Specifically, a MEMDP is a finite set of MDPs that share the same states, actions, and reward function and only differ in their transition functions. Each MDP in a MEMDP is called an *environment*.

Definition 3.9 (MEMDP)**.** A multi-environment MDP (MEMDP) is a tuple $\langle S, s_\iota, A, \{P_i\}_{i \in I}, R \rangle$ where S, s_ι, A, and R are as for MDPs, and $\{P_i \colon S \times A \rightharpoonup \mathcal{D}(S)\}_{i \in I}$ is a set of $I = \{1, \ldots, n\}$ transition functions that are consistent with each other in terms of enabled actions: $\forall (s,a) \in S \times A, \forall i,j \in I \colon P_i(s,a) = \bot \iff P_j(s,a) = \bot$.

A MEMDP is an RMDP $(S, s_\iota, A, \mathcal{P}, R)$ where the uncertainty set $\mathcal{U}$ is discrete: $|\mathcal{U}| \neq \infty$, and $\mathcal{P} = \{P_i\}_{i \in I}$. In general, MEMDPs are *non-rectangular* and follow *static* uncertainty semantics, as nature's choices at each state-action pair must be consistent with, and equivalent to, choosing a single $P_i \in \mathcal{P}$ at the start. As the uncertainty set is discrete, it is also nonconvex. Hence, optimal policies in MEMDPs need to be history-based and randomized; see Table 3.1.

MEMDPs have been studied in both formal methods and AI. In AI, MEMDPs have caught interest because of their applications in robotics, naturally modelling

several possible worlds a robot may act in (Rigter et al., 2021a). Besides being RMDPs, MEMDPs are also a subclass of *partially observable* MDPs (POMDPs), where the agent does not directly observe the states (Kaelbling et al., 1998). In particular, a MEMDP can be transformed into a POMDP by taking the disjoint union of all environments and using the partial observability to hide in which environment the agent is playing (Chatterjee et al., 2020). As a result, quantitative objectives such as reward maximization may be solved by casting the MEMDP to a POMDP and using off-the-shelf POMDP methods.

In formal methods, emphasis has been given to complexity results, especially for *almost-sure* objectives, *i.e.*, objectives that need to be satisfied with probability one. In (Raskin and Sankur, 2014), it is shown that almost-sure parity objectives are in P for MEMDPs of two environments, while (van der Vegt et al., 2023) shows that already for almost-sure reachability, an arbitrary number of environments leads to PSPACE-completeness. Recent work completes the complexity landscape for qualitative objectives in MEMDPs by establishing PSPACE-completeness for almost-sure parity and Rabin objectives (Suilen et al., 2024b). In contrast, almost-sure reachability is already EXPTIME-complete for POMDPs (Chatterjee et al., 2010), and almost-sure parity or Rabin objectives are undecidable (Baier et al., 2008), showing that MEMDPs are an interesting class worth investigating.

3.6 Related Models and Applications

We conclude this chapter by sketching a broader context for RMDPs. In particular, we consider the relation of RMDPs with other models, specifically: parametric MDPs, stochastic games, and models that employ likelihoods of transition functions. Then, we discuss their application to abstraction techniques and give a short summary of the current tool support. Note that robust POMDPs are discussed separately in Chapter 4, and the application of RMDPs in statistical model checking and RL are part of Chapter 5.

Parametric MDPs

Our general definition of RMDPs as given in Definition 3.1 closely resembles that of *parametric* MDPs (pMDPs; Jansen et al., 2022). Indeed, both models assign variables (parameters) to the transitions instead of concrete probabilities, effectively defining a set of possible MDP models. Typically, pMDPs are defined more generally and allow for a rational function over two polynomials on the transitions.

The *parameter synthesis* problem (Dehnert et al., 2016) typically considered in pMDPs is, however, different from the problem of computing robust policies and values in RMDPs. Parameter synthesis asks whether there exists a variable assignment such that *for all* policies, the value of an objective surpasses some threshold. That is, the quantifiers are reversed compared to RMDPs.

Common techniques for parameter synthesis in pMDPs or pMCs include convex optimization approaches (Cubuktepe et al., 2018, 2022), parameter lifting (Quatmann et al., 2016), or exact computation of the solution function (Junges et al., 2024). Tool support for pMDPs can be found in, *e.g.*, Storm (Hensel et al., 2022) or

PROPhESY (Dehnert et al., 2015). Parameter synthesis in pMDPs with memoryless deterministic policies is known to be NP-complete for a fixed number of parameters and ETR-complete for arbitrary numbers of parameters (Junges et al., 2021b).

Stochastic Games

The connection between *stochastic games* (SG) and RMDPs has been noted many times in the RMDP literature. See, *e.g.*, (Goyal and Grand-Clément, 2023; Grand-Clément et al., 2023; Iyengar, 2005; Nilim and Ghaoui, 2005; Wiesemann et al., 2013; Xu and Mannor, 2012). The most explicit game interpretations are given by (Iyengar, 2005; Nilim and Ghaoui, 2005), which both link reward maximization in (s,a)-rectangular RMDPs to turn-based zero-sum stochastic games. This equivalent game is constructed by adding, in addition to the states S, states that correspond with tuples $\langle s,a\rangle$ of states $s \in S$ and actions $a \in A$. Then, the agent controls the original states, and nature controls the tuple states. In a state s, the agent chooses an action a, upon which the game transitions deterministically to the nature state $\langle s,a\rangle$. In this state $\langle s,a\rangle$, nature chooses a variable assignment. Given a nature state $\langle s,a\rangle$ and variable assignment $f \in \mathcal{U}$, the game transitions stochastically to an agent state s' according to $\mathcal{P}(f)(s,a,s')$. The reward function assigns the same value as the reward function of the RMDP in the agent states and zero in nature states.

Changes in the assumptions on nature or the objective require some changes in the translation to stochastic games. Wiesemann et al. (2013) mentions that a similar construction follows for s-rectangular RMDPs, but does not describe the actual game. As s-rectangular RMDPs assume that nature chooses variable assignments without information of the agent's latest action, such games would have to either be partially observable or concurrent. An average reward objective in RMDPs can be linked to zero-sum mean pay-off games (Grand-Clément et al., 2023).

A key difference between RMDPs and SGs is that in RMDPs, it is typically assumed that nature plays memoryless, whereas in SGs, both players are allowed to use history. Grand-Clément et al. (2023) show that the assumption of playing against a memoryless nature is nonrestrictive for discounted reward maximization against a convex and compact s-rectangular uncertainty set.

Likelihoods of Transition Functions

RMDPs describe *sets* of transition functions without any further assumptions or prior knowledge of how likely one transition function over another is. Such prior knowledge may be modeled as a probability distribution over the transition function $\mathcal{P}$ of an RMDP. Models that do account for such prior knowledge, if available, are generally less conservative than RMDPs. Although equivalent, it is often more convenient to model this setting as a pMDP together with a distribution over the parameter values. As such, these models have been named *uncertain parametric MDPs* (upMDPs) in the literature (Badings et al., 2022a). A common verification question is then to obtain a solution that is robust against (for example) at least a 99% probability mass of the distribution.

One important question for upMDPs is whether policies can depend on the

realization of the (uncertain) parameters. For example, one possible assumption is that we can *first* observe the realization of the parameters, and *then* can compute an optimal policy based on this observation. This setting has first been investigated by (Cubuktepe et al., 2020) and in more detail by (Badings et al., 2022a). The other possible assumption is that we *cannot* observe the values of the parameters and instead need to compute a *single* policy that is robust against all weather conditions. This setting has been considered by (Rickard et al., 2023) for upMDPs, and also relates to so-called Bayes-adaptive MDPs, which are commonly used in RL (Costen et al., 2023; Guez et al., 2012; Rigter et al., 2021b). A variant where parameter values cannot be observed and thus must be learned has been studied by (Arming et al., 2018). The latter setting is arguably more difficult to solve due to dependencies between the policies for different weather conditions. However, which of the two assumptions is more appropriate depends on the context.

In practice, it may be unrealistic to have access to an explicit representation of the distributions over transition functions, which motivates sampling-based verification approaches for upMDPs, such as (Cubuktepe et al., 2020; Rickard et al., 2023). A similar setting for continuous-time Markov chains is studied by (Badings et al., 2022b; Rickard et al., 2023). Generally, these approaches assume access to a finite set of parameter samples and aim to compute a solution with statistical (PAC-style) guarantees on its performance on yet another sample from the underlying distribution. For example, (Badings et al., 2022a) obtains PAC guarantees by techniques from *scenario optimization*, which is a methodology to deal with stochastic convex optimization in a data-driven fashion (Campi and Garatti, 2008; Campi et al., 2021).

Applications in Abstraction

RMDPs are commonly used to model abstractions of more complex systems. The general idea is that states of a (non-robust) MDP can be aggregated by overapproximating the transition probabilities in the uncertainty set of an RMDP (Jaeger et al., 2020). This idea at least dates back to (Larsen and Skou, 1991) and has already been identified as an interesting application of RMDPs by Givan et al. (2000). Since then, such abstraction techniques have been used across areas, including formal methods, control theory, and AI.

Game-based abstraction of MDPs in the form of IMDPs has been studied by (Kattenbelt et al., 2010; Kwiatkowska et al., 2006). So-called 3-valued abstractions of Markov chains are developed by (Fecher et al., 2006), who abstract the system into an interval Markov chain whose labelling function has three possible values (true, false, or don't know). A similar approach for 3-valued abstraction of CTMCs is presented by (Katoen et al., 2007). Probabilistic bisimulation of IMDPs to reduce the number of states is considered by (Hashemi et al., 2016).

In control theory, models typically have continuous state and action spaces. A popular approach to synthesizing provably correct control policies is to generate a finite-state abstraction of the continuous model (Abate et al., 2008; Alur et al., 2000; Lahijanian et al., 2015). Under an appropriate simulation relation, satisfaction guarantees of temporal logic formulae carry over from the abstract to the continuous model (Girard and Pappas, 2007). Various papers generate IMDP abstractions of

stochastic systems (Badings et al., 2023a,b; Coppola et al., 2024; Lavaei et al., 2023), and tool support has been developed by, *e.g.*, (Cauchi and Abate, 2019; Wooding and Lavaei, 2024). Similar to game-based abstraction, the general idea is to use the probability intervals to capture uncertainties and abstraction errors. For further details on abstractions in control, we refer to the survey (Lavaei et al., 2022).

Tool Support

General-purpose tool support for RMDPs is still relatively limited compared to other models. The probabilistic model checkers PRISM (Kwiatkowska et al., 2011) and Storm (Hensel et al., 2022) both support basic IMDP model checking, with PRISM's provision slightly more advanced, *e.g.*, in terms of user support for modelling. Storm, on the other hand, has more advanced support for pMDPs, and has been used as a back-end in several of the previously discussed works, *e.g.*, (Badings et al., 2022a,b; Cubuktepe et al., 2022; Suilen et al., 2020). Additionally, there is also the Julia-based IntervalMDP.jl (Mathiesen et al., 2024) for IMDPs.

3.7 Conclusion

This chapter provided an overview of RMDPs and their solution methods, with a detailed account of robust dynamic programming. We also introduced three well-known and commonly used classes of RMDPs: IMDPs, L_1-MDPs, and MEMDPs. We concluded this chapter with a brief summary of the relation of RMDPs to other models and their applications in abstraction. For the relation of RMDPs to robust POMDPs, we refer the reader to Chapter 4. The application of RMDPs in learning settings, specifically statistical model checking and RL, will be discussed in Chapter 5.

4

Finite-Memory Policies for Robust POMDPs

In this chapter, we consider the problem of computing finite-memory policies for robust POMDPs (RPOMDPs). RPOMDPs extend RMDPs with partial observability in the same way POMDPs extend MDPs. Policies for RPOMDPs need not only to be robust against the uncertainty set but also against the state uncertainty stemming from partial observability. As such, we focus on finding finite-memory policies encoded as finite-state controllers. We present two algorithms based on convex optimization approaches to compute such finite-memory policies for RPOMDPs. We state and encode the problem as a nonconvex optimization problem with infinitely many constraints and then present two approaches to solve this optimization problem towards a local optimum. The first approach is to use the convex-concave procedure (CCP) to convexify the problem into a convex quadratically constrained quadratic program and enumerate the vertices of the uncertainty set, yielding a finite convex optimization problem. The second approach uses dualization to make the problem finite and then uses sequential convex programming (SCP) to linearize it into a linear program. Both approaches iteratively convexify or linearize around a previous solution until converging to a local optimum.

4.1 Introduction

Partially observable Markov decision processes (POMDPs) model sequential decision-making problems under stochastic uncertainties and partial information (Kaelbling et al., 1998), and can be seen as an extension of standard MDPs with state uncertainty. Just as with MDPs, POMDPs also rely on the assumption that the transition and observation probabilities are precisely given. This assumption is highly unrealistic in many scenarios.

For example, take an aircraft collision avoidance system that issues advisories to pilots (Kochenderfer, 2015). Modeled as a POMDP, the actions relate to such advice

and concern the flying altitude and the speed. Probabilities enter the model based on data on the reaction time of a pilot and sensor noise in observations regarding the speed and altitude of other aircraft. Depending on the amount and quality of the data, the probabilities derived from the data may be highly inaccurate, leading to poor advisories. Robust POMDPs (RPOMDPs) mitigate precisely such concerns in the same way as RMDPs do: by incorporating an uncertainty set around the transition and observation functions.

While both partially observable Markov decision processes (POMDPs) and robust MDPs have been studied extensively over the last few decades, as discussed in Chapters 2 and 3, their combination, *i.e.*, RPOMDPs, remains relatively under studied. In particular, their semantics have only recently been studied in detail by Bovy et al. (2024), showing that in contrast to (s,a)-rectangular RMDPs, in (s,a)-rectangular RPOMDPs the notions of static and dynamic uncertainty semantics do not coincide in general.

On the algorithmic side, a few approaches to RPOMDPs exist. Osogami (2015) extends standard POMDP approaches of value iteration on the belief space to that of RPOMDPs. Other approaches exist but either consider a different notion of an optimal policy, such as optimal for one instance in the uncertainty set (Itoh and Nakamura, 2007) or optimal given a pessimism level (Saghafian, 2018), or have additional assumptions on the uncertainty set, such as uncertainty in the observation function only (Chamie and Mostafa, 2018) or a distribution over the uncertainty set (Burns and Brock, 2007; Nakao et al., 2021).

4.1.1 Contributions and Approach

Methods that employ finite-memory policies in the form of finite-state controllers (FSCs) for RPOMDPs have long been lacking, and the contribution of this chapter is precisely that. We develop two novel algorithms for efficiently computing robust finite-memory policies for RPOMDPs using robust convex optimization techniques. Specifically, we consider *interval* POMDPs (IPOMDPs), which are the natural extension of IMDPs with partial observability. Nonetheless, the results of this chapter extend to other types of (s,a)-rectangular RPOMDPs. Both algorithms follow the same overall outline and only deviate in one step, as we discuss next and also illustrate in Figure 4.1.

i. *Finite-memory policies via FSCs.* We unfold the memory of an FSC into the state space of a POMDP, following Definition 2.11, which, as we discuss, also applies to IPOMDPs.

ii. *Semi-infinite optimization problems.* We define semi-infinite optimization problems that optimize a memoryless stochastic policy for each objective, *i.e.*, reachability, reach-reward, discounted reward, and stochastic shortest path. These optimization problems are *nonconvex*, since variables of the optimization problems are multiplied, resulting into nonconvex constraints. Furthermore, the optimization problems are *semi-infinite*, as they consist of a finite number of variables but an infinite number of constraints that encode all possible probability distributions in the uncertainty set of the RPOMDP. Such

optimization problems are also known as *robust* optimization problems, and we call the constraints affected by the uncertainty set the *uncertain constraints*.

iii. *Building finite convex optimization problems.* To obtain computationally tractable optimization problems, we need to deal with the nonconvexity and the infinite number of constraints. We resolve these two problems in tandem by employing two approaches from the convex optimization literature: the *convex-concave procedure* (CCP) and *sequential convex programming* (SCP). Both approaches iteratively construct and solve a convex approximation of the original nonconvex problem, yielding efficiently solvable optimization problems at the cost of optimality. Both approaches only converge to a *local* optimum. Applying these two methods to our original semi-infinite optimization problems results in the following two concrete algorithms, which, for convenience, we name after the approximation technique.

(a) *CCP: Convex-concave procedure.* In our first algorithm, we use the *penalty convex-concave procedure* (CCP; Lipp and Boyd, 2016) to build a (still semi-infinite) convex quadratically constrained quadratic program (QCQP) around a previous solution. Robust QCQPs with *polytopic* uncertainty sets, as the ones we have in our setting, can be made finite by enumerating the vertices of the polytope (Löfberg, 2012). Together, the convex-concave procedure and enumeration yield finite, convex QCQPs that are iteratively solved until convergence.

(b) *SCP: Sequential convex programming.* In our second algorithm, we split the concerns of nonconvexity and robustness by first building a *simple* RPOMDP, following the construction of Junges et al. (2018). We obtain constraints that (1) are nonconvex but unaffected by the uncertainty set and (2) constraints that are uncertain but are already linear in return. The uncertain constraints are dualized following standard robust optimization principles (Ben-Tal and Nemirovski, 1998; Bertsimas et al., 2011), while the nonconvex constraints are linearized around a previous solution using *sequential convex programming* (SCP; Mao et al., 2018). Together, dualization and SCP yield finite linear programs that are iteratively solved until convergence.

iv. *Termination.* While both CCP and SCP are guaranteed to eventually converge to a local optimum, the convergence criterion of these techniques can be overly conservative for our purpose of computing robust policies. Therefore, we include a robust policy evaluation step at the end of each iteration. When the performance of the computed policy surpasses a desired threshold, we may already terminate the procedure. An additional benefit is that the resulting value function can also be used to correct numerical inaccuracies of the convex optimization solver.

In our experimental evaluation, we compare both approaches on several benchmarks. Notably, the SCP approach can solve RPOMDPs with hundreds of thousands

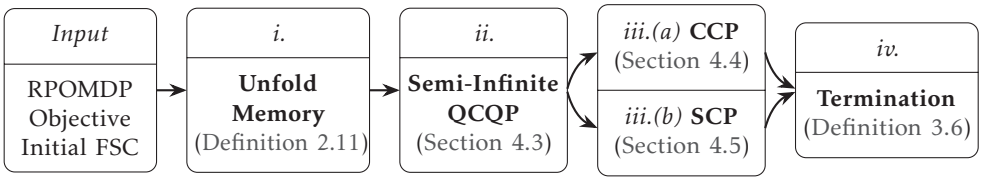


Figure 4.1: Global outline of the approach. More detailed outlines of CCP and SCP are given in their respective sections.

of states, which is out of reach for the belief-based methods of Osogami (2015). In summary, the contribution presented in this chapter is as follows.

Contribution

We present novel algorithms based on convex optimization to compute finite-memory policies for robust POMDPs.

Structure of This Chapter

The remainder of this chapter is structured as follows. We start with the necessary background and definitions in Section 4.2. In Section 4.3, we present the general semi-infinite optimization problems for computing robust policies in RPOMDPs. Then, Section 4.4 details the CCP approach to make the optimization problems tractable, while Section 4.5 details the SCP approach. In Section 4.6, we empirically evaluate and compare both approaches. Finally, in Section 4.7, we conclude the chapter with a discussion of limitations and future work.

4.2 Background: Robust POMDPs

This chapter builds upon POMDPs and RMDPs, as defined in Chapters 2 and 3. For ease of reading, we summarize the definitions here and refer back to the previous chapters for full definitions and discussion. We then briefly introduce robust POMDP semantics and finally define interval POMDPs (IPOMDPs).

4.2.1 Preliminaries

POMDPs. A POMDP (Definition 2.7) is a tuple $\langle S, b_\iota, A, P, R, Z, O\rangle$, where S is a finite set of states, b_ι is the initial belief, A is a finite set of actions, $P\colon S \times A \to \mathcal{D}(S)$ is the transition function, $R\colon S \times A \to \mathbb{R}$ is the reward function, Z is a finite set of observations, and $O\colon S \to Z$ is the observation function. For simplicity, we make the following two assumptions on our POMDPs. First, there is a single, observable, initial state s_ι. Hence, the initial belief is a Dirac distribution, $b_\iota = Dirac(s_\iota)$. Second, the observation function is state-based and deterministic. Recall from Chapter 2 that this second assumption is without loss of generality as any POMDP with a general observation function can be transformed into a (polynomially larger) POMDP with such state-based deterministic observations (Chatterjee et al., 2016).

Policies. A memoryless stochastic policy for a POMDP is a function $\pi\colon S \to \mathcal{D}(A)$. Finite-memory policies are encoded through FSCs (Definition 2.8). An FSC is a tuple $\langle \mathcal{N}, n_\iota, \alpha, \eta \rangle$, where $\mathcal{N}$ is a finite set of memory nodes, n_ι is the initial node, $\alpha\colon \mathcal{N} \times Z \to \mathcal{D}(A)$ is the action mapping, and $\eta\colon \mathcal{N} \times Z \times A \to \mathcal{D}(\mathcal{N})$ is the randomized memory update function. The memory nodes of an FSC can be unfolded into the state space of a POMDP such that computing a memoryless policy on the unfolded POMDP yields an FSC for the original POMDP, see Definition 2.11.

RMDPs. Additionally, recall that an RMDP (Definition 3.1) is a tuple $\langle S, s_\iota, A, \mathcal{P}, R \rangle$, where the uncertain transition function is defined as $\mathcal{P}\colon \mathcal{U} \to (S \times A \rightharpoonup \mathcal{D}(S))$. The uncertainty set $\mathcal{U} = \{f\colon X \to \mathbb{R} \mid \text{ constraints on } f\}$ defines the set of admissible value assignments to the variables in X. An RMDP is (s,a)-*rectangular* if there are no dependencies between the constraints on the variables at different state-action pairs, see Section 3.3 for a more detailed discussion of rectangularity.

IMDPs. An IMDP (Definition 3.7) is an (s,a)-rectangular RMDP of the form $\langle S, s_\iota, A, \underline{P}, \overline{P}, R \rangle$. The transition functions $\underline{P}$ and $\overline{P}$ define lower and upper bounds on each individual transition probability that are required to satisfy the following consistency requirements:

$$\begin{aligned}
&\forall s, s' \in S, a \in A\colon && \underline{P}(s,a,s') = \bot \iff \overline{P}(s,a,s') = \bot, \\
&\forall s, s' \in S, a \in A(s)\colon && \underline{P}(s,a,s') = 0 \iff \overline{P}(s,a,s') = 0, \\
&\forall s, s' \in S, a \in A(s)\colon && \underline{P}(s,a,s') \leq \overline{P}(s,a,s').
\end{aligned} \tag{4.1}$$

Objectives and performance. We consider all four objectives defined in Definition 2.5 in this chapter, *i.e.*, reachability, discounted reward, reach-reward, and stochastic shortest path. We denote these objectives by

$$\varphi \in \Phi = \{\mathbf{P}_{Max}(\Diamond T), \mathbf{R}_{Max}(\Diamond T), \mathbf{R}_{Min}(\Diamond T), \mathbf{R}_{Max}(\gamma)\}.$$

For R(PO)MDPs, we only use the pessimistic extension of these objectives, as defined in Definition 3.4, denoted

$$\underline{\varphi} \in \underline{\Phi} = \{\mathbf{P}_{MaxMin}(\Diamond T), \mathbf{R}_{MaxMin}(\Diamond T), \mathbf{R}_{MinMax}(\Diamond T), \mathbf{R}_{MaxMin}(\gamma)\}.$$

The *performance* of an FSC policy π on a POMDP M for objective $\varphi \in \Phi$ is computed via policy evaluation as defined in Definition 2.9 and denoted by $\rho(\pi, M, \varphi)$. The robust (or worst-case) performance of a memoryless policy $\pi\colon S \to \mathcal{D}(A)$ on R(PO)MDP $\mathcal{M}$ for pessimistic objective $\underline{\varphi} \in \underline{\Phi}$ is denoted $\rho(\pi, \mathcal{M}, \underline{\varphi})$ and computed through robust policy evaluation, see Definition 3.6 and Section 3.4 in general.

Linear programming for MDPs. Recall from Section 2.2.4 that optimizing any of the objectives $\varphi \in \Phi$ in an MDP can be done through a linear program (LP). An LP, and more generally any (non)linear optimization problem consists of a set of *variables* that need to be assigned values in $\mathbb{R}$, an *objective function* that needs

to be maximized or minimized, and *constraints* that need to be satisfied (Boyd and Vandenberghe, 2004). For instance, the reach-reward objective for an MDP $M = \langle S, s_I, A, P, R\rangle$ with target set $T \subset S$ can be encoded into an LP as follows. Let $S^{\infty} \subseteq S$ be the set of states for which there exists a policy that does not reach T almost-surely, and denote the remaining states by $S^{?} = S \setminus (T \cup S^{\infty})$. We define the set of variables $\{v_s \in \mathbb{R} \mid s \in S^{?}\}$. These variables encode the value function $V\colon S \to \mathbb{R}$ at each state $s \in S^{?}$. The encoding for *maximizing* reach-reward is then given by the following *minimzing* LP:

$$\begin{aligned} &\text{Minimize} && v_{s_I} \\ &\text{Subject to} && \\ &\forall s \in T\colon && v_s = 0, \\ &\forall s \in S^{?}, a \in A(s)\colon && v_s \geq R(s,a) + \sum_{s' \in S} P(s' \mid s,a) \cdot v_{s'}. \end{aligned}$$

Solving this LP yields value assignments for each variable, denoted by $\hat{v}_s$, that precisely correspond with the optimal value function: $V^*(s) = \hat{v}_s$. For details on how to encode other objectives for MDPs in LPs we refer back to Section 2.2.4.

Robust POMDP Semantics

We now formally introduce robust POMDPs (RPOMDPs) and their semantics[1]. We start with the general definition of (Bovy et al., 2024). Intuitively, we extend the definition of POMDPs to RPOMDPs the same way as we extended MDPs to RMDPs. However, to maintain the generality of the definition, we separate the sets of observations that belong to the agent, to nature, and those that are visible to both, and similarly split the observation function into three. Finally, we add two more elements to our definition, *stickiness* and *order of play*, which are required to fully specify the semantics of the RPOMDP.

In the following, let $\mathfrak{a}$ denote the agent, $\mathfrak{n}$ nature, and $\mathfrak{an}$ both players.

Definition 4.1 (RPOMDP)**.** An RPOMDP is a tuple $\langle S, s_I, A, \mathcal{P}, R, Z^{\mathfrak{a}}, Z^{\mathfrak{n}}, Z^{\mathfrak{an}}, O^{\mathfrak{a}}, O^{\mathfrak{n}}, O^{\mathfrak{an}}, \mathsf{stick}, \mathsf{order}\rangle$, where $S, s_I, A, \mathcal{P}$ and R are as in RMDPs: a finite set of states, the initial state, a finite set of actions, the uncertain transition function, and the reward function. The finite sets $Z^{\mathfrak{a}}$, $Z^{\mathfrak{n}}$, and $Z^{\mathfrak{an}}$ are the (private) observations for agent and nature and the set of shared (public) observations, respectively. The private and shared observation functions are $O^{\mathfrak{a}}\colon S \to Z^{\mathfrak{a}}$, $O^{\mathfrak{n}}\colon S \to Z^{\mathfrak{n}}$ and $O^{\mathfrak{an}}\colon S \to Z^{\mathfrak{an}}$. The *stickiness* function $\mathsf{stick}\colon X \times Z^{\mathfrak{n}} \times Z^{\mathfrak{an}} \times A \to \{0,1\}$ determines when a variable assignment remains fixed ('sticks') and when nature is free to choose a different value assignment. Finally, the order of play $\mathsf{order} \in \{\mathfrak{a}, \mathfrak{n}\}$ determines which player moves first.

Stickiness generalizes the notions of static and dynamic uncertainty in RMDPs. Recall that static uncertainty essentially requires nature to choose a variable assignment for the RMDP once, while dynamic uncertainty allows nature to change the

[1] Note that while the author contributed to the paper (Bovy et al., 2024), RPOMDP semantics are not a contribution of this thesis and only summarized here as preliminary material.

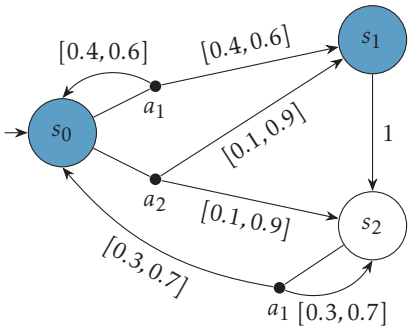


Figure 4.2: An example of an IPOMDP.

variable assignment in the future. These coincide with *full stickiness* and *zero stickiness*, respectively. Other intermediate forms of stickiness can also be constructed; see (Bovy et al., 2024) for discussion and an example.

Together, stickiness and order of play determine the *game semantics* of the RPOMDP. Analogous to RMDPs, RPOMDPs can be fully formalized as a zero-sum partially observable stochastic game (POSG) where the agent selects actions to maximize their reward, and nature selects variable assignments within the uncertainty set to minimize. In RMDPs, nature is typically assumed to play memoryless deterministic. To deal with partial observability, nature plays with memory in RPOMDPs. Stickiness and the order of play determine what variable assignments are available for nature to choose from, while the order of play influences nature's history. For a full account of RPOMDP semantics, including explicit construction of the POSG defining the semantics, we refer to (Bovy et al., 2024).

As with standard POMDPs, the restriction to deterministic state-based observation functions is without loss of generality. For RPOMDPs, too, every (uncertain) observation function can be transformed by expanding the state space and encoding the stochasticity into the transition function (Appendix B; Bovy et al., 2024).

4.2.2 Interval POMDPs

The algorithms presented in this chapter focus on a particular type of RPOMDPs: Interval POMDPs (IPOMDPs) with *full* stickiness, where nature plays first and chooses an unknown but fixed transition model $P \in \mathcal{P}$, after which the agent needs to optimize its policy against this unknown choice. We give nature full observability; hence, there is only a single observation function for the agent. To that end, we define interval POMDPs as follows:

Definition 4.2 (Interval POMDP)**.** An interval POMDP (IPOMDP) is a tuple $\mathcal{M} = \langle S, s_\iota, A, \underline{P}, \overline{P}, R, Z, O \rangle$, where S is a finite set of states, $s_\iota \in S$ is the initial state, A is the finite set of actions, $\underline{P}\colon S \times A \times S \to [0,1]$ and $\overline{P}\colon S \times A \times S \to [0,1]$ are lower and upper bounds on the transition probabilities satisfying the same consistency requirement as IMDPs (Equation 4.1), Z is a finite set of observations, and $O\colon S \to Z$ is the deterministic state-based observation function.

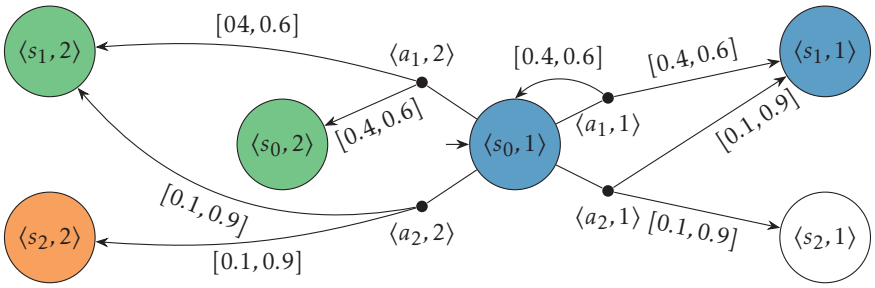


Figure 4.3: Example 2-unfolding of state s_0 of the IPOMDP from Figure 4.2.

4

An example of an IPOMDP with discussion of its semantics is presented in Figure 4.2. For ease of presentation, we write $\mathcal{P}$ for the function mapping transitions to intervals, *i.e.*, $\mathcal{P}(s' \mid s, a) = [\underline{P}(s, a, s'), \overline{P}(s, a, s')]$. For a transition function $P\colon S \times A \rightharpoonup \mathcal{D}(S)$ where each probability lies in its respective interval, we write $P \in \mathcal{P}$.

IPOMDPs are (s, a)-rectangular, we can unfold FSC memory nodes into the state space of the IPOMDP. The construction is the same as for standard POMDPs (Definition 2.11), except that we now copy the uncertainty set instead of the probability distribution at each state-action pair. As an example, Figure 4.3 shows the 2-unfolding of state s_0 of the IPOMDP from Figure 4.2.

Using k-unfolding on IPOMDPs allows us to, just as with standard POMDPs, compute and evaluate FSCs as if they are memoryless policies. Unfolding an IPOMDP, or an (s, a)-rectangular RPOMDP in general, has one important implication for the rectangularity assumption made when evaluating policies.

Remark 4.3 (k-unfolding and rectangularity). To ensure computational tractability of robust policy evaluation, we require (s, a)-rectangularity. That is, there are no dependencies in the uncertainty set between different state-action pairs. When unfolding the IPOMDP, we create copies of state-action pairs and their associated uncertainty sets. These copies represent the agent's internal memory. It would be natural to assume nature cannot exploit any information the agent stores internally. The consequence of that assumption would be, however, that the unfolded IPOMDP is no longer (s, a)-rectangular, as there are now multiple state-action pairs (for instance $(\langle s_0, 1\rangle, \langle a_1, 1\rangle$ and $\langle s_0, 1\rangle, \langle a_1, 2\rangle)$ where nature would have to fix a single choice in the uncertainty set for *both* actions at once. Hence, for computational efficiency, we assume full rectangularity on the unfolded state space, *i.e.*, (s, n, a)-rectangularity, which when used in robust policy evaluation provides a conservative bound on the policy's performance.

4.3 Nonlinear Optimization for Robust Policies

The problem of computing a stationary stochastic policy $\pi\colon S \to \mathcal{D}(A)$ that optimizes one of the robust objectives $\varphi \in \underline{\Phi}$ we consider can be precisely encoded into a *semi-infinite nonlinear optimization problem*, which we shall simply refer to as the nonlinear program (NLP). These NLPs are, in fact, semi-infinite quadratically

constrained quadratic programs (QCQPs) and nonconvex in general. Note that similar finite optimization problems have been used to compute FSCs for ordinary POMDPs (Amato et al., 2010) and parameter synthesis in parametric Markov models (Junges et al., 2024).

We present NLPs below for each of our objectives, *i.e.*, maximizing reachability, maximizing reach-reward, stochastic shortest path, and maximizing discounted reward. Then, in Sections 4.4 and 4.5 we present two methods for dealing with the semi-infiniteness and nonconvexity of the NLPs, either via the *convex-concave procedure* (CCP) or by *sequential convex programming* (SCP).

In all of the following optimization problems, $\{\pi_{s,a} \in (0,1] \mid s \in S, a \in A\}$ are the variables that encode the memoryless stochastic policy $\pi\colon S \to \mathcal{D}(A)$, and $\{v_s \in [0,\infty) \mid s \in S\}$ are the variables that encode the value function $V\colon S \to \mathbb{R}$. Note that the range of values a variable can take is included in the definitions above and not explicitly included in the constraints of the optimization problems below. After solving the optimization problem, the value assigned to a variable x is denoted by $\hat{x}$.

Remark 4.4. The policy variables are bounded away from zero to ensure transitions cannot be removed from the optimization problem. Consequently, any policy computed through such an optimization problem will be stochastic with *full support*, *i.e.*, every action will be chosen with some probability strictly greater than zero. When the optimization problem encodes an IPOMDP with memory unfolded into its state space, the resulting FSC will also have an action mapping with full support, as well as a fully connected memory update function.

Objectives with a target set $T \subseteq S$, *i.e.*, $\mathbf{P}_{Max}(\Diamond T), \mathbf{R}_{Max}(\Diamond T)$, and $\mathbf{R}_{Min}(\Diamond T)$, are preprocessed based on their underlying graph, just as for value iteration in standard MDPs, see Chapter 2. Let $S^{\infty} \subseteq S$ be the set of states for which there exists a policy that does not reach T almost-surely, and denote the remaining states by $S^{?} = S \setminus (T \cup S^{\infty})$. We construct a separate optimization problem for each objective, which we present below.

NLP for Maximizing Reachability

We define the following nonlinear optimization problem that maximizes the reachability probability of the target set:

$$\text{Maximize} \quad v_{s_I} \tag{4.2}$$

$$\text{Subject to}$$

$$\forall s \in T\colon \quad v_s = 1, \tag{4.3}$$

$$\forall s \in S^{?}\colon \quad \sum_{a \in A(s)} \pi_{s,a} = 1, \tag{4.4}$$

$$\forall s, s' \in S^{?}, a \in A(s)\colon \quad O(s) = O(s') \implies \pi_{s,a} = \pi_{s',a}, \tag{4.5}$$

$$\forall s \in S^{?}, \forall P \in \mathcal{P}\colon \quad v_s \leq \sum_{a \in A(s)} \pi_{s,a} \cdot \sum_{s' \in S} P(s' \mid s,a) \cdot v_{s'}. \tag{4.6}$$

This NLP maximizes the probability to reach T from the initial state s_I, as encoded in the objective function (4.2). Constraint (4.3) ensures that states $s \in T$ are assigned

a reachability probability of one. Constraint (4.4) ensures the resulting policy has a valid probability distribution at every state. Constraint (4.5) encodes that the resulting policy adheres to the partial observability. That is, states with the same observation are assigned the same distribution over the actions. Finally, Constraint (4.6) encodes the probability of reaching the target from some state $s \in S \setminus T$. It is this last constraint that makes the problem both nonlinear, as we multiply policy variables $\pi_{s,a}$ with reachability variables $v_{s'}$. The optimization problem is semi-infinite as we have a finite number of optimization variables but an infinite number of constraints since we quantify over all transition functions $P \in \mathcal{P}$. This constraint is also where we exploit the (s,a)-rectangularity of our IPOMDPs.

NLP for Maximizing reach-reward

Maximizing reach-reward objectives follows a similar encoding, presented below.

$$\text{Maximize} \quad v_{s_I} \tag{4.7}$$

$$\text{Subject to}$$

$$\forall s \in T: \quad v_s = 0, \tag{4.8}$$

$$\forall s \in S^?: \quad \sum_{a \in A(s)} \pi_{s,a} = 1, \tag{4.9}$$

$$\forall s, s' \in S^?, \forall a \in A(s): \quad O(s) = O(s') \implies \pi_{s,a} = \pi_{s',a}, \tag{4.10}$$

$$\forall s \in S^?, \forall P \in \mathcal{P}: \quad v_s \leq \sum_{a \in A(s)} \pi_{s,a} \cdot \left(R(s,a) + \sum_{s' \in S} P(s' \mid s,a) \cdot v_{s'} \right). \tag{4.11}$$

Indeed, this NLP is very similar to that for reachability as defined in Equations 4.2 to 4.6, with the only differences in Constraints (4.8) and (4.11). The differences follow the same modifications made when adapting value iteration, that is, Constraint (4.8) now encodes that states inside the target set obtain zero reward, following PRISM semantics (Forejt et al., 2011), and Constraint (4.11) now includes the reward $R(s,a)$ for each state-action pair.

NLP for Stochastic Shortest Path

The nonlinear program for the stochastic shortest path problem, *i.e.*, minimizing the expected cost for reaching a target set T, is encoded analogous to maximizing reach-reward. The NLP encoding is given by:

$$\text{Minimize} \quad v_{s_I} \tag{4.12}$$

$$\text{Subject to}$$

$$\forall s \in T: \quad v_s = 0, \tag{4.13}$$

$$\forall s \in S^?: \quad \sum_{a \in A(s)} \pi_{s,a} = 1, \tag{4.14}$$

$$\forall s, s' \in S^?, \forall a \in A(s): \quad O(s) = O(s') \implies \pi_{s,a} = \pi_{s',a}, \tag{4.15}$$

$$\forall s \in S^?, \forall P \in \mathcal{P}: \quad v_s \geq \sum_{a \in A(s)} \pi_{s,a} \cdot \left(R(s,a) + \sum_{s' \in S} P(s' \mid s,a) \cdot v_{s'} \right). \tag{4.16}$$

The only differences with maximizing reach-reward are in the objective, Equation 4.12, which now minimizes, and Constraint (4.16), where the direction of the constraint has been reversed.

NLP for Discounted Reward

Finally, we also present the nonlinear program for maximizing discounted reward. This NLP is similar to that for maximizing reach-reward and reflects similar changes made to value iteration when considering discounted reward instead of reach-reward, as discussed in Chapter 2. In particular, the discount factor γ is added to Constraint (4.20).

$$\text{Maximize} \quad v_{s_I} \tag{4.17}$$

$$\text{Subject to}$$

$$\forall s \in S: \quad \sum_{a \in A(s)} \pi_{s,a} = 1, \tag{4.18}$$

$$\forall s, s' \in S, \forall a \in A(s): \quad O(s) = O(s') \implies \pi_{s,a} = \pi_{s',a}, \tag{4.19}$$

$$\forall s \in S, \forall P \in \mathcal{P}: \quad v_s \leq \sum_{a \in A(s)} \pi_{s,a} \cdot \left(R(s,a) + \gamma \cdot \sum_{s' \in S} P(s' \mid s,a) \cdot v_{s'} \right). \tag{4.20}$$

4.4 CCP: Convex-Concave Procedure

We now introduce our first approach to solve one of the NLPs defined in Section 4.3. We convexify the semi-infinite NLPs via the *penalty convex-concave procedure* (CCP; Lipp and Boyd, 2016), which iteratively over-approximates a nonconvex optimization problem via linearization. The resulting convex problem can then be solved efficiently, and the process is iterated until a suitable solution is found. This process is illustrated in Figure 4.4.

For ease of presentation, we only show the procedure for the nonconvex quadratic constraints of maximizing reach-reward and minimizing stochastic shortest path, Equations 4.11 and 4.16, as these two optimization problems showcase maximization and minimization. We first detail the convexification step for one iteration. After that, we integrate the convexification step into the iterative procedure and use vertex enumeration on the uncertainty set to make the convex optimization problem finite and ready to be solved.

4.4.1 Convexification

We rewrite the quadratic functions in Equations 4.11 and 4.16 each as a *sum of convex and concave functions* and compute upper bounds for the concave functions. The CCP method starts with any (possibly infeasible) assignment $\hat{v}_s$ and $\hat{\pi}_{s,a}$ to the

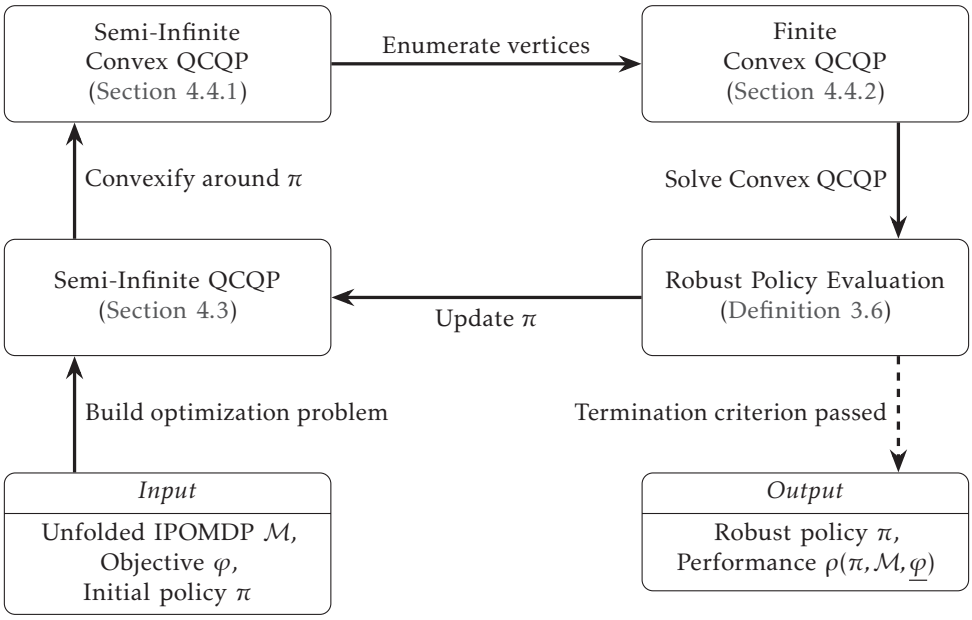


Figure 4.4: Flowchart of the overall CCP approach.

variables v_s and $\pi_{s,a}$. We unfold the brackets in the equations such that we get sums of quadratic functions, each of the form:

$$P(s'|s,a) \cdot \pi_{s,a} \cdot v_{s'},$$

for any $s, s' \in S^?, a \in A(s)$ and $P \in \mathcal{P}$. For readability, we introduce notation for these quadratic functions: $h(s,a,s',P) = P(s'|s,a) \cdot \pi_{s,a} \cdot v_{s'}$. The righthand side of the constraint is then equivalently written as

$$\sum_{a \in A(s)} \pi_{s,a} \cdot R(s,a) + \sum_{s' \in S} h(s,a,s',P).$$

For further ease of reading, we introduce temporary variables $d = 1/2 P(s'|s,a)$, $y = \pi_{s,a}$, and $z = v_{s'}$, such that

$$\begin{aligned} h(s,a,s',P) &= P(s,a,s') \cdot \pi_{s,a} \cdot v_{s'} \\ &= 2 \cdot d \cdot y \cdot z \\ &= 2 \cdot d \cdot y \cdot z + d(y^2 + z^2) - d(y^2 + z^2) \\ &= d(y+z)^2 - d(y^2 + z^2). \end{aligned}$$

After this rewrite, the function consists of a *quadratic convex function* $h_{\text{cvx}}(s,a,s',P) = d(y+z)^2$, and a *concave function* $h_{\text{ccv}}(s,a,s',P) = -d(y^2+z^2)$. It is this concave function that needs to be convexified around a previous solution to obtain a sum of two convex functions, which is also convex.

In particular, we transform $h_{\text{ccv}}(s,a,s',P) = -d(y^2+z^2)$ into

$$\hat{h}_{\text{ccv}}(s,a,s',P) = d(\hat{y}^2+\hat{z}^2) + 2\cdot d(\hat{y}^2+\hat{z}^2 - y\hat{y} - z\hat{z}),$$

where $\hat{y}$ and $\hat{z}$ are any assignments to the variables. After this transformation, the function $\hat{h}_{\text{ccv}}(s,a,s',P)$ is affine in y and z and therefore convex.

After this transformation, the constraint from Equation 4.11 becomes the following convex quadratic constraint:

$$v_s \leq \sum_{a\in A(s)} \pi_{s,a}\cdot R(s,a) + \sum_{s'\in S}\Big(h_{cvx}(s,a,s',P) + \hat{h}_{ccv}(s,a,s',P)\Big).$$

Similarly, Equation 4.16 is transformed into:

$$v_s \geq \sum_{a\in A(s)} \pi_{s,a}\cdot R(s,a) + \sum_{s'\in S}\Big(h_{cvx}(s,a,s',P) + \hat{h}_{ccv}(s,a,s',P)\Big).$$

4

4.4.2 Iterative Over-Approximations Towards Local Optima

The resulting convex QCQP is, however, still semi-infinite. Furthermore, we convexified around variable assignments $\hat{\pi}_{s,a}$ and $\hat{v}_{s'}$. We first discuss how the convex-concave procedure iteratively convexifies around new variable assignments.

As we over-approximate the quadratic functions, any feasible solution to the convex problem is also feasible for the original semi-infinite QCQP. However, due to the over-approximation, the resulting convex problem might be infeasible, even though the original one is not. To find a feasible assignment, we assign a so-called non-negative *penalty variable* k_s for each convexified constraint. To find a solution that induces minimal infeasibility or minimal violations of the convexified constraints, we minimize the sum of the penalty variables scaled by a penalty parameter τ.

If a solution assigns all penalty variables to zero, then the solution to the convex QCQP is feasible for the original non-convex QCQP, as we over-approximate the concave functions by affine functions. If any of the penalty variables k_s are assigned a positive value, we update the penalty parameter τ to $\tau+\mu$ for a $\mu>0$, similar to the approach in (Lipp and Boyd, 2016). We put an upper limit τ_{Max} on τ to avoid numerical problems during the procedure. After getting a new assignment, we convexify the non-convex QCQP by linearizing the concave functions around the new assignment and solve the next convex QCQP. We repeat the procedure until we find a feasible solution. The convergence to a locally optimal solution is guaranteed for a fixed τ, *i.e.*, after $\tau=\tau_{\text{Max}}$, but it may converge to an infeasible point of the original problem (Lipp and Boyd, 2016).

Termination by robust policy evaluation. After each iteration, we use robust policy evaluation to compute the precise, robust state values for the found policy $\pi(a|s)=\hat{\pi}_{s,a}$. These values are then used to linearize around in the next iteration to increase numerical stability. Additionally, we may use the performance of the policy π as a termination criterion. That is, if it surpasses some threshold: $\rho(\pi,\mathcal{M},\underline{\varphi}) \bowtie \lambda$, where $\bowtie \in \{\leq,\geq\}$.

Algorithm 4.1: CCP approach for solving IPOMDPs.

Input: IPOMDP $\mathcal{M}$,
Initialize: weight τ, initial policy π,
while $\tau < \tau_{\text{Max}}$ **do**
  Compute values $v_s = V_\pi(s)$ ▷ *Robust policy evaluation*
  Convexify around π and v ▷ *Convex-concave procedure (Section 4.4.1)*
  Solve convex QCQP ▷ *Enumerate vertices and solve*
  if $\forall s \in S^? \colon k_s = 0$ **then**
    return Converged, the policy π
  end if
  $\tau \leftarrow \tau + \mu$ ▷ *Update penalty variable*
end while
return Termination by τ_{Max}, the policy π

Vertex enumeration. To be solvable, we have to make the semi-infinite convex QCQP finite. For (s,a)-rectangular RPOMDPs where the uncertainty set is given by a *convex polytope* at each state-action pair, as is the case for the IPOMDPs we consider, it is sufficient to enumerate the vertices of that polytope (Löfberg, 2012). This enumeration replaces the constraints with universal quantification $P \in \mathcal{P}$ by a finite number of constraints in which the universal quantifier over all distributions is replaced by all possible combinations of these vertices, which we denote by $\mathsf{Vert}(\mathcal{P})$. QCQP with polytopic uncertainty is still NP-Hard (Bertsimas et al., 2011) as the number of vertices of a convex polytope can be exponential in the number of dimensions. In our application to IPOMDPs, the dimension of each polytope is determined by the number of successor states for each state-action pair. Thus, for sparse models, this enumeration approach is still feasible.

Convex QCQP for Stochastic Shortest Path

The finite, convex QCQP for stochastic shortest path is defined as follows:

$$\text{Minimize} \quad v_{s_\iota} + \tau \sum_{s \in S^?} k_s \tag{4.21}$$

Subject to

$$\forall s \in T \colon \quad v_s = 0, \tag{4.22}$$

$$\forall s \in S^? \colon \quad \sum_{a \in A(s)} \pi_{s,a} = 1, \tag{4.23}$$

$$\forall s, s' \in S^?, \forall a \in A(s) \colon \quad O(s) = O(s') \implies \pi_{s,a} = \pi_{s',a}, \tag{4.24}$$

$$\forall s \in S^?, \forall P \in \mathsf{Vert}(\mathcal{P}) \colon \quad k_s + v_s \geq \sum_{a \in A(s)} \pi_{s,a} \cdot R(s,a) + \sum_{s' \in S} \Big(h_{cvx}(s,a,s',P) + \hat{h}_{ccv}(s,a,s',P) \Big). \tag{4.25}$$

Convex QCQP for reach-reward

The finite, convex QCQP for reach-reward is defined as follows:

$$\text{Maximize} \quad v_{s_I} - \tau \sum_{s \in S^?} k_s \tag{4.26}$$

Subject to

$$\forall s \in T: \quad v_s = 0, \tag{4.27}$$

$$\forall s \in S^?: \quad \sum_{a \in A(s)} \pi_{s,a} = 1, \tag{4.28}$$

$$\forall s, s' \in S^?, \forall a \in A(s): \quad O(s) = O(s') \implies \pi_{s,a} = \pi_{s',a}, \tag{4.29}$$

$$\forall s \in S^?, \forall P \in \mathsf{Vert}(\mathcal{P}): \quad k_s - v_s \geq \sum_{a \in A(s)} \pi_{s,a} \cdot R(s,a) + \sum_{s' \in S} \Big(h_{cvx}(s,a,s',P) + \hat{h}_{ccv}(s,a,s',P) \Big). \tag{4.30}$$

Complete iterative CCP algorithm

Algorithm 4.1 presents the full algorithm for iteratively solving the convex QCQPs to compute a robust policy via the CCP approach. We initialize with an initial guess policy π and a weight $\tau > 0$ for the penalty variables. We use robust policy evaluation to compute the robust value function under π. We convexify around π and the value function and solve the resulting QCQP. If all penalty variables are zero, the CCP method has converged, and we return the policy; otherwise, we increment the weight τ and continue the next iteration.

4.5 SCP: Sequential Convex Programming

In this section, we present our second approach to solving the semi-infinite NLPs from Section 4.3 based on sequential convex programming (SCP; Mao et al., 2018). In the SCP approach, we aim to separate the concerns of nonconvexity and the infinite number of constraints due to the uncertainty set. A *simple IPOMDP* provides such a separation. In a simple IPOMDP, each state either has only a single action enabled or multiple actions, each with at most two successor states. This transformation allows us to define NLPs with separate *uncertain constraints* that are either already linear but need to be robust against the uncertainty or *nonconvex quadratic constraints* that need to be linearized but, in return, are not affected by the uncertainty. The uncertain constraints are dualized to derive a finite nonconvex optimization problem. We then iteratively linearize the nonconvex constraints around a previous solution to construct and solve a finite linear program (LP). The SCP procedure is outlined in Figure 4.5 and converges to a local optimum.

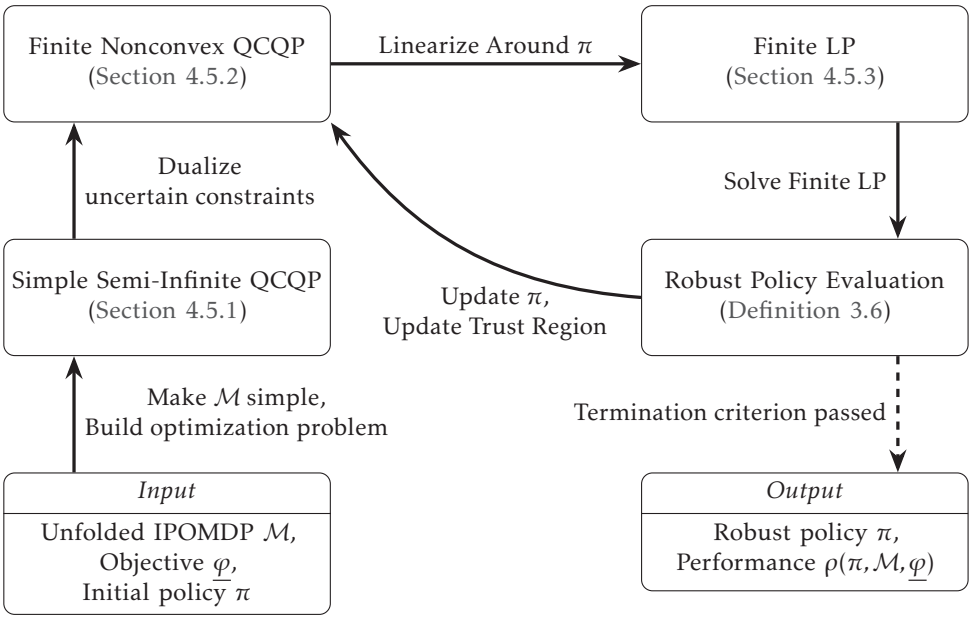


Figure 4.5: Flowchart of the overall SCP approach.

4.5.1 Nonlinear Optimization on Simple IPOMDPs

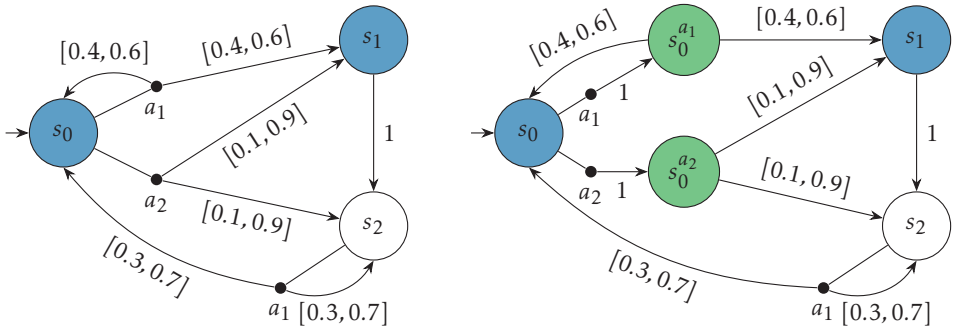
For our SCP approach to work, we wish to separate the concerns of dealing with action choices and the uncertainty on the transition probabilities. A simple IPOMDP provides such a separation. At the cost of increasing the state space (linearly), we obtain an IPOMDP where each state either has (multiple) action choices that all lead to a deterministic successor state with probability one or only one enabled action, which is allowed to have multiple successor states with interval transitions.

Definition 4.5 (Binary and Simple IPOMDP)**.** An IPOMDP is *binary*, if there are at most two enabled actions at every state, *i.e.*, $|A(s)| \leq 2$ for all $s \in S$. An IPOMDP is *simple* if it is binary and for all $s \in S$, the following holds:

$$|A(s)| = 2 \implies \forall a \in A(s) \colon \exists s' \in S \colon \mathcal{P}(s, a, s') = 1.$$

Every IPOMDP can be transformed into a simple IPOMDP by introducing auxiliary states and observations via the same construction as for POMDPs presented in (Junges et al., 2018), and will preserve (s, a)-rectangularity and optimal values. Policies computed for simple IPOMDPs can be mapped back to policies for the original IPOMDP.

> *Example* 8 (Binary and simple IPOMDP). Consider the two IPOMDPs depicted in Figure 4.6. The IPOMDP in Figure 4.6a is already binary, as each state has at most two enabled actions. Figure 4.6b shows the simple IPOMDP obtained



(a) An IPOMDP.

(b) A simple IPOMDP.

Figure 4.6: The IPOMDP from Figure 4.2 and a simple IPOMDP obtained from it.

4

from the IPOMDP in Figure 4.6a using the transformation method of (Junges et al., 2018). We introduce an auxiliary state for every state-action pair with multiple successor states, in this case, (s_0, a_1) and (s_0, a_2). Selecting one of the actions transitions with probability one to its associated auxiliary state. The outgoing interval transitions of the original state-action pair are now assigned to the auxiliary state. Consequently, the transformation preserves the (s,a)-rectangularity of the IPOMDP, as the uncertainty set of a state-action pair is only moved to the auxiliary state, and no new dependencies across states or actions are introduced.

Assuming our IPOMDP has been made simple, we now present slightly modified NLP encodings that exploit the structure of simple IPOMDPs. For ease of readability, we again only present the NLPs for maximizing reach-reward and minimizing stochastic shortest path. The NLPs for reachability and discounted reward can be adapted analogously. Note that these NLPs are specialized versions (adapted to simple IPOMDPs) of the general NLPs defined in Section 4.3.

Recall that $S^\infty \subseteq S$ is the set of states for which there exists a policy that does not reach the target set T almost-surely, and $S^? = S \setminus (T \cup S^\infty)$. In the following, let $S^?_\mathrm{a}$ denote the states with action choices, and $S^?_\mathrm{u}$ denote the states with uncertain outcomes, such that $S^? = S^?_\mathrm{a} \cup S^?_\mathrm{u}$.

NLP for maximizing reach-reward on simple IPOMDPs

The NLP for maximizing reach-reward on simple IPOMDPs is given by

$$\begin{aligned}
&\text{Maximize} && v_{s_\iota} && (4.31)\\
&\text{Subject to} \\
&\quad \forall s \in T: && v_s = 0, && (4.32)\\
&\quad \forall s \in S^?: && \sum_{a \in A(s)} \pi_{s,a} = 1, && (4.33)
\end{aligned}$$

$$\forall s, s' \in S^?, \forall a \in A(s): \quad O(s) = O(s') \implies \pi_{s,a} = \pi_{s',a}, \tag{4.34}$$

$$\forall s \in S^?_{\mathrm{a}}: \quad v_s \leq \sum_{a \in A(s)} \pi_{s,a} \cdot \left(R(s,a) + \sum_{s' \in S} \mathcal{P}(s,a,s') \cdot v_{s'} \right), \tag{4.35}$$

$$\forall s \in S^?_{\mathrm{u}}, \forall P \in \mathcal{P}: \quad v_s \leq R(s) + \sum_{s' \in S} P(s,s') \cdot v_{s'}. \tag{4.36}$$

In particular, note that we now have two constraints, Equations 4.35 and 4.36, instead of Equation 4.11 from the original NLP. The first constraint is nonconvex as we multiply policy variables $\pi_{s,a}$ with value variables $v_{s'}$, but without uncertainty as $\mathcal{P}(s,a,s') = 1$ for precisely one s' by construction. The second constraint is linear, as there is only one action enabled in states $s \in S^?_{\mathrm{u}}$, but uncertain as we have interval transitions to multiple successor states here. Since there is only one action in these states, we also omit it from the transition and reward functions.

4

NLP for Stochastic Shortest Path on simple IPOMDPs

The NLP for minimizing stochastic shortest path on simple IPOMDPs is analogously given by

$$\text{Minimize} \quad v_{s_I} \tag{4.37}$$

$$\text{Subject to}$$

$$\forall s \in T: \quad v_s = 0, \tag{4.38}$$

$$\forall s \in S^?: \quad \sum_{a \in A(s)} \pi_{s,a} = 1, \tag{4.39}$$

$$\forall s, s' \in S^?, \forall a \in A(s): \quad O(s) = O(s') \implies \pi_{s,a} = \pi_{s',a}, \tag{4.40}$$

$$\forall s \in S^?_{\mathrm{a}}: \quad v_s \geq \sum_{a \in A(s)} \pi_{s,a} \cdot \left(R(s,a) + \sum_{s' \in S} \mathcal{P}(s,a,s') \cdot v_{s'} \right), \tag{4.41}$$

$$\forall s \in S^?_{\mathrm{u}}, \forall P \in \mathcal{P}: \quad v_s \geq R(s) + \sum_{s' \in S} P(s,s') \cdot v_{s'}. \tag{4.42}$$

4.5.2 Dualization of the Uncertain Constraints

We start off by summarizing robust LPs with polytopic uncertainty (Ben-Tal and Nemirovski, 1998; Bertsimas et al., 2011). The idea of solving robust LPs with such uncertainty is essential in our approach.

Robust LPs. A robust LP with the variable $x \in \mathbb{R}^n$ is of the form

$$\begin{aligned} &\text{minimize} && c^\top x \\ &\text{subject to} && (d+u)^\top x \leq e \quad \forall u \in \mathcal{U}, \end{aligned}$$

where $c, d \in \mathbb{R}^n$, and $e \in \mathbb{R}$ are given vectors, $u \in \mathbb{R}^n$ is the uncertain parameter, and $\mathcal{U}$ is the *uncertainty set*. We assume that the set $\mathcal{U}$ is a convex polytope, i.e., that it is

an n-dimensional shape defined by the linear inequalities $Cu + g \geq 0$ for $C \in \mathbb{R}^{m \times n}$ and $g \in \mathbb{R}^m$.

Duality. For simplicity, we explain the idea of dualization on a single robust inequality. The idea can be generalized to multiple robust inequalities. With $\mathcal{U}$ defined by the linear inequalities above, *duality* can be used to obtain a tractable solution to the robust LP. Specifically, we write the Lagrangian for the maximization problem over $u^\top x$ with the dual variable $\mu \geq 0$ as

$$L(u, \mu) = u^\top x + \mu^\top (Cu + g).$$

By taking the supremum over u, we obtain

$$\sup_{u \in \mathcal{U}} L(u, \mu) = \begin{cases} \infty & \text{if } C^\top \mu + x \neq 0, \\ \mu^\top g & \text{if } C^\top \mu + x = 0. \end{cases}$$

All inequalities are linear which implies *strong duality*, i.e.,

$$\sup_{u \in \mathcal{U}} u^\top x = \inf_{\mu \geq 0} \{\mu^\top g \mid C^\top \mu + x = 0\}.$$

Since these optimization problems are linear, we know the optimal value is attained at a feasible solution. Therefore, we can replace sup and inf with max and min. The semi-infinite inequality with polytopic uncertainty is equivalent to the following linear constraints

$$d^\top x + \mu^\top g \leq e, \quad C^\top \mu + x = 0, \quad \mu \geq 0.$$

Dualization for Simple IPOMDPs

We now describe the dualization step we use to obtain a finite optimization problem for simple IPOMDPs.

Uncertainty polytopes. The uncertainty set of a simple IPOMDP at a state-action pair is given by a convex polytope defined by the following constraints:

$$\forall s, s' \in S : \ \underline{P}(s, s') \leq u_{s,s'} \leq \overline{P}(s, s'), \qquad \forall s \in S : \ \sum\nolimits_{s' \in S} u_{s,s'} = 1.$$

We denote these constraints in matrix form: $C_s u + g_s \geq 0$.

After obtaining the matrices C_s and vectors g_s characterizing uncertainty sets for each state, we directly use dualization to transform the inequalities from Equation 4.36 of the NLP for maximizing reach-reward into

$$\begin{aligned} \forall s \in S^?_{\mathrm{u}} : &\quad v_s \leq R(s) + \mu_s^\top g_s, \\ \forall s \in S^?_{\mathrm{u}} : &\quad C_s^\top \mu_s - q = 0, \\ \forall s \in S^?_{\mathrm{u}} : &\quad \mu_s \geq 0, \end{aligned}$$

where $\mu_s \in \mathbb{R}^{|S|}$ is the dual variable of the constraint $C_s u + g_s \geq 0$ and q is an $|S|$–dimensional vector denoting the set of value variables v_s for each state $s \in S$.

For stochastic shortest path, Equation 4.42, is reversed compared to that of reach-reward. The resulting constraints are then given by

$$\begin{aligned} \forall s \in S^?_{\mathrm{u}}: &\quad v_s \geq R(s) + \mu_s^\top g_s, \\ \forall s \in S^?_{\mathrm{u}}: &\quad C_s^\top \mu_s + q = 0, \\ \forall s \in S^?_{\mathrm{u}}: &\quad \mu_s \geq 0, \end{aligned}$$

4.5.3 Linearizing the Finite Nonconvex Problem

In this section, we discuss our algorithm to solve the (finite but nonconvex) dual problem from the previous section. Our method is based on the *sequential convex programming* (SCP; Mao et al., 2018; Yuan, 2015). SCP iteratively computes a locally optimal solution to the dual problem from Section 4.5.2 by approximating it as an LP. In every iteration, this approximation is obtained by *linearizing* the quadratic functions around a previous solution.

We again show the linearization step only for reach-reward and stochastic shortest path, *i.e.*, the constraints from Equations 4.35 and 4.41. The linearization step follows a method similar to that of the CCP approach (Section 4.4).

Let $h(s,a,s')$ be the quadratic function of the right-hand side of these constraints for a given $s \in S^?_{\mathrm{a}}, s' \in S$ and $a \in A(s)$:

$$h(s,a,s') = \pi_{s,a} \cdot \mathcal{P}(s,a,s') \cdot v_{s'}$$

and let $d = \mathcal{P}(s,a,s'), y = \pi_{s,a}$, and $z = v_{s'}$. Let $\langle \hat{y}, \hat{z} \rangle$ denote an arbitrary assignment to y and z. We linearize $h(s,a,s') = d \cdot y \cdot z$ around $\langle \hat{y}, \hat{z} \rangle$ into

$$\begin{aligned} h_{\mathrm{aff}}(s,a,s') &= d \cdot \Big(\hat{y} \cdot \hat{z} + \hat{y} \cdot (z - \hat{z}) + \hat{z} \cdot (y - \hat{y}) \Big) \\ &= d \cdot (\hat{y} \cdot z + \hat{z} \cdot y - \hat{z} \cdot \hat{y}). \end{aligned}$$

The resulting function $h_{\mathrm{aff}}(s,a,s')$ is affine in y and z (v_s and $\pi_{s,a}$). After the linearization step, we replace Equation 4.35 for maximizing reach-reward by

$$\forall s \in S^?_{\mathrm{a}}: \quad v_s \leq \sum_{a \in A(s)} \left(\pi_{s,a} \cdot R(s,a) + \sum_{s' \in S} h_{\mathrm{aff}}(s,a,s') \right).$$

Similarly, Equation 4.41 for minimizing stochastic shortest path gets replaced by

$$\forall s \in S^?_{\mathrm{a}}: \quad v_s \geq \sum_{a \in A(s)} \left(\pi_{s,a} \cdot R(s,a) + \sum_{s' \in S} h_{\mathrm{aff}}(s,a,s') \right).$$

The linearized problem may be infeasible, or the optimal solution to the linearized may no longer be feasible to the dual problem, as it is an over-approximation of the original optimization problem. We alleviate these feasibility issues as follows.

First, we add penalty variables to the linearized constraints to ensure that the dual problem is always feasible, and we penalize violations by adding the sum of these variables weighted by penalty parameter τ to the objective function, just as in the CCP approach. Second, we include *trust regions* around the previous solution to ensure we do not deviate too much from that solution. We explain these two additions below. Finally, we use robust dynamic programming to compute the exact robust value function and alleviate any potential approximation errors arising from the linearization. Additionally, the robust performance computed here may also serve as a termination criterion.

Penalty variables. Similar to the CCP approach, we add a non-negative penalty variable k_s for all $s \in S_{\text{a}}^?$ to the linearized constraints, which yields for maximizing reach-reward:

$$\forall s \in S_{\text{a}} : \quad v_s - k_s \leq \sum_{a \in A(s)} \left(\pi_{s,a} \cdot R(s,a) + \sum_{s' \in S} h_{\text{aff}}(s,a,s') \right).$$

When k_s is sufficiently large, these constraints always allow a feasible solution. The objective function becomes to maximize $v_{s_\iota} - \tau \sum_{s \in S_{\text{a}}^?} k_s$. The constraint and objective function for minimizing stochastic shortest path are modified analogously.

Trust regions. We use trust regions by adding the following set of constraints to the resulting linearized problem:

$$\forall s \in S : \quad \frac{\hat{v}_s}{\delta'} \leq v_s \leq \hat{v}_s \cdot \delta', \tag{4.43}$$

$$\forall s \in S_{\text{a}}, \forall a \in A(s) : \quad \frac{\hat{\pi}_{s,a}}{\delta'} \leq \pi_{s,a} \leq \hat{\pi}_{s,a} \cdot \delta', \tag{4.44}$$

where $\delta' = \delta + 1$ and $\delta > 0$ is the size of the trust region, which restricts the set of feasible policies, and $\hat{v}_s$ and $\hat{\pi}_{s,a}$ denote the assigned value and policy variables that are used for linearization.

Complete Finite LPs

We now present the complete finite LPs for simple IPOMDPs with reach-reward and stochastic shortest path objectives.

Complete Finite LP for Reach-Reward

Combining these steps, we now state the resulting finite LP—for some fixed but arbitrary assignment to $\hat{v}_s$ and $\hat{\pi}_{s,a}$ in the definition of h_{aff}, penalty parameter $\tau > 0$ and a trust region $\delta > 0$:

$$\text{Maximize} \quad v_{s_\iota} - \tau \sum_{s \in S_{\text{a}}^?} k_s \tag{4.45}$$

Subject to

$$\forall s \in T: \quad v_s = 0, \tag{4.46}$$

$$\forall s \in S^?: \quad \sum_{a \in A(s)} \pi_{s,a} = 1, \tag{4.47}$$

$$\forall s, s' \in S^?, \forall a \in A(s): \quad O(s) = O(s') \implies \pi_{s,a} = \pi_{s',a}, \tag{4.48}$$

$$\forall s \in S^?_{\mathrm{a}}: \quad v_s - k_s \leq \sum_{a \in A(s)} \pi_{s,a} \cdot R(s,a) + \sum_{s' \in S} h_{\mathrm{aff}}(s,a,s'), \tag{4.49}$$

$$\forall s \in S^?_{\mathrm{u}}: \quad v_s \leq R(s) + \mu_s^\top g_s \tag{4.50}$$

$$\forall s \in S^?_{\mathrm{u}}: \quad C_s^\top \mu_s + q = 0, \tag{4.51}$$

$$\forall s \in S^?_{\mathrm{u}}: \quad \mu_s \geq 0, \tag{4.52}$$

$$\forall s \in S: \quad \hat{v}_s \cdot 1/\delta' \leq v_s \leq \hat{v}_s \cdot \delta', \tag{4.53}$$

$$\forall s \in S_{\mathrm{a}}, \forall a \in A(s): \quad \hat{\pi}_{s,a} \cdot 1/\delta' \leq \pi_{s,a} \leq \hat{\pi}_{s,a} \cdot \delta'. \tag{4.54}$$

4

Complete Finite LP for Stochastic Shortest Path

Analogously, the resulting finite LP for minimizing stochastic shortest path is:

$$\text{Minimize} \quad v_{s_i} + \tau \sum_{s \in S^?_{\mathrm{a}}} k_s \tag{4.55}$$

Subject to

$$\forall s \in T: \quad v_s = 0, \tag{4.56}$$

$$\forall s \in S^?: \quad \sum_{a \in A(s)} \pi_{s,a} = 1, \tag{4.57}$$

$$\forall s, s' \in S^?, \forall a \in A(s): \quad O(s) = O(s') \implies \pi_{s,a} = \pi_{s',a}, \tag{4.58}$$

$$\forall s \in S^?_{\mathrm{a}}: \quad v_s + k_s \geq \sum_{a \in A(s)} \pi_{s,a} \cdot R(s,a) + \sum_{s' \in S} h_{\mathrm{aff}}(s,a,s'), \tag{4.59}$$

$$\forall s \in S^?_{\mathrm{u}}: \quad v_s \leq R(s) + \mu_s^\top g_s \tag{4.60}$$

$$\forall s \in S^?_{\mathrm{u}}: \quad C_s^\top \mu_s + q = 0, \tag{4.61}$$

$$\forall s \in S^?_{\mathrm{u}}: \quad \mu_s \geq 0, \tag{4.62}$$

$$\forall s \in S: \quad \hat{v}_s \cdot 1/\delta' \leq v_s \leq \hat{v}_s \cdot \delta', \tag{4.63}$$

$$\forall s \in S_{\mathrm{a}}, \forall a \in A(s): \quad \hat{\pi}_{s,a} \cdot 1/\delta' \leq \pi_{s,a} \leq \hat{\pi}_{s,a} \cdot \delta'. \tag{4.64}$$

Complete Iterative SCP Algorithm

Algorithm 4.2 details the complete SCP algorithm for computing robust policies in RPOMDPs. We start with an initial guess of a policy π and a trust region radius with $\delta > 0$, and we use robust dynamic programming to compute the robust value function for π, and in particular, its performance $\rho(\pi, \mathcal{M})$. We check whether the obtained value $\rho(\pi, \mathcal{M})$ is improved compared to $\rho(\pi_{\mathrm{old}}, \mathcal{M})$. In this case, we accept the solution and enlarge the trust region by multiplying δ with a parameter $\kappa > 1$.

Algorithm 4.2: SCP with trust region for solving IPOMDPs.

Input: IPOMDP $\mathcal{M}$, $\kappa > 1$, $\omega > 0$
Initialize: trust region δ, weight τ, policy π, $\hat{\rho}_{\text{old}} = 0$
while $\delta > \omega$ **do**
  Extract values v_s, performance $\hat{\rho} = \rho(\pi, \mathcal{M}, \underline{\varphi})$ ▷ *Robust policy evaluation*
  if $\hat{\rho} > \hat{\rho}_{\text{old}}$ **then** ▷ *Reverse for minimizing objectives*
    $\forall s \in S : \hat{v}_s \leftarrow v_s, \hat{\pi} \leftarrow \pi, \hat{\rho}_{old} \leftarrow \hat{\rho}$ ▷ *Accept iteration*
    $\delta \leftarrow \delta \cdot \kappa$ ▷ *Extend trust region*
  else
    $\delta \leftarrow \frac{\delta}{\kappa}$ ▷ *Reject iteration, reduce trust region*
  end if
  Linearize around $\langle \hat{\pi}, \hat{v}_s \rangle$ ▷ *Sequential convex programming (Section 4.5.3)*
  Solve the resulting LP
  Extract policy π: $\pi(a|s) = \hat{\pi}_{s,a}$
end while
return the policy π

If not, we reject the solution and contract the trust region by κ. We then solve the LP with the current parameters. We linearize around previous policy variables $\hat{\pi}_{s,a}$ and value variables $\hat{v}_s$, and solve with parameters δ and κ to get an optimal solution. We iterate this procedure until a feasible solution is found or the radius of the trust region is below a threshold $\omega > 0$. If the trust region size is below ω, the algorithm terminates. In such cases, we can run Algorithm 4.2 with a different initial assignment.

4.6 Experimental Evaluation

We evaluate and compare the QCQP and SCP approaches for computing robust policies on IPOMDPs. Our implementation is built on top of the probabilistic verification tool Storm (Dehnert et al., 2017), together with QCQP and LP solver Gurobi (Gurobi Optimization, 2019). All experiments were run on an i7-10510U CPU with 16GB of RAM.

4.6.1 Setup

We use three POMDP benchmarks to evaluate both algorithms: *aircraft collision avoidance*, *intercepts*, and *spacecraft motion planning*, all detailed below. We transform these POMDPs into IPOMDPs by incorporating intervals around the probabilities of the original POMDP model, which we shall refer to as the *nominal* model. We consider FSCs of sizes $|\mathcal{N}| \in \{1, 2, 3\}$.

Hyperparameters. For both CCP and SCP, we initialize $\hat{\pi}$ to the policy that, for each state, gives a uniform distribution over all available actions. For CCP, we set $\tau = 10^4, \tau_{max} = 10^{10}$, and $\mu = 20$ initially and doubled per iteration. For SCP, we set $\tau = 10^4, \delta = 1.5, \kappa = 1.5$, and $\omega = 10^{-4}$.

Termination criteria. We specify a maximum number of 100 iterations and use a thirty-minute time-out (TO) for both algorithms. We do not set a termination threshold for robust policy evaluation so we may investigate towards which value each algorithm converges within the set timeframe and given iterations.

Environments

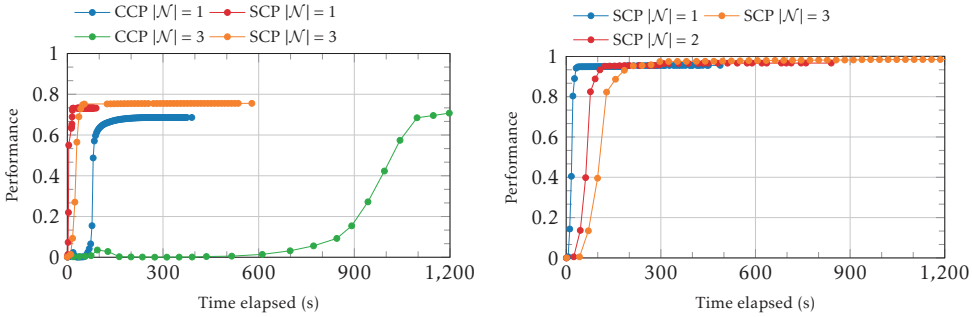
We benchmark our algorithms on the following environments. For each of these environments we have a *nominal model*, *i.e.*, a standard POMDP with fixed probabilities, and two IPOMDPs with *small* and *large* interval sizes, respectively. Furthermore, we unfold the FSC memory into the state space and transform it into a simple IPOMDP, as described in Definition 2.11 and Section 4.5.1, respectively. We consider FSCs of sizes $|\mathcal{N}| = 1$ (memoryless), $|\mathcal{N}| = 2$ and $|\mathcal{N}| = 3$

- *Aircraft Collision Avoidance*. We consider a robust variant of the aircraft collision avoidance problem (Kochenderfer, 2015). The agent's objective is to maximize the probability of avoiding a collision with an intruder aircraft while taking into account sensor errors and uncertainty in the future paths of the intruder, *i.e.*, a reachability objective.

  We consider a two-dimensional instance with a state space consisting of (1) the agent's aircraft position relative to an intruder and (2) the relative speed of the intruder relative to the agent. The relative position is discretized, and the agent cannot precisely observe the intruder's position. In each time step, the action choice reflects changing the acceleration of the own aircraft in two dimensions. The intruder acceleration is chosen with probability 0.5 in the nominal model and intervals $[0.45, 0.55]$ and $[0.2, 0.8]$ in the small and large uncertainty sets. The nominal probability of the pilot being responsive at a given time is 0.9, with uncertainty sets given by the intervals $[0.85, 0.95]$ and $[0.7, 0.98]$. The number of states of this model are 9 100, 27 236, and 45 372, for each FSC size, respectively.

- *Intercept*. This benchmark considers a partially observable grid world in which the agent has to intercept a randomly moving adversary (Junges et al., 2021a). The agent operates on an eight-by-eight grid and can move towards all eight surrounding cells, with a probability of *slipping* two tiles in that direction. Its view is limited to a radius of one, but by using an action to observe, the adversary's position is revealed. The goal is to intercept the adversary before it reaches an exit, expressed through a stochastic shortest path objective. We consider a nominal slip probability of 0.2 and small and large uncertainty sets of $[0.15, 0.25]$ and $[0.1, 0.4]$, respectively. The number of states of this model are 17 570, 57 316, and 97 062, for each FSC size, respectively.

- *Spacecraft Motion Planning*. This case study considers the robust spacecraft motion planning system (Frey et al., 2017; Hobbs and Feron, 2020). The spacecraft orbits the earth along a set of predefined natural motion trajectories (NMTs) (Kim et al., 2007). While the spacecraft follows its current NMT, it does not consume fuel. Upon an imminent close encounter with other objects



(a) Aircraft collision avoidance.

(b) Spacecraft motion planning.

Figure 4.7: Performance of robust policies computed by CCP and SCP against reachability objectives in the Aircraft and Spacecraft environments.

4

in space, the spacecraft may be directed to switch into a nearby NMT at the cost of a certain fuel usage, modeled by a reward function.

For this model, we consider two (separate) objectives: (1) To maximize the probability of avoiding a close encounter with other objects and (2) to minimize fuel consumption, both within successfully finishing an orbiting cycle, *i.e.*, reachability and stochastic shortest path objectives. Uncertainty enters the problem in the form of potential sensing and actuating errors. In particular, there is uncertainty about the spacecraft position and the location of other objects in the current orbit, leading to a failure rate of switching to a nearby NMT. The nominal probability of successfully switching between NMTs is 0.92, with small and large uncertainty sets given by $[0.9, 0.95]$ and $[0.7, 0.98]$. The number of states of this model with the reachability objective are 36 048, 114 406, and 192 764 for each FSC, and for the stochastic shortest path version of the model, we obtain IPOMDPs with 61 545, 191 835, and 322 125 states.

Research questions. Our experimental evaluation centers around the following three research questions.

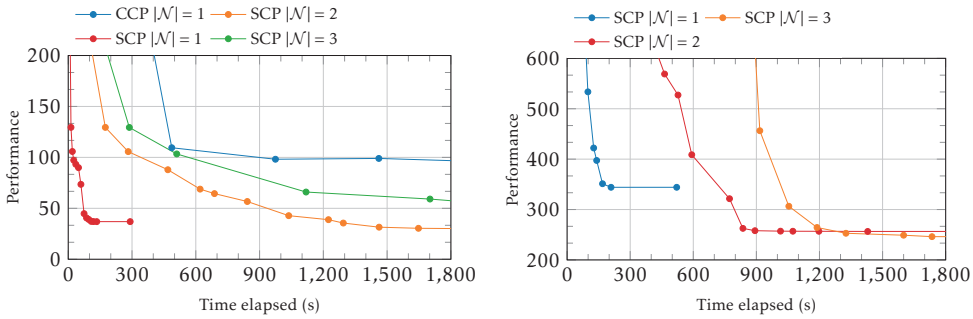
RQ1 How do CCP and SCP compare to each other in terms of achieved performance versus computation time of the robust policies?

RQ2 How does increasing the FSC memory size affect performance and computation time?

RQ3 Do robust policies indeed provide robustness against the uncertainty set of the IPOMDP?

4.6.2 Results and Discussion

In Figures 4.7a, 4.7b, 4.8a and 4.8b we evaluate the performance of CCP and SCP when computing a robust policy against the large uncertainty set of each model.



(a) Intercept.

(b) Spacecraft motion planning.

Figure 4.8: Performance of robust policies computed by CCP and SCP against stochastic shortest path objectives in the Intercept and Spacecraft environments.

Note that certain FSC sizes have been left out, as the results for those sizes are either between the shown results (for FSCs of size $|\mathcal{N}| = 2$), or the algorithm timed-out before finishing the first iteration (CCP on Spacecraft and Intercept). Using these results, we now discuss research questions 1 and 2.

RQ1. In all environments, we observe SCP outperforming CCP. SCP finds better-performing policies faster than CCP. The latter is due to CCP having to enumerate the vertices of the convex polytopes, leading to larger optimization problems than SCP, which employs dualization. Notably, all cases where CCP is missing from the results, were due to either a timeout or running out of memory before the first iteration was completed. We conclude research question 1 with that SCP outperforms CCP in terms of both achieved performance and computation time.

RQ2. We also note that the use of memory is beneficial. Even a small FSC of three nodes manages to perform significantly better than a memoryless policy on the spacecraft environment, as seen in Figures 4.7b and 4.8b. The increase in performance comes at a roughly linear increase in computational cost, which is to be expected since unfolding memory linearly increases the number of state-action pairs in the IPOMDP, answering research question 2.

Evaluating Robustness

Towards an answer to research question 3, we perform an additional experiment to investigate the benefits of robust policies. We use SCP to compute a (memoryless) robust policy and evaluate the performance of this robust policy against both the nominal POMDP and the IPOMDP with small uncertainty. Similarly, we use SCP to compute a memoryless policy for the nominal POMDP and evaluate that policy against the IPOMDPs with small and large uncertainty sets. The results are presented in Tables 4.1 and 4.2.

We note that in three out of four environments, Aircraft and both versions of

Model	Uncertainty size	Nominal policy	Robust policy
Aircraft	Nominal	0.95	**0.97**
	Small	0.93	**0.95**
	Large	0.57	**0.73**
Spacecraft	Nominal	0.996	0.996
	Small	0.994	0.994
	Large	0.952	**0.955**

Table 4.1: Policy evaluation results for reachability objectives.

Model	Uncertainty size	Nominal policy	Robust policy
Intercept	Nominal	**18.50**	22.99
	Small	**20.69**	25.68
	Large	**30.52**	36.93
Spacecraft	Nominal	**149.57**	161.40
	Small	**158.22**	171.23
	Large	384.20	**344.12**

Table 4.2: Policy evaluation results for stochastic shortest path objectives.

Spacecraft, the robust policy performs best against the large uncertainty set. In the Aircraft environment, the robust policy outperforms the nominal policy against all uncertainty sets, including the nominal POMDP itself, while in the Spacecraft environment with reachability objective, their performance is equal on the small uncertainty and nominal models.

While the nominal policy performs better against the nominal POMDP and the small uncertainty set on the Spacecraft environment with rewards, it drops in performance as the uncertainty becomes larger, showcasing that its nominal performance is not indicative of its performance in other environments. The severity of such effects depends on the environment and the uncertainty set, as also seen by the intercept environment, where the nominal policy consistently outperforms the robust policy. Yet, it shows that, generally, when only an uncertain environment is given, computing a robust policy for that environment is the better approach for obtaining a well-performing policy, positively answering research question 3.

4.7 Conclusion

We presented two new algorithms to compute finite-memory policies for RPOMDPs. Both algorithms are based on convex optimization approaches. Their primary difference is how they deal with the nonconvex constraints. CCP convexifies around a previous solution while SCP linearizes. Our experimental evaluation shows that both methods are applicable to large RPOMDPs with varying levels of uncertainty, with SCP being significantly more scalable than CCP.

4.7.1 Limitations and Discussion

The techniques presented in this chapter rely on some assumptions that impose certain limitations. We discuss each of these limitations individually below.

IPOMDP semantics. In Section 4.2, we discussed RPOMDP semantics and the specific semantics we assume for the IPOMDPs used in this chapter. Specifically, we defined IPOMDPs to be (s,a)-rectangular with *full sticky* (*i.e.*, *static* in terms of RMDPs) uncertainty semantics. These semantics imply that nature, who adversarially chooses probability distributions from the uncertainty set, does so only once for each individual state-action pair. In other words, nature selects a single POMDP that is contained in the uncertainty set of the IPOMDP and does not change its choice over time. The use of convex optimization inherently limits us to these semantics. Any other semantics would require the NLPs defined in Section 4.3 to be adaptable to changes in nature's choices.

Unfolding and rectangularity. As noted in Remark 4.3, encoding an FSC's memory nodes into the state space of the IPOMDP prevents an exact robust policy evaluation as we lose (s,a)-rectangularity. Assuming full rectangularity on the unfolded state space, as we do here, provides a conservative bound and thus ensures the soundness of our results. Nonetheless, a more precise policy evaluation could potentially lead to the discovery of other, better-performing, robust policies since the evaluation results feed back into both CCP and SCP. Using techniques for robust dynamic programming on s-rectangular RMDPs (as briefly discussed in Section 3.4) could provide a first direction to alleviate this limitation.

FSC structure. As explained in Remark 4.4, requiring the policy variables to be bounded away from zero will result in FSCs with a fully connected memory update function and action mapping with full support. Both CCP and SCP search the space of all stochastic FSCs that satisfy these conditions, but FSCs outside of this set that perform better may exist. One possible alleviation is to specify some small threshold ε and prune the memory update and action mapping where the probabilities are below this threshold. However, it is unclear if such an approach would be worthwhile, as the pruning would have to be done in each iteration of CCP or SCP. Alternatively, we could fix a specific FSC structure and let the NLPs adhere to that structure. The downside of that approach is the same as that of the current approach: the NLPs still only consider a subset of all possible FSCs.

Future work. Further directions for future work are incorporating learning-based approaches such as recurrent neural networks to construct alternative policy representations (Carr et al., 2021; Galesloot et al., 2024), and extending the presented algorithms to other classes or RPOMDPs. Most notably, we envision adaptations toward RPOMDPs with L_1 uncertainty sets to be feasible with relatively minor changes to the vertex enumeration of CCP or dualization of SCP.

5

Robust Anytime Learning of Markov Decision Processes

This chapter considers an online model-based reinforcement learning setting. Our primary goal is to learn a robust MDP that captures an environment so that robust policies computed on this RMDP are robust against statistical errors made during the learning process. To that end, we introduce a Bayesian inference scheme that continuously learns the transition probabilities of an MDP in a robust anytime learning approach. Our method (1) approximates probabilities as intervals, (2) adapts to new data that may be inconsistent with an intermediate model, and (3) may be stopped at any time to compute a robust policy on the RMDP that represents the data so far. In particular, this learning scheme is capable of adapting to changes in the underlying environment during the learning process.

5.1 Introduction

Sequential decision-making in realistic scenarios is inherently subject to uncertainty, commonly captured via probabilities. *Markov decision processes* (MDPs) are the standard model to reason about such decision-making problems (Bertsekas, 2005; Puterman, 1994). A fundamental requirement for guaranteeing optimality of the policies and values computed on an MDP with regard to some objective is that the probabilities are precisely given. Already, a small misspecification of transition probabilities may lead to significant deterioration in the performance of a policy (Goyal and Grand-Clément, 2023; Mannor et al., 2007).

Inferring probabilities from data naturally introduces statistical errors. As a consequence, methods that learn MDPs from data, such as model-based reinforcement learning (Moerland et al., 2023) or PAC-learning (Strehl et al., 2009) need to be robust against such errors. Additionally, learning methods often assume the underlying environment remains fixed over time, and are not robust to any

disturbances or changes in the environment (Ashok et al., 2019; Jaksch et al., 2010; Strehl and Littman, 2008).

Robust MDPs (RMDPs) may be used to incorporate a layer of additional uncertainty around estimated probabilities via an *uncertainty set* (Goyal and Grand-Clément, 2023; Nilim and Ghaoui, 2005; Rigter et al., 2021a; Wiesemann et al., 2013). Robust policies computed on an RMDP account for the worst-case instance of this uncertainty set, thus inducing a *worst-case performance*, *i.e.*, a conservative bound on, *e.g.*, the reachability probability or expected reward.

5.1.1 Robust RL in Changing Environments

The central contribution presented in this chapter is as follows.

> **Contribution**
>
> We present a new approach to robust reinforcement learning in MDPs where the underlying environment may change over time.

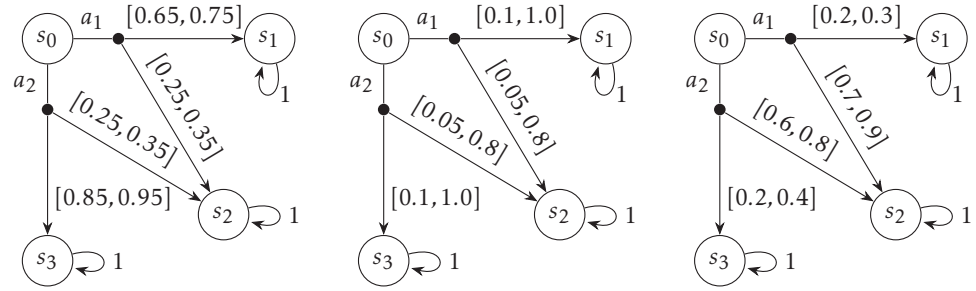
Specifically, we propose an iterative learning method that uses RMDPs as intermediate models and can *adapt to new data*, which may be inconsistent with prior assumptions. Our method learns intervals of probabilities for individual transitions through a Bayesian inference scheme. This *anytime-learning approach* employs *intervals with linearly updating conjugate priors* (Walter and Augustin, 2009) and can iteratively improve upon an RMDP that approximates the underlying environment.

This method not only decreases the size of each interval but may also increase it again in case of a so-called *prior-data conflict* where new data suggests the actual probability lies outside the current interval. Consequently, a newly learned interval does not need to be a subset of its prior interval. This property makes our method especially suitable for learning MDPs where the transition probabilities of the underlying environment change over time.

We summarize the key features of our learning method and what sets it apart from other methods.

- *An anytime approach.* The ability to iteratively update intervals that are not necessarily subsets of each other allows us to design an *anytime-learning approach*. At any time, we may stop the learning and compute a robust policy for the RMDP that the process has yielded thus far, together with the worst-case performance of this policy against a given objective. This performance may often already be sufficient, but when it is not satisfactory, *e.g.*, the worst-case probability of reaching a set of critical states may be below a certain threshold, we continue learning towards a new RMDP that more faithfully captures the true MDP due to the inclusion of further data. Thereby, we ensure that the robust policy gradually gets closer to the optimal policy for the true MDP.

- *Objective-driven.* Our method features the possibility to learn in a task-aware fashion, that is, to learn transitions that matter for a given objective. In



(a) Learned IMDP.

(b) IMDP after prior-data conflicts.

(c) Updated IMDP.

Figure 5.1: Illustration of learning IMDPs through linearly updating intervals in changing environment dynamics.

particular, for reachability or expected reward (temporal logic) objectives which require a certain set of target states to be reached, we only learn and update transitions along paths towards these states. Transitions outside those paths do not affect the satisfaction of the objective.

- *Adaptive to changing environment dynamics.* When using linearly updating intervals, our approach is adaptive to changing environment dynamics. That is, if during the learning process the probability distributions of the underlying MDP change, our method can easily adapt and learns these new distributions.

An illustration of how our anytime learning approach adapts to changing environment dynamics is given in Figure 5.1. In Figure 5.1a, the learning process has nearly converged to the actual environment dynamics, represented by an IMDP with small intervals. When the environment dynamics suddenly change, prior-data conflicts ensure the intervals enlarge again, as seen in Figure 5.1b. As more data is collected, the intervals start to converge again.

Structure of This Chapter

The remainder of this chapter is structured as follows. We start with the necessary preliminaries from Chapters 2 and 3 and background material on (robust) reinforcement learning and PAC learning in Section 5.2. In Section 5.3, we introduce the robust anytime-learning approach. Section 5.4 presents *linearly updating intervals* (LUI) as an effective learning method for our anytime-learning approach. Section 5.5 details how to adapt LUI to operate in changing environments. In Section 5.6, we empirically evaluate our approach, and in Section 5.7, we conclude the chapter with a discussion of limitations and future work.

5.2 Background: Robust Reinforcement Learning

In this section, we first cover the basics of model-based reinforcement learning, with a particular focus on robustness against the epistemic uncertainty stemming

from statistical errors made during the learning process.

5.2.1 Preliminaries

We start by recalling the necessary definitions from Chapters 2 and 3.

MDPs. The Markov decision processes (MDPs, Definition 2.1) considered in this chapter are tuples $\langle S, s_I, A, P, R\rangle$, where S is a finite set of states, $s_I \in S$ is the initial state, A is a finite set of actions, $P\colon S \times A \rightharpoonup \mathcal{D}(S)$ is the transition function, and $R\colon S \times A \rightharpoonup \mathbb{R}_{\geq 0}$ is the (positive) reward function.

RMDPs and IMDPs. The IMDPs in this chapter follow Definition 3.7 from Chapter 3. That is, an IMDP is a tuple $\langle S, s_I, A, \underline{P}, \overline{P}, R\rangle$ where S, s_I, A and R are the same as in the MDPs we consider here, and $\underline{P}\colon S \times A \times S \rightharpoonup [0,1]$ and $\overline{P}\colon S \times A \times S \rightharpoonup [0,1]$ are lower and upper bounds on the transition probabilities such that form consistent intervals. Specifically, $\underline{P}$ and $\overline{P}$ need to satisfy the following conditions.

$$\begin{aligned}
&\forall s, s' \in S, a \in A\colon && \underline{P}(s,a,s') = \bot \iff \overline{P}(s,a,s') = \bot,\\
&\forall s, s' \in S, a \in A(s)\colon && \underline{P}(s,a,s') = 0 \iff \overline{P}(s,a,s') = 0,\\
&\forall s, s' \in S, a \in A(s)\colon && \underline{P}(s,a,s') \leq \overline{P}(s,a,s').
\end{aligned}$$

Recall that an IMDP is a special case of an (s,a)-rectangular RMDP, for which the inner problem of robust dynamic programming can be solved efficiently through Algorithm 3.1.

Objectives and performance. In this chapter we consider all four objectives defined in Definition 2.5: reachability, reach-reward, stochastic shortest path, and discounted reward. We denote these objectives as

$$\varphi \in \{\mathbf{P}_{Max}(\Diamond T), \mathbf{R}_{Max}(\Diamond T), \mathbf{R}_{Min}(\Diamond T), \mathbf{R}_{Max}(\gamma)\}.$$

For IMDPs, we use both pessimistic (*i.e.*, robust) and optimistic objectives as defined in Definition 3.4, respectively:

$$\begin{aligned}
\underline{\varphi} &\in \{\mathbf{P}_{MaxMin}(\Diamond T), \mathbf{R}_{MaxMin}(\Diamond T), \mathbf{R}_{MinMax}(\Diamond T), \mathbf{R}_{MaxMin}(\gamma)\},\\
\overline{\varphi} &\in \{\mathbf{P}_{MaxMax}(\Diamond T), \mathbf{R}_{MaxMax}(\Diamond T), \mathbf{R}_{MinMin}(\Diamond T), \mathbf{R}_{MaxMax}(\gamma)\}.
\end{aligned}$$

All policies considered in this chapter are memoryless, that is, of type $\pi\colon S \to \mathcal{D}(A)$. The value function of an MDP is denoted $V\colon S \to \mathbb{R}$. The optimal value, denoted V^*, is the least fixed point of the Bellman equation for a given objective, as discussed in Section 2.2.1. For IMDPs we have the pessimistic and optimistic value functions $\underline{V}\colon S \to \mathbb{R}$ and $\overline{V}\colon S \to \mathbb{R}$, respectively, and the optimality is again denoted by $\underline{V}^*$ and $\overline{V}^*$. The performance of a policy π for objective φ in MDP M is defined as $\rho(\pi, M, \varphi) = V_{\pi,M,\varphi}$. The pessimistic and optimistic performances of a policy in an IMDP are defined as $\rho(\pi, \mathcal{M}, \underline{\varphi}) = \underline{V}_{\pi,\mathcal{M},\underline{\varphi}}(s_I)$ and $\rho(\pi, \mathcal{M}, \overline{\varphi}) = \overline{V}_{\pi,\mathcal{M},\overline{\varphi}}(s_I)$, respectively. For a complete discussion we refer back to Chapters 2 and 3.

5.2.2 Learning Probabilities

A natural application of RMDPs in both formal methods and AI comes in learning MDPs from data. Naive estimation of the transition probabilities from a finite amount of observations introduces estimation errors. When following a path through a learned MDP, these errors may accumulate, leading to potentially significant differences in values (and possibly optimal policies) between the learned model and the true underlying MDP (Goyal and Grand-Clément, 2023; Mannor et al., 2007). To account for these errors, confidence intervals around the probabilities or distributions may be computed via, *e.g.*, Hoeffding's inequality (Hoeffding, 1963) or the Weissman bound (Weissman et al., 2003), and included in the learned model, yielding an RMDP. Resulting policies and values can then be given a *probably approximately correct* (PAC) guarantee.

Learning MDPs, both with and without PAC guarantees, has been studied extensively in both formal methods and AI, namely in the form of statistical model checking (SMC) and reinforcement learning (RL).

Statistical Model Checking

Statistical model checking is the process of verifying whether a stochastic system satisfies a particular quantitative property through simulations until enough statistical evidence has been collected to accept or reject the property (Legay et al., 2010; Younes and Simmons, 2002). SMC is often used when the system is partially unknown or too large to fit an explicit state space in memory and applies to indefinite or infinite horizon properties such as reachability and reach-reward, or more general temporal logic objectives (Brázdil et al., 2024).

SMC with PAC guarantees (PAC-SMC) has been developed for MDPs and stochastic games (Ashok et al., 2019), and extended to continuous time MDPs (Agarwal et al., 2022). These methods either build IMDPs by deriving confidence intervals through Hoeffding's inequality or construct lower and upper Bellman equations that are updated directly, implicitly performing robust dynamic programming without building the underlying RMDP model (Ashok et al., 2019; Baier et al., 2023; Daca et al., 2016; Kretínský, 2016).

Reinforcement Learning

In contrast, RL is primarily concerned with finding an optimal policy that maximizes discounted or finite horizon reward objectives through efficient exploration (Sutton and Barto, 1998). The RL paradigm is usually split into *model-free* and *model-based* RL. Where model-free RL methods learn through an explicit trial-and-error process, model-based RL methods explicitly construct an estimated model (usually an MDP) of the environment and then perform planning on this model (Moerland et al., 2023; Sutton and Barto, 1998). An RL method is a *robust RL* method when it accounts for uncertainty or disturbances in the environment (Moos et al., 2022). RMDPs, and specifically L_1-MDPs based on the aforementioned Weissman bound, are used in model-based and robust RL to achieve PAC guarantees on the learned model (Strehl and Littman, 2008; Strehl et al., 2009) or efficient exploration through optimistic

policies (Jaksch et al., 2010).

It should be noted that the PAC-MDP framework of (Strehl et al., 2009) explicitly requires the sample efficiency of a learning algorithm to be polynomial in the input to be considered (efficiently) PAC. As a consequence, any non-finitary objective is not PAC-learnable following the PAC-MDP framework, and PAC-SMC methods are said to give anytime or best-effort guarantees (Yang et al., 2022). Recent work investigates techniques to reduce the amount of data required to achieve PAC guarantees in both SMC and RL (Budde et al., 2024; Meggendorfer et al., 2024; Wienhöft et al., 2023).

5.2.3 Learning Point Estimates by Counting

We now describe a general setup for learning point estimates of probabilities via *maximum likelihood estimation* (MLE) or *maximum a-posteriori* (MAP) estimation.

In RL settings, such as the one we also consider in this chapter, we do not have direct sampling access to individual state-action pairs. Instead, we need to sample according to some *exploration policy* that collects trajectories, one per *episode*, of the MDP from the initial state until some termination criterion is met. Since we are only concerned with learning the transition function in this chapter, we omit the rewards from the trajectories.

A dataset is a collection of m trajectories of length n: $\mathbb{D} = \{\langle s_t, a_0, \dots, s_n, a_n \rangle_t\}_{t \in [1:m]}$. We write $\#_{\mathbb{D}}(s,a)$ and $\#_{\mathbb{D}}(s,a,s')$ for the number of times a state-action pair and transition were observed in $\mathbb{D}$, respectively.

Maximum likelihood estimation (MLE) simply estimates probabilities by counting occurrences in the dataset $\mathbb{D}$. Assume a fixed state-action pair (s,a) in some MDP $M = \langle S, s_\iota, A, P, R \rangle$. Let $N = \#_{\mathbb{D}}(s,a) > 0$ be the number of times (s,a) was sampled, and each successor state is observed $k_i = \#_{\mathbb{D}}(s,a,s_i)$ times for $i = 1, \dots, m$ where $m = |Post_M(s,a)|$. The maximum likelihood estimate $\tilde{P}_{\mathrm{MLE}}(s,a)$ of $P(s,a)$ is given by

$$\tilde{P}_{\mathrm{MLE}}(s_i \mid s,a) = \frac{k_i}{N}.$$

When $N = 0$, MLE fails to produce a valid probability distribution for that state-action pair, and a default to fall back on needs to be used, for example, a uniform distribution or by making the state absorbing with a self-loop of probability one.

Another particular issue with MLE is that transitions with count $\#_{\mathbb{D}}(s,a,s') = 0$ will be estimated to have probability zero. In settings where one assumes knowledge of the underlying graph of the MDP, and thus whether (s,a,s') exists or not, the maximum likelihood estimate may contradict this prior knowledge, and again, an ad hoc fix is needed.

Alternatively, we may exploit such prior knowledge by incorporating it in a prior distribution and instead use *Maximum a-posteriori estimation* (MAP-estimation). MAP-estimation derives point estimates from a Dirichlet distribution that is updated based on new data. In an MDP, the observed transition counts of state-action pair

(s,a) follow the *multinomial likelihood*

$$Mn(k_1,\dots,k_n \mid P(\cdot|s,a)) = \frac{N!}{k_1!\cdot\ldots\cdot k_m!}\cdot\prod_{i=1}^{m} P(s_i|s,a)^{k_i}, \tag{5.1}$$

where the transition probabilities $P(s,a,s_i)$ are unknown and given by a Dirichlet distribution with parameters $\alpha_1,\dots,\alpha_m$ (Bishop, 2007):

$$Dir(P(\cdot|s,a) \mid \alpha_1,\dots,\alpha_m) \propto \prod_{i=1}^{m} P(s_i|s,a)^{\alpha_i-1}.$$

The Dirichlet distribution is a *conjugate prior* to the multinomial likelihood, meaning that we do not need to compute the posterior distribution explicitly, but instead, we just update the parameters of the prior Dirichlet distribution. Formally, conjugacy is a closure property, see (Jacobs, 2020).

Given a prior Dirichlet distribution with parameters $\alpha_1,\dots,\alpha_m$, and observing the i-th successor state k_i times, the posterior Dirichlet distribution is given by simply adding k_i to parameter α_i:

$$Dir(P(\cdot|s,a) \mid \alpha_1+k_1,\dots,\alpha_m+k_m).$$

Having computed the posterior Dirichlet distribution, MAP-estimation can be used to infer the probabilities. These estimates are given by the *mode* of each parameter of the posterior distribution:

$$\tilde{P}_{\text{MAP}}(s_i|s,a) = \frac{\alpha_i - 1}{(\sum_{j=1}^{m}\alpha_j) - m}.$$

When all parameters α_i are equal, MAP-estimation yields uniform distributions.

Applying MLE or MAP-estimation for all state-action pairs of the MDP M yields a complete estimate of the transition function of M. The estimated MDP from $\mathbb{D}$, either by MLE or MAP-estimation, is given by $\tilde{M} = \langle S, s_\iota, A, \tilde{P}, R\rangle$ where $\tilde{P}$ is either $\tilde{P}_{\text{MLE}}$ or $\tilde{P}_{\text{MAP}}$, respectively.

5.2.4 Anytime PAC Learning

Given a point estimate of a probability $\tilde{P}(s_i|s,a)$, either derived by MLE or MAP-estimation, we can construct *probably approximately correct* (PAC) intervals via Hoeffding's inequality. Given $N = \#_{\mathbb{D}}(s,a)$ samples and a fixed *confidence level* $1-\delta$, we use Hoeffding's inequality (Hoeffding, 1963) to construct intervals that will define an IMDP $\mathcal{M}$ that is *probably approximately correct* (PAC) for the underlying true MDP M. That is, with high probability $1-\delta$, the optimal value function V^*_M of the MDP is bounded by the optimal pessimistic and optimistic value functions of $\mathcal{M}$, for every state:

$$\forall s \in S\colon \mathbb{P}(\underline{V}^*(s) \le V^*(s) \le \overline{V}^*(s)) > 1-\delta.$$

To construct a PAC guarantee on the entire learned MDP M, and consequently also the optimal value for some objective, we need to distribute δ over all transitions that need to be estimated, *i.e.*, those with probabilities strictly between zero and one. To that end, let

$$\delta_M = \frac{\delta}{\sum_{(s,a)\in S\times A} \mathbb{1}[|Post_M(s,a)| > 1] \cdot |Post_M(s,a)|}.$$

Then, using Hoeffding's inequality to compute the distributed interval size

$$\zeta_M = \sqrt{\frac{\log(2/\delta_M)}{2N}}.$$

Using this ζ_M, we then construct the intervals

$$\underline{P}(s,a,s_i) = \tilde{P}(s_i \,|\, s,a) - \zeta_M, \qquad \overline{P}(s,a,s_i) = \tilde{P}(s_i \,|\, s,a) + \zeta_M, \tag{5.2}$$

such that $\mathcal{M} = \langle S, s_I, A, \underline{P}, \overline{P}, R\rangle$ forms an interval MDP.

To derive a formal PAC guarantee on this IMDP, some conditions need to be met. First, it should be noted that Hoeffding's inequality assumes the samples are *independent and identically distributed* (i.i.d.). This is, in general, not the case when sampling trajectories from an MDP, as shown by (Starre et al., 2023; Strehl and Littman, 2008), as the samples are not independent. However, Strehl and Littman (2008, Appendix A) establish that one can still use the samples *as if* i.i.d. when only providing an upper bound on the probability of specific sequences of successor states occurring.

A second concern is the coverage of samples. Specifically, it should be possible to sample every state-action pair infinitely often. One method to ensure this condition on the sample coverage is satisfied is by requiring that every state can be reached (within a finite number of steps) from every other state under some (randomized) policy (Marjani et al., 2023). If the underlying MDP is episodic, *i.e.*, may be reset, the requirement can be weakened and every state only needs to be reachable from the initial state. Depending on the objective, further processing of *end-components* may be needed (Ashok et al., 2019). An end-component is a strongly connected sub-MDP for which a policy exists such that all paths following that policy remain within the end-component. For details on end-components we refer to (Baier and Katoen, 2008; de Alfaro, 1997).

Proposition 5.1. *Let M be an MDP and $\mathcal{M} = \langle S, s_I, A, \underline{P}, \overline{P}, R\rangle$ the $1-\delta$-correct IMDP with $\underline{P}$ and $\overline{P}$ constructed through Equation 5.2. Let V^* be the optimal value function of M, and $\underline{V}$ and $\overline{V}$ be the optimal pessimistic and optimistic value functions of $\mathcal{M}$, respectively. We then have*

$$\forall s \in S\colon \mathbb{P}(\underline{V}^*(s) \le V^*(s) \le \overline{V}^*(s)) > 1-\delta.$$

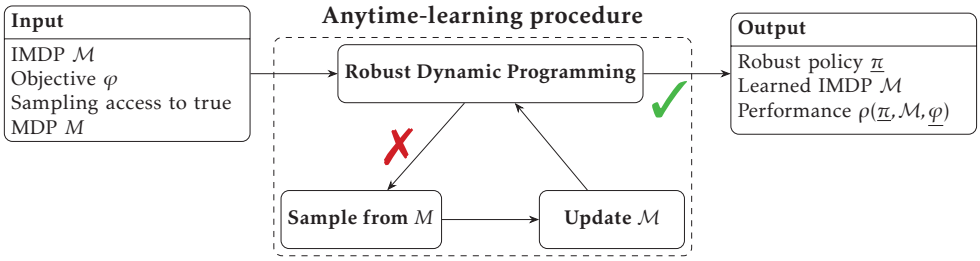


Figure 5.2: Robust anytime-learning procedure outline.

5.3 Robust Anytime Learning

We now present the main contribution of this chapter: a model-based robust RL algorithm that is adaptive to changing environments.

We assume the following setting, as also shown in Figure 5.2. We have an unknown but fixed MDP $M = \langle S, s_\iota, A, P, R\rangle$, which we will refer to as the *true MDP*, an *initial prior IMDP* $\mathcal{M} = \langle S, s_\iota, A, \underline{P}, \overline{P}, R\rangle$, and an objective φ which we want to satisfy. A discussion of prior (and other parameter) choices follows in Section 5.6.

In this setting, we assume the underlying graph of the true MDP M is known.

Assumption 5.2 (Underlying graph)**.** *We assume that the underlying graph of the true MDP M is known and that the IMDP $\mathcal{M}$ always has the same graph structure. That is,*

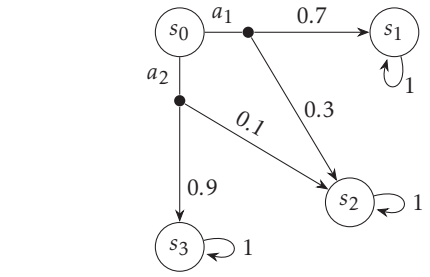
- *transitions that do not exist in M (transitions of probability 0) do also not exist in the IMDP $\mathcal{M}$,* i.e., $\forall s, s' \in S, a \in A\colon P(s'|s,a) = 0 \iff \underline{P}(s,a,s') = \overline{P}(s,a,s') = 0$,
- *transitions of probability 1 in M are assigned the point interval $[1,1]$ in $\mathcal{M}$,* i.e., $\forall s, s' \in S, a \in A\colon P(s'|s,a) = 1 \iff \underline{P}(s,a,s') = \overline{P}(s,a,s') = 1$,
- *any other transition of non-zero probability p has an interval with a lower bound of at least $p_{graph} > 0$ in $\mathcal{M}$,* i.e., $\forall s, s' \in S, a \in A\colon P(s'|s,a) \neq 0 \wedge P(s'|s,a) \neq 1 \implies \underline{P}(s,a,s') \geq p_{graph}$.

Under Assumption 5.2, we construct the initial prior IMDP $\mathcal{M}$ to have transitions of probability 0 and 1 exactly where the true MDP M has these too, and interval transitions $[p_{graph}, 1 - p_{graph}]$ for all other transitions, with $p_{graph} > 0$ free to choose. In particular, our approach does not require p_{graph} to be smaller than the smallest probability $p > 0$ occurring in M, which we also do not assume to be known. Alternatively, if further knowledge is available, one may use any other prior IMDP as long as it satisfies Assumption 5.2.

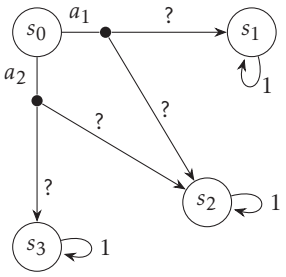
We now outline our anytime-learning procedure as illustrated in Figure 5.2.

i. **Input.** We start with an initial prior IMDP $\mathcal{M}$ and objective φ. We assume (grey box) access to the unknown *true MDP* M to sample trajectories from.

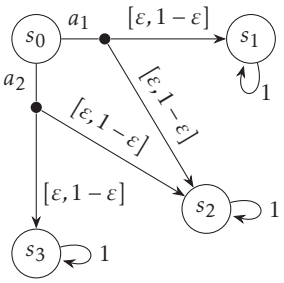
ii. **Robust dynamic programming.** We compute a robust policy $\underline{\pi}$ for the pessimistic extension of φ, *i.e.* $\underline{\varphi}$, in the IMDP $\mathcal{M}$, together with the worst-case performance of the objective: $\rho(\underline{\pi}, \mathcal{M}, \underline{\varphi})$.



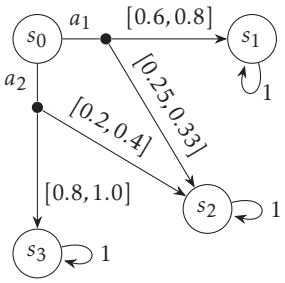
(a) The true MDP.



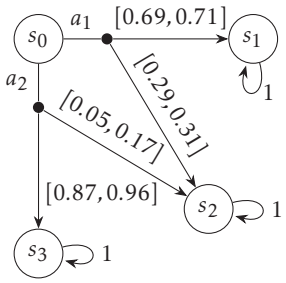
(b) Assumption 5.2.



(c) Initial prior.



(d) Intermediate IMDP.



(e) Converging towards the true MDP.

Figure 5.3: Process flow on an example MDP.

iii. **Anytime learning.** We have the following loop:

(a) **Exploration.** We sample one or more trajectories from the true MDP M, using the *optimism in the face of uncertainty* principle, *i.e.*, according to the optimistic policy $\overline{\pi}$ for $\overline{\varphi}$ in $\mathcal{M}$.

(b) **Update.** We update the intervals of the IMDP $\mathcal{M}$ in accordance with the newly collected data. This update yields a new IMDP that more faithfully captures all collected data up to this point than the previous IMDP.

(c) **Repeat.** We start again at step 2 with this new IMDP.

iv. **Output.** The process may be stopped at any moment and yields the latest IMDP $\mathcal{M}$ together with robust policy $\underline{\pi}$ and the performance $\rho(\underline{\pi}, \mathcal{M}, \underline{\varphi})$.

The effects of this procedure are illustrated in Figure 5.3. In Figure 5.3a, we see an example MDP M to learn, and Figure 5.3b shows the assumed knowledge about M. Figure 5.3c shows the initial IMDP $\mathcal{M}$ constructed from Figure 5.3b, using a (symbolic or explicit) lower bound $p_{graph} > 0$ to ensure that all lower bounds of $\mathcal{M}$ are strictly greater than zero. In Figure 5.3d, we see an intermediate learned IMDP. Some intervals may already have successfully converged towards the probability of that transition in the true MDP, while others may be very inaccurate due to a low sample size and thus a bad estimate. Finally, Figure 5.3e depicts the learned IMDP converging towards the true MDP.

5.4 Linearly Updating Intervals

To be adaptive to changing environments, the *Update* step of our anytime learning procedure needs to be flexible in the presence of new, possibly inconsistent, data. To that end, we employ the Bayesian approach of *intervals with linearly updating conjugate priors* (Walter and Augustin, 2009) to learn intervals of probabilities. These intervals, which we shall simply refer to as *linearly updating intervals* (LUI), are then used to construct and iteratively update IMDPs. We first present LUI in its general form and discuss some modifications to increase robustness against changing environments in Section 5.5.

5.4.1 Learning Linearly Updating Intervals

We have the same setup as for learning probabilities in Section 5.2.2. We assume a dataset $\mathbb{D}$ and counters $N = \#_{\mathbb{D}}(s,a)$ and $k_i = \#_{\mathbb{D}}(s,a,s_i)$ for each $s_i \in Post_M(s,a)$. For further ease of presentation, assume a fixed state-action pair (s,a), and let $\mathcal{P}$ be a shorthand for the intervals at (s,a): $\mathcal{P}_i = [\underline{P}(s,a,s_i), \overline{P}(s,a,s_i)]$.

Each IMDP transition (s,a,s_i) is assigned a *prior interval* $\mathcal{P}_i = [\underline{P}_i, \overline{P}_i]$, and a *prior strength interval* $[\underline{n}_i, \overline{n}_i]$ that represents a minimum and maximum number of samples on which the prior interval is based. The greater the values of the strength interval, the more emphasis is placed on the prior, and the more data is needed to significantly change the prior when computing the posterior. The greater the difference between the $\underline{n}_i$ and $\overline{n}_i$, the greater the difference between a *prior-data conflict* and a *prior-data agreement*.

Definition 5.3 (Posterior interval computation)**.** The interval $[\underline{P}_i, \overline{P}_i]$ can be updated to $[\underline{P}'_i, \overline{P}'_i]$, using $N = \#(s,a)$ and $k_i = \#(s,a,s_i)$, as follows:

$$\underline{P}'_i = \begin{cases} \frac{\overline{n}_i \underline{P}_i + k_i}{\overline{n}_i + N} & \text{if } \forall j\colon \frac{k_j}{N} \geq \underline{P}_j \text{ (prior-data agreement),} \\ \frac{\underline{n}_i \underline{P}_i + k_i}{\underline{n}_i + N} & \text{if } \exists j\colon \frac{k_j}{N} < \underline{P}_j \text{ (prior-data conflict).} \end{cases} \tag{5.3}$$

$$\overline{P}'_i = \begin{cases} \frac{\overline{n}_i \overline{P}_i + k_i}{\overline{n}_i + N} & \text{if } \forall j\colon \frac{k_j}{N} \leq \overline{P}_j \text{ (prior-data agreement),} \\ \frac{\underline{n}_i \overline{P}_i + k_i}{\underline{n}_i + N} & \text{if } \exists j\colon \frac{k_j}{N} > \overline{P}_j \text{ (prior-data conflict).} \end{cases} \tag{5.4}$$

The strength interval is updated straightforwardly by adding the number of samples N: $[\underline{n}'_i, \overline{n}'_i] = [\underline{n}_i + N, \overline{n}_i + N]$.

The initial values for the priors of each state-action pair can be chosen freely subject to the constraints $0 < \underline{P}_i \leq \overline{P}_i \leq 1$, and $\overline{n}_i \geq \underline{n}_i \geq 1$.

Key properties of linearly updating intervals.

- *Convergence in the infinite run.* Under the assumption that the true MDP does not change, each interval will converge to the exact transition probability when the total number of samples processed tends to infinity, regardless of how many samples are processed *per iteration* (Walter and Augustin, 2009). This

assumption is, however, not required for our work. If the true MDP changes over time, or is *adversarial* (*i.e.*, a RMDP), our method is still applicable, but will not converge to a fixed MDP.

- *Prior-data conflict.* When the estimated probability k_i/N lies outside the current interval, a so-called *prior data conflict* occurs. Consequently, if at some point we derive an interval that does not contain the true transition probability, the method will correct itself later on.

- *Closure properties under updating.* Finally, updating is closed in two specific ways. First, any interval of probabilities is updated again to a valid interval of probabilities, and second, any set of intervals at a state-action pair that contains a valid probability distribution over the successor states will again contain a valid distribution over successor states after updating.

A key requirement for computing robust policies on RMDPs is that the lower bound of every interval is strictly greater than zero. This closure property is formalized as follows.

Theorem 5.4 (Closure of intervals under learning)**.** *For any valid prior interval* $[\underline{P}, \overline{P}]$ *with* $0 < \underline{P} \leq \overline{P} \leq 1$*, we have that the posterior* $[\underline{P}', \overline{P}']$ *computed via Definition 5.3 also satisfies* $0 < \underline{P}' \leq \overline{P}' \leq 1$.

The proof of Theorem 5.4 is straightforward and relies on the observation that for any amount of finite data, a single update can only grow closer to zero but not become zero and that iterated updates converge to the true probability which is assumed to be non-zero.

Proof of Theorem 5.4. For any transition (s, a, s_i), the following holds. We assume a valid prior, that is, $0 < \underline{P}_i \leq \overline{P}_i \leq 1$. We have an empirical estimate of k_i/N. Then we also have $0 < \underline{P}'_i \leq \overline{P}'_i \leq 1$, where the first inequality $0 < \underline{P}'_i$ follows from the fact that Equation 5.3 can only be 0 when their nominator is zero, which is not possible when $\underline{n}_i \geq 1$ (or $\overline{n}_i \geq 1$) and $\underline{P}_i > 0$. The second inequality, $\underline{P}'_i \leq \overline{P}'_i$ follows directly from Equations 5.3 and 5.4 together with $\underline{P}_i \leq \overline{P}_i$. The third inequality $\overline{P}'_i \leq 1$ follows again from Equation 5.4 when $\overline{P}_i \leq 1$ and $k_i \leq N$. All reasoning above is independent of prior-data agreement or conflict and thus applies to both cases in each equation. □

Furthermore, we also have closure properties at each state-action pair. In particular, if we choose our prior intervals such that there is at least one valid probability distribution at the state-action pair, then the posterior intervals will again contain a valid probability distribution. We formalize this notion by examining the sum of the lower and upper bounds of the intervals in the following theorem.

Theorem 5.5 (Closure of distributions under learning)**.** *We have the following bounds on the set of posterior distributions. In the case of a prior data agreement, we have the*

sum of the posterior bounds bounded by the sum of the prior bounds and the value 1. *That is,*

$$\sum_i \underline{P}_i \leq \sum_i \underline{P}'_i \leq 1, \qquad 1 \leq \sum_i \overline{P}'_i \leq \sum_i \overline{P}_i. \tag{5.5}$$

In case of a prior-data conflict, the sum of posterior bounds is no longer necessarily bounded by the prior, and we only have

$$\sum_i \underline{P}'_i \leq 1 \leq \sum_i \overline{P}'_i. \tag{5.6}$$

Note, however, that this last constraint (5.6) is already sufficient to ensure that there is a valid probability distribution at the state-action pair.

The proof of Theorem 5.5 uses the following Lemma:

Lemma 5.6. *At a state-action pair with m successor states, there can be at most* $m-1$ *prior-data conflicts on the upper (or lower) bound.*

Proof. We prove the lemma for the upper bound, and a proof for the lower bound follows by symmetry. Assume a valid prior, that is, $\sum_i \overline{P}_i \geq 1$. Suppose there is a prior-data conflict for every interval, *i.e.*, $\forall i : \frac{k_i}{N} > \overline{P}_i$. Then we have

$$1 = \sum_i \frac{k_i}{N} > \sum_i \overline{P}_i \geq 1,$$

which is clearly not possible. The possibility for $m-1$ prior-data conflicts is witnessed in the following example. Take a state-action pair with two successor states, s_1 and s_2. Then $m = 2$. Take one sample, *i.e.*, $N = 1$, and suppose we observe s_1, such that $k_1 = 1$ and $k_2 = 0$. Then for any valid prior intervals $I_1 = [\underline{P}_1, \overline{P}_1]$ and $I_2 = [\underline{P}_2, \overline{P}_2]$ we have $k_1/N = 1 > \overline{P}_1$ and $k_2/N = 0 < \overline{P}_2$. Hence, a prior-data conflict at I_1 but not at I_2, thus $m-1$ conflicts at the state-action pair in total. □

We additionally have the following Lemma:

Lemma 5.7. *A prior-data conflict at the upper bound implies a prior-data agreement at the lower bound and vice versa.*

Proof. Assume a conflict at the upper bound $\overline{P}_i$. Then we have $\frac{k_i}{N} > \overline{P}_i \geq \underline{P}_i$, which is a prior-data agreement with $\underline{P}_i$ by definition. The opposite direction follows by symmetry. □

Now we prove Theorem 5.5.

Proof of Theorem 5.5. We start with the constraints in Equation 5.5.

First, $1 \leq \sum_i \overline{P}'_i$. We use that there is a prior-data agreement, that is, $\forall i\colon \frac{k_i}{N} \leq \overline{P}_i$. Then we derive

$$\sum_i \overline{P}'_i = \sum_i \frac{\overline{n}_i \overline{P}_i + k_i}{\overline{n}_i + N} \quad \geq \quad \sum_i \frac{\overline{n}_i \frac{k_i}{N} + k_i}{\overline{n}_i + N} = \sum_i \frac{\frac{\overline{n}_i k_i + k_i N}{N}}{\overline{n}_i + N}$$
$$= \sum_i \frac{\frac{k_i(\overline{n}_i + N)}{N}}{\overline{n}_i + N} = \sum_i \frac{k_i}{N} = \frac{1}{N} \sum_i k_i = \frac{N}{N} = 1.$$

The bound $\sum_i \overline{P}'_i \leq \sum_i \overline{P}_i$ is derived using that

$$\forall i\colon \frac{k_i}{N} \leq \overline{P}_i \iff \forall i\colon k_i \leq \overline{P}_i N.$$

Then, it follows that

$$\sum_i \overline{P}'_i = \sum_i \frac{\overline{n}_i \overline{P}_i + k_i}{\overline{n}_i + N} \quad \leq \quad \sum_i \frac{\overline{n}_i \overline{P}_i + \overline{P}_i N}{\overline{n}_i + N} = \sum_i \frac{\overline{P}_i(\overline{n}_i + N)}{\overline{n}_i + N} = \sum_i \overline{P}_i.$$

The proof for the bounds

$$\sum_i \underline{P}_i \leq \sum_i \underline{P}'_i \leq 1$$

is symmetrical to the one above.

Next, we consider the case for a prior-data conflict, that is, the bounds from Equation 5.6. The existential condition $\exists j\colon \frac{k_j}{N} \geq \overline{P}_j$ does not have to be unique; hence we make a case distinction on the indexes for which the existential quantification holds and for which it does not. Let $I = \{1, \ldots, m\}$ be the set of indices that enumerates the m successor states at the state-action pair we consider. By Lemma 5.6, we know that there are at most $m-1$ prior-data conflicts. Hence, we can partition I into two non-empty subsets, I_A containing all indices where the point estimate agrees with the prior interval, and I_C the set of indices where there is a prior-data conflict. That is,

$$I_A = \{i \in I \mid \frac{k_i}{N} \leq \overline{P}_i\}, \qquad I_C = \{i \in I \mid \frac{k_i}{N} > \overline{P}_i\}.$$

We use this partition to split the sum over all indices in two:

$$\sum_i \overline{P}'_i = \sum_{i \in I} \frac{\underline{n}_i \overline{P}_i + k_i}{\underline{n}_i + N} = \sum_{i \in I_A} \frac{\underline{n}_i \overline{P}_i + k_i}{\underline{n}_i + N} + \sum_{i \in I_C} \frac{\underline{n}_i \overline{P}_i + k_i}{\underline{n}_i + N}$$

We now reason on each part separately.

For $i \in I_A$ we have $\frac{k_i}{N} \leq \overline{P}_i$, which also means $N \leq \frac{k_i}{\overline{P}_i}$.

$$\sum_{i \in I_A} \frac{\underline{n}_i \overline{P}_i + k_i}{\underline{n}_i + N} \geq \sum_{i \in I_A} \frac{\underline{n}_i \overline{P}_i + k_i}{\left(\underline{n}_i + \frac{k_i}{\overline{P}_i}\right)} = \sum_{i \in I_A} \frac{\underline{n}_i \overline{P}_i + k_i}{\left(\frac{\underline{n}_i \overline{P}_i + k_i}{\overline{P}_i}\right)} = \sum_{i \in I_A} \overline{P}_i \left(\frac{\underline{n}_i \overline{P}_i + k_i}{\underline{n}_i \overline{P}_i + k_i}\right) = \sum_{i \in I_A} \overline{P}_i.$$

Next, for $i \in I_C$ we have $\frac{k_i}{N} > \overline{P}_i$, and thus also $k_i > \overline{P}_i N$. Then we have the following:

$$\sum_{i \in I_C} \frac{\underline{n}_i \overline{P}_i + k_i}{\underline{n}_i + N} > \sum_{i \in I_C} \frac{\underline{n}_i \overline{P}_i + \overline{P}_i N}{\underline{n}_i + N} = \sum_{i \in I_C} \overline{P}_i \frac{\underline{n}_i + N}{\underline{n}_i + N} = \sum_{i \in I_C} \overline{P}_i.$$

Finally, we put the two partitions back together using the inequalities derived on both and conclude by using the assumption that the prior is valid, *i.e.*, $\sum_i \overline{P}_i \geq 1$:

$$\sum_i \overline{P}'_i = \sum_{i \in I_A} \frac{\underline{n}_i \overline{P}_i + k_i}{\underline{n}_i + N} + \sum_{i \in I_C} \frac{\underline{n}_i \overline{P}_i + k_i}{\underline{n}_i + N} > \sum_{i \in I_A} \overline{P}_i + \sum_{i \in I_C} \overline{P}_i = \sum_{i \in I} \overline{P}_i \geq 1.$$

The proof for the lower bounds, that $\sum_i \underline{P}'_i \leq 1$, follows the same reasoning by symmetry. □

5.5 LUI in Changing Environments

Previously, we assumed a fixed unknown true MDP M to learn. But what if the transition probabilities of M suddenly change? Suppose there are two unknown true MDPs, M_1 and M_2, with the same underlying graph but (possibly) different transition probabilities. After an unknown number of interactions with environment M_1, we suddenly continue interacting with M_2.

LUI, as introduced above, is somewhat capable of dealing with this scenario. If the change from M_1 to M_2 happens early on, the prior strengths of the intervals are not extremely large yet, and hence adaptation to new data from M_2 through prior-data conflicts is still feasible. The more data LUI has processed, however, the more future data will be needed to make any significant change to the intervals learnt thus far. Since we do not assume any knowledge on *when* the change in environments will happen, we cannot rely on prior-data conflicts in the standard update rule alone.

We modify the basic LUI by introducing a bound on the strength of the intervals. Such bounds are also sometimes referred to as sliding-window approaches and have been suggested before to be used in such scenarios (McCallum, 1995). Additionally, Gajane et al. (2018) proposes to modify the UCRL2 algorithm with such a window.

We set an upper bound on the strength of the priors, and update the strength in-

tervals up to this bound. Specifically, let $n_{\text{Max}} = [\underline{n}_{\text{Max}}, \overline{n}_{\text{Max}}]$, the posterior strength interval is then given by

$$[\underline{n}'_i, \overline{n}'_i] = [\min(\underline{n}_i + N, \underline{n}_{\text{Max}}), \min(\overline{n}_i + N, \overline{n}_{\text{Max}})].$$

The probability intervals themselves are updated in the same way as before in Definition 5.3. By limiting the prior strength in this way, we trade theoretical convergence for adaptability. The weaker the prior, the greater the effect of a prior-data conflict, and hence adaptability to new data. When suddenly changing environments, new data will likely lead to prior-data conflicts and, thus, a higher adaptability of the overall learning method.

5.5.1 Sampling Policies

Above, we assumed a dataset $\mathbb{D}$ of trajectories was given. To actually obtain the trajectories, we use the well-established *optimism in the face of uncertainty* principle (Munos, 2014). We compute the optimal policy $\overline{\pi}^*$ for the optimistic extension of the objective φ, *i.e.* $\overline{\varphi}$, in the current IMDP and use this policy for exploration. To make exploration objective-driven, we only sample transitions along trajectories toward the target state(s) of the objective. When the last seen state has a probability of zero or one to reach the target for reachability or reward zero or infinity, we restart. States satisfying these conditions can be found by analyzing the graph of the true MDP, which by Assumption 5.2 is given (Baier and Katoen, 2008).

Trajectories and iterations. Our method is iterative in terms of updating the IMDP $\mathcal{M}$ and computing a robust policy. After updating the IMDP, we also compute a new exploration policy, based on the new IMDP. Each iteration consists of processing at least one, but possibly more, trajectories. To determine how many trajectories to collect, we use a doubling-counting scheme, where we keep track of how often every state-action pair and transition is visited during exploration (Jin et al., 2020). An iteration is completed when any of the counters is doubled with respect to the previous iteration. A detailed description of this exploration schedule is given in Algorithm 5.1.

Randomization parameter. Optimism in the face of uncertainty, *i.e.*, using the optimal optimistic policy for exploration, is sufficient for exploration in most scenarios. In our setting of arbitrarily changing environments, however, this deterministic policy may be severely limiting. As the environments change, a different action may become the optimal choice at some state s. However, to detect this change, we need data from all actions at s. Hence, we need to ensure that every action is always sampled with some positive probability.

To that end, we introduce a hyperparameter $\xi \in [0,1]$, and follow with probability ξ the action of the optimistic policy, and distribute the remaining $1-\xi$ uniformly over the other actions, yielding a memoryless randomized policy.

In our experimental evaluation, presented next, we include an ablation study on both upper bounding the prior strength, as well as on using randomization in the exploration policy.

Algorithm 5.1: Exploration scheme.

Input: K : number of trajectories.
Input: H : max trajectory length.
Input: $M = \langle S, s_\iota, A, P, R \rangle$: underlying MDP.
Input: φ : objective.
Input: $\mathcal{A}$: algorithm for exploration and robust verification.

for $s, a \in S \times A$ **do**
  ▷ *Initialize counters*
  $\#(s,a) = 0$
  $\#(s,a,s') = 0 \colon \forall s' \in S$
  $\#_i(s,a) = 0$
  $\#_i(s,a,s') = 0 \colon \forall s' \in S$
end for
for $k \in [1, \cdots, K]$ **do**
  if $\exists s, a \in S \times A \colon \#_i(s,a) >= \#(s,a)$ **then**
    ▷ *Compute new policies.*
    Give iteration counters $\#_i(s,a)$ and $\#_i(s,a,s')$ to $\mathcal{A}$
    Get sampling policy from $\mathcal{A} : \pi_{\text{sampling}}$
    Get robust policy from $\mathcal{A} : \pi_{\text{robust}}$
    Evaluate π_{robust} on M according to φ
    ▷ *Update global counters*
    $\#(s,a)$ += $\#_i(s,a) \colon \forall s, a \in S \times A$
    $\#(s,a,s')$ += $\#_i(s,a,s') \colon \forall s, a \in S \times A \times S$
    ▷ *Reset iteration counters*
    $\#_i(s,a) = 0 \colon \forall s, a \in S \times A$
    $\#_i(s,a,s') = 0 \colon \forall s, a \in S \times A \times S$
  end if
  Sample τ from M following π_{sampling} with max size H
  Increment $\#_i(s,a)$ and $\#_i(s,a,s')$ according to τ.
end for

5.6 Experimental Evaluation

We implement LUI on top of the verification tool PRISM (Kwiatkowska et al., 2011). We compare our method to point estimates derived via MAP-estimation (MAP) and with IMDPs derived from PAC learning (PAC) and with bounds akin to the UCRL2 algorithm (Jaksch et al., 2010) (UCRL). Note that this is not an exact implementation of UCRL2 as our setting is different, as we are interested in robust policies and performance, while UCRL2 only uses optimistic policies. All experiments were performed on a 4GHz Intel Core i9 CPU, using a single core. Each experiment is repeated 100 times and reported with a 95% confidence interval.

Without knowledge about the true MDP apart from Assumption 5.2, we must define an appropriate prior interval for every transition. We set $p_{graph} = 1\text{e-}4$ as constant and define the prior IMDP with intervals $[\underline{P}(s,a,s') = p_{graph}, \overline{P}(s,a,s') = 1 - p_{graph}]$. and strength intervals $[\underline{n}_i, \overline{n}_i] = [5, 10]$ at every transition (s,a,s_t), as in Figure 5.3c. For MAP, we use a prior of $\alpha_i = 10$ for all i. The same prior is used for

the point estimates of PAC and UCRL, together with an error rate of $\delta = 0.01$.

Research questions. This experimental evaluation aims to establish the effectiveness of our approach, LUI, compared with the other methods. To that end, we pose the following research questions for each learning method:

RQ1. How does the learned robust policy perform on the true underlying model?

RQ2. How does the uncertainty set of the IMDP constructed by the learning method change during the learning process?

RQ3. How well does the learned IMDP predict a robust policy's performance on the true MDP?

RQ4. How robust is the learning method against changes in the underlying environment during the learning process?

Evaluation metrics. We consider three metrics to evaluate the learning methods and answer our research questions.

- *Performance.* We evaluate the performance of the robust policy on the true MDP with regard to the objective, *i.e.*, $\rho(\underline{\pi}, M, \varphi)$.

- *Model Error.* We define the model error as the average distance between the true probability and the furthest bound of the interval. Specifically, for each transition (s, a, s'), we compute

$$\max(|P(s' \mid s, a) - \underline{P}(s, a, s')|, |P(s' \mid s, a) - \overline{P}(s, a, s')|),$$

  i.e., the maximum distance between the true probability and the lower and upper bounds of the interval in the IMDP (or the point estimate for MAP-estimation). We then take the average over all these distances.

- *Performance Estimation Error.* We define the performance estimation error as the difference between the performance of the robust policy on the learned IMDP (the worst-case performance) and the performance on the true MDP:

$$\rho(\underline{\pi}, M, \varphi) - \rho(\underline{\pi}, \mathcal{M}, \underline{\varphi}).$$

  While values closer to zero are preferable, a positive estimation error indicates that the robust performance on the learned model of a method is not a lower (conservative) bound on the actual performance of the policy. In particular, an estimation error above zero shows the policy is misleading in terms of predicting its performance.

Benchmark environments. We use the following two well-known benchmarks for detailed analysis of the evaluation presented here.

- *Betting Game* (Bäuerle and Ott, 2011). The agent starts with 10 coins and attempts to maximize the number of coins after 6 bets. When a bet is won, the number of coins placed is doubled; when lost, the number of coins placed is removed. The agent may bet 0, 1, 2, 5, or 10 coins. We consider a version of the game that is favorable to the player, with a win probability of 0.8 per bet. After 6 bets, the player receives a reward equal to the number of coins left. The objective is to maximize the reward: $\mathbf{R}_{Max}(\Diamond T)$, where T is the set of terminal states after six bets.

- *Chain Problem* (Araya-López et al., 2011). We consider a chain of 30 states. There are three actions: one progresses with a probability of 0.95 to the next state, and the other resets the model to the initial state with a probability of 0.05. The second action does the same but with reversed probabilities. The third action has a probability of 0.5 for both cases. Every action gets a reward of 1. The objective is to minimize the reward to reach the last state of the chain, *i.e.*, stochastic shortest path.

We use the following four benchmarks for additional supporting results.

- *Aircraft* (Kochenderfer, 2015). We model a small, simplified instance of the aircraft collision avoidance problem. Two aircraft, one controlled, one adversarial, approach each other. The controlled aircraft can increase, decrease, or stay at the current altitude with a success probability of 0.8. The adversarial aircraft can do the same but does so with probabilities 0.3, 0.3, and 0.4, respectively. The goal is to maximize the probability of the two aircraft passing each other without a collision.

- *Bandit* (Lattimore and Szepesvári, 2020). We consider a 99-armed bandit, where each arm (action) has an increased success probability, $0.01, 0.02, \ldots, 0.99$ respectively. The goal is to find the action that has the highest probability of success, expressed as a reachability objective.

- *Betting Game (Unfavorable)*. We additionally consider an unfavorable version of the Betting Game, where the win probability is 0.2 per bet.

- *Grid*. We consider a slippery 10×10 grid world as in (Derman et al., 2019), where a robot starts in the north-west corner and has to navigate towards a target position. The robot can move in each of the four cardinal directions with a success probability of 0.55, and a probability of 0.15 to move in each other direction. Throughout the grid, the robot has to avoid traps. The model has 100 states and 1450 transitions. The objective is to maximize the probability of reaching the north-east corner without getting trapped.

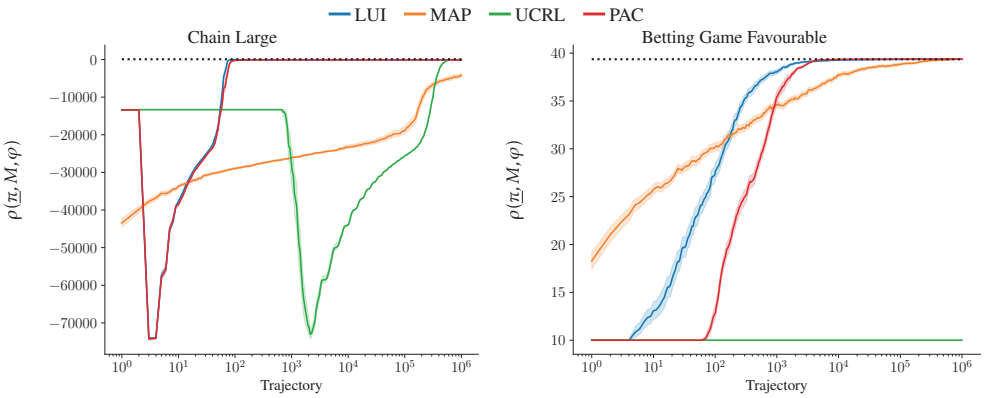


Figure 5.4: Comparison of the performance of robust policies on the Chain and favorable Betting Game environments against the number of trajectories processed (on log-scale). The dashed line indicates the optimal performance.

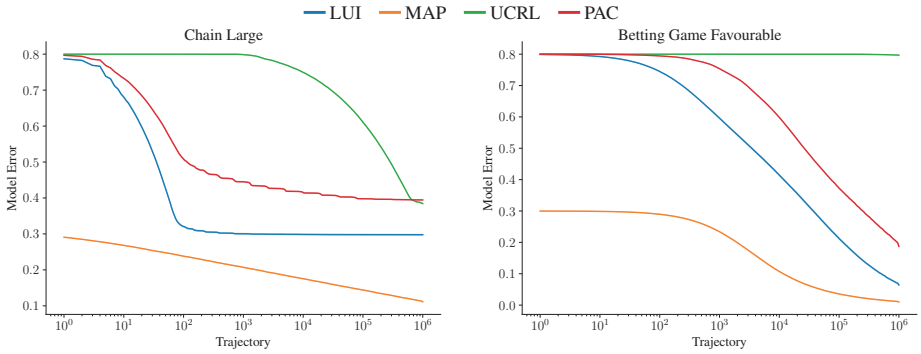


Figure 5.5: Comparison of the model error of the IMDPs learned on the Chain and favorable Betting Game environments.

5.6.1 Results and Discussion

RQ1: Performance robust policies. Figures 5.4 and 5.7 show the performance of the robust policies computed via each learning method against the number of trajectories processed. We first note that in all environments, our LUI method is the first to find an optimal policy. Depending on the environment, the performance of LUI and PAC may be roughly equivalent, as seen in the Chain environment. When also taking the estimation error into account, Figures 5.6 and 5.8, we see LUI outperforming other methods on that metric. UCRL2 is the slowest to converge to an optimal policy. This is due to UCRL2 being a reinforcement learning algorithm, and thus it is slower in reducing the intervals in favor of broader exploration.

RQ2: Recovering from bad estimates. On the Chain environment, we see LUI and PAC (around trajectory 5) and UCRL2 (around trajectory 10^3) choose the wrong

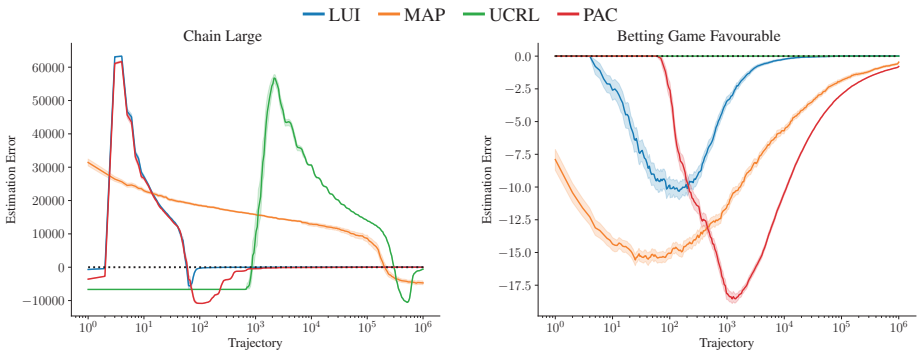


Figure 5.6: Comparison of the estimation error of robust policies on the Chain and favorable Betting Game environments.

action(s), with a decrease in performance as a result. This is most likely due to getting a bad estimate on some of the transitions later on in the chain, leading to many resets to the initial state and thus decreasing the performance significantly. While all three methods manage to recover and then find the optimal policy, UCRL2 takes significantly longer: only after trajectory 10^5, where LUI and PAC only need about 100 trajectories. MAP-estimation typically sits between LUI and PAC in terms of performance. It is less sensitive to mistakes like the one discussed above, but is less reliable in providing a conservative bound on its performance, as will be discussed below. Furthermore, we see that in the unfavorable Betting Game, only MAP-estimation gives sub-optimal performance due to bad estimates. It can recover but needs almost 10^4 trajectories to do so. Due to the low win probability in this Betting Game, a robust policy on the RMDPs is, by default, optimal for the true MDP, and we see that LUI, PAC, and UCRL2 do not change to a sub-optimal policy.

RQ3: Robust policies are conservative. To answer our third research question, consider Figures 5.6 and 5.9. We note the undesirable behavior of having an Estimation Error above zero, which means the performance of the policy on the learned model was higher than its performance on the true MDP. MAP-estimation is particularly susceptible to this behavior, while LUI and PAC yield policies that are conservative in general.

5.6.2 Robustness Against Changing Environments

To answer our fourth research question (RQ4), we investigate the behavior of the learning methods when, after a fixed number of trajectories, the probabilities of the true MDP change, as introduced in Section 5.5.

Setup

Figures 5.10 and 5.11 shows the performance of the robust policy on the true MDP for the environment's objective, *i.e.*, $\rho(\underline{\pi}, M, \varphi)$, after some trajectories for each

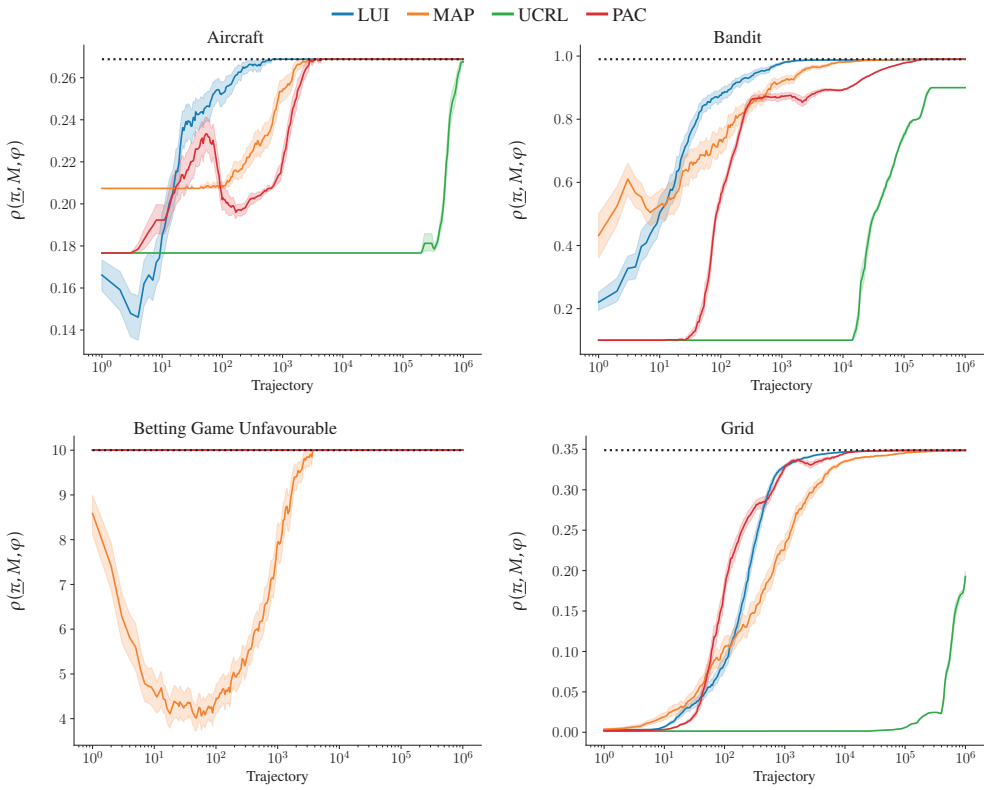


Figure 5.7: Comparison of the performance of robust policies on the Aircraft, Bandit, unfavorable Betting Game, and Grid benchmarks.

learning method on the Chain and Betting Game environments, respectively. After $\dagger \in \{10^2, \ldots, 10^5\}$ trajectories, we change the environment by changing the transition probabilities for three different bounds n_{Max} on the strength intervals, and randomization parameter $\xi \in \{0.8, 1.0\}$ (distribute 0.2 uniformly over the non-optimal actions, or no randomization at all). We similarly bound the MAP-estimation priors for MAP and UCRL, so they also employ a sliding window, with the same exploration randomization parameter ξ. PAC is omitted from this experiment as PAC guarantees lose all meaning when changing the underlying distribution.

For the Chain environment, we swap the transition probabilities for the actions, such that after the change in environment, the new optimal policy has to use the opposite action from the previously optimal policy. This type of change already highlights the potential need for randomization, as previously sub-optimal actions need to be explored again after the change. For the Betting Game environment, we change the game from being favorable, *i.e.*, of having a win probability 0.8, to being unfavorable with a win probability of only 0.2.

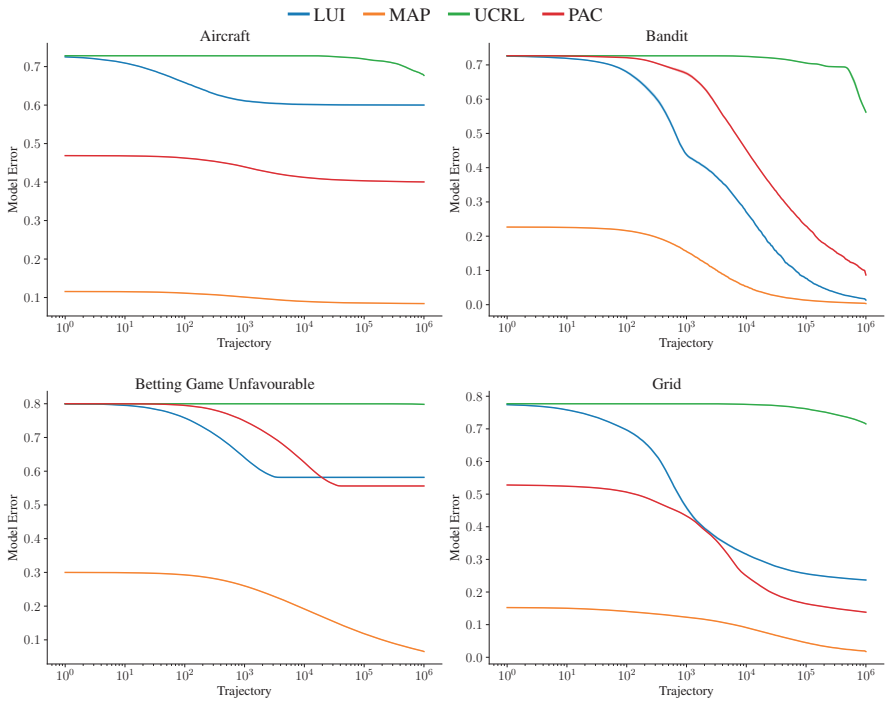


Figure 5.8: Comparison of the model error of the IMDPs learned on the Aircraft, Bandit, unfavorable Betting Game, and Grid benchmarks.

Ablation study. In both Figures 5.10 and 5.11, the top three rows use randomized exploration, while the bottom three rows use purely optimistic exploration. Hence, the bottom three rows can be seen as an ablation for using randomization in the exploration. Rows three and six in both figures use an unbounded sliding window, *i.e.*, $n_{\text{Max}} = [\infty, \infty]$, and are thus an ablation on using a sliding window approach.

Results and Discussion

We observe that LUI is the only method capable of converging to optimal policies both before and after the change in the Chain environment. Furthermore, the lower the bounds n_{Max} on the prior strength, the faster it adapts to the change. The later the change † happens, the more future data is needed to adapt, as seen when comparing columns. When the change happens early enough, *e.g.*, † = 100 or † = 1000, LUI is still very flexible and adapts quickly, even without a sliding window. When a large amount of trajectories has been processed already before the change happens, as with † = 100000, for instance, the need for a sliding window becomes clearly visible, see the last column, rows one and two. MAP and UCRL, both also employing the same sliding window, do not converge to the optimal policy as LUI does. This further highlights the flexibility of LUI and the use of prior-data conflicts in practice.

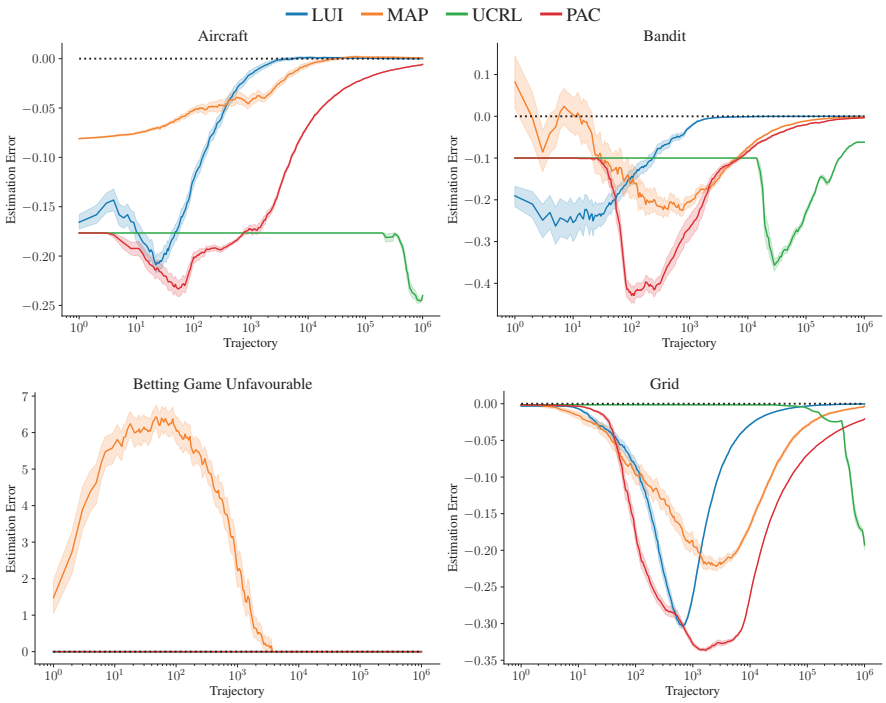


Figure 5.9: Comparison of the estimation error of robust policies on the Aircraft, Bandit, unfavorable Betting Game, and Grid benchmarks.

Comparing the top three rows with the bottom three rows of the Chain environment (Figure 5.10), we clearly note the need for randomization in the exploration on this environment, as we also predicted earlier. The Betting Game (Figure 5.11) is less sensitive to this exploration problem. We conclude that LUI is a robust approach for reinforcement learning in changing environments, while MAP and UCRL are not.

5.7 Conclusion

We presented a new robust reinforcement learning approach that employs linearly updating intervals (LUI) to learn IMDPs. Robust policies computed on these IMDPs are shown to be conservative and reliable in predicting their performance when applied on the MDP that is being learned. LUI is also effective at continuously learning, showcasing robustness against changes in the underlying environment.

5.7.1 Limitations and Discussion

The techniques presented in this chapter rely on some assumptions that impose certain limitations. We discuss each of these limitations individually below.

Formal guarantees. LUI comes with few guarantees. Notably, the only guarantees we have are convergence and closure of intervals and distributions, as dis-

cussed in Section 5.4.1. By setting the prior strength intervals high enough, it is likely possible to enforce PAC guarantees via, *e.g.*, Hoeffding's inequality. More interesting, however, would be to investigate what guarantees can be given regarding robustness against changing environments. A possible avenue could be to assume a model that describes the change in the environment. Such a model could give information on the direction and rate of change that could then be exploited to construct PAC-style guarantees or other theoretical properties such as convergence rate.

Graph assumption. We currently assume the underlying graph of the environment is known and that all intervals have a lower bound of at least ε, as specified in Assumption 5.2. This assumption is primarily made to enable efficient robust value iteration in IMDPs for objectives with target sets, such as reachability. We could weaken this assumption to only assume a known (lower bound on a) minimal transition probability p_{min} and use the lower and upper Bellman equations from (Ashok et al., 2019) to solve the resulting IMDPs.

Other uncertainty sets. In its current form, LUI is inherently limited to IMDPs. Developing similar adaptive techniques for other types of RMDPs, such as L_1-MDPs or s-rectangular RMDPs, would increase the applicability of our techniques and further strengthen RL in changing environments.

Future work. Besides addressing the limitations above, LUI could be extended to other settings. Interesting options include partial observability, *i.e.*, when the underlying environment is given as a POMDP, or to the offline RL setting where only a fixed dataset is given. Recent work already applies LUI as a learning method to the setting of uncertain parametric MDPs (Schnitzer et al., 2024).

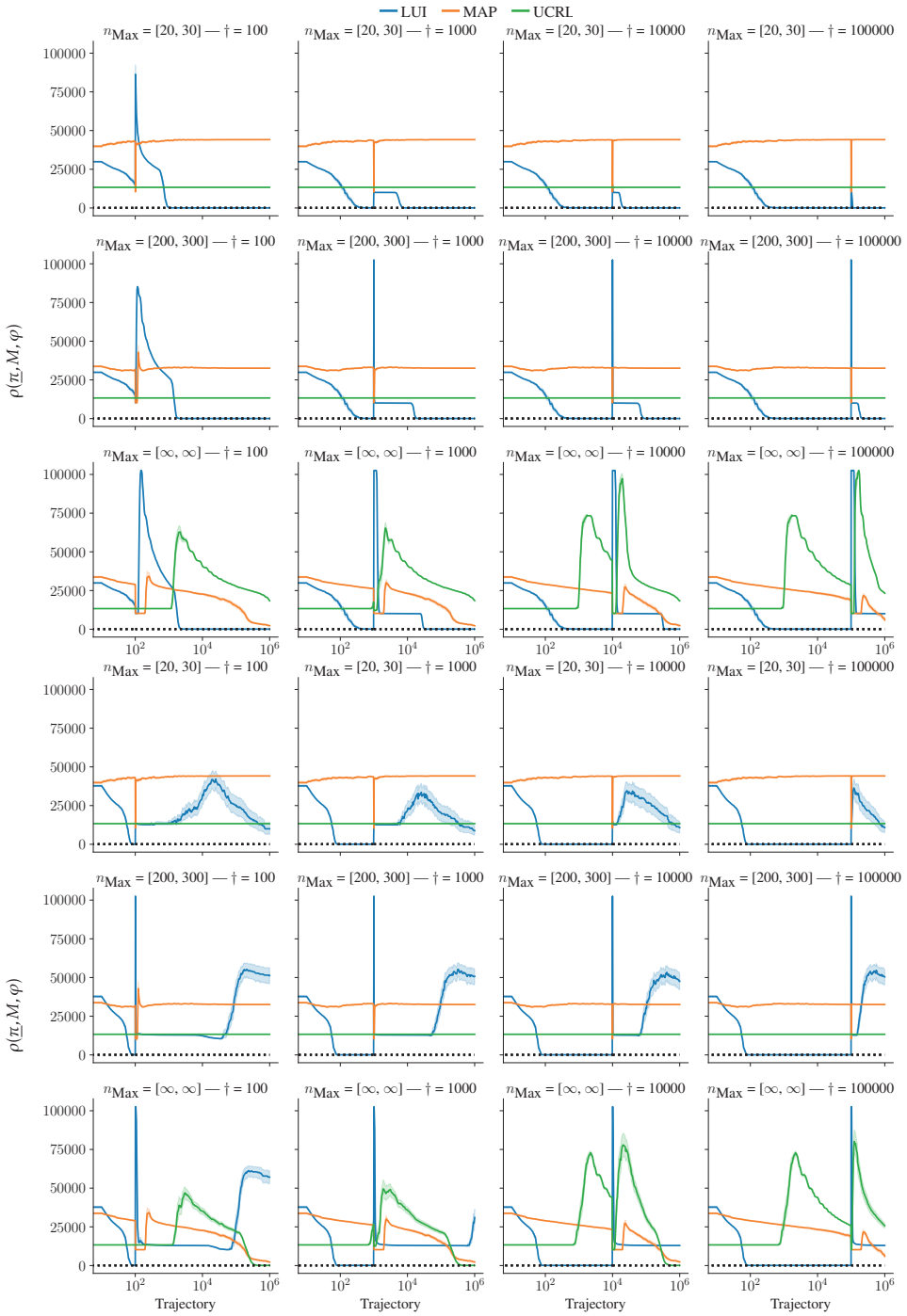


Figure 5.10: Environment change at point † on the Chain environment using randomization parameter $\xi = 0.8$ (top three rows) and $\xi = 1.0$ (bottom three rows) and different n_{Max}.

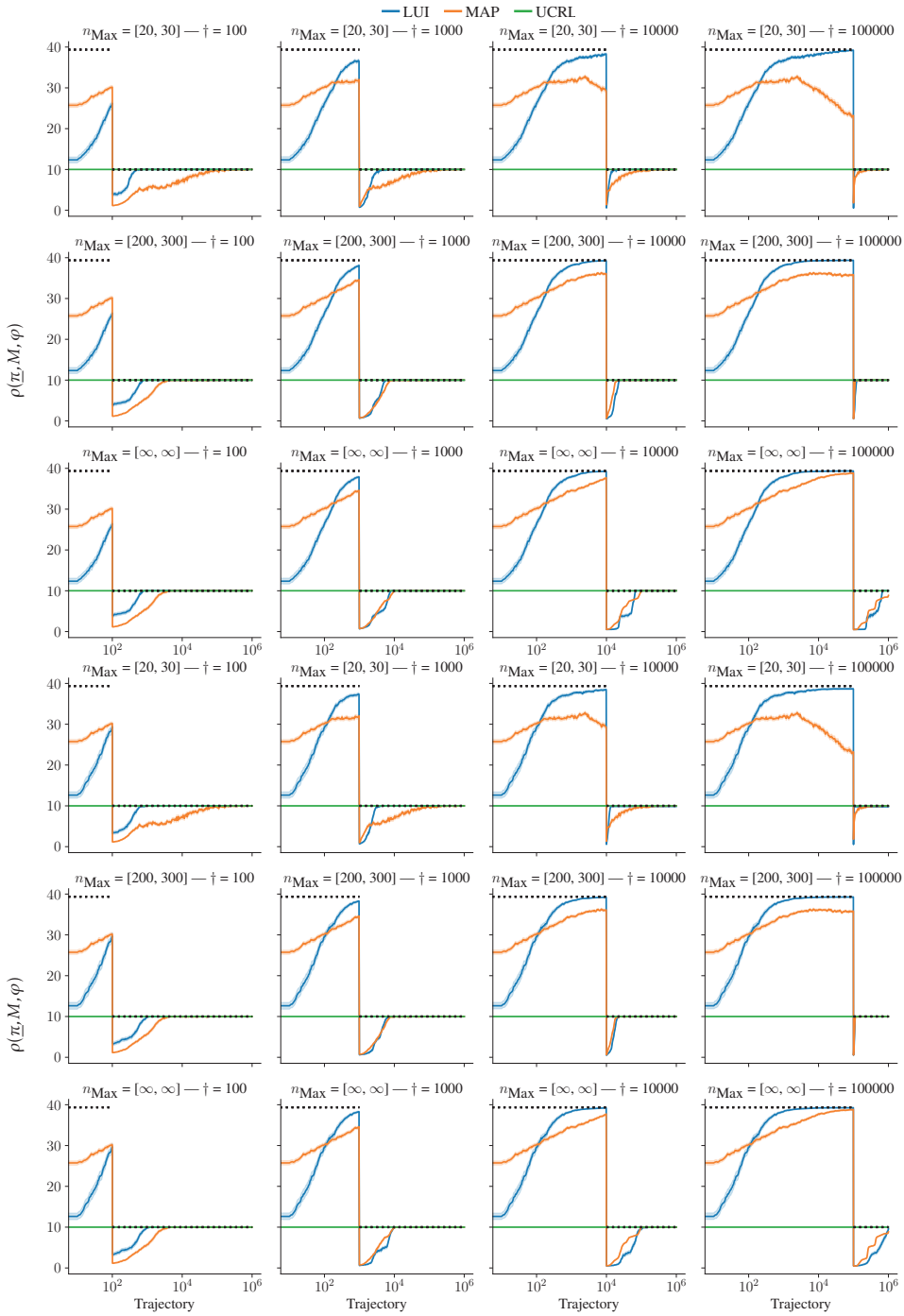


Figure 5.11: Environment change at point † in the Betting Game using randomization parameter $\xi = 0.8$ (top three rows) and $\xi = 1.0$ (bottom three rows) and different n_{Max}.

6

Extending the Scope of Reliable Offline RL

This chapter studies the offline reinforcement learning problem of *safe policy improvement* (SPI). The SPI problem is to improve the performance of a *behavior policy* that was used to collect data about an environment and, while doing so, to provide a formal guarantee on the performance of the resulting policy. Specifically, the improved policy is required to outperform the behavior policy in terms of its performance up to some small error tolerance, with high probability. The safe policy improvement with baseline bootstrapping (SPIBB) algorithm achieves this requirement by exploiting both the available data and the behavior policy. SPIBB constrains the policy space according to the available data, such that the newly computed policy only differs from the behavior policy when sufficient data is available, ensuring the required improvement guarantee. The contributions to the SPI problem presented in this chapter are two-fold. First, we extend SPIBB to work under partially observable environments modeled as POMDPs that are k-Markovian. Second, we introduce a novel approach that significantly reduces the amount of data required to establish the same improvement guarantee within the SPIBB algorithm, thus making the most out of the available data.

6.1 Introduction

Reinforcement learning (RL) solves decision-making problems, in particular when the environment dynamics are unknown (Sutton and Barto, 1998). In an *online* RL setting, an agent aims to learn a decision-making policy that maximizes the expected accumulated reward by interacting with the environment and observing feedback, typically in the form of information about the environment state and reward. While online RL has shown great performance in solving hard problems (Mnih et al., 2015; Silver et al., 2018), the assumption that the agent can always directly interact with the environment is not always realistic. In real-world applications such as

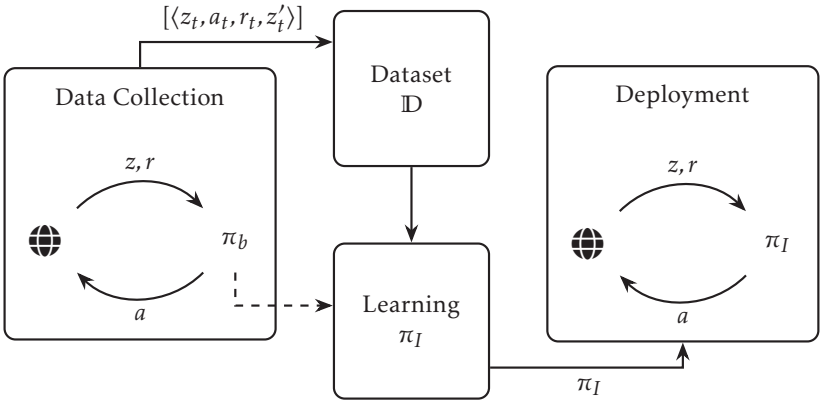


Figure 6.1: Illustration of the offline reinforcement learning problem in partially observable environments (adapted from Levine et al., 2020). Observations z and rewards r are obtained by simulating the behavior policy π_b and stored in a dataset $\mathbb{D}$. After data collection has finished, a new policy π_I is learned from the dataset and deployed. The dashed arrow indicates the setting where the behavior policy is available during learning.

robotics or healthcare, direct interaction can be impractical or dangerous (Levine et al., 2020). Furthermore, alternatives such as simulators or digital twins may not be available or insufficiently capture the nuances of the real-world application for reliable learning (Ramakrishnan et al., 2020; Zhao et al., 2020).

Offline RL (or batch RL) (Lange et al., 2012) mitigates these concerns by restricting the agent to have only access to a fixed dataset of past interactions. As a common assumption, the dataset has been generated by a so-called *behavior policy*. An offline RL algorithm aims to produce a new policy without further interactions with the environment (Levine et al., 2020). Methods that can reliably improve the performance of a policy are key in (offline) RL.

Safe policy improvement (SPI) algorithms address this challenge by providing (probabilistic) correctness guarantees on the reliable improvement of policies (Petrik et al., 2016; Thomas et al., 2015). The *safe policy improvement with baseline bootstrapping* (SPIBB; Laroche et al., 2019) is one of the most used SPI algorithms. SPI(BB) has been studied in various settings, among which multi-objective (Satija et al., 2021), non-stationary (Chandak et al., 2020), factored environments (Simão and Spaan, 2019a,b), and multi-agent (Bianchi et al., 2024) settings. Additionally, SPIBB has been extended to work in settings where the behavior policy is not given (Simão et al., 2020) and in environments with large state spaces via Monte Carlo tree search (Castellini et al., 2023). For a recent overview of SPI methods, we refer to Scholl et al. (2022).

6.1.1 Contributions

In this chapter, we study the SPI problem and extend the applicability of SPIBB. The contributions presented in this chapter are twofold.

Contribution

i. We extend SPIBB to compute finite-memory policies from datasets collected in partially observable environments.

ii. We present new techniques that provide stronger improvement guarantees given the same amount of data in SPI algorithms such as SPIBB.

We now discuss each of the two contributions in more detail.

Contribution I: SPI in Partially Observable MDPs

We contribute a novel extension of SPIBB to partially observable environments modeled by POMDPs. First, to account for the inherent memory requirement in partially observable domains, we consider a behavior policy represented by an FSC. To create a tractable method, we assume that there exists a finite-memory policy for the POMDP that is optimal, also known as the finite-history-window approach (Kaelbling et al., 1996, Section 7.3). This assumption allows us to cast the POMDP as an equivalent, fully observable, *history MDP* that is finite instead of the standard infinite-history MDP (Silver and Veness, 2010). We can then reliably estimate the transition and reward models of this finite-history MDP from the available data. We apply SPIBB to the estimated finite-history MDP to compute an improved policy that outperforms the behavior policy up to an *admissible performance loss* with high probability. Compared to the approach for mere MDPs (Laroche et al., 2019), we derive an improved bound on this admissible performance loss by exploiting the specific structure of the history MDP. Figure 6.1 illustrates our approach.

Contribution II: Tighter Improvement Bounds for SPI

The guarantees of SPI algorithms depend on the size of the dataset and adhere to a conservative bound on the minimal amount of samples required. Since this bound often turns out to be too large for practical applications of SPI, it is instead often turned into a hyperparameter (see, *e.g.*, Laroche et al., 2019)). The offline nature of SPI prevents further data collection, which steers the key requirements of SPI in practical settings: (1) exploit the dataset as efficiently as possible and (2) compute better policies from smaller datasets.

Our contribution provides the theoretical foundations to improve the understanding of SPI algorithms in general. Specifically, in a general SPI setting, we can guarantee a higher performance for significantly less data. Our main technical contribution is a transformation of the underlying MDP model into a *two-successor MDP* (2sMDP) along with adjustments to the dataset, which allows us to prove these tighter bounds. A 2sMDP is an MDP where each state-action pair has at most two successors. These transformations preserve (the optimal) performance of policies and are reversible. In the context of SPI these transformations are implicit, *i.e.*, do not have to be computed explicitly. Hence, we can apply standard SPI algorithms such as SPIBB, and use our novel improvement guarantees without any algorithmic changes necessary, as also illustrated in Figure 6.2.

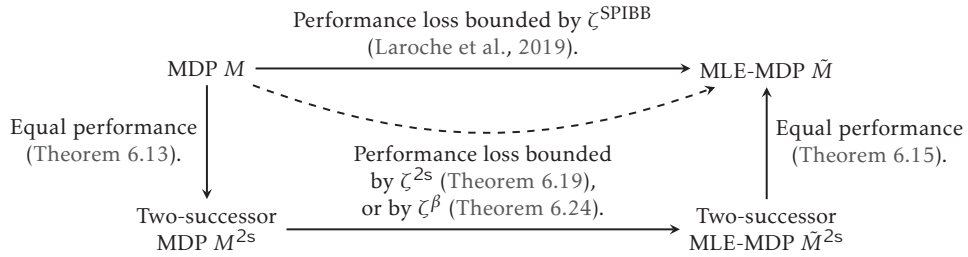


Figure 6.2: Overview of the tighter improvement bounds for SPI. The solid arrows indicate how the full derivation of the improvement guarantees is done, while the dashed line indicates that the transformations are only used in the proofs and that, in practice, we can immediately use ζ^{2s} or ζ^{β} as bounds.

Following the theoretical foundations for the MDP and dataset transformations (Section 6.4), we present two different methods to compute the new performance guarantees (Section 6.4.3). The first uses Weissman's bound (Weissman et al., 2003), as also used in, *e.g.*, standard SPIBB, while the second uses the inverse incomplete beta function (Temme, 1992). Our experimental results show a significant reduction in the amount of data required for equal performance guarantees (Section 6.5). Concretely, where the number of samples required at each state-action pair of standard SPIBB grows linearly in the number of states; our approach only grows logarithmic in the number of states for both methods. We also demonstrate the impact on three well-known benchmarks in practice by comparing them with standard SPIBB across multiple hyperparameters.

6

Structure of This Chapter

The remainder of this chapter is structured as follows. First, we recall the necessary preliminaries from Chapter 2 and provide the technical background on SPI and SPIBB in Section 6.2. In Section 6.3 we present the theoretical results for our first contribution, SPI in partially observable environments. In Section 6.4, we present the theoretical results for our second contribution. Section 6.5 contains the experimental evaluation of each of our two contributions individually in Sections 6.5.1 and 6.5.2, respectively. Finally, in Section 6.6, we conclude the chapter with a discussion of limitations and future work.

6.2 Background: Safe Policy Improvement

In this section, we briefly recap the basic tools from Chapter 2 we need and then review safe policy improvement (SPI) for MDPs.

6.2.1 Preliminaries

Discounted reward MDPs. The MDPs we consider in this chapter are tuples $\langle S, s_\iota, A, P, R\rangle$ as generally defined in Definition 2.1, where S is a finite set of states,

$s_\iota \in S$ is the initial state, A is a finite set of actions, $P\colon S \times A \rightharpoonup \mathcal{D}(S)$ is the transition function and R is the reward function. In the context of this chapter, we assume the reward function is allowed to map to negative values and to be bounded by some known value r_{max}, that is, $R\colon S \times A \rightharpoonup [-r_{max}, r_{max}]$. The objective is discounted reward maximization with a discount factor γ, *i.e.*, $\mathbf{R}_{Max}(\gamma)$ as defined in Definition 2.5. Since this chapter considers multiple different discount factors, we often explicitly include these into the MDP tuple and write, *e.g.*, $M = \langle S, s_\iota, A, P, R, \gamma \rangle$.

Discounted reward POMDPs. A POMDP is a tuple $\langle S, b_\iota, A, P, R, Z, O \rangle$ (see Definition 2.7), where S, A, P, R are as for MDPs, $b_\iota \in \mathcal{D}(S)$ is the initial belief, Z is a finite set of observations, and $O\colon S \times A \rightharpoonup \mathcal{D}(Z)$ is the observation function.

Paths and histories. Recall that for an MDP M, $Post_M(s,a)$ is the set of successor states reachable with positive probability from the state-action pair (s,a) in M. A *path* in M is a finite sequence $\langle s_1, a_1, \ldots, a_{n-1}, s_n \rangle \in (S \times A)^* \times S$ where $s_i \in Post_M(s_{i-1}, a_{i-1})$ for all $i \in [2{:}n]$. The probability of following a path $\langle s_1, a_1, \ldots, a_{n-1}, s_n \rangle$ in the MDP M given a deterministic sequence of actions is written as $\mathbb{P}_M(\langle s_1, a_1, \ldots, a_{n-1}, s_n \rangle)$ and can be computed by repeatedly applying the transition probability function: $\mathbb{P}_M(\langle s_1, a_1, \ldots, a_{n-1}, s_n \rangle) = \prod_{i=1}^{n-1} P(s_{i+1} \,|\, s_i, a_i)$. A *history* for a POMDP $\mathcal{M} = \langle S, b_\iota, A, P, R, Z, O \rangle$ is a sequence of observations and actions: $h \in (Z \times A)^* \times Z$. We denote the set of all histories by Hists, and Hists_k denotes all histories of maximal length k, where the length $|h|$ is the number of observations in the history h.

Policies. This chapter considers memoryless policies, *i.e.*, policies of type $\pi\colon S \to \mathcal{D}(A)$ for MDPs. For POMDPs, we consider both memoryless and finite-memory policies represented by finite-state controllers (FSCs; Definition 2.8) for POMDPs. An FSC is a tuple $\langle \mathcal{N}, n_\iota, \alpha, \eta \rangle$, where $\mathcal{N}$ is a finite set of memory nodes, n_ι is the initial node, $\alpha\colon \mathcal{N} \times Z \to \mathcal{D}(A)$ is the action mapping, and $\eta\colon \mathcal{N} \times Z \times A \to \mathcal{D}(\mathcal{N})$ is the memory update function. The *performance*, also called the *expected return*, $\rho(\pi, M, \gamma)$ of a policy π in an MDP M with discount factor γ is defined as the value in the initial state $s_\iota \in S$, *i.e.*, $\rho(\pi, M, \gamma) = V_{M,\pi,\gamma}(s_\iota)$. Since we include γ in the (PO)MDP tuples in this chapter, we omit it from the performance and simply write $\rho(\pi, M)$. Furthermore, we write $V_{\max}$ for a known upper bound on the absolute value of the performance: $V_{\max} \leq {}^{r_{max}}/_{1-\gamma}$.

Discounted reward policy iteration. The policy iteration algorithm (Section 2.2.3) can be used to compute a memoryless deterministic policy that maximizes the expected discounted reward in an MDP. It consists of two steps: (1) policy evaluation and (2) policy improvement. For the policy evaluation step, compute the state-action values under π as

$$Q_\pi(s,a) = R(s,a) + \gamma \sum_{s'} P(s' \,|\, s, a) V^*_\pi(s'), \quad \forall s \in S, a \in A(s).$$

The policy improvement step updates the policy π to a new policy π' by

$$\pi'(s) = \operatorname*{argmax}_{a \in A} Q_\pi(s,a), \quad \forall s \in S, \tag{6.1}$$

which is guaranteed to have a value at least as good as the previous policy, *i.e.*, $V^*_{\pi'} \geq V^*_\pi$. This process starts with any initial policy and terminates as soon as the policy does not change anymore: $\pi' = \pi$, after which π is guaranteed to be optimal.

6.2.2 Safe Policy Improvement

A *dataset* is a sequence $\mathbb{D}$ of trajectories collected under a *behavior policy* π_b in an MDP M^*. For MDPs, the datasets we consider for SPI are of the form $\mathbb{D} = \langle s_t, a_t, r_t \rangle_{t \in [1:m]}$, and we write $\#_{\mathbb{D}}(x)$ for the number of times x occurs in $\mathbb{D}$. The goal of SPI is to compute a new policy π_I based on $\mathbb{D}$ that outperforms π_b with an allowed performance loss $\zeta \in \mathbb{R}$ with high probability $1 - \delta$.

SPI operates on a set of *admissible MDPs* Ξ, that is, MDPs $M = \langle S, s_I, A, P, R \rangle$ which are 'close' to an MDP $\tilde{M} = \langle S, s_I, A, \tilde{P}, \tilde{R} \rangle$ estimated from a dataset $\mathbb{D}$ by maximum likelihood estimation (MLE).

Definition 6.1 (MLE-MDP)**.** The MLE-MDP of an unknown true MDP $M^* = \langle S, s_I, A, P, R \rangle$ and a dataset $\mathbb{D}$ is a tuple $\tilde{M} = \langle S, s_I, A, \tilde{P}, \tilde{R} \rangle$ with transition and reward functions $\tilde{P}$ and $\tilde{R}$ derived from $\mathbb{D}$ via maximum likelihood estimation:

$$\tilde{P}(s' \mid s,a) = \frac{\#_{\mathbb{D}}(s,a,s')}{\#_{\mathbb{D}}(s,a)}, \text{ and } \tilde{R}(s,a) = \frac{\sum_{(s_t,a_t,r_t) \in \mathbb{D}} \mathbb{1}[s_t = s \wedge a_t = a] \cdot r_t}{\#_{\mathbb{D}}(s,a)}.$$

For an MLE-MDP $\tilde{M}$ and error function $e\colon S \times A \to \mathbb{R}$, we define the set of admissible MDPs $\Xi_e^{\tilde{M}}$ with a transition function P that has L_1-distance to the estimated transition function $\tilde{P}$ bounded by the error function e:

$$\Xi_e^{\tilde{M}} = \left\{ M = \langle S, s_I, A, P, R, \gamma \rangle \mid \forall (s,a)\colon \|P(\cdot \mid s,a) - \tilde{P}(\cdot \mid s,a)\|_1 \leq e(s,a) \right\}.$$

Remark 6.2. The set of admissible MDPs $\Xi_e^{\tilde{M}}$ forms an L_1-MDP, see Definition 3.8.

The general idea behind SPI methods is to define the error function e such that $\Xi_e^{\tilde{M}}$ includes the true MDP M^* with high probability $1 - \delta$ (Petrik et al., 2016, Proposition 9). Then one can compute a new policy which is an improvement *for all* MDPs within $\Xi_e^{\tilde{M}}$. An alternative is to simply solve the MLE-MDP, but this could lead to arbitrarily poor policies when the amount of data is insufficient. If, however, the amount of data is sufficient for all state-action pairs, then we can guarantee with high probability that the improved policy computed on the MLE-MDP has a higher performance. The amount of data required to achieve a ζ^{SPI}-approximately safe policy improvement with probability $1 - \delta$ (recall Equation 6.3) for all state-action pairs has been established by (Laroche et al., 2019) as

$$\#_{\mathbb{D}}(s,a) \geq N_\wedge^{\text{SPI}} = \frac{8 V_{max}^2}{(\zeta^{\text{SPI}})^2 (1-\gamma)^2} \log \frac{2|S||A|2^{|S|}}{\delta}. \tag{6.2}$$

Then, with probability $1-\delta$, an optimal policy π_I for $\tilde{M}$ is ζ-approximately safe with respect to the true MDP M^* for some *admissible performance loss* $\zeta \in \mathbb{R}$. That is,

$$\rho(\pi_I, M^*) \geq \rho(\pi^*, M^*) - \zeta \geq \rho(\pi_b, M^*) - \zeta, \tag{6.3}$$

where π^* is an optimal policy in the true MDP M^*. Intuitively, the constraint defined in Equation 6.2 ensures that the estimated transition function is close enough to the true MDP to guarantee that the policy computed in the MLE-MDP approximately outperforms the behavior policy in the underlying true MDP.

6.2.3 SPI with Baseline Bootstrapping on MDPs

The bound in Equation 6.2 needs to hold for every state-action pair, which impairs the practical use of optimizing the MLE-MDP. The *SPI with baseline bootstrapping* (SPIBB; Laroche et al., 2019) algorithm overcomes this limitation, allowing the constraint in (6.2) to be violated on some state-action pairs. These state-action pairs are collected in $\mathbb{U}$, the set of *unknown* state-action pairs with counts smaller than a given hyperparameter $N_\wedge^{\text{SPIBB}}$:

$$\mathbb{U} = \left\{(s,a) \in S \times A \mid \#_{\mathbb{D}}(s,a) \leq N_\wedge^{\text{SPIBB}}\right\}.$$

SPIBB then determines an improved policy π_I similar to (standard) SPI, except that if $(s,a) \in \mathbb{U}$, π_I is required to follow the behavior policy π_b:

$$\forall (s,a) \in \mathbb{U}\colon\ \pi_I(a \mid s) = \pi_b(a \mid s).$$

Then, π_I is an improved policy as in Equation 6.3, where $N_\wedge^{\text{SPIBB}}$ is treated as a hyperparameter and ζ is given by

$$\zeta^{\text{SPIBB}} = \frac{4V_{max}}{1-\gamma}\sqrt{\frac{2}{N_\wedge^{\text{SPIBB}}}\log\frac{2|S||A|2^{|S|}}{\delta}} - \rho(\pi_I, \tilde{M}) + \rho(\pi_b, \tilde{M}).$$

We can rearrange this equation to compute the number of necessary samples for a ζ^{SPIBB}-approximate improvement. As $\rho(\pi_I, \tilde{M})$ is only known at runtime, we have to employ an under-approximation $\rho(\pi_b, \tilde{M})$ to a priori compute

$$N_\wedge^{\text{SPIBB}} = \frac{32V_{max}^2}{(\zeta^{\text{SPIBB}})^2(1-\gamma)^2}\log\frac{2|S||A|2^{|S|}}{\delta}. \tag{6.4}$$

Thus, the sample size constraint $N_\wedge^{\text{SPIBB}}$ grows approximately linearly in terms of the size of the MDP. The term $2^{|S|}$ is an over-approximation of the maximum branching factor of the MDP since, worst-case, the MDP can be fully connected.

Algorithmically, SPIBB performs a *constrained* version of policy iteration on the MLE-MDP. SPIBB computes stochastic policies by modifying the policy improvement step, Equation 6.1. Recall that $\mathcal{U}$ is the set of uncertain state-action pairs, and let $\mathcal{A}(s) = \{a \in A \mid (s,a) \notin \mathcal{U}\}$ be the set of *free* state-action pairs. Specifically, the new

policy follows π_b with the same probability mass in state-action pairs in $\mathcal{U}$, while the remaining probability mass gets assigned to the free action that maximizes Q_π:

$$\pi'(a|s) = \begin{cases} \pi_b(a|s) & \forall (s,a) \in \mathcal{U}, \\ \sum_{a' \in \mathcal{A}(s)} \pi_b(a'|s) & a = \operatorname*{argmax}_{a' \in \mathcal{A}(s)} Q_\pi(s,a'), \\ 0 & \text{otherwise.} \end{cases}$$

As with standard policy iteration, this process terminates when $\pi' = \pi$, and the resulting policy is the improved policy π_I that solves the SPI problem.

6.3 Safe Policy Improvement in POMDPs

Now, we detail our approach to applying SPIBB to POMDPs.

Formal problem statement. Given a POMDP $M = \langle S, b_\iota, A, P, R, Z, O \rangle$ of which the transition and observation functions are unknown, some initial belief $b_\iota \in \mathcal{D}(S)$, and a finite-memory behavior policy represented as a κ-FSC $\pi_b = \langle \mathcal{N}, n_\iota, \alpha, \eta \rangle$, the goal is to apply SPIBB to construct a new κ-FSC $\pi_I = (\mathcal{N}, n_\iota, \alpha', \eta)$ with the same nodes and memory structure η, *i.e.*, $\pi_b, \pi_I \in \Pi_k^\eta$, such that with high probability $1-\delta$, π_I is a ζ-approximately safe improvement over π_b with respect to M. That is, with a probability of at least $1-\delta$ we have

$$\rho(\pi_I, M) \geq \rho(\pi_b, M) - \zeta.$$

6.3.1 From POMDP to Finite-History MDP

As already briefly mentioned but not worked out in detail yet, every POMDP is equivalent to a fully observable *history MDPs* (Silver and Veness, 2010).

Definition 6.3 (History MDP)**.** Let $M = \langle S, b_\iota, A, P, R, Z, O \rangle$ be a POMDP. The *history MDP* of M is a tuple $\langle \mathsf{Hists}, z_\iota, A, P_{\mathsf{Hists}}, R_{\mathsf{Hists}} \rangle$, where the states are given by all histories of M, $z_\iota \in \mathcal{D}(Z)$ is the initial distribution over the first observation, the actions remain A, and $P_{\mathsf{Hists}} \colon \mathsf{Hists} \times A \rightharpoonup \mathcal{D}(\mathsf{Hists})$ and $R_{\mathsf{Hists}} \colon \mathsf{Hists} \times A \to \mathbb{R}$ are given by

$$P_{\mathsf{Hists}}(h : a : z | h, a) = \sum_{s \in S} b(s|h) \cdot \sum_{s' \in S} P(s'|s,a) \cdot O(z|s',a),$$
$$R_{\mathsf{Hists}}(h, a) = \sum_{s \in S} b(s \mid h) \cdot R(s,a).$$

While a POMDP can be mapped to a fully observable history MDP (Definition 6.3), this MDP has infinitely many states, making a direct application of SPI(BB) methods infeasible. To mitigate this issue, we assume the structure of the history MDP (and inherently on the POMDP) that implies that the history MDP is equivalent to a smaller, finite MDP. We formalize this assumption via stochastic bisimulation (Givan et al., 2003). Intuitively, this bisimulation is an equivalence relation that relates (history) states that *behave* similarly according to reward signals.

Definition 6.4 (Bisimilarity of history states)**.** A stochastic bisimulation relation $E \subseteq \mathcal{H} \times \mathcal{H}$ on history states $h_1, h_2 \in \mathcal{H}$ is an equivalence relation satisfying

$$E(h_1, h_2) \iff \forall a \in A: \quad R_{\mathcal{H}}(h_1, a) = R_{\mathcal{H}}(h_2, a) \text{ and}$$
$$\forall h_1', h_2' \in \mathcal{H} \text{ with } E(h_1', h_2') \text{ we have}$$
$$P_{\mathcal{H}}(h_1' \mid h_1, a) = P_{\mathcal{H}}(h_2' \mid h_2, a).$$

The largest stochastic bisimulation relation is called (stochastic) bisimulation, denoted by $\sim$. We write $[h]_\sim$ for the equivalence class of history h under $\sim$, and $\mathcal{H}/_\sim$ for the set of equivalence classes.

Assumption 6.5 (Sufficiency of finite histories)**.** *Every history state h of size $|h| > k$ in the history MDP is bisimilar to a history state h' of size $|h'| \leq k$. That is, $h \sim h'$.*

As a consequence, the history MDP satisfying Assumption 6.5 has a finite bisimulation quotient MDP (Givan et al., 2003), and we call it a *finite-history MDP* instead. This finite-history MDP consists of states with the equivalence classes of histories under $\sim$. Note that belief remains a sufficient statistic in this case, *i.e.*, $b(s \mid [h]_\sim) = b(s \mid h)$.

Definition 6.6 (Finite-history MDP)**.** A POMDP satisfying Assumption 1 is a fully observable finite-state MDP $M = \langle \mathsf{Hists}/_\sim, A, P_H, R_H \rangle$ where the states are given by the set of equivalence classes, the actions and discount factor from the POMDP, and transition and reward functions are defined as

$$P_H([haz]_\sim \mid [h]_\sim, a) = \sum_{s \in S} b(s \mid [h]_\sim) \sum_{s' \in S} P(s' \mid s, a) O(z \mid s', a),$$
$$R_H([h]_\sim, a) = \sum_{s \in S} b(s \mid [h]_\sim) R(s, a).$$

Under bisimulation equivalence, the finite-history MDP and the POMDP are related in the following fundamental way.

Theorem 6.7 (Optimal finite-memory policies under bisimilarity)**.** *An optimal policy π^* in the finite-history MDP is an optimal finite-memory policy for the POMDP.*

Theorem 6.7 is a direct result of bisimilarity (Givan et al., 2003). We may number the equivalence classes in the finite-history MDP so that they correspond to memory nodes of an FSC. As a result, the finite-history MDP can be defined on a state space consisting of memory nodes and observations rather than histories.

Definition 6.8 (Finite-history MDP via FSC)**.** A POMDP satisfying Assumption 6.5 is a fully observable finite-state MDP $M = \langle \mathcal{N} \times Z, \langle n_\iota, z_\iota \rangle, A, P_H, R_H \rangle$ where the states are given by pairs of memory nodes from an FSC and observations, the initial state $\langle n_\iota, z_\iota \rangle$ is the initial memory node, and the first observation z_ι, the actions from the POMDP, and transition and reward functions defined as

$$P_H(\langle n', z' \rangle \mid \langle n, z \rangle, a) = \sum_{s \in S} b(s \mid \langle n, z \rangle) \sum_{s' \in S} P(s' \mid s, a) O(z' \mid s', a) \eta(n' \mid n, z', a),$$
$$R_H(\langle n, z \rangle, a) = \sum_{s \in S} b(s \mid \langle n, z \rangle) R(s, a),$$

where $b(s|\langle n, z\rangle)$ is the belief of being in state s of the POMDP, given memory node n and observation z.

Recall that η is the memory update function of the FSC. This finite-history MDP will serve as the (unknown) true MDP M^* in our application of SPIBB.

6.3.2 Estimating the Finite-History MDP

Next, we describe how to estimate the true finite-history MDP M^* by an MLE-MDP $\tilde{M}$. The approach is similar to that of SPI for MDPs, except that the dataset $\mathbb{D}$ is different. Here, $\mathbb{D}$ is collected from simulating the POMDP M under (FSC) policy π_b. This yields a dataset of the form

$$\mathbb{D} = \langle\langle n_t, z_t\rangle, a_t, r_t\rangle_{t\in[1:m]}, \tag{6.5}$$

where the observations z_t come from the observation function and the memory nodes n_t are observed from the FSC.

Definition 6.9 (Finite-history MLE-MDP)**.** The MDP from Definition 6.8 can be estimated from a dataset $\mathbb{D}$ of the form (6.5), following the same approach for estimating a standard MLE-MDP as in Definition 6.1:

$$\tilde{P}_H(\langle n', z'\rangle|\langle n, z\rangle, a) = \frac{\#_{\mathbb{D}}(\langle n, z\rangle, a, \langle n', z'\rangle)}{\#_{\mathbb{D}}(\langle n, z\rangle, a)}, \text{ and}$$
$$\tilde{R}_H(\langle n, z\rangle, a) = \frac{\sum_{(\langle n_t, z_t\rangle, a_t, r_t)\in\mathbb{D}} \mathbb{1}(\langle n_t, z_t\rangle = \langle n, z\rangle \wedge a_t = a)\cdot r_t}{\#_{\mathbb{D}}(\langle n, z\rangle, a)}.$$

6.3.3 Applying SPIBB to the Finite-History MDP

In this section, we apply the theory of SPIBB, as introduced in Section 6.2, to our setting. In particular, we have just defined a true MDP M^* (the finite-history MDP, Definition 6.6) and an MLE-MDP $\tilde{M}$ estimating M^* (Definition 6.9). Let

$$\mathbb{U} = \{(\langle n, z\rangle, a) \in \mathcal{N}\times Z\times A \mid \#_{\mathbb{D}}(\langle n, z\rangle, a) \le N_\wedge\}$$

be the set of tuples $(\langle n, z\rangle, a)$ which occur less than $N_\wedge$ times in the dataset $\mathbb{D}$ for some hyperparameter $N_\wedge$. Just as in SPIBB for MDPs, we compute a new policy $\pi_I \in \Pi_k^\eta$ for the MLE-MDP $\tilde{M}$ that estimates the finite-history MDP, constrained to follow the behavior policy π_b used to collect $\mathbb{D}$ for all $(\langle n, z\rangle, a) \in \mathbb{U}$.

Theorem 6.10 (ζ-bound on history MDP)**.** *Let Π_β be the set of policies under the constraint of following π_b when $(\langle n, z\rangle, a) \in \mathbb{U}$. Then, the policy π_I computed by the SPIBB algorithm on the history MDP (Definition 6.3) is a ζ-approximate safe policy improvement over the behavior policy π_b with high probability $1-\delta$, where:*

$$\zeta = \frac{4V_{\max}}{1-\gamma}\sqrt{\frac{2}{N_\wedge}\log\frac{2|\mathcal{H}||A|2^{|Z|}}{\delta}} - \rho(\pi_I, \tilde{M}) + \rho(\pi_b, \tilde{M}).$$

The proof replaces the regular MDP from the SPIBB algorithm with the (infinite) history MDP. We can reduce the exponent from $|\mathcal{H}|$, which would be the result of naively applying the SPIBB algorithm, to $|Z|$ because of the structure of the transition function of the history MDP. In particular, the transition function of the history MDP is defined for histories h, which are appended by an action a and an observation z to haz, see Definition 6.3. As such, the successor states of h in the history MDP are fully determined by the observation z instead of the full state space, and thus we may replace $2^{|S|}$ from Equation 6.2 by $2^{|Z|}$.

Proof of Theorem 6.10. First, we restate Theorem 2.1 by Weissman et al. (2003) that bounds the L1-error of an empirical probability distribution $\tilde{P}$ given a finite number of samples.

Proposition (Theorem 2.1 by Weissman et al. (2003))**.** *Given m i.i.d. random variables distributed according to P and given an estimated probability distribution $\tilde{P}$ computed from those m variables, we have*

$$\Pr(\|P - \tilde{P}\|_1 \geq \epsilon) \leq (2^n - 2)e^{-m\epsilon^2/2}.$$

To bound the L_1-error of the estimated transition function of given a (h,a) pair of the history MDP, given $\#_{\mathbb{D}}(h,a)$ samples for $(h,a) \in \mathbb{D}$ we can apply the Theorem 2.1 of (Weissman et al., 2003) to obtain

$$\Pr\left(\|P(\cdot|h,a) - \tilde{P}(\cdot|h,a)\|_1 \geq \epsilon\right) \leq (2^{|Z|} - 2)e^{-\#_{\mathbb{D}}(h,a)\epsilon^2/2}.$$

Choosing an appropriate $N_\wedge$, we can do a union bound over all history-action pairs. We get that the probability of having an arbitrary history-action pair (h,a) of which the L_1-error is larger than ϵ is bounded by δ:

$$\begin{aligned}&\Pr\left(\exists (h,a) \in \mathcal{H} \times A \colon \|P(\cdot|h,a) - \tilde{P}(\cdot \mid h,a)\|_1 \geq \epsilon\right)\\ &\quad\leq \sum_{h,a \in \mathcal{H}\times A} \Pr\left(\|P(\cdot|h,a) - \tilde{P}(\cdot \mid h,a)\|_1 \geq \epsilon\right) \leq |\mathcal{H}||A|(2^{|Z|} - 2)e^{-N_\wedge \epsilon^2/2} = \delta.\end{aligned}$$

Reordering the equation above, we get

$$\epsilon = \sqrt{\frac{2}{N_\wedge} \log \frac{2|\mathcal{H}||A|2^{|Z|}}{\delta}}, \tag{6.6}$$

and then

$$\Pr\left(\exists h,a \in \mathcal{H} \times A \colon \|P(\cdot|h,a) - \tilde{P}(\cdot \mid h,a)\|_1 \geq \epsilon\right) \leq \delta.$$

The last step of the proof is to replace the ϵ from Equation 36 of (Laroche et al., 2019) by the value from Equation 6.6, resulting in

$$\zeta = \frac{4V_{\max}}{1-\gamma}\sqrt{\frac{2}{N_\wedge}\log\frac{2|\mathcal{H}||A|2^{|Z|}}{\delta}} - \rho(\pi_I, \tilde{M}) + \rho(\pi_b, \tilde{M}),$$

as stated in the Theorem. □

While Theorem 6.10 and its proof reason over the full history MDP, these results extend to the finite-history MDP when Assumption 6.5 is satisfied. We have the following corollary.

Corollary 6.11 (ζ-bound on finite-history MDP)**.** *Let Π_β and π_b be as in Theorem 6.10. Then, the policy π_I computed by the SPIBB algorithm in the finite-history MDP M^* of a POMDP satisfying Assumption 6.5 is a ζ-approximate safe policy improvement over the behavior policy π_b with high probability $1-\delta$, where the admissible performance loss ζ is given by*

$$\zeta = \frac{4V_{\max}}{1-\gamma}\sqrt{\frac{2}{N_\wedge}\log\frac{2|\mathcal{N}\times Z||A|2^{|Z|}}{\delta}} - \rho(\pi_I, \tilde{M}) + \rho(\pi_b, \tilde{M}).$$

Since bisimilarity is an equivalence relation, the finite-history MDP is equivalent to the full history MDP, and thus also the POMDP, see Theorem 6.7. As a consequence, the proof of Corollary 6.11 follows immediately from Theorem 6.10 and the fact that bisimulation is an equivalence relation.

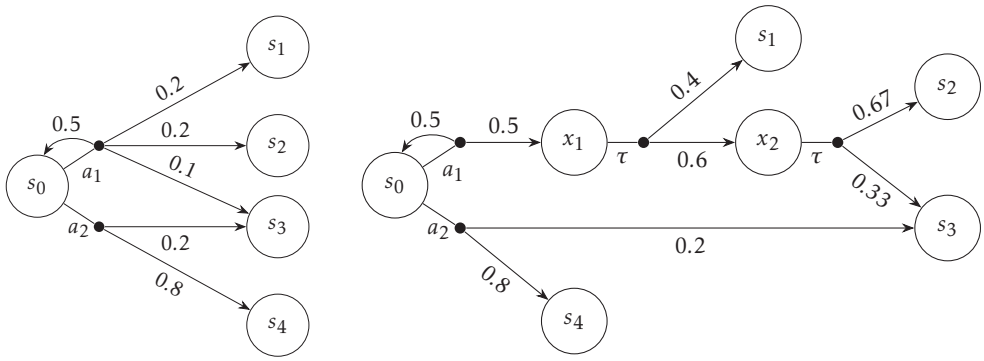
6.4 Tighter Improvement Bounds for SPI

Our second contribution to the SPI problem is to derive tighter improvement guarantees for the same dataset compared to the state-of-the-art (Laroche et al., 2019). To achieve our new guarantees, we rely on a new transformation of the dataset and MLE-MDP into a *two-successor MDP* where each state-action pair only has at most two successor states, as well as a transformation of a dataset collected on an arbitrary MDP to a dataset for the two-successor MDP. These transformations preserve (optimal) performance of policies and are reversible. Hence, we can apply SPI(BB) on the transformed environment and dataset and then transform the resulting improved policy back to the original setting, as illustrated in Figure 6.2.

In the following, we present the technical construction of two-successor MDPs and the dataset transformation that allows us to derive these tighter performance guarantees in SPI.

6.4.1 From MDP to Two-Successor MDP

Definition 6.12. A *two-successor MDP* (2sMDP) is an MDP M^{2s} where each state-action pair (s,a) has at most two possible successors states, *i.e.*, $|Post_{M^{2s}}(s,a)| \leq 2$.



(a) An MDP M. (b) The 2sMDP M^{2s}.

Figure 6.3: Example of a transformation from an MDP to a 2sMDP, where the single and double arc indicate different actions.

Every MDP $M = \langle S, s_I, A, P, R, \gamma \rangle$ can be transformed into a 2sMDP $M^{2s} = \langle S \cup S_{aux}, s_I, A \cup \{\tau\}, P^{2s}, R^{2s}, \gamma^{2s} \rangle$. To do so, we introduce a set of *auxiliary* states S_{aux} along with the *main* states S of the MDP M. Further, we include an additional action τ and adapt the probability and reward functions.

For readability, we now detail the transformation for a fixed state-action pair (s, a) with three or more successors. The transformation of the whole MDP follows from repeatedly applying this transformation to all such state-action pairs.

We enumerate the successor states of (s, a), *i.e.*, $Post_M(s, a) = \{s_1, \ldots, s_k\}$ and define $p_i = P(s_i \,|\, s, a)$ for all $i = 1, \ldots, k$. Further, we introduce $k - 2$ auxiliary states $S^{s,a}_{aux} = \{x_2, \ldots, x_{k-1}\}$, each with one available action with a binary outcome. Concretely, the two possible outcomes in state x_i are *move to state* s_i or *move to one of the states* $s_{i+1}, \ldots, s_k$, where the latter is represented by moving to an auxiliary state x_{i+1}, unless $i = k - 1$ in which case we immediately move to s_k. Formally, the new transition function $P^{2s}(\cdot \,|\, s, a)$ is:

$$P^{2s}(s_1 \,|\, s, a) = p_1, \quad P^{2s}(x_2 \,|\, s, a) = 1 - p_1.$$

For the transition function P^{2s} in the auxiliary states we define a new action τ that will be the only enabled action in these states. For $i > 1$, the transition function P^{2s} is then

$$P^{2s}(s_i \,|\, x_i, \tau) = \frac{p_i}{1 - (p_1 + \cdots + p_{i-1})},$$
$$P^{2s}(x_{i+1} \,|\, x_i, \tau, i < k - 1) = 1 - \frac{p_i}{1 - (p_1 + \cdots + p_{i-1})},$$
$$P^{2s}(s_k \,|\, x_{k-1}, \tau) = 1 - \frac{p_k}{1 - (p_{i-1} + p_k)}.$$

An example of this transformation is shown in Figure 6.3, where Figure 6.3a shows the original MDP and Figure 6.3b shows the resulting 2sMDP. As we introduce

$|Post_M(s,a)|$ auxiliary states for a state-action pair (s,a), and $k \leq |S|$ in the worst-case of a fully connected MDP, we can bound the number of states in the 2sMDP by $|S \cup S_{\text{aux}}| \leq |S| + |S||A|(|S|-2) \leq |S|^2|A|$. Note that we did not specify a particular order to enumerate the successor states. Further, other transformations utilizing auxiliary states with a different structure (*e.g.*, a balanced binary tree) are possible. However, neither the structure of the auxiliary states nor the order of successor states changes the total number of states in the 2sMDP, which is the deciding factor for applying this transformation in the context of SPI algorithms.

The extension of the reward function is straightforward, *i.e.*, the agent receives the same reward as in the original MDP when in main states and no reward when in auxiliary states:

$$R^{2s}(s,a) = \begin{cases} R(s,a) & s \in S, a \in A, \\ 0 & \text{otherwise.} \end{cases}$$

Any policy π for the MDP M can be extended into a policy π^{2s} for the 2sMDP M^{2s} by copying π for states in S and choosing τ otherwise:

$$\pi^{2s}(a|s) = \begin{cases} \pi(a|s) & s \in S, \\ \mathbb{1}[a = \tau] & s \in S_{\text{aux}}. \end{cases}$$

Finally, to preserve discounting correctly, we introduce a state-dependent discount factor γ^{2s}, such that discounting only occurs in the main states, *i.e.*,

$$\gamma^{2s}(s) = \begin{cases} \gamma & s \in S, \\ 1 & s \in S_{\text{aux}}. \end{cases}$$

This yields the following value function for the 2sMDP M^{2s}:

$$V_{M^{2s},\pi^{2s}}(s) = \sum_{a \in A} \pi^{2s}(a|s) \Big(R^{2s}(s,a) + \gamma^{2s}(s) \sum_{s' \in S} P^{2s}(s'|s,a) V_{M^{2s},\pi^{2s}}(s') \Big).$$

The performance of policy π^{2s} on M^{2s} uses the value function defined above and is denoted by $\rho^{2s}(\pi^{2s}, M^{2s}) = V_{M^{2s},\pi^{2s}}(s_\iota)$, for the initial state $s_\iota \in S$. Our transformation described above, together with the adjusted value function, indeed preserves the performance of the original MDP and policy:

Theorem 6.13 (Preservation of transition probabilities)**.** *For every transition (s,a,s') in the original MDP M, there exists a unique path $\langle s,a,x_2,\tau,\ldots,x_i,\tau,s' \rangle$ in the 2sMDP M^{2s} with the same probability. That is,*

$$\mathbb{P}_M(\langle s,a,s' \rangle) = \mathbb{P}_{M^{2s}}(\langle s,a,x_2,\tau,\ldots,x_i,\tau,s' \rangle).$$

Proof of Theorem 6.13. Fix a state $s \in S$ and action $a \in A(s)$ that is enabled in s and assume $Post_M(s,a) = \{s_1,\ldots,s_k\}$. We define $p_i = P(s_i|s,a)$ for all $i = 1,\ldots,k$. By definition, for each $s_i \in Post_M(s,a)$ with $i < k$, there exists a path $\langle s,a,x_2,\tau,\ldots,x_i,\tau,s_i \rangle$ that has positive probability. For $i = k$ the path is instead

given by $\langle s, a, x_2, \tau, \ldots, x_{k-1}, \tau, s_k \rangle$. Note that there is precisely one incoming transition for each auxiliary state $x_i \in S_{\text{aux}}^{s,a}$ and each $s_i \in Post_M(s,a)$ has precisely one predecessor in $S_{\text{aux}}^{s,a}$. Further, $Post_{M^{2s}}(s,a) \subseteq S_{\text{aux}}^{s,a} \cup Post_M(s,a)$ and also $Post_{M^{2s}}(x,\tau) \subseteq S_{\text{aux}}^{s,a} \cup Post_M(s,a)$ for all $x \in S_{\text{aux}}^{s,a}$, *i.e.*, any path $\langle s, a, \ldots \rangle$ must stay in the set $S_{\text{aux}}^{s,a}$ until it reaches a state in $Post_M(s,a)$. Thus, the path from s to s_i must be unique.

We now show that the probabilities pre- and post-transformation match:

$$\mathbb{P}_M(\langle s, a, s_i \rangle) = \mathbb{P}_{M^{2s}}(\langle s, a, x_2, \tau, \ldots, x_i, \tau, s_i \rangle).$$

For $s_i = s_1$, the statement holds by definition. Now, consider the case $2 \leq i \leq k-1$. Then, we can compute the probability of the path as

$$\begin{aligned}
&\mathbb{P}_{M^{2s}}(\langle s, a, x_2, \tau, \ldots, x_i, \tau, s_i \rangle) \\
&= P^{2s}(x_2 \,|\, s, a) \left(\prod_{j=2}^{i-1} P^{2s}(x_{j+1} \,|\, x_j, \tau) \right) P^{2s}(s_i \,|\, x_i, \tau) \\
&= (1 - p_1) \left(\prod_{j=2}^{i-1} 1 - \frac{p_j}{1 - (p_1 + \cdots + p_{j-1})} \right) \frac{p_i}{1 - (p_1 + \cdots + p_{i-1})} \\
&= (1 - p_1) \left(\prod_{j=2}^{i-1} \frac{1 - (p_1 + \cdots + p_j)}{1 - (p_1 + \cdots + p_{j-1})} \right) \frac{p_i}{1 - (p_1 + \cdots + p_{i-1})} \\
&= p_i \\
&= \mathbb{P}_M(\langle s, a, s_i \rangle).
\end{aligned}$$

For the case $i = k$, the proof is analogous to the case $i = k-1$, replacing the last factor by $P^{2s}(s_k \,|\, x_{k-1}, \tau)$. □

Corollary 6.14 (Preservation of performance)**.** *Let M be an MDP, π a policy for M, and M^{2s} the two-successor MDP with policy π^{2s} constructed from M and π as described above. Then $\rho(\pi, M) = \rho^{2s}(\pi^{2s}, M^{2s})$.*

Proof of Corollary 6.14. Recall that the auxiliary states can only choose action τ, which does not yield any reward. Further, since no discount is applied in auxiliary states, for all states $s \in S$, we have

$$\begin{aligned}
V_{M^{2s},\pi^{2s}}(s) &= \sum_{a \in A} \pi^{2s}(a \,|\, s) \Big(R^{2s}(s,a) + \gamma^{2s}(s) \sum_{s' \in S} P^{2s}(s' \,|\, s, a) V_{M^{2s},\pi^{2s}}(s') \Big) \\
&= \sum_{a \in A} \pi(a \,|\, s) \Big(R(s,a) + \gamma \sum_{s' \in S} \mathbb{P}_{M^{2s}}(s_i \,|\, s, a, x_2, \tau, \ldots, x_i, \tau) V_{M^{2s},\pi^{2s}}(s') \Big) \\
&= \sum_{a \in A} \pi(a \,|\, s) \Big(R(s,a) + \gamma \sum_{s' \in S} P(s,a) V_{M^{2s},\pi^{2s}}(s') \Big).
\end{aligned}$$

This is exactly the defining Bellman equation for $V_{M,\pi}$, *i.e.*, by definition, we have

$$V_{M,\pi}(s) = \sum_{a \in A} \pi(a|s)\Big(R(s,a) + \gamma \sum_{s' \in S} P(s,a) V_{M,\pi}(s')\Big).$$

As the Bellman equation has a unique least fixed point for a discount factor $\gamma < 1$ (Puterman, 1994), we must have $V_{M,\pi}(s) = V_{M^{2s},\pi^{2s}}(s)$ for all $s \in S$. In particular, this holds for $s = s_\iota$, and thus $\rho(\pi, M) = \rho(\pi^{2s}, M^{2s})$. □

6.4.2 Dataset Transformation

In the previous section, we discussed how to transform an MDP into a 2sMDP. However, for SPI, we do not have access to the underlying MDP, but only to a dataset $\mathbb{D}$ and the behavior policy π_b used to collect this data. In this section, we present a transformation similar to the one from MDP to 2sMDP, but now for the dataset $\mathbb{D}$. This dataset transformation allows us to estimate a 2sMDP from the transformed data via maximum likelihood estimation (MLE).

Assume a dataset $\mathbb{D}$ of observed states and actions of the form $\mathbb{D} = \langle s_t, a_t \rangle_{t \in [1:m]}$ from an MDP M. We transform the dataset $\mathbb{D}$ into a dataset $\mathbb{D}^{2s}$ that we use to estimate a two-successor MLE-MDP $\tilde{M}^{2s}$. Each sample (s_t, a_t, s_{t+1}) in $\mathbb{D}$ is transformed into a set of samples, each corresponding to a path from s_t to s_{t+1} via states in S_{aux} in M^{2s}. Importantly, the dataset transformation only relies on $\mathbb{D}$ and not on any additional knowledge about M.

Similar to the notation in Section 6.2, let $\#_{\mathbb{D}}(x)$ denote the number of times x occurs in $\mathbb{D}$. For each state-action pair $(s,a) \in S \times A$ we denote its successor states in $\tilde{M}$ as $Post_{\tilde{M}}(s,a) = \{s_i \mid \#_{\mathbb{D}}(s,a,s_i) > 0\}$, which are again enumerated by $\{s_1, \ldots, s_k\}$. Similarly as for the MDP transformation, we define $Post_{\tilde{M}^{2s}}(s,a) = Post_{\tilde{M}}(s,a)$ if $k \leq 2$ and $Post_{\tilde{M}^{2s}}(s,a) = \{s_1, x_2\}$ otherwise. For auxiliary states $x_i \in S^{s,a}_{\text{aux}}$, we define $Post_{\tilde{M}^{2s}}(x_i, \tau) = \{s_i, x_{i+1}\}$ for $i < k-1$ and $Post_{\tilde{M}^{2s}}(x_{k-1}, \tau) = \{s_{k-1}, s_k\}$. We then define the transformed dataset $\mathbb{D}^{2s}$ from $\mathbb{D}$ for each $s \in S$ and $s' \in Post_{\tilde{M}^{2s}}(s,a)$ as follows:

$$\#_{\mathbb{D}^{2s}}(s,a,s') = \begin{cases} \#_{\mathbb{D}}(s,a,s') & s' \in S, \\ \sum_{j=2}^{k} \#_{\mathbb{D}}(s,a,s_j) & s' = x_2 \in S^{s,a}_{\text{aux}}, \\ 0 & \text{otherwise.} \end{cases}$$

Further, for each $x_i \in S^{s,a}_{\text{aux}}$ and $s' \in Post_{\tilde{M}^{2s}}(s,a)$

$$\#_{\mathbb{D}^{2s}}(x_i, \tau, s') = \begin{cases} \#_{\mathbb{D}}(s,a,s') & s' \in S, \\ \sum_{j=i+1}^{k} \#_{\mathbb{D}}(s,a,s_j) & s' = x_{i+1} \in S^{s,a}_{\text{aux}}, \\ 0 & \text{otherwise.} \end{cases}$$

The following preservation results for data generated MLE-MDPs are in the line of Theorem 6.13 and Corollary 6.14. See Figure 6.2 for an overview of the relationships between theorems.

Theorem 6.15 (Preservation of estimated transition probabilities)**.** *Let $\mathbb{D}$ be a dataset and $\mathbb{D}^{2s}$ be the dataset obtained by the transformation above. Further, let $\tilde{M}$ and $\tilde{M}^{2s}$ be the MLE-MDPs constructed from $\mathbb{D}$ and $\mathbb{D}^{2s}$, respectively. Then for every transition (s,a,s') in $\tilde{M}$ there is a unique path $\langle s,a,x_2,\tau,\ldots,x_i,\tau,s'\rangle$ in $\tilde{M}^{2s}$ with the same probability:*

$$\mathbb{P}_{\tilde{M}}(\langle s,a,s'\rangle) = \mathbb{P}_{\tilde{M}^{2s}}(\langle s,a,x_2,\tau,\ldots,x_i,\tau,s'\rangle).$$

The proof of this theorem is similar to that of Theorem 6.13.

Proof of Theorem 6.15. Fix a state $s \in S$ and action $a \in A(s)$ that is enabled in s and assume $Post_{\tilde{M}}(s,a) = \{s_1,\ldots,s_k\}$. For the case $k \leq 2$ the statement trivially holds as $\#_{\mathbb{D}}(s,a,\cdot) = \#_{\mathbb{D}^{2s}}(s,a,\cdot)$.

We now consider the case $k > 2$. As we have $\tilde{P}^{2s}(s'|s,a) > 0$ if and only if $\#_{\mathbb{D}^{2s}}(s,a,s') > 0$ for all $s,s' \in S \cup S_{\text{aux}}$, and for each $s' \in S^{s,a}_{\text{aux}} \cup Post_{\tilde{M}}(s,a)$, by definition there is exactly one state $s \in S^{s,a}_{\text{aux}} \cup \{s\}$ for which $\#_{\mathbb{D}^{2s}}(s,a,s') > 0$. Thus, for each $i \in \{1,\ldots,k\}$ there must be exactly one unique path $\langle s,a,x_2,\tau,\ldots,x_i,\tau,s_i\rangle$ in MLE-2sMDP $\tilde{M}^{2s}$.

We now show that $\mathbb{P}_{\tilde{M}}(\langle s,a,s_i\rangle) = \mathbb{P}_{\tilde{M}^{2s}}(\langle s,a,x_2,\tau,\ldots,x_i,\tau,s_i\rangle)$. For $s_i = s_1$, the statement holds by definition. Now consider the case $2 \leq i \leq k-1$. Then, we can compute the probability of the path as

$$\begin{aligned}
&\mathbb{P}_{\tilde{M}^{2s}}(\langle s,a,x_2,\tau,\ldots,x_i,\tau,s_i\rangle)\\
&= P_{\tilde{M}^{2s}}(x_2|s,a)\cdot\left(\prod_{j=2}^{i-1} P_{\tilde{M}^{2s}}(x_{j+1}|x_j,\tau)\right)\cdot P_{\tilde{M}^{2s}}(s_i|x_i,\tau)\\
&= \frac{\#_{\mathbb{D}^{2s}}(s,a,x_2)}{\sum_{s'}\#_{\mathbb{D}^{2s}}(s,a,s')}\cdot\left(\prod_{j=2}^{i-1}\frac{\#_{\mathbb{D}^{2s}}(x_i,\tau,x_{j+1})}{\sum_{s'}\#_{\mathbb{D}^{2s}}(x_j,\tau,s')}\right)\cdot\frac{\#_{\mathbb{D}^{2s}}(x_i,\tau,s_i)}{\sum_{s'}\#_{\mathbb{D}^{2s}}(x_i,\tau,s')}\\
&= \frac{\#_{\mathbb{D}^{2s}}(s,a,x_2)}{\#_{\mathbb{D}^{2s}}(s,a,s_1)+\#_{\mathbb{D}^{2s}}(s,a,x_2)}\cdot\left(\prod_{j=2}^{i-1}\frac{\#_{\mathbb{D}^{2s}}(x_i,\tau,x_{j+1})}{\#_{\mathbb{D}^{2s}}(x_j,\tau,s_j)+\#_{\mathbb{D}^{2s}}(x_j,\tau,x_{j+1})}\right)\\
&\qquad\cdot\frac{\#_{\mathbb{D}^{2s}}(x_i,\tau,s_i)}{\#_{\mathbb{D}^{2s}}(x_i,\tau,s_i)+\#_{\mathbb{D}^{2s}}(x_i,\tau,x_{i+1})}\\
&= \frac{\sum_{m=2}^{k}\#_{\mathbb{D}}(s,a,s_m)}{\#_{\mathbb{D}}(s,a,s_1)+\sum_{m=2}^{k}\#_{\mathbb{D}}(s,a,s_m)}\cdot\left(\prod_{j=2}^{i-1}\frac{\sum_{m=j+1}^{k}\#_{\mathbb{D}}(s,a,s_m)}{\#_{\mathbb{D}}(s,a,s_j)+\sum_{m=j+1}^{k}\#_{\mathbb{D}}(s,a,s_m)}\right)\\
&\qquad\cdot\frac{\#_{\mathbb{D}}(s,a,s_i)}{\#_{\mathbb{D}}(s,a,s_i)+\sum_{m=i+1}^{k}\#_{\mathbb{D}}(s,a,s_m)}\\
&= \frac{\sum_{m=2}^{k}\#_{\mathbb{D}}(s,a,s_m)}{\sum_{m=1}^{k}\#_{\mathbb{D}}(s,a,s_m)}\cdot\left(\prod_{j=2}^{i-1}\frac{\sum_{m=j+1}^{k}\#_{\mathbb{D}}(s,a,s_m)}{\sum_{m=j}^{k}\#_{\mathbb{D}}(s,a,s_m)}\right)\cdot\frac{\#_{\mathbb{D}}(s,a,s_i)}{\sum_{m=i}^{k}\#_{\mathbb{D}}(s,a,s_m)}
\end{aligned}$$

$$= \frac{\#_{\mathbb{D}}(s,a,s_i)}{\sum_{m=1}^{k} \#_{\mathbb{D}}(s,a,s_m)}$$
$$= \frac{\#_{\mathbb{D}}(s,a,s_i)}{\sum_{s'} \#_{\mathbb{D}}(s,a,s')}$$
$$= \mathbb{P}_{\tilde{M}}(\langle s,a,s_i\rangle).$$

The case where $i = k$ is analogous to the case $i = k-1$. □

Corollary 6.16 (Preservation of estimated performance)**.** *Let $\tilde{M}$ and $\tilde{M}^{2s}$ be the MLE-MDPs as above, constructed from $\mathbb{D}$ and $\mathbb{D}^{2s}$, respectively. Further, let $\tilde{\pi}$ be an arbitrary policy on $\tilde{M}$ and $\tilde{\pi}^{2s}$ the policy that extends π for $\tilde{M}^{2s}$ by choosing τ in all auxiliary states. Then $\rho(\tilde{\pi},\tilde{M}) = \rho^{2s}(\tilde{\pi}^{2s},\tilde{M}^{2s})$.*

Proof of Corollary 6.16. Using Theorem 6.15, the proof is analogous to the proof of Corollary 6.14. □

We want to emphasize that while $\mathbb{D}^{2s}$ may contain more samples than $\mathbb{D}$, it does not yield any additional information. Rather, instead of viewing each transition sample as an atomic data point, in $\mathbb{D}^{2s}$ transition samples should be considered a multi-step process, *e.g.*, the sample $(s,a,s_3) \in \mathbb{D}$ would be transformed into the samples $\{(s,a,x_2),(x_2,\tau,x_3),(x_3,\tau,s_3)\} \in \mathbb{D}^{2s}$, which in the construction of the MLE-MDP is used to estimate the probabilities $P(s' \neq s_1 \,|\, s,a), P(s' \neq s_2 \,|\, s,a,s' \neq s_1)$ and $P(s' = s_3 \,|\, s,a,s' \neq s_1, s' \neq s_2)$, respectively. The probabilities of these events are mutually independent, but when multiplied, give precisely $P(s_3 \,|\, s,a)$.

6.4.3 SPI in Two-Successor MDPs

In this section, we discuss how SPI can benefit from two-successor MDPs as constructed following our new transformation. The dominating term in the bound $N_{\wedge}^{\text{SPIBB}}$ obtained by (Laroche et al., 2019) is the branching factor of the MDP, which, without any prior information, has to necessarily be over-approximated by $|S|$, see Equation 6.4 and subsequent discussion. We use our transformation from regular MDP to two-successor MDP to bound the branching factor to $k = 2$, which allows us to provide stronger guarantees with the same dataset (or, conversely, require less data to guarantee a set maximum performance loss). Note that bounding the branching factor by any other constant can be achieved by a similar transformation as in Section 6.4, but $k = 2$ leads to an optimal bound, as we shall establish later.

Let $\tilde{M}$ and $\tilde{M}^{2s}$ be the MLE-MDPs inferred from datasets $\mathbb{D}$ and $\mathbb{D}^{2s}$, respectively. Further, let $\pi_{\odot}$ and $\pi_{\odot}^{2s}$ denote the optimal policies in these MLE-MDPs, constrained to the set of policies that follow the behavior policy π_b for state-action pairs $(s,a) \in \mathbb{U}$. Note that these optimal policies can easily be computed using, *e.g.*, standard value iteration, see Chapter 2. First, we show how to improve the admissible performance loss ζ in SPI on two-successor MDPs.

Lemma 6.17. *Let M^{2s} be a two-successor MDP with behavior policy π_b. Then, $\pi^{2s}_{\odot}$ is a ζ-approximately safe policy improvement over π_b with high probability $1-\delta$, where:*

$$\zeta = \frac{4V_{max}}{1-\gamma}\sqrt{\frac{2}{N_\wedge}\log\frac{8|S||A|}{\delta}} - \rho^{2s}(\pi^{2s}_{\odot}, \tilde{M}^{2s}) + \rho^{2s}(\pi_b, \tilde{M}^{2s}).$$

Proof of Lemma 6.17. The proof follows a similar argumentation as (Laroche et al., 2019, Theorem 2). First, note that we cannot directly apply the cited theorem as we deal with a two-successor MDP for which $\rho(\cdot,\cdot)$ is defined in a different manner. More precisely, the discount rate γ is not constant. We now show that the results can still be applied in our setting by adapting the proof of (Laroche et al., 2019, Theorem 2).

Let $M^{2s} = \langle S \cup S_{\text{aux}}, s_\iota, A \cup \{\tau\}, P^{2s}, R^{2s}, \gamma^{2s}\rangle$ be the given two-successor MDP (Definition 6.12), $\mathbb{D}$ a dataset of trajectories over M^{2s} and $N_\wedge \in \mathbb{N}$ the sample size requirement given as a parameter. The set of bootstrapped state-action pairs is defined as $\mathbb{U} = \{(s,a) \mid \#_{\mathbb{D}^{2s}}(s,a) < N_\wedge\} \cup \{(s,\tau) \mid s \in S\}$. Note that actions in auxiliary states are always bootstrapped. The behavior policy π_b is decomposed into $\tilde{\pi}_b$, for bootstrapped and $\dot{\pi}_b$ for non-bootstrapped actions:

$$\tilde{\pi}_b = \begin{cases} \pi_b(a|s) & (s,a) \in \mathbb{U}, \\ 0 & \text{else}, \end{cases}$$

and $\dot{\pi}_b(a|s) = \pi_b(a|s) - \tilde{\pi}_b(a|s)$.

The remainder of the proof relies on the use of *semi-MDPs* (Sutton and Barto, 1998; Sutton et al., 1999).

Definition 6.18 (Semi-MDP)**.** A *semi-MDP* is a tuple $\langle S, s_\iota, \Omega_A, P, R, \Gamma\rangle$, where S, s_ι, P and R are as in standard MDPs, Ω_A is the set of *options*, and $\Gamma\colon S \times A \to (0,1)$ is the state-action dependent discount factor. An option is a tuple $\langle I, \pi, \beta\rangle \in \Omega_A$ that consists of an initiation set $I \subseteq S$, a policy $\pi\colon S \to \mathcal{D}(A)$ and a termination condition $\beta\colon S \to [0,1]$.

Semantically, a semi-MDP works as follows. An option $\langle I, \pi, \beta\rangle$ is available in states $s_t \in I$. If the agent takes this option, actions are chosen stochastically in accordance with π until the termination condition β is met. That is, from $s_t \in I$, action a_t is taken with probability $\pi(a_t|s_t)$, the environment transitions to s_{t+1}, after which the option terminates with probability $\beta(s_{t+1})$.

Following the same strategy as the original proof in (Laroche et al., 2019), we now transform M^{2s} into its corresponding bootstrapped semi-MDP counterpart $\ddot{M}$. We define this semi-MDP as a tuple $\ddot{M} = \langle S \cup S_{\text{aux}}, s_\iota, \Omega_A, \ddot{P}, \ddot{R}, \Gamma\rangle$

with $\Omega_A = \{\omega_a\}_{a\in A}$ where for each $a \in A$

$$\omega_a = \langle I_a, a\colon \pi_b, \beta(s)\rangle = \begin{cases} I_a = \{s \mid (s,a) \notin \mathbb{U}\}, \\ a\colon \pi_b, \\ \beta(s) = \sum_{a'\in A\cup\{\tau\}} \dot{\pi}_b(s,a'), \end{cases}$$

where $a\colon \pi_b$ is the option where the agent first uses action a, and then plays according to π_b, and

$$\Gamma(s,a) = \begin{cases} \gamma^{2\mathrm{s}} & s \in S, \\ 1 & s \in S_{\mathrm{aux}}. \end{cases}$$

P and R are naturally extended to options, *i.e.*, $\ddot{P}(s,\omega_a) = P^{2s}(s,a)$ and $\ddot{R}(s,\omega_a) = R^{2s}(s,a)$. In the same fashion we transform the MLE-MDP $\tilde{M}^{2\mathrm{s}}$ into its bootstrapped counterpart $\tilde{\tilde{M}}^{2s} = \langle S \cup S_{\mathrm{aux}}, s_\iota, \Omega_A, \tilde{\tilde{P}}, \ddot{R}, \Gamma\rangle$.

Using (Weissman et al., 2003, Theorem 2.1) on all $(s,a) \notin \mathbb{U}$ we obtain that for all $a \in A$ and $s \in I_a$, with probability at least $1-\delta$ we have

$$\begin{aligned} \|\gamma^{2\mathrm{s}}\ddot{P}(s,a) - \gamma^{2\mathrm{s}}\tilde{\tilde{P}}(s,a)\|_1 &\le \sqrt{\frac{2}{\min\limits_{a\in A, s\in I_a} \#_{\mathbb{D}^{2\mathrm{s}}}(s,a)} \log \frac{2|S||A|2^{\max_{a\in A, s\in I_a}|Post(s,a)|}}{\delta}} \\ &\le \sqrt{\frac{2}{N_\wedge} \log \frac{8|S||A|}{\delta}} \end{aligned}$$

This means we can apply Lemma 1 from (Laroche et al., 2019) to $M^{2\mathrm{s}}$ and $\tilde{M}^{2\mathrm{s}}$ for arbitrary bootstrapped policies to obtain

$$|\rho(\pi^{2\mathrm{s}}_\odot, M^{2\mathrm{s}}) - \rho(\pi^{2\mathrm{s}}_\odot, \tilde{M}^{2\mathrm{s}})| \le \frac{2V_{max}}{1-\gamma}\sqrt{\frac{2}{N_\wedge}\log\frac{8|S||A|}{\delta}},$$

$$|\rho(\pi^{2\mathrm{s}}_b, M^{2\mathrm{s}}) - \rho(\pi^{2\mathrm{s}}_b, \tilde{M}^{2\mathrm{s}})| \le \frac{2V_{max}}{1-\gamma}\sqrt{\frac{2}{N_\wedge}\log\frac{8|S||A|}{\delta}}.$$

Combining these inequalities, we obtain the final result

$$\begin{aligned} \zeta^{2s} &= \rho(\pi^{2\mathrm{s}}_b, M^{2\mathrm{s}}) - \rho(\pi^{2\mathrm{s}}_\odot, M^{2\mathrm{s}}) \\ &\le \frac{4V_{max}}{1-\gamma}\sqrt{\frac{2}{N_\wedge}\log\frac{8|S||A|}{\delta}} - \rho(\pi^{2\mathrm{s}}_\odot, \tilde{M}^{2\mathrm{s}}) + \rho(\pi^{2\mathrm{s}}_b, \tilde{M}^{2\mathrm{s}}). \end{aligned}$$ □

For a general MDP M, we can utilize this result by first applying the transformation from Section 6.4.1.

Theorem 6.19 (Weissman-based tighter improvement guarantee)**.** *Consider an MDP $M = \langle S, s_\iota, A, P, R, \gamma\rangle$ with behavior policy π_b. Then, $\pi_\odot$ is a $\zeta^{2\mathrm{s}}$-approximate safe policy*

improvement over π_b with high probability $1-\delta$, where:

$$\zeta^{2s} = \frac{4V_{max}}{1-\gamma}\sqrt{\frac{2}{N_\wedge^{2s}}\log\frac{8|S|^2|A|^2}{\delta}} - \rho(\pi_\odot, \tilde{M}) + \rho(\pi_b, \tilde{M}). \tag{6.7}$$

Proof of Theorem 6.19. The proof follows straightforwardly from Theorem 6.13 and Theorem 6.15.

$$\begin{aligned}\rho(\pi_\odot, M) &= \rho(\pi_\odot^{2s}, M^{2s}) \\ &\geq \rho(\pi_b^{2s}, M^{2s}) - \frac{4V_{max}}{1-\gamma}\sqrt{\frac{2}{N_\wedge}\log\frac{8|S|^2|A|^2}{\delta}} - \rho(\pi_\odot^{2s}, \tilde{M}^{2s}) + \rho(\pi_b^{2s}, \tilde{M}^{2s}) \\ &= \rho(\pi_b, M) - \zeta^{2s}\end{aligned}$$

Both equalities follow by applying Theorem 6.13 and Theorem 6.15. The inequality is obtained by applying Lemma 6.17 to M^{2s} where the size of the state space of M^{2s} is bounded by $|S \cup S_{\text{aux}}| \leq |S|^2|A|$, see Section 6.4.1 □

Just as done for ζ^{SPIBB}, we can rearrange the Equation 6.7 to compute the sample size requirement $N_\wedge^{2s}$ for a ζ^{2s}-safe improvement:

$$N_\wedge^{2s} = \frac{32V_{max}^2}{(\zeta^{2s})^2(1-\gamma)^2}\log\frac{8|S|^2|A|^2}{\delta}.$$

Remark 6.20. Note that ζ^{2s} and $N_\wedge^{2s}$ only depend on parameters of M and policy performances on $\tilde{M}$, which follows from Corollary 6.16 yielding $\rho(\pi_\odot, \tilde{M}) = \rho^{2s}(\pi_\odot, \tilde{M}^{2s})$. Hence, it is not necessary to explicitly compute the transformed MLE-MDP $\tilde{M}^{2s}$.

k-successor MDPs

We saw that the $N_\wedge^{\text{SPIBB}}$ grows linearly in $|S|$ whereas $N_\wedge^{2s}$ grows logarithmically in $|S|$. The contributing factor to this was that for any state-action pair (s,a) the L_1-norm between the true successor distribution $P(s,a)$ and its maximum likelihood estimate can only be bounded linearly in terms of the branching factor $k = |Post_M(s,a)|$ when applying the bound obtained by (Weissman et al., 2003). Hence, the idea behind our 2sMDP transformation was to bound the branching factor by a constant. To achieve this, we necessarily needed to introduce auxiliary states, essentially creating a trade-off between the branching factor and the size of the state space. One question that may arise is whether it may be beneficial to allow for a larger branching factor k than 2, possibly harvesting the advantage of introducing fewer auxiliary states. In Section 6.4.1 we hint at the fact that for SPI algorithms, $k = 2$ is indeed optimal. We now outline why this is the case.

First, notice that we can easily adapt the transformation outlined in Section 6.4.1 towards k-successor MDPs by structuring the auxiliary nodes in a tree with branching factor 2 rather than in a binary tree, resulting in up to $|S|/k-1$ auxiliary states in

each tree. Thus, when transforming an arbitrary MDP into a k-successor MDP, we can bound the L_1-error in the same fashion as in Theorem 6.19 to obtain

$$N_\wedge^{ks} = \frac{32V_{max}^2}{\zeta^2(1-\gamma)^2}\log\frac{2|S|^2|A|^2 2^k}{(k-1)\delta}$$

As all terms are positive, his expression is minimal if and only if $\frac{2^k}{k-1}$ is minimal, which is the case for $k=2$ and $k=3$. Hence, 2sMDPs are optimal for SPIBB when utilizing the L_1-norm bound by (Weissman et al., 2003).

The main reason why we choose a branching factor of $k=2$ over $k=3$ is that for $k=2$, we can give even tighter bounds on the L_1-norm by computing integrals over the probability density function of the transition probabilities that are given by a beta distribution as described in Section 6.4.4. Next, we provide proofs for the Lemma and Theorem in that section.

6.4.4 Uncertainty in Two-Successor MDPs

So far, the methods we outlined relied on a bound of the L_1-distance between a probability vector and its estimate based on a number of samples (Weissman et al., 2003). In this section, we outline a second method to tighten this bound for two-successor MDP and how to apply it to obtain a smaller admissible performance loss ζ^β for a fixed sample size requirement $N_\wedge^\beta$ based on the *inverse incomplete Beta function* (Temme, 1992).

In the following, we use that every two-successor MDP is also a regular MDP to simplify notation. Given a 2sMDP (represented as a standard MDP) $M^{2s} = \langle S, s_\iota, A, P, R\rangle$ and an error tolerance δ, we construct an error function $e\colon S\times A\to\mathbb{R}$ that ensures with probability $1-\delta$ that $\|P(s,a)-\hat{P}(s,a)\|_1 \le e(s,a)$ for all (s,a). To achieve this, we distribute δ uniformly over all states to obtain $\delta_T = \delta/|S|$, independently ensuring that for each state-action pair (s,a) the condition $\|P(s,a)-\hat{P}(s,a)\|_1 \le e(s,a)$ is satisfied with probability at least $1-\delta_T$.

We now fix a state-action pair (s,a). Since we are dealing with a two-successor MDP, there are only two successor states, s_1 and s_2. To bound the error function, we view each sample of action a in state s as a Bernoulli trial. As shorthand notation, let $p = P(s,a,s_1)$ and $1-p = P(s,a,s_2)$. Using a uniform prior over p and given a dataset $\mathbb{D}$ in which (s,a,s_1) occurs k_1 times and (s,a,s_2) occurs k_2 times, the posterior probability over p is given by a beta distribution with parameters k_1+1 and k_2+1, *i.e.*, $\Pr(p\,|\,\mathbb{D}) \sim B(k_1{+}1, k_2{+}1)$ (Jaynes, 2003). We can express the error function in terms of the probability of p being contained in a given interval $[\underline{p},\overline{p}]$ as $e(s,a) = \overline{p}-\underline{p}$.

The task that remains is to find such an interval $[\underline{p},\overline{p}]$ for which we can guarantee with probability δ_T that p is contained within it. Formally, we can express this via the incomplete regularized beta function I, which in turn is defined as the cumulative density function of the beta distribution B:

$$\Pr(p\in[\underline{p},\overline{p}]) = I_{\underline{p},\overline{p}}(k_1{+}1, k_2{+}1).$$

We show that we can define the smallest such interval in terms of the inverse incomplete beta function (Temme, 1992), denoted as I_δ^{-1}.

Lemma 6.21. *Let $k \sim Bin(n,p)$ be a random variable according to a binomial distribution. Then the smallest interval $[\underline{p},\overline{p}]$ for which*

$$\mathbb{P}\left(p \in \left[\underline{p},\overline{p}\right]\right) \geq 1-\delta_T$$

holds, has size

$$\overline{p}-\underline{p} \leq 1-2I^{-1}_{\delta_T/2}\left(\frac{n}{2}+1,\frac{n}{2}+1\right).$$

Proof of Lemma 6.21. We first show that for $k=\frac{n}{2}$ the statement holds.

Let $B(\cdot,\cdot)$ denote the beta function and $f_B(\cdot\,|\,a,b)$ the probability density function of the Beta distribution with parameters a and b.

Consider the interval $[\underline{p},\overline{p}] = [\frac{1}{2}-h,\frac{1}{2}+h]$ with $h=\frac{1}{2}\left(1-2I^{-1}_{\delta_T/2}\left(\frac{n}{2}+1,\frac{n}{2}+1\right)\right)$. The interval clearly has size $\overline{p}-\underline{p}=1-2I^{-1}_{\delta_T/2}\left(\frac{n}{2}+1,\frac{n}{2}+1\right)$.

Now we show that it contains p with probability $1-\delta_T$.

$$\begin{aligned}
\mathbb{P}\left(p \in \left[\underline{p},\overline{p}\right]\right) &= \int_{\frac{1}{2}-h}^{\frac{1}{2}+h} f_B(u\,|\,\frac{n}{2}+1,\frac{n}{2}+1) && (6.8)\\
&= \int_{\frac{1}{2}-h}^{\frac{1}{2}+h} \frac{u^{\frac{n}{2}}(1-u)^{\frac{n}{2}}}{B\left(\frac{n}{2}+1,\frac{n}{2}+1\right)}du && (6.9)\\
&= I_{\frac{1}{2}+h}\left(\frac{n}{2}+1,\frac{n}{2}+1\right) - I_{\frac{1}{2}-h}\left(\frac{n}{2}+1,\frac{n}{2}+1\right) && (6.10)\\
&= 1-2\cdot I_{\frac{1}{2}-h}\left(\frac{n}{2}+1,\frac{n}{2}+1\right) && (6.11)\\
&= 1-2I_{I^{-1}_{\delta_T/2}\left(\frac{n}{2}+1,\frac{n}{2}+1\right)}\left(\frac{n}{2}+1,\frac{n}{2}+1\right) && (6.12)\\
&= 1-2\frac{\delta_T}{2} && (6.13)\\
&= 1-\delta_T && (6.14)
\end{aligned}$$

Note that Equation 6.8 assumes a uniform prior. Equation 6.10 is obtained by definition and Equation 6.11 by using the symmetry of $I(a,a)$.

Due to symmetry of $f_B(\cdot\,|\,\frac{n}{2},\frac{n}{2})$ and its monotonicity on the intervals $[0,\frac{1}{2})$ and $(\frac{1}{2},1]$ we have for all $r \in [0,\frac{1}{2}-h]\cup[\frac{1}{2}+h,1]$ and $s\in[\frac{1}{2}-h,\frac{1}{2}+h]$ that

$$f_B(r\,|\,\frac{n}{2},\frac{n}{2}) \leq f_B(\frac{1}{2}-h\,|\,\frac{n}{2},\frac{n}{2}) = f_B(\frac{1}{2}+h\,|\,\frac{n}{2},\frac{n}{2}) \leq f_B(s\,|\,\frac{n}{2},\frac{n}{2}).$$

Further, as $f_B(\cdot\,|\,\frac{n}{2},\frac{n}{2})$ is positive on $[0,1]$, we conclude that for any interval $[\underline{p}',\overline{p}']$ with $\overline{p}'-\underline{p}'=\overline{p}-\underline{p}$ we have $\mathbb{P}\left(p\in\left[\underline{p}',\overline{p}'\right]\right) \leq \mathbb{P}\left(p\in\left[\underline{p},\overline{p}\right]\right)$. As all steps in the chain above are equalities, $\left[\underline{p},\overline{p}\right]$ is indeed the smallest interval for which we can guarantee $\mathbb{P}\left(p\in\left[\underline{p},\overline{p}\right]\right)\geq 1-\delta_T$.

Next, we consider arbitrary $1 \leq k \leq n-1$. In this case, we construct the following interval, which contains p with probability $1-\delta_T$ by definition:

$$\left[\underline{p}_k, \overline{p}_k\right] = \left[I^{-1}_{\delta_T/2}(k, n-k), I^{-1}_{1-\delta_T/2}(k, n-k)\right].$$

Note that for $k = n/2$ the intervals $\left[\underline{p}, \overline{p}\right]$ and $\left[\underline{p}_k, \overline{p}_k\right]$ coincide.

We now show that the size of the interval $\overline{p}_k - \underline{p}_k$ is maximal if $k = n/2$. We do this by computing the derivative with respect to k. Substituting $a = k$ and $b = n-k$, using symmetry, applying the multi-variable chain rule, and renaming integration variables, we obtain:

$$\begin{aligned}
&\frac{\partial}{\partial k}(I^{-1}_{1-\delta_T/2}(k, n-k) - I^{-1}_{\delta_T/2}(k, n-k)) \\
&= \frac{\partial}{\partial k}(I^{-1}_{1-\delta_T/2}(a, b) - I^{-1}_{\delta_T/2}(a, b)) \\
&= \frac{\partial}{\partial k}(1 - I^{-1}_{\delta_T/2}(b, a) - I^{-1}_{\delta_T/2}(a, b)) \\
&= \frac{\partial}{\partial b} I^{-1}_{\delta_T/2}(b, a) - \frac{\partial}{\partial a} I^{-1}_{\delta_T/2}(b, a) - \frac{\partial}{\partial a} I^{-1}_{\delta_T/2}(a, b) + \frac{\partial}{\partial b} I^{-1}_{\delta_T/2}(a, b) \\
&= \frac{\partial}{\partial c} I^{-1}_{\delta_T/2}(c, a) - \frac{\partial}{\partial c} I^{-1}_{\delta_T/2}(b, c) - \frac{\partial}{\partial c} I^{-1}_{\delta_T/2}(c, b) + \frac{\partial}{\partial c} I^{-1}_{\delta_T/2}(a, c) \\
&= \frac{\partial}{\partial c}\left(I^{-1}_{\delta_T/2}(c, a) - I^{-1}_{\delta_T/2}(c, b)\right) - \frac{\partial}{\partial c}\left(I^{-1}_{\delta_T/2}(b, c) - I^{-1}_{\delta_T/2}(a, c)\right) \\
&= \frac{\partial}{\partial c}\left(I^{-1}_{\delta_T/2}(c, b) - I^{-1}_{\delta_T/2}(c, a)\right) + \frac{\partial}{\partial c}\left(I^{-1}_{1-\delta_T/2}(c, b) - I^{-1}_{1-\delta_T/2}(c, a)\right)
\end{aligned}$$

Clearly, for $a = b$ the expression equals 0, *i.e.*, for $k = n/2$ the interval size reaches an extreme point. As the function $a \mapsto I^{-1}_p(a, b)$ for any $p \in (0, 1)$ and $b \geq 1$ is concave on $[1, \infty)$ (Askitis, 2021), both $\frac{\partial}{\partial c}(I^{-1}_{1-\delta_T/2}(b, c) - I^{-1}_{1-\delta_T/2}(a, c))$ and $\frac{\partial}{\partial c}(I^{-1}_{\delta_T/2}(b, c) - I^{-1}_{\delta_T/2}(a, c))$ are positive if and only if $b > a$. That is, exactly when $k/2 < n$. Analogously, the expression is negative for $k/2 > n$. Thus, $k = n/2$ is the only extreme point in the interval $[0, 1]$ and a maximum.

As a result, for every k we have an interval $\left[\underline{p}_k, \overline{p}_k\right]$ that contains p with probability at least $1-\delta_T$ and has size bounded by

$$\overline{p}_k - \underline{p}_k \leq 1 - 2I^{-1}_{\delta_T/2}\left(\frac{n}{2} + 1, \frac{n}{2} + 1\right),$$

and for $k = n/2$ no smaller interval exists. □

Remark 6.22. Note that in Equation 6.8, we assumed a uniform prior in the binomial distribution. However, we can easily generalize this result for other choices of beta-distributed priors. Observe that the proof only relies on the parameters of the

beta distribution but not how the parameters are composed, *i.e.*, to which extent the prior hyperparameters or the samples contributed. Thus, we can generalize the result as follows:

Corollary 6.23. *Let $k \sim Bin(n,p)$ be a random variable according to a binomial distribution. Assume a a beta-distributed prior $p \sim B(\alpha_1, \alpha_2)$ Then, the smallest interval $[\underline{p}, \overline{p}]$ for which*

$$\mathbb{P}\left(p \in \left[\underline{p}, \overline{p}\right]\right) \geq 1 - \delta_T$$

holds, has size

$$\overline{p} - \underline{p} \leq 1 - 2I^{-1}_{\delta_T/2}\left(\frac{n+\alpha_1+\alpha_2}{2}, \frac{n+\alpha_1+\alpha_2}{2}\right).$$

Note that the interval size decreases monotonically as $n + \alpha_1 + \alpha_2$ increases. As $\alpha_1, \alpha_2 > 0$, in case no information about the prior distribution of p is present, we can still give a lower bound of the interval size, namely $1 - 2I^{-1}_{\delta_T/2}\left(\frac{n}{2}, \frac{n}{2}\right)$ by underapproximating $\alpha_1 = \alpha_2 = 0$.

Next, we show how to utilize this bound for the interval size in MDPs with arbitrary topology. The core idea is the same as in Theorem 6.19: We transform the MDP into a 2sMDP and apply the error bound $e(s,a) = \overline{p} - \underline{p}$ from Lemma 6.21.

Theorem 6.24 (Beta-based tighter improvement guarantee)**.** *Let M be an MDP with behavior policy π_b. Then, $\pi_\odot$ is a ζ^β-approximate safe policy improvement over π_b with high probability $1-\delta$, where:*

$$\zeta^\beta = \frac{4V_{max}}{1-\gamma}\left(1 - I^{-1}_{\delta_T/2}\left(\frac{N^\beta_\wedge}{2}+1, \frac{N^\beta_\wedge}{2}+1\right)\right) - \rho(\pi_\odot, \tilde{M}) + \rho(\pi_b, \tilde{M}), \tag{6.15}$$

with $\delta_T = \frac{\delta}{|S|^2|A|^2}$.

Proof of Theorem 6.24. Let $M^{2s} = \langle S \cup S_{\text{aux}}, s_\iota, A \cup \{\tau\}, P^{2s}, R^{2s}, \gamma^{2s}\rangle$ be the 2s-MDP obtained by transforming M. We then define the set of bootstrapped state-action pairs in M^{2s} as $\mathfrak{B}^{2s} = \{(s,a) \mid s \in S, a \in A(s) : \#_{\mathbb{D}^{2s}}(s,a) < N^\beta_\wedge\} \cup \{(s,\tau) \mid s \in S_{\text{aux}}\}$. As a consequence, the set of non-bootstrapped actions is of size at most $|S||A|$, *i.e.*, $|\{(s,a) \mid (s,a) \notin \mathfrak{B}^{2s}\}| \leq |S||A|$. Distributing the error tolerance δ uniformly over all state-action pairs and by applying Corollary 6.23 we can ensure with high probability $1-\delta$ that $e(s,a) \leq 1 - 2I^{-1}_{\delta_T/2}\left(\frac{n}{2}+1, \frac{n}{2}+1\right)$ for all $(s,a) \notin \mathfrak{B}$. Note that although we assumed uniform priors for all transitions in M, we do not necessarily have uniform priors for all transitions in M^{2s}. Precisely, by the marginal distributions of the Dirichlet distribution, all transitions (s,a) in M^{2s} have a prior of $B(1,m)$ where m is the number of states reachable from s by action a through only auxiliary states. This is why we have to apply Corollary 6.23 rather than Lemma 6.21. However, the bound from Lemma 6.21 is still as tight as possible since there are transitions with $m = 1$, namely all τ-transitions from the last auxiliary state after the transformation and all transitions that were already binary before the transformation. We

Method	Admissible performance loss ζ	Sample size requirement $N_\wedge$
Standard SPI (Petrik et al., 2016)	$\zeta^{\text{SPI}} = \frac{2\gamma V_{max}}{1-\gamma}\sqrt{\frac{2}{N_\wedge^{\text{SPI}}}\log\frac{2\lvert S\rvert\lvert A\rvert 2^{\lvert S\rvert}}{\delta}}$	$N_\wedge^{\text{SPI}} = \frac{8V_{max}^2}{{\zeta^{\text{SPI}}}^2(1-\gamma)^2}\log\frac{2\lvert S\rvert\lvert A\rvert 2^{\lvert S\rvert}}{\delta}$ $(\star)$
Standard SPIBB (Laroche et al., 2019)	$\zeta^{\text{SPIBB}} = \frac{4V_{max}}{1-\gamma}\sqrt{\frac{2}{N_\wedge^{\text{SPIBB}}}\log\frac{2\lvert S\rvert\lvert A\rvert 2^{\lvert S\rvert}}{\delta}} + \tilde{\rho}$	$N_\wedge^{\text{SPIBB}} = \frac{32V_{max}^2}{{\zeta^{\text{SPIBB}}}^2(1-\gamma)^2}\log\frac{2\lvert S\rvert\lvert A\rvert 2^{\lvert S\rvert}}{\delta}$
Two-Successor SPIBB (Theorem 6.19)	$\zeta^{2\text{s}} = \frac{4V_{max}}{1-\gamma}\sqrt{\frac{2}{N_\wedge^{2\text{s}}}\log\frac{8\lvert S\rvert^2\lvert A\rvert^2}{\delta}} + \tilde{\rho}$	$N_\wedge^{2\text{s}} = \frac{32V_{max}^2}{(\zeta^{2\text{s}})^2(1-\gamma)^2}\log\frac{8\lvert S\rvert^2\lvert A\rvert^2}{\delta}$
Inverse beta SPIBB (Theorem 6.24)	$\zeta^{\beta} = \frac{4V_{max}}{1-\gamma}\left(1 - 2I^{-1}_{\delta_T/2}\left(\frac{N_\wedge^\beta}{2}+1, \frac{N_\wedge^\beta}{2}+1\right)\right) + \tilde{\rho}$	No closed formula available (use binary search to compute)

Table 6.1: Overview of the different ζ and $N_\wedge$ we obtain, where $\delta_T = \frac{\delta}{\lvert S\rvert^2\lvert A\rvert^2}$ and $\tilde{\rho} = -\rho(\pi_\odot, \tilde{M}) + \rho(\pi_b, \tilde{M})$ is the difference in performance between optimal and behavior policy on the MLE-MDP. $(\star)$ Standard SPI requires at least $N_\wedge^{\text{SPI}}$ samples in *all* state-action pairs.

then finish this proof in the same fashion as the proof of Theorem 6.19 and Lemma 6.17 but use the above-mentioned bound for the L_1-error instead of the bound obtained in (Weissman et al., 2003). □

6

There is no closed formula to directly compute $N_\wedge^\beta$ for a given ζ^β. However, for a given admissible performance loss ζ, we can perform a binary search to obtain the smallest natural number $N_\wedge^\beta$ such that $\zeta^\beta \leq \zeta$ given in Theorem 6.24.

6.4.5 Comparison of Different Minimal Sample Thresholds $N_\wedge$

In the context of SPI, finding an $N_\wedge$ that is as small as possible while still guaranteeing ζ-approximate improvement is the main objective. An overview of the different ζ and $N_\wedge$ that are available is given in Table 6.1. Comparing the equations for different $N_\wedge$, we immediately see that $N_\wedge^{2\text{s}} \leq N_\wedge^{\text{SPIBB}}$ if and only if $2^{\lvert S\rvert} \geq 4\lvert S\rvert\lvert A\rvert$. This means the only MDPs where standard SPIBB outperforms our 2sMDP approaches are environments with a small state space but a large action space. By Lemma 6.21, we have that the error term $e(s,a)$ used to compute ζ^β is minimal in the 2sMDP, and in particular, it is smaller than the error term used to compute $\zeta^{2\text{s}}$. Thus we always have $N_\wedge^\beta \leq N_\wedge^{2\text{s}}$. In case $2^{\lvert S\rvert} < 4\lvert S\rvert\lvert A\rvert$ it is also possible to compute both $N_\wedge^{\text{SPIBB}}$, and $N_\wedge^\beta$ and simply choose the smaller one.

6.5 Experimental Evaluation

We experimentally evaluate the theoretical results presented in this chapter, stating with SPI in POMDPs, Section 6.3, and moving to tighter improvement bounds for SPI, Section 6.4 after.

6.5.1 SPI in Partially Observable Environments

We now empirically evaluate our first contribution, extending SPIBB to partially observable environments, as detailed in Section 6.3. We first describe the setup of the experiments and then present and analyze the results.

Setup

Research questions. We pose the following research questions to evaluate our first contribution to SPI.

RQ1 Can our approach show empirical improvement in the policy, even in environments that do not satisfy Assumption 6.5?

RQ2 Does the FSC memory size affect the improvement?

Environments. We consider three POMDP environments:

i. CheeseMaze (McCallum, 1993): An agent navigates a maze, moving in the four cardinal directions, but in each state, it only perceives whether or not there is a barrier in each direction. The agent is placed at a random location at the beginning of an episode and receives a positive reward (+1) if it reaches the goal and a small negative reward (−0.01) otherwise. The episode ends when the agent reaches the goal.

ii. Tiger (Kaelbling et al., 1998): An agent is in front of two doors, and a tiger is randomly positioned behind one of them at the beginning of each episode. The agent has three actions: Listening or opening one of the doors. Listening gives a noisy observation of the position of the tiger and a small negative reward (−1). Opening the door with the tiger gives a large negative reward (−100), while opening the other door gives a positive reward (+10).

iii. Voicemail (Williams and Young, 2007): An agent controls a voicemail machine. At the beginning of the episode, the user listens to a message and decides if they want to keep it. This information is hidden from the agent, which has three actions: *ask*, *save*, and *delete*. Asking the user if they want to keep the message gives the agent a small negative reward (−1) and a noisy observation of the user's intention. Correctly saving the message gives a positive reward (+5), and a negative reward (−10) otherwise. Correctly deleting the message gives a positive reward (+5), and a negative reward (−20) otherwise.

All experiments use a discount factor $\gamma = 0.95$, with at most 300 steps in an episode.

Satisfaction of Assumption 6.5. Note that the Maze environment is close to satisfying Assumption 6.5 for memory that looks back two steps, *i.e.*, $k = 2$, with the exceptions of histories with equal observations. Tiger and Voicemail do not satisfy the assumption for any k.

		CheeseMaze	Tiger	Voicemail
Initial exploration rate	ϵ_0	0.500	0.500	0.500
Initial learning rate	α_0	1.000	1.000	1.000
Decay rate	λ	0.002	0.002	0.002
Softmax temperature	τ	15.000	0.050	0.300

Table 6.2: Hyperparameters to generate the behavior policies.

Data collection. We generate behavior policies via Q-learning using the memory of an FSC that keeps track of the last $k \in \{1, 2\}$ observations as the state. After convergence, we extract a softmax policy to ensure we sample different actions during data collection. We consider datasets of different sizes, namely: 1, 2, 5, 10, 20, 50, $\cdots$, 5 000, and 10 000 trajectories, and generate 500 datasets for each environment, number of trajectories, and behavior policy.

Training the behavior policies. To train the behavior policy, we use Q-learning over 5 000 episodes with learning rate α and exploration rate ϵ decaying exponentially after each episode:

$$\alpha_i = \alpha_0 \exp(-\lambda * i), \qquad \epsilon_i = \epsilon_0 \exp(-\lambda * i),$$

where i is the episode index, λ is the decay rate, α_0 and ϵ_0 are the initial learning rate and initial exploration rate, respectively. After training, we extract the behavior policy π_b using a softmax function.

$$\pi_b(a \mid s) = \frac{e^{-\tau Q(s,a)}}{\sum_{a' \in A} e^{-\tau Q(s,a')}},$$

where $Q(s, a)$ is the final value of the state-action pair s, a and τ a hyperparameter to control the randomness of the behavior policy. See Table 6.2 for all hyperparameters per environment.

Learning algorithms. We consider two algorithms to compute a new policy: *SPIBB*, and *Basic RL*. Both algorithms operate on the finite-history MLE-MDP (Definition 6.9) related to the finite-history MDP of the POMDP. We implement Basic RL as an unconstrained SPIBB where $N_\wedge = 0$; that is, it solves the MLE-MDP using value iteration. For each dataset, we compute new policies π_I using each offline RL algorithm, considering different hyperparameters: $N_\wedge \in \{5, 7, 10, 15, 20, 30, 50, 70, 100\}$ and $k' \in \{k, k+1\}$, where k' is the history size encoded in the FSC of π_I.

Evaluation metrics. Each policy is evaluated over 10 000 episodes to estimate the performance of the improved policy $\rho(\pi_I, M^*)$. We also consider the normalized policy improvement:

$$\bar{\rho}(\pi_I) = \frac{\rho(\pi_I, M^*) - \rho(\pi_b, M^*)}{\rho(\pi_{\max}, M^*) - \rho(\pi_b, M^*)},$$

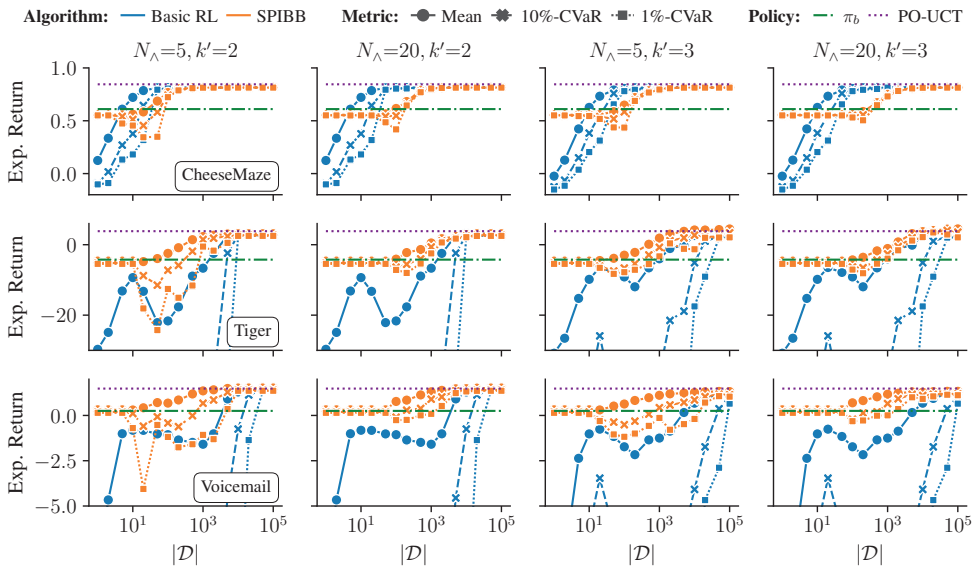


Figure 6.4: Policy improvement on the environments Maze, Tiger, and Voicemail (first, second and third row, respectively) for datasets collected by a behavior policy with history size $k = 2$, varying the hyperparameters pairs column-wise: $(N_\wedge = 5, k' = 2)$, $(N_\wedge = 20, k' = 2)$, $(N_\wedge = 5, k' = 3)$, and $(N_\wedge = 20, k' = 3)$. The plots show the mean (solid line), 10%-CVaR (dashed line), and 1%-CVaR (dotted line). The performance of the behavior policy is shown in green (dash-dotted line).

where $\pi_{\max}$ is the policy with the highest expected return in each environment. To aggregate the results across the 500 repetitions, we compute the mean and Conditional Value at Risk (CVaR; Rockafellar and Uryasev, 2000). We use x%-CVaR to indicate the mean of the x% lowest performances. To approximate the optimal value, we show the performance of PO-UCT (Silver and Veness, 2010), which uses the environment as a simulator to compute a policy.

Computing specifications. All experiments were performed on a 4GHz Intel Core i9 CPU and 64Gb of memory, using a single core for each experiment.

Results and Discussion

Figure 6.4 shows results on the three environments (ordered by row). The data was collected using a behavior policy with $k = 2$. The first column shows the results where SPIBB uses a low threshold to consider a history-action pair known and the same memory size as the behavior policy ($N_\wedge = 5$ and $k' = 2$). The second column shows the results with a higher threshold ($N_\wedge = 20$ and $k' = 2$). The third column shows the results for increased memory ($N_\wedge = 5$ and $k' = 3$). Finally, the fourth column shows the results with a higher threshold and increased memory ($N_\wedge = 20$

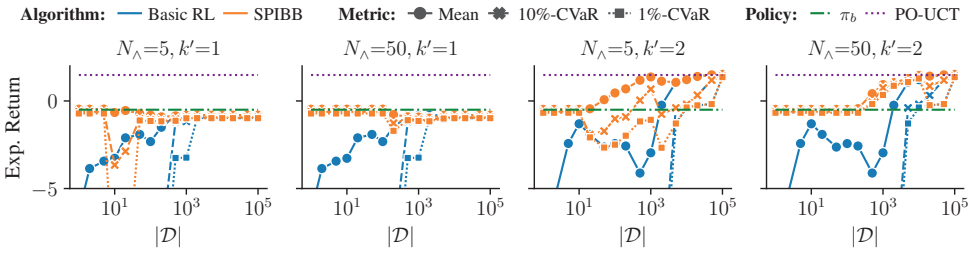


Figure 6.5: Policy improvement on the Voicemail environment for datasets collected with a memoryless policy ($k = 1$), varying the hyperparameters pairs column-wise ($N_\wedge = 5, k' = 1$), ($N_\wedge = 50, k' = 1$), ($N_\wedge = 5, k' = 2$), and ($N_\wedge = 50, k' = 2$). The plots show the mean (solid line), 10%-CVaR (dashed line), and 1%-CVaR (dotted line). The performance of the behavior policy is shown in green (dash-dotted line).

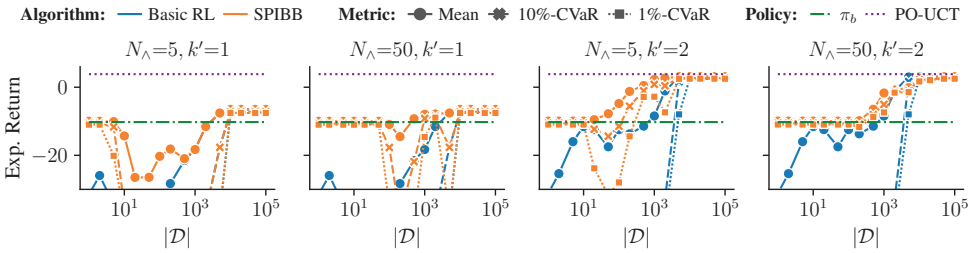


Figure 6.6: Policy improvement on the Tiger environment for datasets collected with a memoryless policy ($k = 1$), varying the hyperparameters pairs column-wise: ($N_\wedge = 5, k' = 1$), ($N_\wedge = 50, k' = 1$), ($N_\wedge = 5, k' = 2$), and ($N_\wedge = 50, k' = 2$). The plots show the mean (solid line), 10%-CVaR (dashed line), and 1%-CVaR (dotted line). The performance of the behavior policy is shown in green (dash-dotted line).

and $k' = 3$). Basic RL is included everywhere to give a perspective on the influence of different hyperparameters.

Figures 6.5 and 6.7 extend the empirical analysis on the Voicemail and Tiger environments for memoryless behavior policy ($k = 1$), since they demonstrated to be more challenging for the safe policy improvement problem. Figure 6.5 considers the Voicemail environment, while Figures 6.7 to 6.11 show the normalized results for a range of thresholds and different memory sizes in the three environments.

Basic RL is unreliable. Across all environments, the Basic RL algorithm shows a considerable performance drop compared to the behavior policy, even in terms of the mean performance for smaller datasets. Notice that for Tiger and Voicemail, the CVaR metrics are often outside the graph.

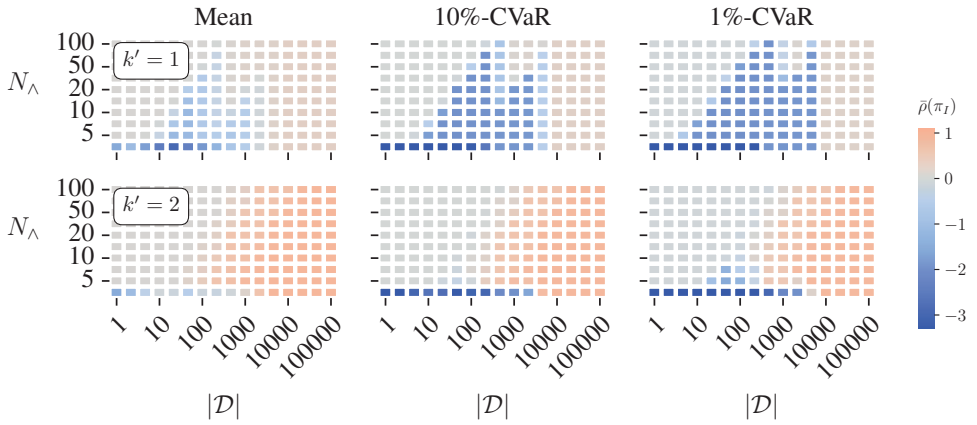


Figure 6.7: Normalized performance $\tilde{\rho}(\pi_I)$ on the Tiger environment ($k = 1$). The left, middle, and right columns show the mean, 10%-CVaR and 1%-CVaR, respectively. The first row shows the results where the improved policy uses the same memory as the behavior policy ($k' = k$), while the second row shows the results for an improved policy with more memory ($k' = k + 1$).

SPIBB outperforms Basic RL. In the environments Tiger and Voicemail (Figure 6.4), the SPIBB algorithm shows better performance than the Basic RL across all dataset sizes. This is likely due to the SPIBB algorithm retaining the randomization of the behavior policy when insufficient data is available.

SPIBB is reliable when Assumption 6.5 is satisfied. Analyzing the results for the Maze environment (Figure 6.4, first row), we observe that SPIBB shows reliably outperforms the behavior policy even for a small $N_\wedge$ (first column), for which only the 1%-CVaR shows a performance drop.

More memory improves the reliability. SPIBB shows slightly unreliable behavior for small values of $N_\wedge$ in the Tiger and Voicemail environments (Figure 6.4), as evidenced by both the CVaR curves, which can be alleviated by increasing the $N_\wedge$ or the memory of the new policy (second, third and fourth column). When Assumption 6.5 is violated, the performance drop may be significant, as seen in the first two columns of Figure 6.5. In this case, merely increasing the $N_\wedge$ threshold is insufficient to guarantee a policy improvement. Increasing the memory size, however, allows the SPIBB algorithm to improve the behavior policy, as Figure 6.5 (last column) and Figure 6.7 (second row) show. Similar results are observed in Figures 6.8 to 6.11. We thus positively answer RQ1 and RQ2: under the right circumstances, our approach can show empirical improvements, even when Assumption 6.5 is not satisfied, and increasing the FSC memory size positively affects the improvements made over the behavior policy.

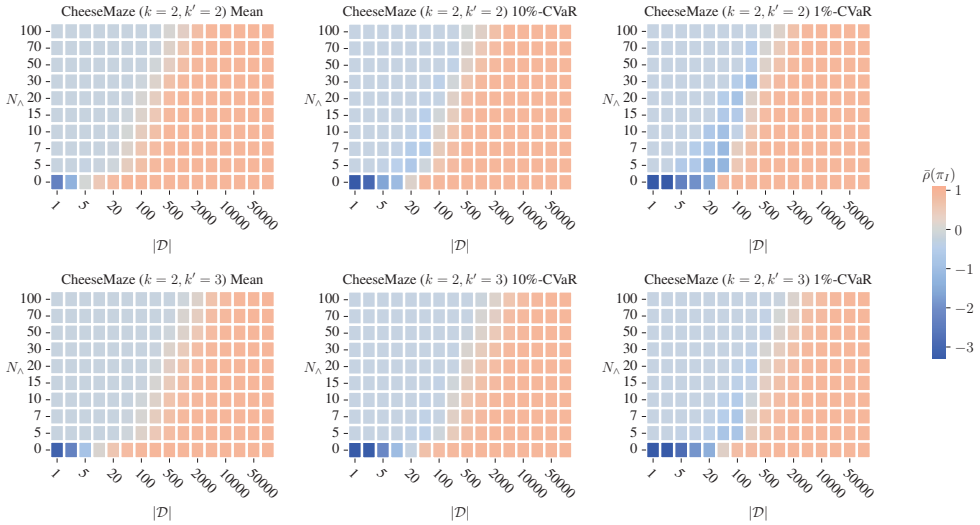


Figure 6.8: Normalized results ($\bar{\rho}(\pi_I)$) on the Maze environment ($k = 2$). The left, middle, and right columns show the mean, 10%-CVaR and 1%-CVaR, respectively. The first row shows the results where the improved policy uses the same memory as the behavior policy ($k' = k$), while the second row shows the results for an improved policy with more memory ($k' = k + 1$).

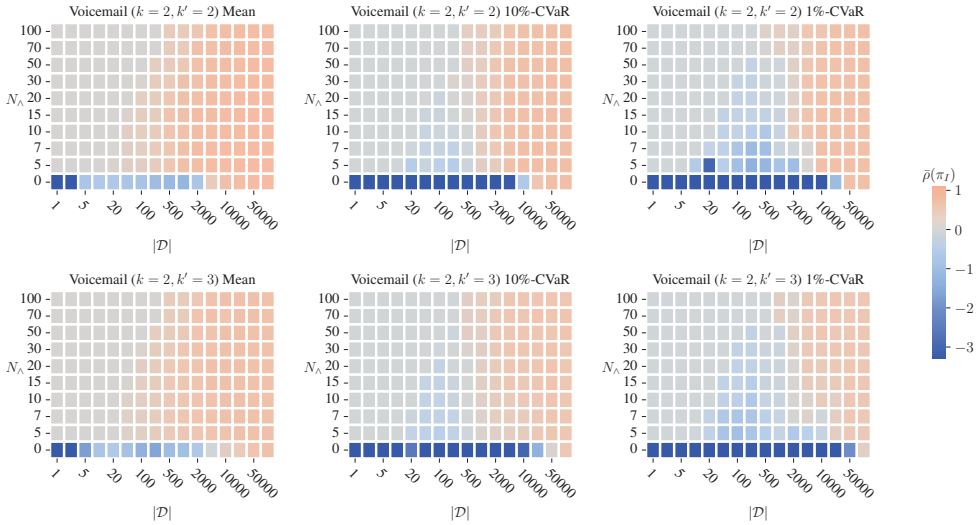


Figure 6.9: Normalized result on the Voicemail environment ($k = 2$). The left, middle, and right columns show the mean, 10%-CVaR and 1%-CVaR, respectively. The first row shows the results where the improved policy uses the same memory as the behavior policy ($k' = k$), while the second row shows the results for an improved policy with more memory ($k' = k + 1$).

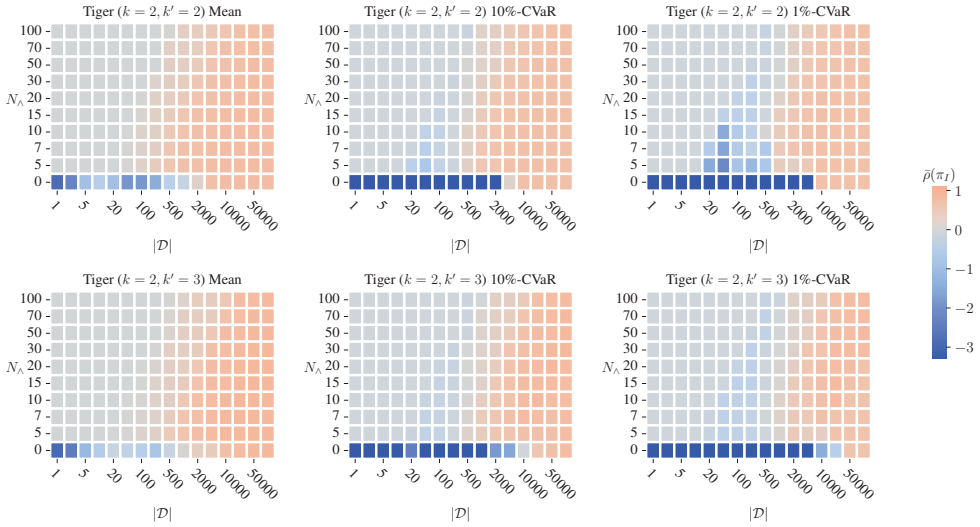


Figure 6.10: Normalized results ($\bar{\rho}(\pi_I)$) on the Tiger environment ($k = 2$). The left, middle, and right columns show the mean, 10%-CVaR and 1%-CVaR, respectively. The first row shows the results where the improved policy uses the same memory as the behavior policy ($k' = k$), while the second row shows the results for an improved policy with more memory ($k' = k + 1$).

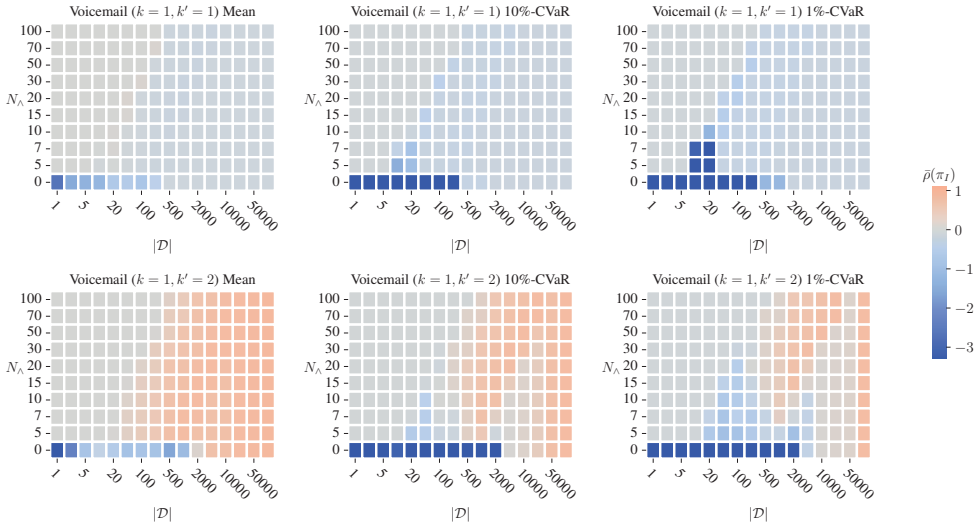
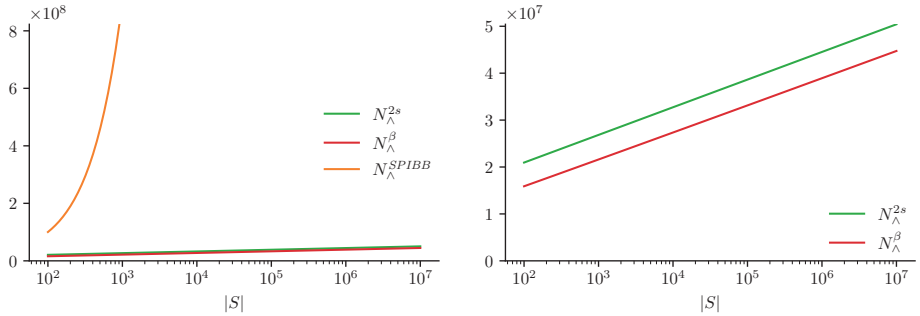


Figure 6.11: Normalized results ($\bar{\rho}(\pi_I)$) on the Voicemail environment ($k = 1$). The left, middle, and right columns show the mean, 10%-CVaR and 1%-CVaR, respectively. The first row shows the results where the improved policy uses the same memory as the behavior policy ($k' = k$), while the second row shows the results for an improved policy with more memory ($k' = k + 1$).



(a) $N_\wedge^{\text{SPIBB}}, N_\wedge^{2s}$ and $N_\wedge^{\beta}$. (b) $N_\wedge^{2s}$ and $N_\wedge^{\beta}$.

Figure 6.12: Required number of samples for different $|S|$ with $|A| = 4, V_{max} = 1, \gamma = 0.95, \delta = 0.1$ and $\zeta = 0.1$.

Deterministic policies may require more memory. Figure 6.5 shows an interesting phenomenon. In partially observable settings, the stochastic behavior policy might perform better than the new deterministic policy since randomization can trade-off some amount of memory. We observe that when $k = 1$, SPIBB and Basic RL converge to deterministic policies with an expected return lower than the behavior policy. When SPIBB has sufficient data, it is not constrained to follow the behavior policy and thus does not inherit any randomization from that policy. As stated in the previous paragraph, more memory can, in that case, yield a new deterministic policy with a higher performance than the behavior policy.

6

6.5.2 SPI With Stronger Performance Guarantees

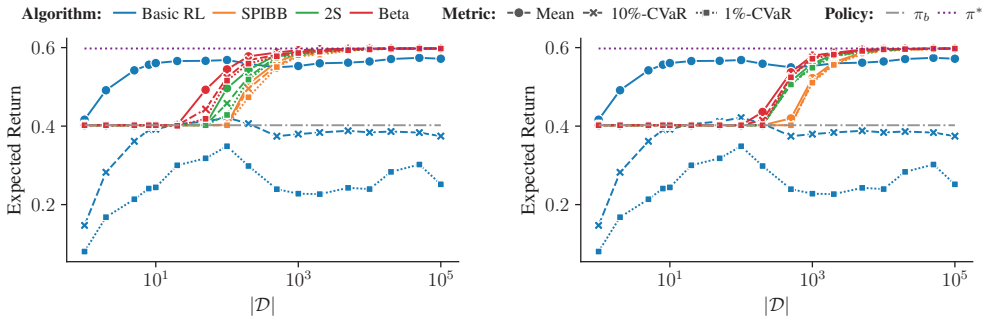
We now empirically evaluate our second contribution of tighter improvement guarantees to the SPI problem, as detailed in Section 6.4. We provide an evaluation of our approach from two different perspectives. First, a theoretical evaluation of how the different $N_\wedge$ depend on the size of a hypothetical MDP, and second, a practical evaluation to investigate how smaller $N_\wedge$ values translate to the performance of the improved policies.

Setup

Reserach questions. We pose the following two research questions to evaluate our second contribution to SPI.

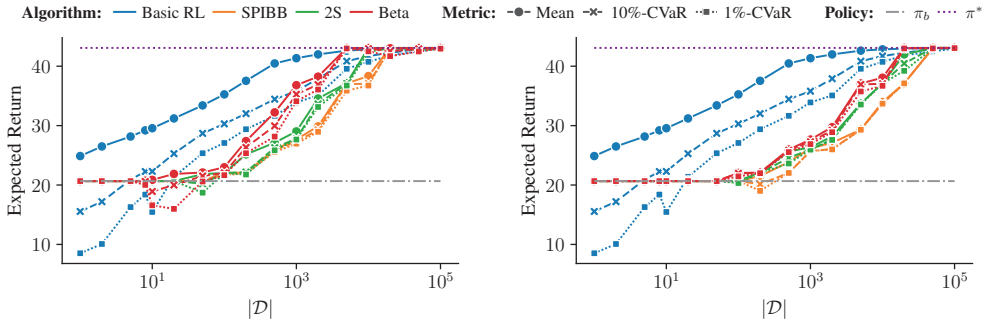
RQ3 How does the size of the state space of the underlying MDP affect the values for $N_\wedge^{SPIBB}$, $N_\wedge^{2s}$, and $N_\wedge^{\beta}$?

RQ4 Do the new theoretical bounds for smaller $N_\wedge$ values also translate to the performance of the improved policies?



(a) $N_\wedge^{\text{SPIBB}} = 100, N_\wedge^{2s} = 55, N_\wedge^{\beta} = 27.$ (b) $N_\wedge^{\text{SPIBB}} = 400, N_\wedge^{2s} = 219, N_\wedge^{\beta} = 146.$

Figure 6.13: Safe policy improvement on the Gridworld environment.



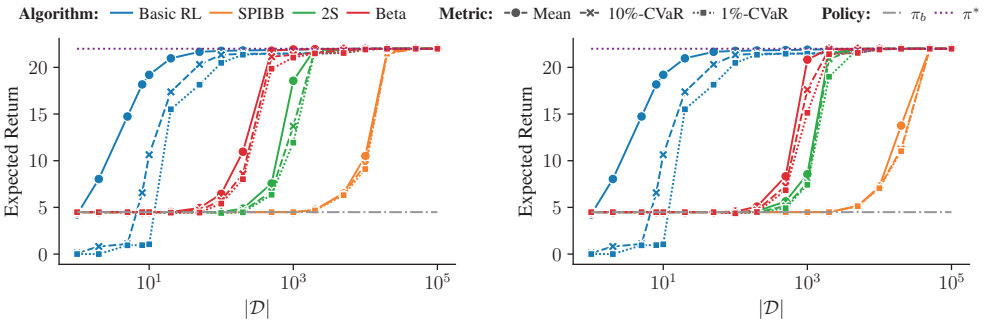
(a) $N_\wedge^{\text{SPIBB}} = 60, N_\wedge^{2s} = 34, N_\wedge^{\beta} = 10.$ (b) $N_\wedge^{\text{SPIBB}} = 180, N_\wedge^{2s} = 101, N_\wedge^{\beta} = 61.$

Figure 6.14: Safe policy improvement on the Wet Chicken environment.

Example Comparison of Different $N_\wedge$

To render the theoretical differences between the possible $N_\wedge$ discussed at the end of Section 6.4.3 more tangible and answer RQ3, we now give a concrete example.

We assume a hypothetical MDP with $|A| = 4$, $V_{max} = 1$, $\gamma = 0.95$, and SPIBB parameters $\delta = 0.1$ and $\zeta = 0.1$. For varying sizes of the state space, we compute all three sample size constraints: $N_\wedge^{\text{SPIBB}}$, $N_\wedge^{2s}$, and $N_\wedge^{\beta}$. The results are shown in Figure 6.12, where Figure 6.12a shows the full plot and Figure 6.12b provides an excerpt to differentiate between the $N_\wedge^{2s}$ and $N_\wedge^{\beta}$ plots by scaling down the y-axis. Note that the x-axis, the number of states in our hypothetical MDP, is on a log scale. We see that $N_\wedge^{\text{SPIBB}}$ grows linearly with the number of states, whereas $N_\wedge^{2s}$ and $N_\wedge^{\beta}$ are logarithmic in the number of states. Further, we note that $N_\wedge^{\beta}$ is significantly below $N_\wedge^{2s}$, which follows from Lemma 6.17. Finally, the difference between $N_\wedge^{\text{SPIBB}}$ and $N_\wedge^{2s}$ is for small MDPs of around a hundred states already a factor 10.



(a) $N_\wedge^{\text{SPIBB}} = 600, N_\wedge^{2s} = 43, N_\wedge^{\beta} = 12$.

(b) $N_\wedge^{\text{SPIBB}} = 1000, N_\wedge^{2s} = 71, N_\wedge^{\beta} = 37$.

Figure 6.15: Safe policy improvement on the Resource Gathering environment.

Discussion. While we show that a significant reduction of the required number of samples per state-action pair $N_\wedge$ is possible via our two approaches, we note that even for small MDPs (*e.g.*, $|S| = 100$) we still need over 10 million samples per state-action pair to guarantee that an improved policy is safe with respect to the behavior policy. That is, with probability $1 - \delta = 0.9$, an improved policy will have an admissible performance loss of at most $\zeta = 0.1$, which is infeasible in practice. Nevertheless, a practical evaluation of our approaches is possible by taking on a different perspective, which we will address next.

6

Evaluation in SPIBB

To answer RQ4, we integrate our novel results for computing $\zeta^{2s}, \zeta^{\beta}, N_\wedge^{2s}$, and $N_\wedge^{\beta}$ into the implementation of SPIBB (Laroche et al., 2019).

Environments. We consider two standard benchmarks used in SPI and one other well-known MDP: the 25-state *Gridworld* proposed by Laroche et al. (2019), the 25-state *Wet Chicken* benchmark (Hans and Udluft, 2009), which was used to evaluate SPI approaches by Scholl et al. (2022), and a 376-state instance of *Resource Gathering* proposed by Barrett and Narayanan (2008).

Behavior policies. For the Gridworld, we use the same behavior policy as (Laroche et al., 2019). For the Wet Chicken environment, we use Q-Learning with a softmax function to derive a behavior policy. The behavior policy of Resource Gathering was derived from the optimal policy by selecting each non-optimal action with a probability of 1e-5.

Methodology. Recall that in the standard SPIBB approach, $N_\wedge$ is used as a hyperparameter since the actual $N_\wedge$ for reasonable δ and ζ are infeasible. While our methods improve significantly on $N_\wedge$, the values we obtain are still infeasible in practice, as discussed in Section 6.5.2. We still use $N_\wedge^{\text{SPIBB}}$ as a hyperparameter,

and then run the SPIBB algorithm and compute the resulting ζ^{SPIBB}. This ζ^{SPIBB} is consequently used to compute the values $N_\wedge^{\text{2s}}$ and $N_\wedge^{\beta}$ that ensure the same performance loss. We then run SPIBB again with these two values for $N_\wedge$. As seen in the previous experiment, and detailed at the end of Section 6.4.3, for most MDPs – including our examples – we have $N_\wedge^{\beta} \leq N_\wedge^{\text{2s}} \leq N_\wedge^{\text{SPIBB}}$ for a fixed ζ.

Evaluation metrics. For each dataset size, we repeat each experiment 1000 times and report the mean performance of the learned policy, as well as the 10% and 1% conditional value at risk (CVaR) values (Rockafellar and Uryasev, 2000), indicating the mean performance of the worst 10% and 1% runs. We also include the performance of basic RL (dynamic programming on the MLE-MDP (Sutton and Barto, 1998)), the behavior policy π_b, and the optimal policy π^* of the underlying MDP.

Results. We present the results for the Gridworld, Wet Chicken, and Resource Gathering environments in Figures 6.13 to 6.15 for three different hyperparameters $N_\wedge^{\text{SPIBB}}$, respectively. In all instances, we see similar and improved behaviors as we presumed by sharpening the sampling bounds with our new approaches. Smaller values for $N_\wedge$ typically require smaller datasets for a policy to start improving, and this is precisely what our methods set out to do. In particular, we note that our methods (2S and Beta) are quicker to converge to an optimal policy than standard SPIBB. Beta is, as expected, the fastest and has started to improve over the behavior policy for datasets, about half the size compared to SPIBB in Gridworld. Further, while theoretically, the factor between the different $N_\wedge$ does not directly translate to the whole dataset size, we see that in practice on all three benchmarks, this is roughly the case. Finally, we note that Basic RL is unreliable compared to the SPI methods, as seen by the CVaR values being significantly below the baseline performance for several dataset sizes in all three environments. This is as expected and in accordance with well-established results.

6.6 Conclusion

This chapter presented two key contributions to the offline RL problem of safe policy improvement (SPI). First, we presented a new approach to SPI in partially observable environments modeled as POMDPs by relying on a reduction to finite-history MDPs. Our experiments show the applicability of our approach, even in cases where finite-history is not sufficient to obtain optimal results.

Second, we presented a new approach to SPI that significantly reduces the sample size requirements of datasets. We derived new performance guarantees and applied them to the well-established SPIBB algorithm. Specifically, we introduced a novel transformation to the underlying MDP model that limits the branching factor and provided two new ways of computing the admissible performance loss ζ and the sample size constraint $N_\wedge$, both exploiting the limited branching factor in SPI(BB). This contribution improves the overall performance of SPI algorithms, leading to more efficient use of a given dataset.

6.6.1 Limitations and Discussion

The techniques presented in this chapter rely on some assumptions that impose certain limitations. We discuss each of these limitations individually below.

Finite history MDP assumption. Our first contribution, extending SPIBB to work in POMDPs, relies on the assumption that the POMDP is equivalent to a finite-history MDP, see Assumption 6.5 and Definition 6.6. This assumption poses a strong limitation on this contribution, even though we empirically established that it is still possible to apply SPIBB in practice without it. Natural directions for future work are ways to relax this assumption. One approach would be to consider (bisimulation) distance metrics (Ferns et al., 2004, 2005; García and Fernández, 2015) instead of the exact bisimulation we currently use. We conjecture that assuming a bound on such a distance could be included in the performance guarantees of SPI methods such as SPIBB. Other methods to relax the assumption could be to consider other subclasses of POMDPs such as *regular decision processes* (Brafman and Giacomo, 2024), or abstraction techniques for RL as presented in, *e.g.*, (Starre et al., 2023).

Practical applicability of SPIBB. Our second contribution does not rely on any assumptions other than those made in standard SPIBB. Yet, the overall applicability of SPIBB, even with our newly contributed performance guarantees, remains limited. While we reduce the required number of samples per state-action pair required compared to standard SPIBB, data efficiency of offline RL with reliability guarantees remains a fundamental problem.

Future work. There are a few other directions for future work besides the ones already mentioned. SPI in partially observable environments could benefit from adaptively learning a memory structure as done in, *e.g.*, model-free RL for POMDPs (McCallum, 1995). Our second contribution, on a more abstract level, is that we showed that even in a setting where the underlying model is unknown, it is still possible to reason over the structure of that model and the data generated from it. Hence, a clear direction for future work is the application of the techniques presented to other learning settings such as online model-based RL (Jaksch et al., 2010; Moerland et al., 2023; Moos et al., 2022), data-driven abstractions (Badings et al., 2023b), or statistical model checking (Ashok et al., 2019). The latter has already been considered in recent work (Meggendorfer et al., 2024).

7

Conclusion and Outlook

This thesis presented several contributions towards making decision-making under uncertainty more robust and reliable, each in their own way and highlight the many facets involved. Nonetheless, they all share the same throughline: robustness and reliability can be ensured by using model-based approaches that explicitly account for uncertainty. We now summarize our contributions and their potential impact.

A Tutorial on Robust Markov Decision Processes. In Chapter 3, we presented a short tutorial on robust MDP theory and robust dynamic programming. RMDPs form the backbone of (robust) RL and other applications such as statistical model checking. Hence, making RMDPs accessible to a wider audience is important for both the formal methods and AI communities.

Finite-Memory Policies for Robust POMDPs. RPOMDPs are the extension of RMDPs with partial observability. While POMDPs themselves are well-studied and many scalable planning algorithms exist, that is not the case for RPOMDPs. In Chapter 4, we contributed two algorithms based on convex optimization to compute finite-memory policies in robust POMDPs. Given the extensive use of RMDPs in (robust) reinforcement learning, having access to efficient planning algorithms for RPOMDPs is the first step towards developing RL methods under partial observability.

Robust Anytime Learning of Markov Decision Processes. In Chapter 5, we presented a new approach to robust reinforcement learning in MDPs where the underlying environments may change over time. Robustness against changing environments is a key challenge in RL, and our contribution addresses this challenge by combining a sliding window approach with linearly updating intervals, a Bayesian scheme that updates prior intervals given observations. This combination outperforms existing approaches that only use a sliding window.

Extending the Scope of Reliable Offline RL. Finally, in Chapter 6, we presented our last two contributions: An extension to the safe policy improvement with baseline bootstrapping (SPIBB) algorithm to compute finite-memory policies from datasets collected in partially observable environments and new techniques that provide stronger improvement guarantees given the same amount of data in safe policy improvement algorithms, such as SPIBB. The former brings offline RL with reliability guarantees to partially observable environments, where policies usually depend on a sufficient amount of memory to yield an acceptable performance. The later contribution reduces the sample size requirements to achieve the same reliability guarantees. This contribution is of interest in a broader context as a natural direction to investigate whether such techniques can be applied to other RL and statistical model checking settings to reduce sample complexities.

7.1 Directions for Future Work

To conclude, we present three general directions for future work, going beyond the possible directions already mentioned at the end of each chapter. These directions directly build on top of several of the contributions presented in this thesis, highlighting their potential impact.

Robust POMDP Theory and Algorithms

In Chapter 4, we presented two algorithms to compute finite-memory policies in RPOMDPs under several structural and semantic assumptions. Most notably, we assumed *fully stickiness*, *i.e.*, static uncertainty semantics, where nature chooses (non-deterministically) a complete transition model $P \in \mathcal{P}$ at the start. Our algorithms are, to the best of our knowledge, the first to tackle robust planning in RPOMDPs under these semantics. Conversely, other existing algorithms for RPOMDPs, such as the value iteration-based approaches of Osogami (2015), all assume *zero stickiness*, *i.e.*, dynamic uncertainty semantics, and that the agent plays first. See (Bovy et al., 2024) for a full classification of existing RPOMDP algorithms. A natural question that arises is whether any of these algorithms can be adapted to work under different RPOMDP semantics. Second, all existing RPOMDP methods assume (s,a)-rectangularity. New approaches to s-rectangular uncertainty sets would further expand the applicability of RPOMDPs.

Our algorithms, based on convex optimization, may be interpreted as methods that search the space of (fixed size) finite-memory policies. Integration with learning methods and alternative policy representations, such as recurrent neural networks (RNNs), may provide a feasible alternative to finding robust policies. This direction has proven successful for standard POMDPs already (Carr et al., 2019, 2021).

Theoretically, the partially observable stochastic game (POSG) semantics of an RPOMDP, as defined by Bovy et al. (2024), raise several new questions. Complexity theoretically, RPOMDPs are at least as hard as standard POMDPs, but whether an additional jump in complexity classes occurs here too, for instance between policy evaluation in MDPs and RMDPs (Table 3.1, Chapter 3) is still open. Additionally, as shown by Bovy et al. (2024), a Nash equilibrium must exist in the semantical game

for an optimal robust policy to exist in the RPOMDP. The existence of such a Nash equilibrium has only been shown for finite horizon reward maximization and is thus an open problem for other objectives. Finally, on the algorithmic side, it would be interesting to see whether algorithms for POSGs can be applied to solve RPOMDPs in practice and, conversely, whether RPOMDP algorithms can be adapted to solve (subcases of) POSGs.

Robust RL in Partially Observable Environments

In Chapter 5, we presented an RL algorithm that is robust against changing environments, while in Chapter 6, we presented an approach to deal with partially observable environments in the offline setting of safe policy improvement. There, we presented changing environments and partial observability as two separate concerns. Yet, they may be closely related.

A direction worthwhile to investigate would be to assume some form of a known model of the environment change, as also discussed in the conclusion of Chapter 5. Alternatively, a change in distributions may be caused by partial observability. When the states consist of several features, *i.e.*, may be factorized into a Cartesian product of sets, one of these features may not be observable. Yet, the transition function, and thus the observed samples, depends on all features. In such a setting, being robust against changes in the environment corresponds to being robust against partial observability. The need for memory in POMDPs is well-established, as also seen in the experimental evaluation of Chapter 6. Hence, when the change of environments is caused by unobserved features, learning (robust) finite-memory policies may already be sufficient to ensure robustness and reliability against such changes.

RL in partially observable environments is, of course, also a topic of interest in its own right. Robust RL in fully observable environments often relies on robust planning methods being applied to an estimated model from data. Our contributions to computing finite-memory policies for POMDPs in Chapter 4 may thus be of use to build model-based RL methods for RPOMDPs.

Abstraction and Approximation of POMDPs

Finally, we envision a direction relevant to both planning and RL in POMDPs. POMDPs are notoriously difficult to solve, as evidenced by many computability and complexity results (Baier et al., 2008; Madani et al., 2003; Papadimitriou and Tsitsiklis, 1987). Yet, many real-world problems are structured in ways that the general POMDP framework does not account for. Identifying more tractable subclasses of POMDPs, such as MEMDPs (Raskin and Sankur, 2014) or regular decision processes (Brafman and Giacomo, 2024), provides models that are more tailored to specific problems.

When a problem does not fit such a POMDP subclass, it may still be used through abstraction. In particular, given an arbitrary POMDP, we could ask whether this POMDP is contained in a specific subclass, and if not, can we define some notion of distance between the POMDP and the closest tractable model? Such a distance could be used to bound approximation errors when abstracting the POMDP into

the subclass and transferring the results back to the original model. In essence, a rudimentary version, without any such distance or guarantees, is what we do in practice in the experimental evaluation of Chapter 6 where we consider POMDP environments that do not satisfy our finite-history MDP quotient assumption. As our results show, a certain amount of memory provides a sufficient abstraction of the underlying partially observable model.

7.2 Final Remarks

The contributions of this thesis rely on a strong combination of methods ranging across theoretical computer science, in particular formal methods and AI. Fundamental questions around robustness and reliability in decision-making under uncertainty, and AI in general, are too important to be left to a single scientific disciple. Multidisciplinary research into all aspects of robustness, reliability, safety, explainability, and ethics is paramount to ensure the just and ethical deployment of AI technology into our society.

Bibliography

Alessandro Abate, Maria Prandini, John Lygeros, and Shankar Sastry. Probabilistic Reachability and Safety for Controlled Discrete Time Stochastic Hybrid Systems. *Automatica*, 44(11):2724 – 2734, 2008.

Chaitanya Agarwal, Shibashis Guha, Jan Kretínský, and Pazhamalai Muruganandham. PAC Statistical Model Checking of Mean Payoff in Discrete- and Continuous-Time MDP. In *CAV (2)*, volume 13372 of *Lecture Notes in Computer Science*, pages 3–25. Springer, 2022.

Gul Agha and Karl Palmskog. A Survey of Statistical Model Checking. *ACM Trans. Model. Comput. Simul.*, 28(1):6:1–6:39, 2018.

Rajeev Alur, Thomas A. Henzinger, Gerardo Lafferriere, and George J. Pappas. Discrete Abstractions of Hybrid Systems. *Proceedings of the IEEE*, 88(7):971–984, 2000.

Christopher Amato, Daniel S. Bernstein, and Shlomo Zilberstein. Optimizing Fixed-Size Stochastic Controllers for POMDPs and Decentralized POMDPs. *Auton. Agents Multi Agent Syst.*, 21(3):293–320, 2010.

Charalampos P. Andriotis and Konstantinos G. Papakonstantinou. Deep Reinforcement Learning Driven Inspection and Maintenance Planning Under Incomplete Information and Constraints. *Reliab. Eng. Syst. Saf.*, 212:107551, 2021.

Mauricio Araya-López, Olivier Buffet, Vincent Thomas, and François Charpillet. Active Learning of MDP Models. In *EWRL*, volume 7188 of *LNCS*, pages 42–53. Springer, 2011.

Sebastian Arming, Ezio Bartocci, Krishnendu Chatterjee, Joost-Pieter Katoen, and Ana Sokolova. Parameter-Independent Strategies for pMDPs via POMDPs. In *QEST*, volume 11024 of *LNCS*, pages 53–70. Springer, 2018.

Pranav Ashok, Jan Kretínský, and Maximilian Weininger. PAC Statistical Model Checking for Markov Decision Processes and Stochastic Games. In *CAV (1)*, volume 11561 of *LNCS*, pages 497–519. Springer, 2019.

Dimitris Askitis. Logarithmic Concavity of the Inverse Incomplete Beta Function with Respect to the First Parameter. *Mathematica Scandinavica*, 127(1):111–130, 2021.

Karl Johan Åström. Optimal Control of Markov Processes with Incomplete State Information. *Journal of mathematical analysis and applications*, 10(1):174–205, 1965.

Thom S. Badings, Arnd Hartmanns, Nils Jansen, and Marnix Suilen. Balancing Wind and Batteries: Towards Predictive Verification of Smart Grids. In *NFM*, volume 12673 of *LNCS*, pages 1–18. Springer, 2021.

Thom S. Badings, Murat Cubuktepe, Nils Jansen, Sebastian Junges, Joost-Pieter Katoen, and Ufuk Topcu. Scenario-Based Verification of Uncertain Parametric MDPs. *Int. J. Softw. Tools Technol. Transf.*, 24(5):803–819, 2022a.

Thom S. Badings, Nils Jansen, Sebastian Junges, Mariëlle Stoelinga, and Matthias Volk. Sampling-Based Verification of CTMCs with Uncertain Rates. In *CAV (2)*, volume 13372 of *LNCS*, pages 26–47. Springer, 2022b.

Thom S. Badings, Licio Romao, Alessandro Abate, and Nils Jansen. Probabilities Are Not Enough: Formal Controller Synthesis for Stochastic Dynamical Models with Epistemic Uncertainty. In *AAAI*, pages 14701–14710. AAAI Press, 2023a.

Thom S. Badings, Licio Romao, Alessandro Abate, David Parker, Hasan A. Poonawala, Mariëlle Stoelinga, and Nils Jansen. Robust Control for Dynamical Systems with Non-Gaussian Noise via Formal Abstractions. *J. Artif. Intell. Res.*, 76:341–391, 2023b.

Thom S. Badings, Thiago D. Simão, Marnix Suilen, and Nils Jansen. Decision-Making Under Uncertainty: Beyond Probabilities. *Int. J. Softw. Tools Technol. Transf.*, 25(3):375–391, 2023c.

Arash Bahrammirzaee. A Comparative Survey of Artificial Intelligence Applications in Finance: Artificial Neural Networks, Expert System and Hybrid Intelligent Systems. *Neural Comput. Appl.*, 19(8):1165–1195, 2010.

Christel Baier and Joost-Pieter Katoen. *Principles of Model Checking*. MIT Press, 2008.

Christel Baier, Nathalie Bertrand, and Marcus Größer. On Decision Problems for Probabilistic Büchi Automata. In *FoSSaCS*, volume 4962 of *LNCS*, pages 287–301. Springer, 2008.

Christel Baier, Joachim Klein, Linda Leuschner, David Parker, and Sascha Wunderlich. Ensuring the Reliability of Your Model Checker: Interval Iteration for Markov Decision Processes. In *CAV (1)*, volume 10426 of *LNCS*, pages 160–180. Springer, 2017.

Christel Baier, Holger Hermanns, and Joost-Pieter Katoen. The 10, 000 Facets of MDP Model Checking. In *Computing and Software Science*, volume 10000 of *LNCS*, pages 420–451. Springer, 2019.

Christel Baier, Clemens Dubslaff, Patrick Wienhöft, and Stefan J. Kiebel. Strategy Synthesis in Markov Decision Processes Under Limited Sampling Access. In *NFM*, volume 13903 of *Lecture Notes in Computer Science*, pages 86–103. Springer, 2023.

Leon Barrett and Srini Narayanan. Learning All Optimal Policies with Multiple Criteria. In *ICML*, pages 41–47. ACM, 2008.

Nicole Bäuerle and Jonathan Ott. Markov Decision Processes with Average-Value-at-Risk criteria. *Math. Methods Oper. Res.*, 74(3):361–379, 2011.

Bahram Behzadian, Marek Petrik, and Chin Pang Ho. Fast Algorithms for L_∞-Constrained S-Rectangular Robust MDPs. In *NeurIPS*, pages 25982–25992, 2021.

Richard Bellman. A Markovian Decision Process. *Journal of mathematics and mechanics*, pages 679–684, 1957.

Aharon Ben-Tal and Arkadi Nemirovski. Robust Convex Optimization. *Mathematics of operations research*, 23(4):769–805, 1998.

Dimitri P. Bertsekas. *Dynamic Programming and Optimal Control, 3rd Edition*. Athena Scientific, 2005.

Dimitris Bertsimas, David B. Brown, and Constantine Caramanis. Theory and Applications of Robust Optimization. *SIAM Rev.*, 53(3):464–501, 2011.

Federico Bianchi, Edoardo Zorzi, Alberto Castellini, Thiago D. Simão, Matthijs T. J. Spaan, and Alessandro Farinelli. Scalable Safe Policy Improvement for Factored Multi-Agent MDPs. In *ICML*. OpenReview.net, 2024.

Andrea Bianco and Luca de Alfaro. Model Checking of Probabalistic and Nondeterministic Systems. In *FSTTCS*, volume 1026 of *LNCS*, pages 499–513. Springer, 1995.

Christopher M. Bishop. *Pattern Recognition and Machine Learning, 5th Edition*. Information science and statistics. Springer, 2007.

Blai Bonet. An Epsilon-Optimal Grid-Based Algorithm for Partially Observable Markov Decision Processes. In *ICML*, pages 51–58. Morgan Kaufmann, 2002.

Alexander Bork, Debraj Chakraborty, Kush Grover, Jan Kretínský, and Stefanie Mohr. Learning Explainable and Better Performing Representations of POMDP Strategies. In *TACAS (2)*, volume 14571 of *LNCS*, pages 299–319. Springer, 2024.

Eline M. Bovy, Marnix Suilen, Sebastian Junges, and Nils Jansen. Imprecise Probabilities Meet Partial Observability: Game Semantics for Robust POMDPs. In *IJCAI*, pages 6697–6706. ijcai.org, 2024.

Stephen Boyd and Lieven Vandenberghe. *Convex optimization*. Cambridge university press, 2004.

Ronen I. Brafman and Giuseppe De Giacomo. Regular Decision Processes. *Artif. Intell.*, 331:104113, 2024.

Tomás Brázdil, Krishnendu Chatterjee, Martin Chmelik, Vojtech Forejt, Jan Kretínský, Marta Z. Kwiatkowska, David Parker, and Mateusz Ujma. Verification of Markov Decision Processes Using Learning Algorithms. In *ATVA*, volume 8837 of *LNCS*, pages 98–114. Springer, 2014.

Tomás Brázdil, Krishnendu Chatterjee, Martin Chmelik, Vojtech Forejt, Jan Kretínský, Marta Kwiatkowska, Tobias Meggendorfer, David Parker, and Mateusz Ujma. Learning Algorithms for Verification of Markov Decision Processes. *CoRR*, abs/2403.09184, 2024.

Carlos E. Budde, Arnd Hartmanns, Tobias Meggendorfer, Maximilian Weininger, and Patrick Wienhöft. Sound Statistical Model Checking for Probabilities and Expected Rewards. *CoRR*, abs/2411.00559, 2024.

Brendan Burns and Oliver Brock. Sampling-Based Motion Planning With Sensing Uncertainty. In *ICRA*, pages 3313–3318. IEEE, 2007.

Marco C. Campi and Simone Garatti. The Exact Feasibility of Randomized Solutions of Uncertain Convex Programs. *SIAM J. Optim.*, 19(3):1211–1230, 2008.

Marco C. Campi, Algo Carè, and Simone Garatti. The Scenario Approach: A Tool at the Service of Data-Driven Decision Making. *Annu. Rev. Control.*, 52:1–17, 2021.

Steven Carr, Nils Jansen, Ralf Wimmer, Alexandru Constantin Serban, Bernd Becker, and Ufuk Topcu. Counterexample-Guided Strategy Improvement for POMDPs Using Recurrent Neural Networks. In *IJCAI*, pages 5532–5539. ijcai.org, 2019.

Steven Carr, Nils Jansen, and Ufuk Topcu. Task-Aware Verifiable RNN-Based Policies for Partially Observable Markov Decision Processes. *J. Artif. Intell. Res.*, 72:819–847, 2021.

Alberto Castellini, Federico Bianchi, Edoardo Zorzi, Thiago D. Simão, Alessandro Farinelli, and Matthijs T. J. Spaan. Scalable Safe Policy Improvement via Monte Carlo Tree Search. In *ICML*, volume 202 of *Proceedings of Machine Learning Research*, pages 3732–3756. PMLR, 2023.

Nathalie Cauchi and Alessandro Abate. StocHy: Automated Verification and Synthesis of Stochastic Processes. In *TACAS (2)*, volume 11428 of *LNCS*, pages 247–264. Springer, 2019.

Iadine Chadès, Tara G Martin, Samuel Nicol, Mark A Burgman, Hugh P Possingham, and Yvonne M Buckley. General Rules for Managing and Surveying Networks of Pests, Diseases, and Endangered Species. *Proceedings of the National Academy of Sciences*, 108(20):8323–8328, 2011.

Iadine Chadès, Josie Carwardine, Tara G. Martin, Samuel Nicol, Régis Sabbadin, and Olivier Buffet. MOMDPs: A Solution for Modelling Adaptive Management Problems. In *AAAI*, pages 267–273. AAAI Press, 2012.

Mahmoud El Chamie and Hala Mostafa. Robust Action Selection in Partially Observable Markov Decision Processes with Model Uncertainty. In *CDC*, pages 5586–5591. IEEE, 2018.

Yash Chandak, Scott M. Jordan, Georgios Theocharous, Martha White, and Philip S. Thomas. Towards Safe Policy Improvement for Non-Stationary MDPs. In *NeurIPS*, pages 9156–9168. Curran Associates, Inc., 2020.

Krishnendu Chatterjee, Laurent Doyen, and Thomas A. Henzinger. Qualitative Analysis of Partially-Observable Markov Decision Processes. In *MFCS*, volume 6281 of *LNCS*, pages 258–269. Springer, 2010.

Krishnendu Chatterjee, Martin Chmelik, and Jessica Davies. A Symbolic SAT-Based Algorithm for Almost-Sure Reachability with Small Strategies in POMDPs. In *AAAI*, pages 3225–3232. AAAI Press, 2016.

Krishnendu Chatterjee, Martin Chmelík, Deep Karkhanis, Petr Novotný, and Amélie Royer. Multiple-Environment Markov Decision Processes: Efficient Analysis and Applications. In *ICAPS*, pages 48–56. AAAI Press, 2020.

Krishnendu Chatterjee, Ehsan Kafshdar Goharshady, Mehrdad Karrabi, Petr Novotný, and Dorde Zikelic. Solving Long-run Average Reward Robust MDPs via Stochastic Games. *CoRR*, abs/2312.13912, 2023.

Taolue Chen, Tingting Han, and Marta Z. Kwiatkowska. On the Complexity of Model Checking Interval-Valued Discrete Time Markov Chains. *Inf. Process. Lett.*, 113(7):210–216, 2013.

Wang Chi Cheung, David Simchi-Levi, and Ruihao Zhu. Reinforcement Learning for Non-Stationary Markov Decision Processes: The Blessing of (More) Optimism. In *ICML*, volume 119 of *Proceedings of Machine Learning Research*, pages 1843–1854. PMLR, 2020.

Edmund M. Clarke and E. Allen Emerson. Design and Synthesis of Synchronization Skeletons Using Branching-Time Temporal Logic. In *Logic of Programs*, volume 131 of *LNCS*, pages 52–71. Springer, 1981.

Edmund M. Clarke, William Klieber, Milos Novácek, and Paolo Zuliani. Model Checking and the State Explosion Problem. In *LASER Summer School*, volume 7682 of *LNCS*, pages 1–30. Springer, 2011.

Rudi Coppola, Andrea Peruffo, Licio Romao, Alessandro Abate, and Manuel Mazo Jr. Data-driven Interval MDP for Robust Control Synthesis. *CoRR*, abs/2404.08344, 2024.

Clarissa Costen, Marc Rigter, Bruno Lacerda, and Nick Hawes. Planning with Hidden Parameter Polynomial MDPs. In *AAAI*, pages 11963–11971. AAAI Press, 2023.

Murat Cubuktepe, Nils Jansen, Sebastian Junges, Joost-Pieter Katoen, and Ufuk Topcu. Synthesis in pMDPs: A Tale of 1001 Parameters. In *ATVA*, volume 11138 of *LNCS*, pages 160–176. Springer, 2018.

Murat Cubuktepe, Nils Jansen, Sebastian Junges, Joost-Pieter Katoen, and Ufuk Topcu. Scenario-Based Verification of Uncertain MDPs. In *TACAS (1)*, volume 12078 of *LNCS*, pages 287–305. Springer, 2020.

Murat Cubuktepe, Nils Jansen, Sebastian Junges, Ahmadreza Marandi, Marnix Suilen, and Ufuk Topcu. Robust Finite-State Controllers for Uncertain POMDPs. In *AAAI*, pages 11792–11800. AAAI Press, 2021.

Murat Cubuktepe, Nils Jansen, Sebastian Junges, Joost-Pieter Katoen, and Ufuk Topcu. Convex Optimization for Parameter Synthesis in MDPs. *IEEE Trans. Autom. Control.*, 67(12):6333–6348, 2022.

Przemyslaw Daca, Thomas A. Henzinger, Jan Kretínský, and Tatjana Petrov. Faster Statistical Model Checking for Unbounded Temporal Properties. In *TACAS*, volume 9636 of *LNCS*, pages 112–129. Springer, 2016.

Josh C. D'aeth, Shubhechyya Ghosal, Fiona Grimm, David J. Haw, Esma Koca, Krystal Lau, Huikang Liu, Stefano Moret, Dheeya Rizmie, Peter C. Smith, Giovanni Forchini, Marisa Miraldo, and Wolfram Wiesemann. Optimal Hospital Care Scheduling During the SARS-CoV-2 Pandemic. *Manag. Sci.*, 69(10):5923–5947, 2023.

David Dalrymple, Joar Skalse, Yoshua Bengio, Stuart Russell, Max Tegmark, Sanjit Seshia, Steve Omohundro, Christian Szegedy, Ben Goldhaber, Nora Ammann, Alessandro Abate, Joe Halpern, Clark W. Barrett, Ding Zhao, Tan Zhi-Xuan, Jeannette Wing, and Joshua B. Tenenbaum. Towards Guaranteed Safe AI: A Framework for Ensuring Robust and Reliable AI Systems. *CoRR*, abs/2405.06624, 2024.

Luca de Alfaro. *Formal Verification of Probabilistic Systems*. PhD thesis, Stanford University, USA, 1997.

Christian Dehnert, Sebastian Junges, Nils Jansen, Florian Corzilius, Matthias Volk, Harold Bruintjes, Joost-Pieter Katoen, and Erika Ábrahám. PROPhESY: A PRObabilistic ParamEter SYnthesis Tool. In *CAV (1)*, volume 9206 of *LNCS*, pages 214–231. Springer, 2015.

Christian Dehnert, Sebastian Junges, Nils Jansen, Florian Corzilius, Matthias Volk, Joost-Pieter Katoen, Erika Ábrahám, and Harold Bruintjes. Parameter Synthesis for Probabilistic Systems. In *MBMV*, pages 72–74. Albert-Ludwigs-Universität Freiburg, 2016.

Christian Dehnert, Sebastian Junges, Joost-Pieter Katoen, and Matthias Volk. A Storm is Coming: A Modern Probabilistic Model Checker. In *CAV (2)*, volume 10427 of *LNCS*, pages 592–600. Springer, 2017.

Esther Derman, Daniel J. Mankowitz, Timothy A. Mann, and Shie Mannor. A Bayesian Approach to Robust Reinforcement Learning. In *UAI*, volume 115 of *Proceedings of Machine Learning Research*, pages 648–658. AUAI Press, 2019.

Yann Dujardin, Tom Dietterich, and Iadine Chadès. Three New Algorithms to Solve N-POMDPs. In *AAAI*, pages 4495–4501. AAAI Press, 2017.

Gabriel Dulac-Arnold, Nir Levine, Daniel J. Mankowitz, Jerry Li, Cosmin Paduraru, Sven Gowal, and Todd Hester. Challenges of Real-World Reinforcement Learning: Definitions, Benchmarks and Analysis. *Mach. Learn.*, 110(9):2419–2468, 2021.

Harald Fecher, Martin Leucker, and Verena Wolf. *Don't Know* in Probabilistic Systems. In *SPIN*, volume 3925 of *LNCS*, pages 71–88. Springer, 2006.

Norm Ferns, Prakash Panangaden, and Doina Precup. Metrics for Finite Markov Decision Processes. In *UAI*, pages 162–169. AUAI Press, 2004.

Norm Ferns, Prakash Panangaden, and Doina Precup. Metrics for Markov Decision Processes with Infinite State Spaces. In *UAI*, pages 201–208. AUAI Press, 2005.

Nathanaël Fijalkow, Nathalie Bertrand, Patricia Bouyer-Decitre, Romain Brenguier, Arnaud Carayol, John Fearnley, Hugo Gimbert, Florian Horn, Rasmus Ibsen-Jensen, Nicolas Markey, Benjamin Monmege, Petr Novotný, Mickael Randour, Ocan Sankur, Sylvain Schmitz, Olivier Serre, and Mateusz Skomra. Games on Graphs. *CoRR*, abs/2305.10546, 2023.

Vojtech Forejt, Marta Z. Kwiatkowska, Gethin Norman, and David Parker. Automated Verification Techniques for Probabilistic Systems. In *SFM*, volume 6659 of *LNCS*, pages 53–113. Springer, 2011.

Gregory R. Frey, Christopher D. Petersen, Frederick A. Leve, Ilya V. Kolmanovsky, and Anouck R. Girard. Constrained Spacecraft Relative Motion Planning Exploiting Periodic Natural Motion Trajectories and Invariance. *Journal of Guidance, Control, and Dynamics*, 40(12):3100–3115, 2017.

Uri Gadot, Esther Derman, Navdeep Kumar, Maxence Mohamed Elfatihi, Kfir Levy, and Shie Mannor. Solving Non-Rectangular Reward-Robust MDPs via Frequency Regularization. In *AAAI*, pages 21090–21098. AAAI Press, 2024.

Pratik Gajane, Ronald Ortner, and Peter Auer. A Sliding-Window Algorithm for Markov Decision Processes with Arbitrarily Changing Rewards and Transitions. *CoRR*, abs/1805.10066, 2018.

Maris F. L. Galesloot, Marnix Suilen, Thiago D. Simão, Steven Carr, Matthijs T. J. Spaan, Ufuk Topcu, and Nils Jansen. Pessimistic Iterative Planning for Robust POMDPs. *CoRR*, abs/2312.06344, 2024.

Javier García and Fernando Fernández. A Comprehensive Survey on Safe Reinforcement Learning. *J. Mach. Learn. Res.*, 16:1437–1480, 2015.

Antoine Girard and George J. Pappas. Approximation Metrics for Discrete and Continuous Systems. *IEEE Trans. Autom. Control.*, 52(5):782–798, 2007.

Robert Givan, Sonia M. Leach, and Thomas L. Dean. Bounded-Parameter Markov Decision Processes. *Artif. Intell.*, 122(1-2):71–109, 2000.

Robert Givan, Thomas L. Dean, and Matthew Greig. Equivalence Notions and Model Minimization in Markov Decision Processes. *Artif. Intell.*, 147(1-2):163–223, 2003.

Vineet Goyal and Julien Grand-Clément. Robust Markov Decision Processes: Beyond Rectangularity. *Math. Oper. Res.*, 48(1):203–226, 2023.

Julien Grand-Clément and Marek Petrik. Reducing Blackwell and Average Optimality to Discounted MDPs via the Blackwell Discount Factor. In *NeurIPS*, 2023.

Julien Grand-Clément, Marek Petrik, and Nicolas Vieille. Beyond Discounted Returns: Robust Markov Decision Processes with Average and Blackwell Optimality. *CoRR*, abs/2312.03618, 2023.

Arthur Guez, David Silver, and Peter Dayan. Efficient Bayes-Adaptive Reinforcement Learning using Sample-Based Search. In *NIPS*, pages 1034–1042, 2012.

LLC Gurobi Optimization. Gurobi Optimizer Reference Manual, 2019. URL `http://www.gurobi.com`.

Serge Haddad and Benjamin Monmege. Reachability in MDPs: Refining Convergence of Value Iteration. In *RP*, volume 8762 of *LNCS*, pages 125–137. Springer, 2014.

Alexander Hans and Steffen Udluft. Efficient Uncertainty Propagation for Reinforcement Learning with Limited Data. In *ICANN (1)*, pages 70–79. Springer, 2009.

Hans Hansson and Bengt Jonsson. A Logic for Reasoning about Time and Reliability. *Formal Aspects Comput.*, 6(5):512–535, 1994.

Arnd Hartmanns and Benjamin Lucien Kaminski. Optimistic Value Iteration. In *CAV (2)*, volume 12225 of *LNCS*, pages 488–511. Springer, 2020.

Arnd Hartmanns, Sebastian Junges, Tim Quatmann, and Maximilian Weininger. A Practitioner's Guide to MDP Model Checking Algorithms. In *TACAS (1)*, volume 13993 of *LNCS*, pages 469–488. Springer, 2023.

Vahid Hashemi, Holger Hermanns, Lei Song, K. Subramani, Andrea Turrini, and Piotr Wojciechowski. Compositional Bisimulation Minimization for Interval Markov Decision Processes. In *LATA*, volume 9618 of *LNCS*, pages 114–126. Springer, 2016.

Milos Hauskrecht and Hamish S. F. Fraser. Planning Treatment of Ischemic Heart Disease With Partially Observable Markov Decision Processes. *Artif. Intell. Medicine*, 18(3):221–244, 2000.

Christian Hensel, Sebastian Junges, Joost-Pieter Katoen, Tim Quatmann, and Matthias Volk. The Probabilistic Model Checker Storm. *Int. J. Softw. Tools Technol. Transf.*, 24(4):589–610, 2022.

Chin Pang Ho, Marek Petrik, and Wolfram Wiesemann. Fast Bellman Updates for Robust MDPs. In *ICML*, volume 80 of *Proceedings of Machine Learning Research*, pages 1984–1993. PMLR, 2018.

Chin Pang Ho, Marek Petrik, and Wolfram Wiesemann. Partial Policy Iteration for L1-Robust Markov Decision Processes. *J. Mach. Learn. Res.*, 22:275:1–275:46, 2021.

Chin Pang Ho, Marek Petrik, and Wolfram Wiesemann. Robust ϕ-Divergence MDPs. In *NeurIPS*, 2022.

Kerianne L. Hobbs and Eric M. Feron. A Taxonomy for Aerospace Collision Avoidance with Implications for Automation in Space Traffic Management. In *AIAA Scitech 2020 Forum*, page 0877, 2020.

Wassily Hoeffding. Probability Inequalities for Sums of Bounded Random Variables. *Journal of the American Statistical Association*, 58(301):13–30, 1963.

Eyke Hüllermeier and Willem Waegeman. Aleatoric and Epistemic Uncertainty in Machine Learning: An Introduction to Concepts and Methods. *Mach. Learn.*, 110 (3):457–506, 2021.

Hideaki Itoh and Kiyohiko Nakamura. Partially Observable Markov Decision Processes with Imprecise Parameters. *Artif. Intell.*, 171(8-9):453–490, 2007.

Garud N. Iyengar. Robust Dynamic Programming. *Math. Oper. Res.*, 30(2):257–280, 2005.

Bart Jacobs. A Channel-Based Perspective on Conjugate Priors. *Math. Struct. Comput. Sci.*, 30(1):44–61, 2020.

Manfred Jaeger, Giorgio Bacci, Giovanni Bacci, Kim Guldstrand Larsen, and Peter Gjøl Jensen. Approximating Euclidean by Imprecise Markov Decision Processes. In *ISoLA (1)*, volume 12476 of *LNCS*, pages 275–289. Springer, 2020.

Thomas Jaksch, Ronald Ortner, and Peter Auer. Near-optimal Regret Bounds for Reinforcement Learning. *J. Mach. Learn. Res.*, 11:1563–1600, 2010.

Nils Jansen, Sebastian Junges, and Joost-Pieter Katoen. Parameter Synthesis in Markov Models: A Gentle Survey. In *Principles of Systems Design*, volume 13660 of *LNCS*, pages 407–437. Springer, 2022.

Edwin T. Jaynes. *Probability Theory: The Logic of Science*. Cambridge University Press, 2003.

Shihao Ji, Ronald Parr, Hui Li, Xuejun Liao, and Lawrence Carin. Point-Based Policy Iteration. In *AAAI*, pages 1243–1249. AAAI Press, 2007.

Fei Jiang, Yong Jiang, Hui Zhi, Yi Dong, Hao Li, Sufeng Ma, Yilong Wang, Qiang Dong, Haipeng Shen, and Yongjun Wang. Artificial Intelligence in Healthcare: Past, Present and Future. *Stroke and vascular neurology*, 2(4), 2017.

Chi Jin, Tiancheng Jin, Haipeng Luo, Suvrit Sra, and Tiancheng Yu. Learning Adversarial Markov Decision Processes with Bandit Feedback and Unknown Transition. In *ICML*, pages 4860–4869. PMLR, 2020.

Bengt Jonsson and Kim Guldstrand Larsen. Specification and Refinement of Probabilistic Processes. In *LICS*, pages 266–277. IEEE Computer Society, 1991.

Sebastian Junges, Nils Jansen, Ralf Wimmer, Tim Quatmann, Leonore Winterer, Joost-Pieter Katoen, and Bernd Becker. Finite-State Controllers of POMDPs using Parameter Synthesis. In *UAI*, pages 519–529. AUAI Press, 2018.

Sebastian Junges, Nils Jansen, and Sanjit A. Seshia. Enforcing Almost-Sure Reachability in POMDPs. In *CAV (2)*, volume 12760 of *LNCS*, pages 602–625. Springer, 2021a.

Sebastian Junges, Joost-Pieter Katoen, Guillermo A. Pérez, and Tobias Winkler. The Complexity of Reachability in Parametric Markov Decision Processes. *J. Comput. Syst. Sci.*, 119:183–210, 2021b.

Sebastian Junges, Erika Ábrahám, Christian Hensel, Nils Jansen, Joost-Pieter Katoen, Tim Quatmann, and Matthias Volk. Parameter Synthesis for Markov Models: Covering the Parameter Space. *Formal Methods Syst. Des.*, 62(1):181–259, 2024.

Leslie Pack Kaelbling, Michael L. Littman, and Andrew W. Moore. Reinforcement Learning: A Survey. *J. Artif. Intell. Res.*, 4:237–285, 1996.

Leslie Pack Kaelbling, Michael L. Littman, and Anthony R. Cassandra. Planning and Acting in Partially Observable Stochastic Domains. *Artif. Intell.*, 101(1-2): 99–134, 1998.

Joost-Pieter Katoen. The Probabilistic Model Checking Landscape. In *LICS*, pages 31–45. ACM, 2016.

Joost-Pieter Katoen, Daniel Klink, Martin Leucker, and Verena Wolf. Three-Valued Abstraction for Continuous-Time Markov Chains. In *CAV*, volume 4590 of *LNCS*, pages 311–324. Springer, 2007.

Joost-Pieter Katoen, Daniel Klink, Martin Leucker, and Verena Wolf. Three-Valued Abstraction for Probabilistic Systems. *J. Log. Algebraic Methods Program.*, 81(4): 356–389, 2012.

Mark Kattenbelt, Marta Z. Kwiatkowska, Gethin Norman, and David Parker. A Game-Based Abstraction-Refinement Framework for Markov Decision Processes. *Formal Methods Syst. Des.*, 36(3):246–280, 2010.

David L. Kaufman and Andrew J. Schaefer. Robust Modified Policy Iteration. *INFORMS J. Comput.*, 25(3):396–410, 2013.

Susan C. Kim, Stanley W. Shepperd, H Lee Norris, Hannah R. Goldberg, and Mark S. Wallace. Mission Design and Trajectory Analysis for Inspection of a Host Spacecraft by a Microsatellite. In *2007 IEEE Aerospace Conference*, pages 1–23. IEEE, 2007.

Volker Klingspor, John Demiris, and Michael Kaiser. Human-Robot Communication and Machine Learning. *Applied Artificial Intelligence*, 11(7):719–746, 1997.

Mykel J. Kochenderfer. *Decision Making Under Uncertainty: Theory and Application*. MIT press, 2015.

Jan Kretínský. Survey of Statistical Verification of Linear Unbounded Properties: Model Checking and Distances. In *ISoLA (1)*, volume 9952 of *Lecture Notes in Computer Science*, pages 27–45, 2016.

Navdeep Kumar, Esther Derman, Matthieu Geist, Kfir Y. Levy, and Shie Mannor. Policy Gradient for Rectangular Robust Markov Decision Processes. In *NeurIPS*, 2023.

Hanna Kurniawati, David Hsu, and Wee Sun Lee. SARSOP: Efficient Point-Based POMDP Planning by Approximating Optimally Reachable Belief Spaces. In *Robotics: Science and Systems*. The MIT Press, 2008.

Marta Z. Kwiatkowska, Gethin Norman, and David Parker. Game-based Abstraction for Markov Decision Processes. In *QEST*, pages 157–166. IEEE Computer Society, 2006.

Marta Z. Kwiatkowska, Gethin Norman, and David Parker. Stochastic Model Checking. In *SFM*, volume 4486 of *LNCS*, pages 220–270. Springer, 2007.

Marta Z. Kwiatkowska, Gethin Norman, and David Parker. PRISM 4.0: Verification of Probabilistic Real-Time Systems. In *CAV*, volume 6806 of *LNCS*, pages 585–591. Springer, 2011.

Morteza Lahijanian, Sean B. Andersson, and Calin Belta. Formal Verification and Synthesis for Discrete-Time Stochastic Systems. *IEEE Trans. Autom. Control.*, 60 (8):2031–2045, 2015.

Sascha Lange, Thomas Gabel, and Martin A. Riedmiller. Batch Reinforcement Learning. In *Reinforcement Learning*, volume 12 of *Adaptation, Learning, and Optimization*, pages 45–73. Springer, 2012.

Romain Laroche, Paul Trichelair, and Remi Tachet des Combes. Safe Policy Improvement with Baseline Bootstrapping. In *ICML*, pages 3652–3661. PMLR, 2019.

Kim Guldstrand Larsen and Arne Skou. Bisimulation through Probabilistic Testing. *Inf. Comput.*, 94(1):1–28, 1991.

Tor Lattimore and Csaba Szepesvári. *Bandit Algorithms*. Cambridge University Press, 2020.

Abolfazl Lavaei, Sadegh Soudjani, Alessandro Abate, and Majid Zamani. Automated Verification and Synthesis of Stochastic Hybrid Systems: A Survey. *Autom.*, 146: 110617, 2022.

Abolfazl Lavaei, Sadegh Soudjani, Emilio Frazzoli, and Majid Zamani. Constructing MDP Abstractions Using Data With Formal Guarantees. *IEEE Control. Syst. Lett.*, 7:460–465, 2023.

Axel Legay, Benoît Delahaye, and Saddek Bensalem. Statistical Model Checking: An Overview. In *RV*, volume 6418 of *LNCS*, pages 122–135. Springer, 2010.

Sergey Levine, Aviral Kumar, George Tucker, and Justin Fu. Offline Reinforcement Learning: Tutorial, Review, and Perspectives on Open Problems. *CoRR*, abs/2005.01643, 2020.

Thomas Lipp and Stephen Boyd. Variations and Extension of the Convex–Concave Procedure. *Optimization and Engineering*, 17(2):263–287, 2016.

Johan Löfberg. Automatic Robust Convex Programming. *Optim. Methods Softw.*, 27 (1):115–129, 2012.

Omid Madani, Steve Hanks, and Anne Condon. On the Undecidability of Probabilistic Planning and Related Stochastic Optimization Problems. *Artif. Intell.*, 147 (1-2):5–34, 2003.

Shie Mannor, Duncan Simester, Peng Sun, and John N. Tsitsiklis. Bias and Variance Approximation in Value Function Estimates. *Manag. Sci.*, 53(2):308–322, 2007.

Yuanqi Mao, Michael Szmuk, Xiangru Xu, and Behcet Acikmese. Successive Convexification: A Superlinearly Convergent Algorithm for Non-convex Optimal Control Problems. *arXiv preprint arXiv:1804.06539*, 2018.

Aymen Al Marjani, Andrea Tirinzoni, and Emilie Kaufmann. Active Coverage for PAC Reinforcement Learning. In *COLT*, volume 195 of *Proceedings of Machine Learning Research*, pages 5044–5109. PMLR, 2023.

Frederik Baymler Mathiesen, Morteza Lahijanian, and Luca Laurenti. IntervalMDP.jl: Accelerated Value Iteration for Interval Markov Decision Processes. *CoRR*, abs/2401.04068, 2024.

Andrew McCallum. Overcoming Incomplete Perception with Utile Distinction Memory. In *ICML*, pages 190–196. Morgan Kaufmann, 1993.

Andrew McCallum. Instance-Based Utile Distinctions for Reinforcement Learning with Hidden State. In *ICML*, pages 387–395. Morgan Kaufmann, 1995.

Tobias Meggendorfer, Maximilian Weininger, and Patrick Wienhöft. What Are the Odds? Improving the Foundations of Statistical Model Checking. *CoRR*, abs/2404.05424, 2024.

Nicolas Meuleau, Kee-Eung Kim, Leslie Pack Kaelbling, and Anthony R. Cassandra. Solving POMDPs by Searching the Space of Finite Policies. In *UAI*, pages 417–426. Morgan Kaufmann, 1999.

Volodymyr Mnih, Koray Kavukcuoglu, David Silver, Andrei A. Rusu, Joel Veness, Marc G. Bellemare, Alex Graves, Martin A. Riedmiller, Andreas 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. *Nat.*, 518(7540): 529–533, 2015.

Thomas M. Moerland, Joost Broekens, Aske Plaat, and Catholijn M. Jonker. Model-Based Reinforcement Learning: A Survey. *Found. Trends Mach. Learn.*, 16(1): 1–118, 2023.

Janosch Moos, Kay Hansel, Hany Abdulsamad, Svenja Stark, Debora Clever, and Jan Peters. Robust Reinforcement Learning: A Review of Foundations and Recent Advances. *Mach. Learn. Knowl. Extr.*, 4(1):276–315, 2022.

Rémi Munos. From Bandits to Monte-Carlo Tree Search: The Optimistic Principle Applied to Optimization and Planning. *Found. Trends Mach. Learn.*, 7(1):1–129, 2014.

Hideaki Nakao, Ruiwei Jiang, and Siqian Shen. Distributionally Robust Partially Observable Markov Decision Process with Moment-Based Ambiguity. *SIAM J. Optim.*, 31(1):461–488, 2021.

Arnab Nilim and Laurent El Ghaoui. Robust Control of Markov Decision Processes with Uncertain Transition Matrices. *Oper. Res.*, 53(5):780–798, 2005.

Takayuki Osogami. Robust Partially Observable Markov Decision Process. In *ICML*, volume 37 of *JMLR Workshop and Conference Proceedings*, pages 106–115. JMLR.org, 2015.

Wenfan Ou and Sheng Bi. Sequential Decision-Making under Uncertainty: A Robust MDPs Review. *CoRR*, abs/2305.10546, 2024.

Christos H. Papadimitriou and John N. Tsitsiklis. The Complexity of Markov Decision Processes. *Math. Oper. Res.*, 12(3):441–450, 1987.

Marek Petrik, Mohammad Ghavamzadeh, and Yinlam Chow. Safe Policy Improvement by Minimizing Robust Baseline Regret. In *NIPS*, pages 2298–2306, 2016.

Joelle Pineau, Geoffrey J. Gordon, and Sebastian Thrun. Point-Based Value Iteration: An Anytime Algorithm for POMDPs. In *IJCAI*, pages 1025–1032. Morgan Kaufmann, 2003.

Amir Pnueli. The Temporal Logic of Programs. In *FOCS*, pages 46–57. IEEE Computer Society, 1977.

Pascal Poupart and Craig Boutilier. Bounded Finite State Controllers. In *NIPS*, pages 823–830. MIT Press, 2003.

Alberto Puggelli, Wenchao Li, Alberto L. Sangiovanni-Vincentelli, and Sanjit A. Seshia. Polynomial-Time Verification of PCTL Properties of MDPs with Convex Uncertainties. In *CAV*, volume 8044 of *LNCS*, pages 527–542. Springer, 2013.

Martin L. Puterman. *Markov Decision Processes: Discrete Stochastic Dynamic Programming*. Wiley Series in Probability and Statistics. Wiley, 1994.

Tim Quatmann and Joost-Pieter Katoen. Sound Value Iteration. In *CAV (1)*, volume 10981 of *LNCS*, pages 643–661. Springer, 2018.

Tim Quatmann, Christian Dehnert, Nils Jansen, Sebastian Junges, and Joost-Pieter Katoen. Parameter Synthesis for Markov Models: Faster Than Ever. In *ATVA*, volume 9938 of *LNCS*, pages 50–67, 2016.

Ramya Ramakrishnan, Ece Kamar, Debadeepta Dey, Eric Horvitz, and Julie Shah. Blind Spot Detection for Safe Sim-to-Real Transfer. *J. Artif. Intell. Res.*, 67:191–234, 2020.

Jean-François Raskin and Ocan Sankur. Multiple-Environment Markov Decision Processes. In *FSTTCS*, volume 29 of *LIPIcs*, pages 531–543. Schloss Dagstuhl - Leibniz-Zentrum für Informatik, 2014.

Luke Rickard, Alessandro Abate, and Kostas Margellos. Learning Robust Policies for Uncertain Parametric Markov Decision Processes. *CoRR*, abs/2312.06344, 2023.

Marc Rigter, Bruno Lacerda, and Nick Hawes. Minimax Regret Optimisation for Robust Planning in Uncertain Markov Decision Processes. In *AAAI*, pages 11930–11938. AAAI Press, 2021a.

Marc Rigter, Bruno Lacerda, and Nick Hawes. Risk-Averse Bayes-Adaptive Reinforcement Learning. In *NeurIPS*, pages 1142–1154, 2021b.

R. Tyrrell Rockafellar and Stanislav Uryasev. Optimization of Conditional Value-at-Risk. *Journal of risk*, 2(3):21–41, 2000.

Stuart Russell and Peter Norvig. *Artificial Intelligence: A Modern Approach (4th Edition)*. Pearson, 2020.

Soroush Saghafian. Ambiguous Partially Observable Markov Decision Processes: Structural Results and Applications. *J. Econ. Theory*, 178:1–35, 2018.

Harsh Satija, Philip S. Thomas, Joelle Pineau, and Romain Laroche. Multi-Objective SPIBB: Seldonian Offline Policy Improvement with Safety Constraints in Finite MDPs. In *NeurIPS*, pages 2004–2017, 2021.

Yannik Schnitzer, Alessandro Abate, and David Parker. Learning Provably Robust Policies in Uncertain Parametric Environments. *CoRR*, abs/2408.03093, 2024.

Philipp Scholl, Felix Dietrich, Clemens Otte, and Steffen Udluft. Safe Policy Improvement Approaches on Discrete Markov Decision Processes. In *ICAART (2)*, pages 142–151. SCITEPRESS, 2022.

David Silver and Joel Veness. Monte-Carlo Planning in Large POMDPs. In *NIPS*, pages 2164–2172. Curran Associates, Inc., 2010.

David Silver, Julian Schrittwieser, Karen Simonyan, Ioannis Antonoglou, Aja Huang, Arthur Guez, Thomas Hubert, Lucas Baker, Matthew Lai, Adrian Bolton, Yutian Chen, Timothy P. Lillicrap, Fan Hui, Laurent Sifre, George van den Driessche, Thore Graepel, and Demis Hassabis. Mastering the Game of Go Without Human Knowledge. *Nat.*, 550(7676):354–359, 2017.

David Silver, Thomas Hubert, Julian Schrittwieser, Ioannis Antonoglou, Matthew Lai, Arthur Guez, Marc Lanctot, Laurent Sifre, Dharshan Kumaran, Thore Graepel, et al. A General Reinforcement Learning Algorithm that Masters Chess, Shogi, and Go Through Self-Play. *Science*, 362(6419):1140–1144, 2018.

Thiago D. Simão and Matthijs T. J. Spaan. Safe Policy Improvement with Baseline Bootstrapping in Factored Environments. In *AAAI*, pages 4967–4974. AAAI Press, 2019a.

Thiago D. Simão and Matthijs T. J. Spaan. Structure Learning for Safe Policy Improvement. In *IJCAI*, pages 3453–3459. ijcai.org, 2019b.

Thiago D. Simão, Romain Laroche, and Rémi Tachet des Combes. Safe Policy Improvement with an Estimated Baseline Policy. In *AAMAS*, pages 1269–1277. IFAAMAS, 2020.

Thiago D. Simão, Marnix Suilen, and Nils Jansen. Safe Policy Improvement for POMDPs via Finite-State Controllers. In *AAAI*, pages 15109–15117. AAAI Press, 2023.

Richard D. Smallwood and Edward J. Sondik. The Optimal Control of Partially Observable Markov Processes over a Finite Horizon. *Oper. Res.*, 21(5):1071–1088, 1973.

Trey Smith and Reid G. Simmons. Heuristic Search Value Iteration for POMDPs. In *UAI*, pages 520–527. AUAI Press, 2004.

Christian Soize. *Uncertainty quantification*. Springer, 2017.

Matthijs T. J. Spaan. Partially Observable Markov Decision Processes. In *Reinforcement Learning*, volume 12 of *Adaptation, Learning, and Optimization*, pages 387–414. Springer, 2012.

Matthijs T. J. Spaan and Nikos Vlassis. Perseus: Randomized Point-based Value Iteration for POMDPs. *J. Artif. Intell. Res.*, 24:195–220, 2005.

Rolf A. N. Starre, Marco Loog, Elena Congeduti, and Frans A. Oliehoek. An Analysis of Model-Based Reinforcement Learning From Abstracted Observations. *Trans. Mach. Learn. Res.*, 2023, 2023.

Alexander L. Strehl and Michael L. Littman. An analysis of Model-Based Interval Estimation for Markov Decision Processes. *J. Comput. Syst. Sci.*, 74(8):1309–1331, 2008.

Alexander L. Strehl, Lihong Li, Eric Wiewiora, John Langford, and Michael L. Littman. PAC Model-Free Reinforcement Learning. In *ICML*, volume 148 of *ACM International Conference Proceeding Series*, pages 881–888. ACM, 2006.

Alexander L. Strehl, Lihong Li, and Michael L. Littman. Reinforcement Learning in Finite MDPs: PAC Analysis. *J. Mach. Learn. Res.*, 10:2413–2444, 2009.

Marnix Suilen, Nils Jansen, Murat Cubuktepe, and Ufuk Topcu. Robust Policy Synthesis for Uncertain POMDPs via Convex Optimization. In *IJCAI*, pages 4113–4120. ijcai.org, 2020.

Marnix Suilen, Thiago D. Simão, David Parker, and Nils Jansen. Robust Anytime Learning of Markov Decision Processes. In *NeurIPS*, volume 35, pages 28790–28802. Curran Associates, Inc., 2022.

Marnix Suilen, Thom Badings, Eline M. Bovy, David Parker, and Nils Jansen. Robust Markov Decision Processes: A Place Where AI and Formal Methods Meet. In *Principles of Verification: Cycling the Probabilistic Landscape: Essays Dedicated to Joost-Pieter Katoen on the Occasion of His 60th Birthday, Part III*, volume 15262 of *LNCS*, pages 126–154. Springer, 2024a.

Marnix Suilen, Marck van der Vegt, and Sebastian Junges. A PSPACE Algorithm for Almost-Sure Rabin Objectives in Multi-Environment MDPs. In *CONCUR*, volume 311 of *LIPIcs*, pages 40:1–40:17. Schloss Dagstuhl - Leibniz-Zentrum für Informatik, 2024b.

Richard S. Sutton and Andrew G. Barto. *Reinforcement Learning - An Introduction*. Adaptive computation and machine learning. MIT Press, 1998.

Richard S. Sutton, Doina Precup, and Satinder Singh. Between MDPs and Semi-MDPs: A Framework for Temporal Abstraction in Reinforcement Learning. *Artif. Intell.*, 112(1-2):181–211, 1999.

Nico M. Temme. Asymptotic Inversion of the Incomplete Beta Function. *Journal of computational and applied mathematics*, 41(1-2):145–157, 1992.

Philip S. Thomas, Georgios Theocharous, and Mohammad Ghavamzadeh. High Confidence Policy Improvement. In *ICML*, pages 2380–2388. PMLR, 2015.

Sebastian Thrun, Wolfram Burgard, and Dieter Fox. *Probabilistic Robotics*. Intelligent robotics and autonomous agents. MIT Press, 2005.

Leslie G. Valiant. A Theory of the Learnable. *Commun. ACM*, 27(11):1134–1142, 1984.

Marck van der Vegt, Nils Jansen, and Sebastian Junges. Robust Almost-Sure Reachability in Multi-Environment MDPs. In *TACAS (1)*, volume 13993 of *LNCS*, pages 508–526. Springer, 2023.

Nikos Vlassis, Michael L. Littman, and David Barber. On the Computational Complexity of Stochastic Controller Optimization in POMDPs. *ACM Trans. Comput. Theory*, 4(4):12:1–12:8, 2012.

Gero Walter and Thomas Augustin. Imprecision and Prior-Data Conflict in Generalized Bayesian Inference. *Journal of Statistical Theory and Practice*, 3(1):255–271, 2009.

Qiuhao Wang, Chin Pang Ho, and Marek Petrik. Policy Gradient in Robust MDPs with Global Convergence Guarantee. In *ICML*, volume 202 of *Proceedings of Machine Learning Research*, pages 35763–35797. PMLR, 2023.

Tsachy Weissman, Erik Ordentlich, Gadiel Seroussi, Sergio Verdú, and Marcelo J. Weinberger. Inequalities for the L1 Deviation of the Empirical Distribution. *Hewlett-Packard Labs, Tech. Rep*, 2003.

Patrick Wienhöft, Marnix Suilen, Thiago D. Simão, Clemens Dubslaff, Christel Baier, and Nils Jansen. More for Less: Safe Policy Improvement with Stronger Performance Guarantees. In *IJCAI*, pages 4406–4415. ijcai.org, 2023.

Wolfram Wiesemann, Daniel Kuhn, and Berç Rustem. Robust Markov Decision Processes. *Math. Oper. Res.*, 38(1):153–183, 2013.

Jason D. Williams and Steve J. Young. Partially Observable Markov Decision Processes for Spoken Dialog Systems. *Comput. Speech Lang.*, 21(2):393–422, 2007.

Eric M. Wolff, Ufuk Topcu, and Richard M. Murray. Robust Control of Uncertain Markov Decision Processes with Temporal Logic Specifications. In *CDC*, pages 3372–3379. IEEE, 2012.

Ben Wooding and Abolfazl Lavaei. IMPaCT: Interval MDP Parallel Construction for Controller Synthesis of Large-Scale Stochastic Systems. *CoRR*, abs/2401.03555, 2024.

Huan Xu and Shie Mannor. Distributionally Robust Markov Decision Processes. *Math. Oper. Res.*, 37(2):288–300, 2012.

Cambridge Yang, Michael L. Littman, and Michael Carbin. On the (In)Tractability of Reinforcement Learning for LTL Objectives. In *IJCAI*, pages 3650–3658. ijcai.org, 2022.

Jared Yeager, J. Eliot B. Moss, Michael Norrish, and Philip S. Thomas. Mechanizing Soundness of Off-Policy Evaluation. In *ITP*, volume 237, pages 32:1–32:20, 2022.

Håkan L. S. Younes and Reid G. Simmons. Probabilistic Verification of Discrete Event Systems Using Acceptance Sampling. In *CAV*, volume 2404 of *Lecture Notes in Computer Science*, pages 223–235. Springer, 2002.

Ya-Xiang Yuan. Recent Advances in Trust Region Algorithms. *Math. Program.*, 151 (1):249–281, 2015.

Wenshuai Zhao, Jorge Peña Queralta, and Tomi Westerlund. Sim-to-Real Transfer in Deep Reinforcement Learning for Robotics: A Survey. In *SSCI*, pages 737–744. IEEE, 2020.

Research Data Management

This thesis research has been carried out under the research data management policy of the Institute for Computing and Information Science of Radboud University, The Netherlands.[1]

The following research datasets have been produced during this PhD research:

- Chapter 4:
  - Murat Cubuktepe, Nils Jansen, Sebastian Junges, Ahmadreza Marandi, Marnix Suilen, and Ufuk Topcu. Finite-Memory Policies for Robust POMDPs via Convex Optimization (implementation). Zenodo, 2021. https://doi.org/10.5281/zenodo.14675305.

    Also available at https://github.com/LAVA-LAB/ipomdp_cvx.
- Chapter 5:
  - Marnix Suilen, Thiago D. Simão, David Parker, and Nils Jansen. Robust Anytime Learning of Markov Decision Processes (implementation). Zenodo, 2022. https://doi.org/10.5281/zenodo.14675274.

    Also available at https://github.com/LAVA-LAB/luiaard.
- Chapter 6:
  - Thiago D. Simão, Marnix Suilen, and Nils Jansen. Safe Policy Improvement for POMDPs via Finite-State Controllers (implementation). Zenodo, 2023. https://doi.org/10.5281/zenodo.14675351.

    Also available at https://github.com/LAVA-LAB/spi_pomdp.
  - Patrick Wienhöft, Marnix Suilen, Thiago D. Simão, Clemens Dubslaff, Christel Baier, and Nils Jansen. More for Less: Safe Policy Improvement With Stronger Performance Guarantees (implementation). Zenodo, 2023. https://doi.org/10.5281/zenodo.14675359.

    Also available at https://github.com/LAVA-LAB/improved_spi.

No research data or code has been produced for the other chapters.

[1] ru.nl/icis/research-data-management/, last accessed August 8th, 2024.

List of Publications

Related to This Thesis

Marnix Suilen, Nils Jansen, Murat Cubuktepe, and Ufuk Topcu. Robust Policy Synthesis for Uncertain POMDPs via Convex Optimization. In *IJCAI*, pages 4113–4120. ijcai.org, 2020.

Murat Cubuktepe, Nils Jansen, Sebastian Junges, Ahmadreza Marandi, Marnix Suilen, and Ufuk Topcu. Robust Finite-State Controllers for Uncertain POMDPs. In *AAAI*, pages 11792–11800. AAAI Press, 2021.

Marnix Suilen, Thiago D. Simão, David Parker, and Nils Jansen. Robust Anytime Learning of Markov Decision Processes. In *NeurIPS*, volume 35, pages 28790–28802. Curran Associates, Inc., 2022.

Thiago D. Simão, Marnix Suilen, and Nils Jansen. Safe Policy Improvement for POMDPs via Finite-State Controllers. In *AAAI*, pages 15109–15117. AAAI Press, 2023.

Patrick Wienhöft, Marnix Suilen, Thiago D. Simão, Clemens Dubslaff, Christel Baier, and Nils Jansen. More for Less: Safe Policy Improvement With Stronger Performance Guarantees. In *IJCAI*, pages 4406–4415. ijcai.org, 2023.

Marnix Suilen, Thom Badings, Eline M. Bovy, David Parker, and Nils Jansen. Robust Markov Decision Processes: A Place Where AI and Formal Methods Meet. In *Principles of Verification: Cycling the Probabilistic Landscape: Essays Dedicated to Joost-Pieter Katoen on the Occasion of His 60th Birthday, Part III*, volume 15262 of *LNCS*, pages 126–154. Springer, 2024.

Other Peer-Reviewed Publications

Thom S. Badings, Arnd Hartmanns, Nils Jansen, and Marnix Suilen. Balancing Wind and Batteries: Towards Predictive Verification of Smart Grids. In *NFM*, volume 12673 of *LNCS*, pages 1–18. Springer, 2021.

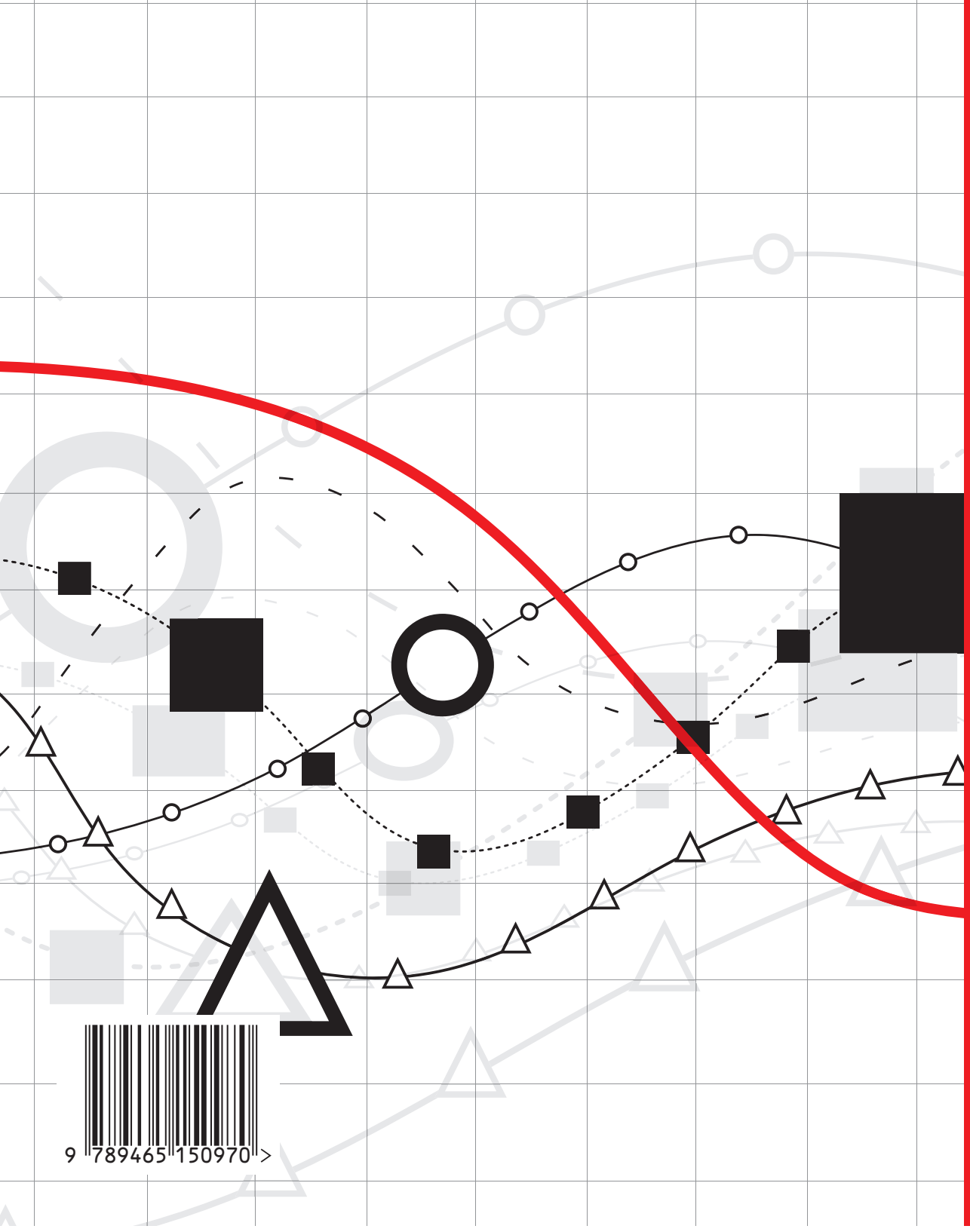
Thom S. Badings, Thiago D. Simão, Marnix Suilen, and Nils Jansen. Decision-Making Under Uncertainty: Beyond Probabilities. *Int. J. Softw. Tools Technol. Transf.*, 25(3): 375–391, 2023.

Eline M. Bovy, Marnix Suilen, Sebastian Junges, and Nils Jansen. Imprecise Probabilities Meet Partial Observability: Game Semantics for Robust POMDPs. In *IJCAI*, pages 6697–6706. ijcai.org, 2024.

Marnix Suilen, Marck van der Vegt, and Sebastian Junges. A PSPACE Algorithm for Almost-Sure Rabin Objectives in Multi-Environment MDPs. In *CONCUR*, volume 311 of LIPIcs, pages 40:1–40:17. Schloss Dagstuhl - Leibniz-Zentrum für Informatik, 2024.

About the Author

Marnix Suilen was born on February 2, 1996, in Roermond, the Netherlands. From a young age, he developed an interest in computers. In 2014, he started his studies in computing science at Radboud University in Nijmegen, the Netherlands. He obtained his bachelor's degree in 2018 and his master's degree with a specialisation in mathematical foundations of computing science in 2020. He stayed in Nijmegen to pursue a Ph.D. in computer science between 2020 and 2024. Since 2024, he has been a postdoctoral researcher at the University of Antwerp, Belgium.



9 789465 150970

Radboud Universiteit