# BAYES-ADAPTIVE SEMI-MARKOV DECISION PROCESSES

A Pathway to Optimal Learning in Sequential Decision Making with Time under Uncertainty

# PROCESSUS DE DÉCISION SEMI-MARKOVIENS ADAPTATIFS DE BAYES

Un chemin vers l'apprentissage optimal dans la prise de décision séquentielle avec le temps sous incertitude

A Thesis Submitted to the Division of Graduate Studies
of the Royal Military College of Canada
by

Richard Kohar, BSc (Hons), MSc

In Partial Fulfillment of the Requirements for the Degree of
Doctor of Philosophy

June 2023

*To my parents*

*Rick*

*&*

*Anneliese*

*who*

*have*

*always*

*been*

*proud*

*of*

*me.*

# Acknowledgments

I would like to express gratitude to the following who have made the completion of this thesis possible.

This work straddles two fields, and thus required two excellent supervisors. Dr François Rivest was the one who first hired me to work on a machine learning project and got me hooked. We have spent many hours discussing problems and working on papers together. Without his patience, I do not know if I would have been able to accomplish any programming task; he helped me to become a better programmer. Thank you to Dr Alain Gosselin who agreed to co-supervise the mathematical aspect of this work. He has the fantastic ability to view problems differently from other people. Simply put, he is the randomness that perturbs me out of a local solution. Both of these men have contributed to me becoming an expert in this field.

Thank you to my examining committee: Dr Daniel Lagacé-Roy (Interim Dean of Social Sciences and Humanities), Dr René Sturgeon (Head of Mathematics and Computer Science Department, Royal Military College), Dr Aboelmagd Noureldin (Professor of Electrical and Computer Engineering, Royal Military College), and Dr Patrick Simen (Chair of Neuroscience, Oberlin College) for their valuable comments, questions, and criticism.

I wish to dedicate this thesis to my father and mother—Rick & Anneliese Kohar—who have supported me from the very beginning. Whilst I enjoy the sound of deadlines as they whiz past my head, they were the ones who reminded me to get this finished. My father and mother, along with my brother Dr Christopher Kohar, dealt tactfully with the kitchen table always covered in piles of pens, reams of paper, and stacks of books. It is very fitting that as Chris and I grew up that we did our homework at the kitchen table; we began at the kitchen table, and it ended with us both writing our PhDs there.

I appreciate the welcoming environment of the Department of Mathematics & Computer Science at the Royal Military College of Canada, and I hope to see the lasting tradition I started of consuming fish—raw or deep-fried—continues every Friday.

The following graphics are from the `thenounproject.com`: *Bus* by Rainbow Designs, *Bus Stop Square* by Rihards Gromuls, *man and bicycle* by Gan Khoon Lay, *clock* and *Time* by kiddo.

While completing a PhD is exhausting under the best of times, the sojourn time from the beginning to the completion of this thesis unfolded during the unprecedented 2020 pandemic made it a truly gruelling experience. This was also difficult as I helped take care of my mother before she passed in 2021—she would have been the happiest of all to have seen this completed. For the reader who actually reads acknowledgments, let me give you a small morsel of wisdom: Being an introvert,

you would think that social distancing would be ideal to allow one to focus intensely on work. But month after month, I missed being in the department and in-person interactions—even passport smiles. Joni’s words ring true: you don’t know what you’ve got ‘til it’s gone.

# Abstract

In many real-world situations, sequential decision-making demands not only managing environmental uncertainty but also accounting for the stochastic nature of timing, as events unravel unpredictably over time. Traditional reinforcement learning models often struggle in these contexts, specifically in effectively learning the sojourn time distributions and grappling with uncertainties about the environment's dynamics. This thesis addresses these challenges by delivering two key contributions: a novel solver for partially observable semi-Markov decision processes (POSMDPs) and the development of the Bayes-adaptive semi-Markov decision processes (BA-SMDPs) framework. Both tools significantly extend reinforcement learning capabilities, catering to problems where timing is crucial.

Our first major contribution is CHRONOSPERSEUS, an innovative POSMDP solver that combines the strengths of point-based value iteration and importance sampling to effectively handle a wide array of problem types. We designed this solver to handle episodic and non-episodic problems, mixed observability, discrete or continuous observation spaces, and a mixture of fixed and stochastic continuous sojourn times. CHRONOSPERSEUS represents a significant stride in tackling reinforcement learning problems involving timing.

Simultaneously, we introduce BA-SMDPs, a novel Bayesian reinforcement learning framework specifically designed for optimally learning the policy in problems of sequential decision-making under uncertainty. The heart of BA-SMDPs lies in the intricate interplay between timing, environmental exploration, and exploitation of current knowledge. Notably, we demonstrate that BA-SMDPs can be expressed as POSMDPs, enabling the application of CHRONOSPERSEUS for BA-SMDPs and thereby further broadening its utility.

Within the BA-SMDPs framework, we explore four distinct approaches: learning the sojourn time distribution parameters from a finite set of sojourn times; learning mixtures of known sojourn time distributions with unknown proportions; learning mixtures of known SMDPs with unknown proportions; and learning unknown continuous sojourn-time distribution parameters. Further, we contribute a conjugate prior for the mean parameter of the inverse Gaussian sojourn-time distribution, enhancing our ability to track the uncertainty of SMDP parameters whilst finding an optimal policy.

This thesis substantively contributes to reinforcement learning research, particularly in contexts where timing is essential. It provides a robust foundation for further exploration, such as examining the implications of an unknown reward function within the BA-SMDP framework. It also encourages the application of the developed frameworks and algorithms to a wider array of complex real-world problems, inviting both theoretical advancements and practical applications that can significantly impact various societal sectors. In essence, this work proposes powerful new tools and frameworks for reinforcement learning when timing is stochastic, opening the door to more nuanced and effective solutions in the field.

*Keywords*: Bayesian reinforcement learning, Bayes-adaptive semi-Markov decision process, sequential decision making, partial observability, sojourn-time distribution.

# Résumé

Dans de nombreuses situations réelles la prise de décision séquentielle exige non seulement de gérer l'incertitude environnementale mais aussi de tenir compte de la nature stochastique du temps puisque les événements se déroulent de manière temporellement imprévisible. Les modèles traditionnels d'apprentissage par renforcement ont souvent des difficultés dans ces contextes, notamment en ce qui concerne l'apprentissage efficace des distributions de temps de séjour et la prise en compte des incertitudes relatives à la dynamique de l'environnement. Cette thèse aborde ces défis en apportant deux contributions clés : un nouveau solveur pour les processus de décision semi-markoviens partiellement observables (POSMDP) et le développement d'un cadre décisionnel pour les processus de décision semi-markoviens adaptatifs de Bayes (BA-SMDP). Ces deux outils étendent considérablement les capacités d'apprentissage par renforcement en s'adressant aux problèmes où le temps est crucial.

Notre première contribution majeure est ChronosPerseus, un solveur POSMDP innovant qui combine les forces de l'itération de la valeur par évaluations ponctuelles et de l'échantillonnage par importance pour traiter efficacement un large éventail de types de problèmes. Nous avons conçu ce solveur pour traiter les problèmes épisodiques et non épisodiques, l'observabilité mixte, les espaces d'observation discrets ou continus, et un mélange de temps de séjour fixes et stochastiques continus. ChronosPerseus représente une avancée significative dans la résolution des problèmes d'apprentissage par renforcement impliquant le temps. Simultanément, nous présentons le BA-SMDP, un nouveau cadre d'apprentissage par renforcement bayésien spécialement conçu pour l'apprentissage optimal de la politique décisionnelle dans les problèmes de prise de décision séquentielle en présence d'incertitude. Le cœur des BA-SMDP réside dans l'interaction complexe entre l'élément temporel, l'exploration de l'environnement et l'exploitation des connaissances actuelles. Nous démontrons notamment que les BA-SMDP peuvent être exprimés comme des POSMDP, ce qui permet d'appliquer ChronosPerseus aux BA-SMDP et d'élargir ainsi son utilité.

Dans le cadre des BA-SMDP nous explorons quatre approches distinctes : l'apprentissage des paramètres de la distribution des temps de séjour à partir d'un nombre fini d'échantillonnage ; l'apprentissage des proportions inconnues de l'apport de distributions de temps de séjour connues ; l'apprentissage de mélanges de SMDP connus avec des proportions inconnues ; et l'apprentissage des paramètres inconnus d'une distribution de temps de séjour continus. De plus, nous apportons une loi a priori conjuguée pour le paramètre moyen de la distribution inverse gaussienne du temps de séjour, améliorant ainsi notre capacité de gérer l'incertitude des paramètres du SMDP tout en trouvant une politique décisionnelle optimale.

Cette thèse apporte une contribution substantielle à la recherche sur l'apprentissage par renforcement en particulier dans les contextes où le temps est essentiel. Elle fournit une base solide pour des explorations ultérieures tel que l'examen des impactes d'une fonction de récompense inconnue dans le cadre du BA-SMDP. Elle encourage également l'application des cadres et algorithmes existants à un plus large éventail de problèmes complexes du monde réel, invitant à la fois à des avancées théoriques et des applications pratiques qui peuvent avoir un impact important dans de nombreux secteurs de la société. Fondamentalement, ce travail propose de nouveaux outils et cadres puissants pour l'apprentissage par renforcement lorsque le temps est stochastique ouvrant ainsi la voie à des solutions plus nuancées et plus efficaces dans ce domaine.

*Mots-clés*: Apprentissage par renforcement bayésien, processus de décision semi-markoviens adapté à Bayes, prise de décision séquentielle, observabilité partielle, distribution du temps de séjour.

# Contents

# List of Figures

# List of Tables

# Summary of Notation

Notation varies from different authors in reinforcement learning. The following notation that has been selected has allowed me to be consistent throughout this thesis.

Capital letters are used for random variables, whereas lower case letters are used for the values of random variables and for scalar functions. Quantities that are required to be real-valued vectors are written in bold and in lower case (even if random variables). Matrices are bold capitals.

If continuous functions are dealing with probability, capital letters denote cumulative distribution functions, and lowercase letters denote probability density functions.

Set notation:

| | |
|---|---|
| $x$ | element |
| $E$ | set |
| $\forall$ | for all |
| $x \in E$ | element $x$ belongs to set $E$ |
| $x \notin E$ | element $x$ does not belong to set $E$ |
| $E \subset F$ | set $E$ is a strict subset of $F$ |
| $E \not\subset F$ | set $E$ is not a strict subset of $F$ |
| $E \cup F$ | set $E$ union set $F$ |
| $E \setminus F$ | remove all elements of $F$ in $E$ |
| $\mathcal{X}$ | space |
| $\lvert\mathcal{X}\rvert$ | cardinality of the space (if finite, then it is the number of elements in the space) |
| $(a, b]$ | the real interval between $a$ and $b$, including $b$, but not including $a$ |
| $\infty$ | infinity |
| $\mathbb{N}$ | set of natural numbers: $\{1, 2, 3, \ldots\}$ |
| $\mathbb{W}$ | set of whole numbers: $\{0, 1, 2, 3, \ldots\}$ |
| $\mathbb{R}$ | set of real numbers |

General mathematical notation:

| | |
|---|---|
| $f : \mathcal{A} \to \mathcal{B}$ | function $f$ from elements of set $\mathcal{A}$ (domain) to elements of set $\mathcal{B}$ (range) |
| $\triangleq$ | equality relationship that is true by definition |
| $\approx$ | approximately equal |
| $\propto$ | proportional to |
| $\epsilon$ | small positive real value |

General mathematical functions:

| | |
|---|---|
| e | Euler's constant: $\mathrm{e} \approx 2.71828$ |

| | |
|---|---|
| $\ln(x)$ | natural logarithm of $x$; $\ln(x) \triangleq \log_e(x)$ |
| $e^x$ | natural exponential of $x$; $e^{\ln x} = x$ |

General mathematical operators:

| | |
|---|---|
| $\sum_{i=1}^{n} x_i$ | $x_1 + x_2 + \cdots + x_n$ |
| $\prod_{i=1}^{n} x_i$ | $x_1 \times x_2 \times \cdots \times x_n$ |
| $\min_{x\in\mathcal{X}} f(x)$ | minimal value of the function $f(x)$ takes in the domain $\mathcal{X}$ |
| $\max_{x\in\mathcal{X}} f(x)$ | maximal value of the function $f(x)$ takes in the domain $\mathcal{X}$ |
| $\sup_{x\in\mathcal{X}} f(x)$ | the supremum of function $f(x)$ on the domain $\mathcal{X}$ |
| $\arg\min_{x\in\mathcal{X}} f(x)$ | value of $x$ in the domain $\mathcal{X}$ at which the function $f(x)$ attains its minimal value |
| $\arg\max_{x\in\mathcal{X}} f(x)$ | value of $x$ in the domain $\mathcal{X}$ at which the function $f(x)$ attains its maximal value |

General probability theory:

| | |
|---|---|
| $\mathbf{P}(X = x)$ | probability that a random variable $X$ takes on the value $x$ |
| $f_X$ | probability density function for random variable $X$; $f_X(x) \triangleq \mathbf{P}(X = x)$ |
| $F_X$ | cumulative probability function for random variable $X$; $F_X(x) \triangleq \mathbf{P}(X \leq x)$ |
| $X \sim p$ | random variable $X$ selected from distribution $p(x)$ Pr $X = x$ |
| $\mathbf{E}(X)$ | expectation of random variable $X$ |
| $\sigma_X^2$ | variance of random variable $X$ |

Algorithms:

| | |
|---|---|
| $\triangleright$ | the remainder of the line is a comment |
| $\leftarrow$ | assignment |

In a Decision Processes:

| | |
|---|---|
| $s, s'$ | states |
| $a$ | an action |
| $r_1(s,a)$ | immediate reward or lump sum reward for state $s$ and action $a$ |
| $r_2(t \mid s,a,s')$ | continuous reward rate over the sojourn time $t$ from state $s$ to $s'$ under action $a$ |
| $\mathcal{S}$ | state space (set of all states) |
| $\mathcal{A}(s)$ | set of actions available in state $s$ |
| $\mathcal{A}$ | action space (set of all actions), $\cup_{s\in\mathcal{S}}\mathcal{A}(s)$ |
| $\mathcal{K}$ | the set of admissible state-action pairs |
| $\lvert\mathcal{S}\rvert$ | number of states in the finite state space $\mathcal{S}$ |
| $\lvert\mathcal{A}\rvert$ | number of actions in the finite action space $\mathcal{A}$ |
| $\xi$ | belief state |
| $\triangle$ | belief simplex |

| | |
|---|---|
| $n$ | decision epoch |
| $N$ | planning horizon; it can be finite or $N = \infty$ |
| $A_n$ | action at decision epoch $n$ |
| $S_n$ | state at decision epoch $n$ |
| $R_n$ | reward at decision epoch $n$ |
| $\pi$ | policy (decision-making rule) |

| | |
|---|---|
| $\pi(s)$ | action taken in state $s$ under *deterministic* policy $\pi$ |
| | |
| $P(s' \mid s, a)$ | probability of transition from state $s$ to state $s'$ under action $a$ |
| $R(s, a)$ | expected immediate reward from state $s$ after action $a$ |
| $R(s, a, s')$ | expected immediate reward on transition from $s$ to $s'$ under action $a$ |
| $Q(t, s' \mid s, a)$ | probability of sojourn time $t$ and state $s'$ given that agent starts from state $s$ under action $a$ |
| $G(o \mid a, s')$ | probability of observing $o$ given that agent does action $a$ and lands in state $s'$ |
| | |
| $V^{\pi}(s)$ | value of state $s$ under policy $\pi$ (expected return) |
| $V^{*}(s)$ | value of state $s$ under the optimal policy |

# 1
# Introduction

We regard decision making under uncertainty as one of the attributes of human intelligence. Consequently, if we can get a computer to make decisions under uncertainty we can feel that it is imitating one of the aspects of human intelligence.

—Bellman (1978, p. 48)

Suppose that you need to travel by bus to a destination that is four bus stops away, and you have the option of bringing a bicycle along with you. The catch is that the bus journey is subject to different levels of traffic intensity. In low traffic, the bus can travel unimpeded, quickly reaching its goal; on the other hand, high traffic can significantly delay the bus's arrival.

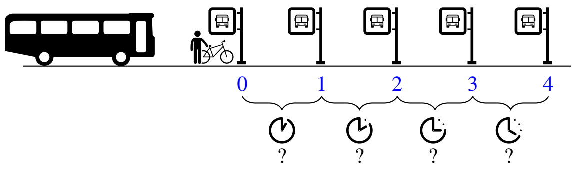


Fig. 1.1 Waiting for the bus.

At each bus stop, you face a decision: you can either stay on the bus, hoping that it will get to your destination more quickly despite the unknown traffic, or you can disembark and ride your bicycle the rest of the way. Once you make the decision to get off the bus and ride the bicycle, you are committed to this mode of transport for the rest of your journey; there is no option to reboard the bus at subsequent stops. The faster you get to your destination, the better; think of it as getting a higher reward for speedy arrival. If you decide to stay on the bus, the time it will take to get the next stop will depend on the traffic intensity. In contrast, if you decide to hop off and ride your bicycle, the travel time will be fixed, regardless of the traffic intensity.

The challenge here is to find the best policy that will maximize your reward (getting to your destination as quickly as possible) while taking into account your uncertainty about the traffic conditions. In other words, how can you make the best possible decision at each bus stop, given that you need to know how heavy the traffic is going to be?

Now, let us make this story a bit more interesting. Imagine you have just moved to the city and you need to become more familiar with the bus route or how the traffic affects travel times. You know the bus stops and the traffic levels, but you need to know how long it will take to get from one stop to the next under different traffic conditions.

In this situation, you start your journey with a lot of uncertainty. At each bus stop, you need to decide whether to stay on the bus or ride your bike, but with a good understanding of the travel times, this decision is easier.

So, what is the best way to navigate this situation? The answer involves a combination of making the best possible decision at each stop, given what you currently believe about the travel times (this is the *exploitation* part of your knowledge), and also learning from new experiences to improve your understanding of the travel times for future decisions (this is the *exploration* part of your environment).

This process of learning and adjustment, so fundamental to our own decision-making, serves as an insightful guide in the quest to teach machines to make intelligent decisions. As we grapple with uncertainty and balance the known with the exploration of the unknown, so must the algorithms we design. The more experience they accumulate, the more refined their internal models of the world become, aiding their decisions in the same way you would approach learning when to ride the bus and when to ride your bike.

As Mitchell (1997, p. 2) so eloquently puts it, learning constitutes the ability to improve one's performance of a task through experience. Within the realm of artificial intelligence, *machine learning* signifies the ambitious endeavour of devising algorithms and computational models that empower computers to learn from data and experience. This enables them to improve their performance on tasks, including making informed decisions on what actions to take, without relying upon explicit programming. However, in machine learning, the majority of research has focused on *what* to do, rather than *when* to do it.

## 1.1 Timing

The ability to make predictions lies at the very heart of the artificial intelligence problem (Schmidhuber, 2007). The crux of many planning dilemmas, including our bus and bicycle conundrum, hinges upon the use of timing mechanisms. These mechanisms aid us in foreseeing the cadence of imminent events, crucial for synchronizing strategies, decisions, and actions within our milieu. Robots, too, require a capacity for learning timing (Maniadakis and Trahanias, 2011). Alas, transmuting a continuous stream of observations into actionable insights for real-time decision-making proves to be an arduous computational challenge (Schmidhuber, 2007).

Computationally, the challenge of predicting when the next event or decision epoch boils down to discerning which fragment of a limited past ought to be employed for formulating a prediction. If we restrict ourselves to a singular stream of binary events (either a 0 or 1 at each time step), the sheer quantity of potential subsequences throughout the previous $n$ time steps is the powerset of $n$, which is exponential! For instance, state-of-the-art artificial neural networks necessitate a myriad of trials, numbering in the thousands, to learn a straightforward association separated by a mere handful of time steps (Rivest *et al.*, 2010). Astonishingly, at a hundred time steps apart, the endeavour demands millions of trials, with success remaining elusive (Gers *et al.*, 2002). Intensifying the sampling frequency exacerbates the problem's complexity as much as augmenting the interval length itself. Bereft of any foreknowledge regarding the sequence's architecture, employing artificial neural networks for predicting a binary event is apt to be $\mathcal{NP}$-complete (Blum and Rivest, 1992), signifying a potential need for a trial count exponential in relation to the temporal distance, as measured in time steps, between the essential inputs and the subsequent prediction.

In a rather delightful contrast, learning when events will occur in the natural world appears to be quite effortless. Animals are capable of grasping stimulus-reward associations within merely tens or hundreds of trials, spanning intervals from mere milliseconds to several minutes (Gallistel and Gibbon, 2000). Intriguingly, as long as the ratio between the interstimulus interval (the time between a stimulus announcing a reward and the reward itself) and the intertrial interval (the time between two conditioning trials) remains constant, the number of trials required for conditioning is also constant ($\mathcal{O}(1)$) (Gallistel and Gibbon, 2000). Even more fascinating is that precise timing is already mastered when a conditioned response emerges (Balsam *et al.*, 2002; Balci *et al.*, 2009)! Evolution has indeed devised ways for animals to learn timing in a constant number of trials, irrespective of the time scale, suggesting an alternative computational approach to learning timing. In fact, animals can encode in their brains these temporal relations between reward and events from just a single experience (Balsam and Gallistel, 2009). Exciting recent developments employing new learning rules focused on timing have shown substantial improvements in learning speed, approaching the performance of animals (Rivest and Bengio, 2011; Simen *et al.*, 2011; Luzardo *et al.*, 2013, 2017; Rivest and Kohar, 2020).

Timing encompasses the following two aspects:

(a) *temporal learning*—the tracking of time when an event occurs (Church, 2012);
(b) *temporal control*—the control of when an action is performed by the agent.

**Example 1.1.1 — Temporal learning.** Should we take the elevator or take the stairs? This problem requires us to have some experience to estimate how long we will have to wait before the elevator arrives on average; we need to track when the elevator arrives over time. We also can track how long it takes us to climb the stairs on average. With these two averages, we can decide as soon as we arrive in the lobby to either wait or take the stairs by selecting the minimum of the two averages.

**Example 1.1.2 — Temporal control.** When should we stop waiting for an elevator and take the stairs? In this case, we are going to wait initially in the lobby for the elevator. We are exerting control over when we will switch from one action (waiting) to another action (taking the stairs).

Temporal learning and temporal control are indeed vital facets of the intelligence problem. It is of paramount importance to not only contemplate the essence of *what* we should do, but also considerable thought to *when* we ought to execute our carefully crafted plans. While both temporal learning and temporal control are integral to the intelligence problem, this thesis primarily focuses on temporal learning. In this thesis, we delve into understanding how an agent can track and learn from the timing of events in its environment. However, it's important to note that temporal control is not entirely overlooked. We touch upon this aspect, particularly in the context of when an agent should switch from one action to another, though it is not the main focus of our study.

## 1.2 Sequential decision processes

This optimization of a problem over time arises in many areas. It is found in engineering for the control of heating systems and landing aircraft; managing inventories of drugs with limited shelf life; acquiring and selling assets on the market; scheduling and routing of deliveries or information in a network. These are all sequential decision processes. The name is aptly suited as each of these problems or processes involves making not only decisions but decisions made in the correct order. While sequential decision processes can be easy to formulate (if not subtle), it is a whole different matter to solve them.

Sequential decision processes are conventional in modern life. For instance, the law requires two persons $A$ and $B$, otherwise known as lawyers, to argue about hypothesis $H$ (the guilt of the accused). Person $A$ is paid to pretend that the probability of $H$ is 0, whilst person $B$ pretends the probability of $H$ is 1. Experiments are performed that consist of asking witnesses questions. A sequential decision process ensues as $A$ and $B$'s further questions are influenced by previous answers as well as whom to call to the stand next. The jury is left to decide whether to accept or to reject the hypothesis based on their final probability estimate of $H$ (Good, 1952, p. 112).

The theory of sequential decision processes was created to solve problems that arise from the study of multi-stage planning, which can be described informally in the following manner: We have a physical environment whose state at a point in time is specified by what we call a *state* variable. In this environment, we have an *agent*—either a person or a machine—that at certain times predetermined by the environment, the agent must decide what *action* to take which in turn affects the environment; these points in time are called *decision epochs*. The agent's action will cause a transition from one state to another, and therefore facilitates the agent's movement through the system as it accumulates *rewards* or costs. The goal of the agent is to find a sequence of state-actions, called a *policy*, that will optimize the desired performance measure such as maximizing reward or minimizing cost (Bellman, 1954, p. 503).

Following the development of sequential decision process theory, Bellman (1957b) laid the foundations of the Markov Decision Process (MDP), which emerged as a significant mathematical framework for handling these decision-making problems. Bellman (1957a) introduced the idea of dynamic programming as a method to solve MDPs, enabling the optimization of actions over time based on a known model of the environment. This process set the stage for the field of Reinforcement Learning (RL) in the 1980s (Sutton, 1984, 1988; Sutton and Barto, 1998). Unlike traditional dynamic programming, RL focused on situations where the MDP was not fully known to the agent, in particular, the state transition probabilities and the reward function. This introduced a new layer of complexity, as the agent now needed to learn about its environment while also trying to optimize its actions. This shift marked a critical step in the evolution of decision-making algorithms, as it moved us closer to solving real-world problems where uncertainty and incomplete information are the norm.

## 1.3 Thesis Objective

The objective of this thesis is to expand the capabilities of reinforcement learning, enabling it to tackle more complex, real-world problems where timing plays a critical role. Existing models have addressed Markov Decision Processes (MDPs) with time considerations, known as Semi-Markov Decision Processes (SMDPs), and MDPs with partial observability (POMDPs). However, the work on models that incorporate both time and partial observability, namely Partially Observable Semi-Markov Decision Processes (POSMDPs), has been limited. In particular, few efforts have been made to address scenarios where time is the key observation.

Moreover, while the trade-off between exploration and exploitation has been thoroughly investigated in the MDP context, this essential balance has not been extensively studied in the SMDP context. This thesis addresses these gaps by presenting new mathematical models and algorithmic developments. The central contributions of this work enable reinforcement learning in time-sensitive and partially observable problem domains, marking a significant step forward in expanding the reach of reinforcement learning techniques into new realms of complexity and practical relevance.

## 1.4 Thesis Contributions

The first contribution of this thesis is the development of a novel algorithm, ChronosPerseus, which utilizes randomized point-based value iteration with importance sampling to solve Partially Observable Semi-Markov Decision Processes (POSMDPs). By maintaining a set of sampled times and weighting them by their likelihood within a single backup, ChronosPerseus significantly reduces computational complexity. Demonstrated through the lens of both episodic and non-episodic problems, the algorithm exhibits its capacity to handle a diverse array of issues, including mixed-observability, discrete or continuous observation space, and a mixture of fixed and stochastic continuous sojourn times.

The second key contribution of this work is the introduction of a novel framework in Bayesian Reinforcement Learning—the Bayes-adaptive semi-Markov decision process (BA-SMDP). This framework provides a means for learning sojourn time distributions, offering a dynamic approach to solving problems where timing is paramount. The BA-SMDP framework is explored through four unique approaches:

(a) Learning the sojourn time distribution parameters using a count array to record the number of each sojourn time occurrence for a particular $(s, a, s')$ transition, where sojourn times come from a finite set;
(b) Learning the mixture of known sojourn time distributions with unknown proportions;
(c) Learning the mixture of known SMDPs with unknown proportions; and
(d) Learning the unknown continuous sojourn-time distribution parameters.

These approaches enable optimal learning, providing a means to find the policy that strikes the best balance between exploration and exploitation. This optimal tradeoff allows for superior sample efficiency, thus leading to more effective decision-making under uncertainty. Consequently, the breadth of applicability and effectiveness of reinforcement learning in time-sensitive problem domains is substantially enhanced.

The third contribution of this work is the creation of a novel conjugate prior for the mean parameter of the inverse Gaussian distribution. This advancement is significant as it enables us to effectively track the uncertainty surrounding the parameter of the inverse Gaussian distribution, a vital aspect when dealing with unknown continuous sojourn-time distributions.

Altogether, these contributions take a step forward in the field of reinforcement learning, specifically addressing problems involving timing.

## 1.5 Outline

This thesis unfolds over seven chapters. It begins with Chap. 2, where we lay the foundation with an in-depth discussion of the Markov Decision Process. The narrative then advances to Chap. 3, where the concept of MDP is extended to scenarios where states are only partially observable. In Chap. 4, this knowledge base is extended to encompass the semi-Markov decision process, a variant of MDP that accounts for random sojourn times. Our journey then leads us to Chap. 5, where we encounter the framework for the partially observable semi-Markov decision process (POSMDP) and unveil our novel ChronosPeresus algorithm. Chapter 6 then presents our pioneering Bayes-adaptive semi-Markov decision process (BA-SMDP) framework, along with four distinct methods for managing a BA-SMDP. Our journey culminates in Chap. 7, where we summarize the key contributions of the thesis and outline potential paths for future research. For readers interested in a deeper dive, Appendix A contains documented code for all the algorithms discussed in this thesis.

# 2
# The Markov Decision Process (MDP) Model

In this chapter, we undertake an examination of the Markov Decision Process (MDP) framework. This framework serves as a cornerstone in the field of reinforcement learning, particularly in cases where decisions are made sequentially under uncertain conditions (Bellman, 1957a; Howard, 1971; Puterman, 1994; Bertsekas and Tsitsiklis, 1996; Sutton and Barto, 2018). Our exposition commences with a careful delineation of the MDP model, followed by a detailed exploration of the computational strategies associated with it. Furthermore, in order to highlight the constraints of MDPs when dealing with temporal considerations, we introduce an illustrative problem scenario—the elevator problem. This serves as an illustrative example of the challenges associated with the integration of time in the classical MDP framework.

## 2.1 The Framework

We begin with the definition of an MDP, which was first introduced by Bellman (1957b).

**Definition 2.1.1 — MDP.** A Markov decision process (MDP) is an 7-tuple

$$\langle \mathcal{S}, \mathcal{A}, \mathcal{K}, P, R, \gamma, N \rangle$$

where

| | |
|---|---|
| $\mathcal{S}$ | is the Borel *state space*, and the elements of $\mathcal{S}$ are called *states* |
| $\mathcal{A}$ | is the Borel *action space*, and the elements of $\mathcal{A}$ are called *actions*. To each $s \in \mathcal{S}$, we associate a nonempty Borel-measurable subset $\mathcal{A}(s) \subseteq \mathcal{A}$, whose elements are the *admissible actions* for the agent when the process is in state $s$ |
| $\mathcal{K}$ | is the set of admissible state-action pairs, and it is assumed to be a Borel subset $\mathcal{K} \subseteq \mathcal{S} \times \mathcal{A}$. In other words, $$\mathcal{K} \triangleq \{(s,a) \mid s \in \mathcal{S}, a \in \mathcal{A}(s)\}.$$ |
| $P(\mathrm{d}s' \mid s, a)$ | is the state transition Borel-measurable stochastic kernel (or conditional probability measure) on $\mathcal{S}$ given $\mathcal{K}$ |
| $R(s,a)$ | is the per-stage (bounded Borel-measurable) reward function given $\mathcal{K}$ |
| $\gamma$ | is the discounting rate where $\gamma \in [0, 1)$ |
| $N$ | is the planning horizon. It could be finite, or $N = \infty$. |

Given the model in Definition 2.1.1, the dynamics of the MDP proceed according to Algorithm 1. This involves at each decision epoch $n$ choosing an action $a_n$, accruing a lump sum reward $R(s_n, a_n)$ as the state changes from $s_n$ to $s_{n+1}$.

**Algorithm 1** Dynamics of MDP

```
In the beginning $n = 0$, the agent observes it is in state $s_0$.
For each decision epoch $n = 1, 2, \dots, N$:
    (a) Based on the history
            $h_0 = (s_0)$
            $h_n = (s_0, a_1, \dots, s_{n-1}, a_n, s_n)$,
        the agent performs action
            $a_n = \pi_n(h_n) \in \mathcal{A} \qquad n = 1, 2, \dots, N.$
        Here $\pi_n$ denotes a policy that the agent uses at decision epoch $n$.
    (b) The agent obtains a reward $R(s_{n-1}, a_n)$ for choosing action $a_n$ at decision
        epoch $n$.
    (c) The state evolves randomly with transition probability
            $P(s' \mid s, a) = \mathbf{P}(S_{n+1} = s' \mid S_n = s, A_n = a)$
        to the next state $s_{n+1}$ that decision epoch $n + 1$.
    (d) The agent updates its history as
            $h_{n+1} = (h_n, a_{n+1}, s_{n+1})$.
        If $n < N$, then set $n$ to $n + 1$, and go back to step (a).
        If $n = N$, then the agent receives the last reward and the process terminates.
```

## 2.2 Stochastic Kernels

A stochastic kernel is a mathematical function that defines transition probabilities between states in a stochastic process. Essentially, given a state and an action, it provides the probability that the process transitions to that event from the given state, satisfying certain measure-theoretic properties.

**Definition 2.2.1 — Stochastic kernel.** A stochastic kernel is a function $K$ defined on $\mathcal{X} \times \mathcal{B}(\mathcal{X})$ such that

(a) for every $x \in \mathcal{X}$, $K(x, \cdot)$ is a probability measure; and
(b) for every $E \in \mathcal{B}(\mathcal{X})$, $K(\cdot, E)$ is measurable.

### 2.2.1 Discrete Spaces

If the state space $\mathcal{S}$ and action space $\mathcal{A}$ are discrete, then the stochastic kernel $P(\cdot \mid s, a)$ is defined by the values on the singletons $P(s' \mid s, a)$, where

$$P(s' \mid s, a) = \mathbf{P}(S_{n+1} = s' \mid S_n = s, A_n = a). \tag{2.2.1}$$

Since each $P(s' \mid s, a)$ is a probability, it must satisfy

$$0 \le P(s' \mid s, a) \le 1 \qquad \forall (s, a) \in \mathcal{K}, \forall s' \in \mathcal{S}. \tag{2.2.2}$$

Additionally, since the agent must occupy one of the states in the process,

$$\sum_{s' \in \mathcal{S}} P(s' \mid s, a) = 1, \qquad \forall (s, a) \in \mathcal{K}. \tag{2.2.3}$$

The stochastic matrix $P$ is also known as *transition probability matrix*. The $|\mathcal{S}|^2$ probabilities for an action $a$ can be represented by a square $|\mathcal{S}| \times |\mathcal{S}|$ matrix

$$P(\cdot \mid \cdot, a) = \begin{array}{c} \\ s_1 \\ s_2 \\ \\ s \\ \\ s_{|\mathcal{S}|} \end{array} \begin{array}{c} \begin{array}{ccccccc} s_1 & s_2 & & s' & & s_{|\mathcal{S}|} \end{array} \\ \begin{bmatrix} P(s_1 \mid s_1, a) & P(s_2 \mid s_1, a) & \cdots & P(s' \mid s_1, a) & \cdots & P(s_{|\mathcal{S}|} \mid s_1, a) \\ P(s_1 \mid s_2, a) & P(s_2 \mid s_2, a) & \cdots & P(s' \mid s_2, a) & \cdots & P(s_{|\mathcal{S}|} \mid s_2, a) \\ \vdots & \vdots & & \vdots & & \vdots \\ P(s_1 \mid s, a) & P(s_2 \mid s, a) & \cdots & P(s' \mid s, a) & \cdots & P(s_{|\mathcal{S}|} \mid s, a) \\ \vdots & \vdots & & \vdots & & \vdots \\ P(s_1 \mid s_{|\mathcal{S}|}, a) & P(s_2 \mid s_{|\mathcal{S}|}, a) & \cdots & P(s' \mid s_{|\mathcal{S}|}, a) & \cdots & P(s_{|\mathcal{S}|} \mid s_{|\mathcal{S}|}, a) \end{bmatrix} \end{array} \tag{2.2.4}$$

Each entry in the matrix must in the interval [0, 1] (Eq. (2.2.2)), and the sum of the rows must be one (Eq. (2.2.3)).

### 2.2.2 Continuous Spaces

We will see later in Chap. 3 that a partially observable MDP (POMDP) can be reformulated as a continuous state MDP, so this formalism that we will present now will have some importance later.

If the state space $\mathcal{S}$ is continuous, then the stochastic kernel $P(\cdot \mid s, a)$ is a conditional density $p$ such that

$$P(S \mid s, a) = \int_{s' \in S} p(s' \mid s, a)\, \mathrm{d}s' \tag{2.2.5}$$

where $S$ denotes any measurable set in $\mathcal{S}$.

Notice that we cannot just calculate the probability of landing on a point in continuous space (like we did in the discrete case in Eq. (2.2.1)), because the probability of landing on a point in a continuous space is zero. We must think of landing in an area or a set; then, we can apply our probability measure which is essentially integrating over that landing set.

To ensure that $P(\cdot \mid s, a)$ is a probability measure, it is also required that

$$P(\mathcal{S} \mid s, a) = \int_{s' \in \mathcal{S}} p(s' \mid s, a)\, \mathrm{d}s' = 1 \tag{2.2.6}$$

and

$$P(\varnothing \mid s, a) = 0. \tag{2.2.7}$$

## 2.3 The Reward Function

The reward function is a scalar signal from the environment, which serves as a means for feedback to the agent about the outcomes of its actions. The goal of the agent is to maximize its reward by adjusting its actions accordingly.

Mathematically, the reward function $R(s, a)$ maps state-action pairs $(s, a) \in \mathcal{K}$ to the real numbers $\mathbb{R}$. When $R(s, a)$ is negative, it is often referred to as a cost function. From the perspective of the MDP, it does not matter how the reward is accrued between decision epochs. We only require that the reward value or expected value be known before selecting an action, and that it is not affected by future actions. For MDPs, it is standard to think of the reward as a lump sum received prior to the next decision epoch (Puterman, 1994, p. 24). Later, when we look at SMDPs (Sec. 4) and POSMDPs (Sec. 5), in addition to the lump sum, the reward can be continuously accumulated during the sojourn time.

If the reward depends on the landing state $s' \in \mathcal{S}$, we let $R(s, a, s')$ denote the value of the reward received when starting from state $s$, performs action $a \in \mathcal{A}(s)$, and then lands in state $s'$. If $\mathcal{S}$ is a continuous state space, then the reward $R(s,a)$ can be computed as the expected value given by

$$R(s,a) \triangleq \int_{s'\in\mathcal{S}} P(\mathrm{d}s' \mid s,a) R(s,a,s'), \tag{2.3.1}$$

or if $\mathcal{S}$ is a discrete space then

$$R(s,a) \triangleq \sum_{s'\in\mathcal{S}} P(s' \mid s,a) R(s,a,s'). \tag{2.3.2}$$

## 2.4 The Value Function

While the reward function in Sec. 2.3 tells us what is good in the immediate sense, we need another function—the value function—which will tell us what is good in the long run (Sutton and Barto, 1998, p. 8). The value function evaluates the expected future reward as a function of the current state. Using the value function, the agent can predict long-term consequences of available actions, and it can use this as a planning tool to generate optimal decisions.

The Principle of Optimality is stated as follows by Bellman:

> An optimal policy has the property that whatever the initial state and initial decisions are, the remaining decisions must constitute an optimal policy with regard to the state resulting from the first decisions (Bellman, 1954, p. 504).

What Bellman is requiring is the markov property.

**Definition 2.4.1 — Markov property.** Given a stochastic kernel $\kappa$, a controlled stochastic process $(S_n, A_n)_{n\in\mathbb{N}\cup\{0\}}$ is said to have the Markov property if for any $n$, the conditional distribution of $S_n$ given $s_0, a_0, s_1, a_1, \ldots, s_{n-1}, a_{n-1}$, is the same as the distribution $S_n$ given $s_{n-1}, a_{n-1}$; that is,

$$\begin{aligned}&\mathbf{P}(S_{n+1}\in E \mid S_0 = s_0, A_0 = a_0, \ldots, S_{n-1} = s_{n-1}, A_{n-1} = a_{n-1})\\ =&\mathbf{P}(S_{n+1}\in E \mid S_{n-1} = s_{n-1}, A_{n-1} = a_{n-1})\\ =&\int_E P(\mathrm{d}s \mid s_{n-1}, a_{n-1})\end{aligned}$$

where $E \in \mathcal{B}(\mathcal{S})$.

A stochastic process is said to have the *Markov property* if the conditional probability distribution of the next state of the process depends only upon the current state and action. In other words, the probability of the next state does not depend on the entire history of the process, but only on the current state and action.

In the case where $\mathcal{S}$ is a discrete state space (with the discrete $\sigma$-algebra), and a discrete number of decision epochs,

$$\begin{aligned}\mathbf{P}(S_{n+1} = s_{n+1} \mid S_n = s_n, A_n = a_n) = \mathbf{P}(S_{n+1} = s_{n+1} \mid S_0 = s_0, A_0 = a_0,\\ \ldots, S_n = s_n, A_n = a_n)\end{aligned}$$

### 2.4.1 The General Form

The value of a state $s$ for an MDP following a policy $\pi$ is

$$V^\pi(s) \triangleq \underbrace{R\big(s,\pi(s)\big)}_{\text{immediate reward}} + \underbrace{\gamma \int_{s'\in\mathcal{S}} P\big(\mathrm{d}s' \mid s,\pi(s)\big) V^\pi(s')}_{\text{sum of discounted future rewards}} \qquad \forall s \in \mathcal{S}, \tag{2.4.1}$$

where $\gamma \in [0,1]$ is the discount rate.

### 2.4.2 Discrete states and discrete actions

The value of a state $s$ for an MDP following a policy $\pi$ is

$$V^\pi(s) = \underbrace{R\big(s, \pi(s)\big)}_{\text{immediate reward}} + \underbrace{\gamma \sum_{s' \in \mathcal{S}} P\big(s' \mid s, \pi(s)\big) V^\pi(s')}_{\text{sum of discounted future rewards}} \qquad \forall s \in \mathcal{S}, \tag{2.4.2}$$

where $\gamma \in [0, 1]$ is the discount rate.

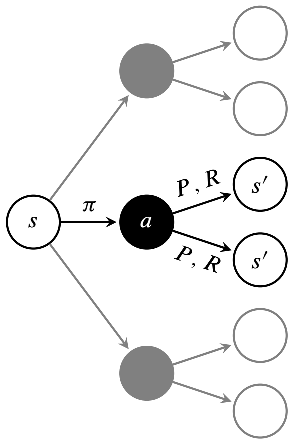


Fig. 2.1 Backup diagram for the value function $V^\pi$ following policy $\pi$.

Figure 2.1 is an example of a *backup diagram* and it represents Eq. (2.4.2). It helps to visualize the relations that form part of the update or backup operations: these operations transfer value information back to a state (or state-action pair) from its successor states (or state-action pairs). Each white node represents a state, and each black node represents a state-action pair. Beginning on the left with state $s$, the agent can take any action $a$ available in $s$—there are three shown in the diagram—but it will select the action corresponding to policy $\pi$ (the rest of the action paths are in gray). The environment will respond with a successor state $s'$—two are shown in the diagram—according to the environment's dynamics given by $P(s' \mid s, a)$, along with a reward $r(s, a, s')$.

The value function for a policy (Eq. (2.4.2)) can be expressed concisely using matrices:

$$\begin{bmatrix} \vdots \\ V^\pi(s) \\ \vdots \end{bmatrix} = \begin{bmatrix} \vdots \\ R\big(s, \pi(s)\big) \\ \vdots \end{bmatrix} + \gamma \begin{bmatrix} \ddots & & \\ & P\big(s' \mid s, \pi(s)\big) & \\ & & \ddots \end{bmatrix} \begin{bmatrix} \vdots \\ V^\pi(s') \\ \vdots \end{bmatrix} \tag{2.4.3}$$

or simply,

$$V^\pi = R^\pi + \gamma P^\pi V^\pi. \tag{2.4.4}$$

Note that there is a different matrices $R\big(\cdot, \pi(\cdot)\big)$ and $P\big(\cdot \mid \cdot, \pi(\cdot)\big)$ for each policy $\pi$. With the notation of Eq. (2.4.4), we can write

$$\begin{aligned} V^\pi &= R^\pi + \gamma P^\pi V^\pi \\ V^\pi - \gamma P^\pi V^\pi &= R^\pi \\ (I - \gamma P^\pi) V^\pi &= R^\pi. \end{aligned}$$

Since $P^\pi$ is a stochastic matrix, then all of its associated eignvalues are less than one. Therefore, the eignvalues of the matrix $(I - \gamma P^\pi)$ are bounded by being greater than $1-\gamma$. This guarantees that $(I - \gamma P^\pi)$ is invertible[1] and makes it possible to compute

$$V^\pi = (I - \gamma P^\pi)^{-1} R^\pi. \tag{2.4.5}$$

The optimal Bellman equation, or the optimal value of a state $s$ for an MDP following an optimal policy $\pi^*$ is given by

$$V^*(s) = \max_{a \in \mathcal{A}(s)} \left[ R(s,a) + \gamma \sum_{s' \in \mathcal{S}} P(s' \mid s,a) V^*(s') \right] \qquad \forall s \in \mathcal{S}. \tag{2.4.6}$$

Figure 2.2 gives the corresponding backup diagram for Eq. (2.4.6). We can also

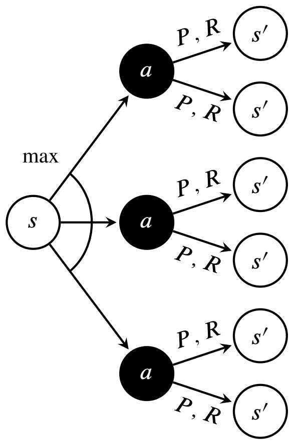


Fig. 2.2 Backup diagram for the value function $V^*$

express the optimal Bellman equation, or the optimal value function using matrices:

$$V^* = \max_{\pi \in \Pi} [R^\pi + \gamma P^\pi V^\pi]. \tag{2.4.7}$$

## 2.5 Computational Methods

Value iteration and policy iteration are two standard approaches to solving a completely specified MDP model.

### 2.5.1 Value iteration

The value iteration algorithm is an application of Bower's fixed point theorem. The value iteration algorithm finds the stationary $\epsilon$-optimal policy by approximating the fixed point of the value function?

The value iteration algorithm is due to Bellman (1957a).

If we are working with a lookup table representation, for an MDP with $|\mathcal{S}|$ states and $|\mathcal{A}|$ actions, then value iteration requires $O(|\mathcal{S}|^2|\mathcal{A}|)$ computations per iteration.

We stop the algorithm when

$$||V_{k+1} - V_k||_{\infty,\mathcal{S}} < \frac{\epsilon(1-\gamma)}{2\gamma} \tag{2.5.1}$$

where $||V||_{\infty,\mathcal{S}}$ is the supremum norm defined by

$$||V||_{\infty,\mathcal{S}} = \sup_{s \in \mathcal{S}} |V(s)|. \tag{2.5.2}$$

[1] Inverting the matrix can be accomplished with Gauss-Jordan elimination with computational complexity $\mathcal{O}(N^3)$, or Strassen's algorithm with $\mathcal{O}(N^{2.807})$.

**Algorithm 2** Value iteration algorithm for the infinite horizon, Bellman (1957a)

```
1: Select an arbitrary value function $V_0 \in \mathcal{V}$, specify $\epsilon > 0$ and $0 < \gamma < 1$, and set
   $n = 0$.
2: repeat
3:     for $s \in \mathcal{S}$ do
4:         $V_{k+1}(s) \leftarrow \max_{a \in \mathcal{A}(s)} \left[ R(s,a) + \gamma \sum_{s' \in \mathcal{S}} P(s' \mid s,a) V_k(s') \right]$
5:     end for
6:     $k \leftarrow k + 1$
7: until $||V_{k+1} - V_k||_{\infty,\mathcal{S}} < \frac{\epsilon(1-\gamma)}{2\gamma}$
8: for $s \in \mathcal{S}$ do
9:     $\pi_\epsilon(s) \leftarrow \arg\max_{a \in \mathcal{A}(s)} \left[ R(s,a) + \gamma \sum_{s' \in \mathcal{S}} P(s' \mid s,a) V_{k+1}(s') \right]$
10: end for
```

### 2.5.2 Policy iteration

The policy iteration algorithm, given in Algorithm 3, is due to Howard (1958). The algorithm finds an optimal policy in a two-step iteration cycle:

(a) the *policy evaluation* step calculates the value function using the current policy $\pi$; and

(b) the *policy improvement* step acts greedy by selecting the action that maximizes the value function from the previous step.

The cycle repeats until the algorithm does not make any more changes to the policy; see Fig. 2.3. So in contrast with the value iteration algorithm, the policy iteration algorithm computes a policy at each iteration instead of just once at the end.

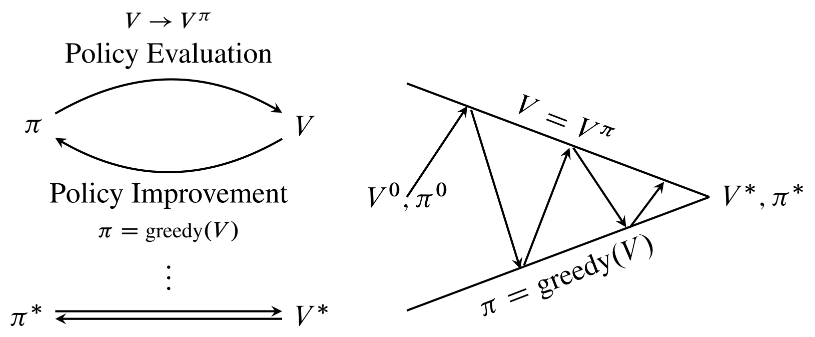


Fig. 2.3 Policy iteration algorithm. Adapted from Sutton and Barto (1998).

## 2.6 $Q$-Learning

Numerical solution was considered the last resort of an incompetent mathematician. The opposite, of course, is true. Once working in the area, it is very quickly realized that far more ability and sophistication is required to obtain a numerical solution than to establish the usual existence and uniqueness theorems. It is far more difficult to obtain an effective algorithm than one that stops with a demonstration of validity.

—Bellman (1984, pp. 184–185)

**Algorithm 3** Policy iteration algorithm, Howard (1958)

```
1: Select an arbitrary policy $\pi' \in \Pi$.
2: repeat
3:     $\pi \leftarrow \pi'$
4:     for $s \in \mathcal{S}$ do    ▷ Policy Evaluation: Compute the value function of policy $\pi$
5:         $V^{\pi}(s) \leftarrow R\big(s, \pi(s)\big) + \gamma \sum_{s' \in \mathcal{S}} P(s' \mid s, a) V^{\pi}(s')$
6:     end for
7:     for $s \in \mathcal{S}$ do    ▷ Policy Improvement: Select the best action for each state
8:         $\pi'(s) \leftarrow \arg\max_{a \in \mathcal{A}(s)} \Big[ R(s, a) + \gamma \sum_{s' \in \mathcal{S}} P(s' \mid s, a) V^{\pi}(s') \Big]$
9:     end for
10: until $\pi = \pi'$
```

First proposed by Watkins (1989), $\mathcal{Q}$-learning is a simple learning algorithm that an agent can use to discover how to act optimally in a Markov environment without initially knowing the dynamics of the environment.

This learning algorithm is a numerical method for estimating a cost function or finding an optimal policy when analytical methods are intractable.

This numerical approach is attractive especially when parts of the Markov decision process, such as the transition probability matrix or the reward function, is unknown but either the process can be simulated with chosen actions, or the physical system can be observed with chosen controls. For this reason, Watkins and Dayan (1992, p. 279) calls $\mathcal{Q}$-learning a model-free based approach to finding the optimal policy in a Markov decision process.

An agent in a particular state attempts to perform an action, and evaluates its consequences in terms of an immediate reward or cost it receives *and* the agent estimates the value of the state that it is now in. By repeatedly visiting every state *and* trying all the possible actions, over time, the agent gradually learns the best overall policy judged by long-term discounted reward (Watkins and Dayan, 1992, p. 279).

For some arbitrary policy $\pi$, we can define the state-action value function or $Q$-values as

$$Q^{\pi}\big(s, \pi(s)\big) = R\big(s, \pi(s)\big) + \gamma \sum_{s' \in \mathcal{S}} P\big(s' \mid s, \pi(s)\big) Q^{\pi}\big(s', \pi(s')\big). \tag{2.6.1}$$

The $Q$-value is the total expected discounted reward in state $s$, performing action $a$, and then following the policy $\pi$ afterwards. The corresponding optimal state-action value function is

$$\mathcal{Q}^*(s, a) = \mathcal{Q}^{\pi^*}(s, a) \quad \forall (s, a) \in \mathcal{K} \tag{2.6.2}$$

$$= \max_{a \in \mathcal{A}_s} \Big[ R(s, a) + \gamma \sum_{s' \in \mathcal{S}} P\big(s' \mid s, a\big) Q^{\pi}(s', a') \Big]. \tag{2.6.3}$$

In $\mathcal{Q}$-learning, the agent explores the world in an episode in a sequence of steps. (This can be viewed as an iterative stochastic algorithm.) Once the agent receives the reward, the episode restarts.

At step $n$ of the episode, the agent

(a) observes the current state $s$
(b) selects and performs an action $a$
(c) observes the next state $s'$
(d) receives an immediate reward $r$

(e) adjusts the previous $\mathcal{Q}$ values using a learning rate $\alpha_n$ by the following

$$\mathcal{Q}_{n+1}(s_n, a_n) = (1-\alpha_n)\mathcal{Q}_n(s,a) + \alpha_n \left[ r_n(a_t) + \gamma \max_{a \in \mathcal{A}(s_{n+1})} \mathcal{Q}_n(s_{n+1}, a) \right] \tag{2.6.4}$$

This forms a look-up representation for all of the values $\mathcal{Q}_n(s,a)$.

**Algorithm 4** $\mathcal{Q}$-Learning, Watkins (1989)

```
Require: Discount factor $\gamma$, and learning parameter $\alpha$
 1: Initialize an arbitrary $\mathcal{Q}$-function, such as $\mathcal{Q}(s,a) = 0, \forall (s,a) \in \mathcal{K}$.
 2: for each episode do
 3:     Randomly select a starting state $s$
 4:     repeat
 5:         Randomly select and perform action $a \in \mathcal{A}_s$
 6:         Receive immediate reward $r$ and observe next state $s'$
 7:         $\mathcal{Q}(s,a) \leftarrow \mathcal{Q}(s,a) + \alpha \left[ r + \gamma \max_{a' \in \mathcal{A}_{s'}} \mathcal{Q}(s', a') - \mathcal{Q}(s,a) \right]$
 8:         $s \leftarrow s'$
 9:     until $s$ is the goal state
10: end for
```

The $\mathcal{Q}$-Learning algorithm is a stochastic iterative form of value iteration. We can see that (2.6.4) has the form of a Robbins and Monro (1951) algorithm, which suggests the application of stochastic approximation theory.

The $\mathcal{Q}$-Learning value iteration algorithm is

$$\mathcal{Q}_{t+1}(s_n, a_n) = (1-\gamma_t)\mathcal{Q}_t(s_n, a_n) + \gamma_t \left( r(s_t, a_t, s_{t+1}) + \alpha \max_{a_{t+1} \in \mathcal{A}(s_{t+1})} \mathcal{Q}_t(s_{t+1}, a_{t+1}) \right) \tag{2.6.5}$$

where $\alpha$ is the discount factor (Watkins and Dayan, 1992, p. 57).

## 2.7 Optimal Stopping Problems

An optimal stopping problem—a form of temporal control—is a sequential decision process where an agent must determine the optimal time to halt the ongoing process. At each decision epoch, the agent observes the current state and must decide whether to continue or stop the process. Stopping incurs a certain cost, represented as a negative reward. The crux of the decision is not solely about which action to take, but critically about when to execute it. The objective is to pinpoint the optimal stopping time that will maximize the total expected reward over the decision horizon. This concept aligns with the notion of temporal control, which emphasizes the timing of an action in the decision-making process.

One possible way to solve this with a Markov decision process is to augment the state space with discrete time. However, this increases the state space dimensions exponentially, which leads to the curse of dimensionality.[2] Because of the Markovian assumption, any information that is needed to make a decision must be in the state information. For example, if there was information that was needed from a previous stage, that information must be dragged along into the new state. Not only have we increased the state space, we have also increased the computational time required to find an optimal policy.

[2]This was a term Richard Bellman had coined in dynamic programming when the size of the state space grows so large that it is no longer tractable computationally.

Let us consider a relatable example that we have crafted to illustrate the concept of an optimal stopping problem and how the state space expands with each time step.

**Example 2.7.1 — Elevator Problem (MDP version).** Suppose that when an agent arrives at the lobby (on floor 1), $N$ minutes remain before its meeting begins. If the elevator arrives before the meeting starts, it earns reward $r$. However, the agent does not enjoy waiting, so to reflect its dislike it incurs a cost $c$ each minute it waits. The agent can see on which floor, $s$, the elevator is on currently by the floor indicator above the elevator door. It also knows the conditional probability $\mathbf{P}(S_{n-1} = s' \mid S_n = s, \text{wait})$, which represents the probability that the elevator will descend to floor $s'$ in the next minute given that the elevator is currently on floor $s$. At each minute $n$, the agent must decide either to wait for the elevator, or to give up and to take the stairs; this decision to stop or to continue is what makes this an optimal stopping problem. The action of taking the stairs incurs a heavy cost of $C$ units. Suppose that when the agent arrives and pushes the elevator button, the descending elevator is on floor $S$. What is the optimal policy that the agent should follow?

**Solution** *Using dynamic programming*. We can model this problem using the MDP framework. The state space consists of the floor numbers and the time in minutes (or the decision epoch index),

$$\mathcal{S} = \underbrace{\{1, 2, \ldots, S\}}_{\text{floor}} \times \underbrace{\{N-1, N-2, \ldots, 2, 1, 0\}}_{\text{remaining time}},$$

and the number of states is

$$|\mathcal{S}| = S \times N.$$

A typical state would be the pair $(s, n) \in \mathcal{S}$, where the elevator is on floor $s$, and the remaining time $n$ until the meeting. The action space consists of two actions

$$\mathcal{A} = \{\text{wait}, \text{stairs}\}.$$

The transition probability matrix is given by

$$P\big((s', n-1) \mid (s, n), \text{wait}\big) = \mathbf{P}(S_{n-1} = s' \mid S_n = s, A_n = \text{wait}), \tag{2.7.1}$$

and

$$P\big((s', n-1) \mid (s, n), \text{stairs}\big) = \mathbf{P}(S_{n-1} = 1 \mid S_n = s, A_n = \text{stairs}). \tag{2.7.2}$$

The reward function is

$$R\big((s,n), a\big) = \begin{cases} r - C, & (a = \text{stairs}); \\ rP\big((1, n-1) \mid (s,n), \text{wait}\big) - c, & (a = \text{wait}), \end{cases} \tag{2.7.3}$$

for all $s \in \{1, 2, \ldots, S\}$ and $n \in \{N-1, N-2, \ldots, 2, 1, 0\}$. Intuitively, this follows since if the agent chooses to take the stairs, it earns reward $r$ but incurs the heavy cost $C$. On the other hand, if the agent waits for a minute, it will incur a waiting cost of $c$ and will receive reward $r$ with probability $P\big((1, n-1) \mid (s,n), \text{wait}\big)$. Thus, if the agent waits, its net reward is $rP\big((1, n-1) \mid (s,n), \text{wait}\big) - c$. We let $n = 0$ be the beginning of the problem and the last decision epoch $n = N-1$ be the end of the problem; $N < \infty$: therefore, it is a finite horizon problem. This completes our description of the elevator problem as an MDP.

Since this is an MDP, we can use dynamic programming to determine an optimal policy that will maximize its expected net reward (reward minus waiting costs). In this case, the agent knows the reward $r$ and the transition probability matrix $P$. We will assume that the amount of time between each decision epoch is one minute. To define the value function, let

$V^*(s,n) =$ the maximum expected net reward that the agent receives when the elevator is on floor $s$ from decision epoch $n$ to $N$.

Due to the state space being augmented by time, this requires a slight modification to Eq. (2.4.6); instead, we will use

$$V^*(s,N-1) = \max_{a\in\mathcal{A}} R\big((s,N-1),a\big), \tag{2.7.4}$$

and if $0 \le n < N-1$,

$$V^*(s,n) = \max_{a\in\mathcal{A}} \left[ R\big((s,n),a\big) + \gamma \sum_{s'\in\mathcal{S}} P\big((s',n-1) \mid s,a\big) V(s',n-1) \right] \tag{2.7.5}$$

for all $s \in \{1,2,\dots,S\}$. We will start with $V(s,N-1)$, which is the last decision that the agent can make. Substituting Eq. (2.7.3) into Eq. (2.7.4), the value function at $n = N-1$ is

$$V^*(s,N-1) = \max_{a\in\mathcal{A}} \begin{cases} r - C, & (a = \text{stairs}); \\ rP\big((1,N-2) \mid (s,N-1),\text{wait}\big) - c, & (a = \text{wait}). \end{cases}$$

for all $s \in \{1,2,\dots,S\}$. Substituting Eq. (2.7.1) and Eq. (2.7.3) into Eq. (2.7.5), the value function at $n < N-1$ is

$$V^*(s,n) = \max_{a\in\mathcal{A}} \begin{cases} r - C, & (a = \text{stairs}); \\ \underbrace{rP\big((1,n-1) \mid (s,n),\text{wait}\big) - c}_{\text{(a) elevator arrives}} + \\ \underbrace{\sum_{1<s'<s} P\big((s',n-1) \mid (s,n),\text{wait})\big) V^*(s',n-1)}_{\text{(b) elevator does not arrive}}, & (a = \text{wait}). \end{cases}$$

Intuitively, the previous equation follows since if the agent waits there are two possibilities.

(a) The elevator arrives within the next minute: $rP\big((1,n-1) \mid (s,n),\text{wait}\big) - c$.

(b) The elevator does not arrive in the next minute: the amount of reward depends on what will happen in future decision epochs. In this case, the net reward from decision epoch $n+1$ to $N$ is $V(s',n+1)$. Since $V(s',n+1)$ depends on $s'$, and the agent does not know which floor the elevator will be in the next time step, we sum over the remaining floors and their associated probability to get the expected net reward from decision epoch $n+1$ to $N$: $\sum_{1<s'<s} P(s' \mid s)V(s',n+1)$.

To determine the agent's optimal waiting policy, we work backwards from $V(S,N)$ until $V(1,0)$ is computed. There exists an optimal policy, and it is defined by

$$\pi^*(s,n) = \begin{cases} \text{stairs}, & \text{if } V^*(s,n) = r - C; \\ \text{wait}, & \text{otherwise.} \end{cases} \tag{2.7.6}$$

We give a numerical example by setting $N = 10$, $S = 5$, $r = 25$, $c = 3$, $C = 10$, and the state transition probability matrix as

$$P(s' \mid s) = \begin{array}{c|ccccc} & 1 & 2 & 3 & 4 & 5 \\ \hline 1 & 1 & 0 & 0 & 0 & 0 \\ 2 & \frac{1}{2} & \frac{1}{2} & 0 & 0 & 0 \\ 3 & 0 & \frac{1}{2} & \frac{1}{2} & 0 & 0 \\ 4 & 0 & 0 & \frac{1}{2} & \frac{1}{2} & 0 \\ 5 & 0 & 0 & 0 & \frac{1}{2} & \frac{1}{2} \end{array}.$$

We have calculated the optimal value function $V^*(s, n)$ along with the optimal policy $\pi^*$ in Fig. 2.4. The policy $\pi^*$ is an example of a *threshold policy*: it is optimal to stop when the value function crosses a specified level.

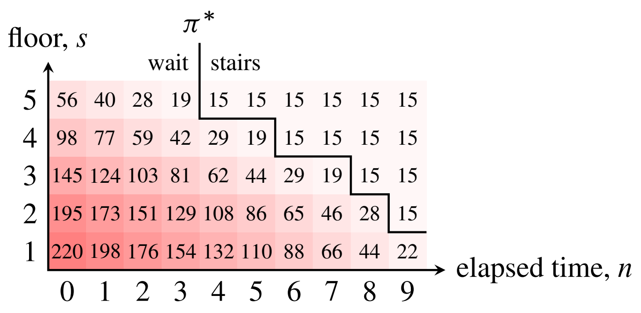


Fig. 2.4 The optimal value function $V^*(s, n)$ calculated for the numerical example for the elevator problem. The optimal policy $\pi^*$ is shown by a threshold: if the state is to the left of the threshold, then the agent should wait; if the state is to the right of the threshold, then the agent should take the stairs.

The Python code is provided in Appendix A.1. ◀

In the previous example, we saw an agent in a time-critical situation waiting for an elevator. As each minute passes, the agent must decide based on the environment's current state, adding a layer of complexity to the state space. This problem not only exemplifies optimal stopping and temporal control, but also demonstrates how the state space grows with each decision epoch. In later sections, we will explore how the problem can become more realistic by incorporating partial observability and random sojourn times, further enhancing the challenge and applicability of the model. We will use this as our running example to explore how the optimal stopping problem can become more realistic by incorporating partial observability and random sojourn times, further enhancing the challenge and applicability.

## 2.8 Bibliographic Remarks

Bellman (1957b) first coined the term Markov decision processes, of which this and other early work is summarized in his book (Bellman, 1957a). Shapley (1953) is also credited with fundamental work in this area of two-person stochastic games that were very close to MDPs.

In Sec. 2.5, we discussed two computational approaches to solving MDPs. Another computational approach is to convert Bellman's equation as a linear programming problem (d'Epenoux, 1960, 1963).

Backup diagrams were introduced in Sutton and Barto (1998). We will extend these diagrams to represent other models in the next chapters.

In Sec. 2.4, the value function accumulates reward additively over time. However, there are other value functions that accumulates over time multiplicatively, where the reward function has an exponential function form (Bertsekas, 2022, pp. 187–190).

# 3

# The Partially Observable Markov Decision Process (POMDP) Model

In this chapter, we embark on a formal introduction and definition of a Partially Observable Markov Decision Process (POMDP) within the context of a Borel space, serving as a generalized extension of a Markov Decision Process (MDP); see Fig. 3.1.

$$\begin{tikzcd}
\text{MDP} \arrow[d, "+\text{ partially observability}"] \\
\text{POMDP}
\end{tikzcd}$$

Fig. 3.1 The relationship between MDP and POMDP

Subsequent to the model's delineation, we will delve into the computational strategies applicable for solving a POMDP. Notably, we will explore the PERSEUS algorithm, a randomized point-based value iteration method. This algorithm is particularly significant as it lays the groundwork for addressing Partially Observable Semi-Markov Decision Processes (POSMDPs) in Chap. 5.

Before we go further with this discussion, we will discuss the importance of partial observability, and show how we can include it in an MDP from Chap. 2 to yield the partially observable Markov decision process (POMDP). The belief state is introduced, and then we discuss further the quote from Sutton and Barto along with alternative representations of POMDPs. Lastly, we discuss exact and approximate solutions of POMDPs and their respective algorithms from which these algorithms form the basis of our algorithms for Chap. 5.

## 3.1 Partial Observability

Non-Markovian problems manifest in a variety of situations. Consider an agent operating in an environment where it lacks perfect state information. Although the environment itself follows Markovian properties, the agent's observations do not fully reveal the state of the environment, rendering it only *partially observable*. The agent may infer these hidden states from past time steps, but if this information is not available in the current state, the problem remains non-Markovian.

**Example 3.1.1 — In a darkroom.** Suppose you are standing in a room like Fig. 3.2. Your friend turns out the light and places you randomly somewhere inside of the room; you assume now that you could be starting at any one of the tiles in the room. You remember the interior of the room (the agent knows the state space), but you do not know where you are in the room (the agent does not know where it is in the state

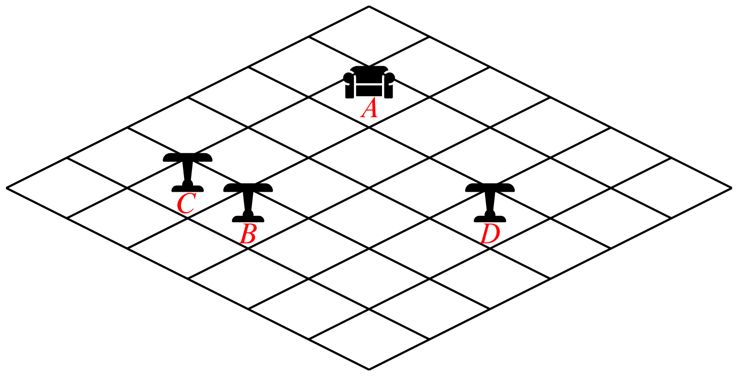


Fig. 3.2 A tiled room grid with furniture, where tiles marked $A$ represent a sofa, tiles marked $B$, $C$, and $D$ represent tables, and empty tiles can be any of the unmarked ones. In a dark room, the observer can distinguish the different tiles by feeling the presence or absence of furniture in them.

space). But, as you begin to move blindly from tile to tile in the room, you can feel (observe) in that tile if there is furniture or not. This will help you distinguish the different tiles.

(a) If there is nothing in that tile, then you know that you can be in any one of the empty tiles.

(b) If you come across a sofa and because you know this is the only sofa in the room, then you know with certainty that you are in tile $A$.

(c) If you came across a table, then you could be in either tile $B$, $C$, or $D$. If you continue, and move into the next tile and feel another table, then you can restrict your belief that you are in either tile $B$ or $C$.

◀

In the previous example, you are creating a conditional probability of where you believe you are in the state space given your past observations. Partially observable Markov processes can be thought of as an ordinary Markov decision process where the state is the agent's belief (or subjective probability) of being in a particular state. An agent knows the entire state space, but it does not know exactly where it is in the state space. Hence, the agent creates a probability distribution over the state space of where it believes it is located within the state space from its observations.

## 3.2 The POMDP Model

The Partially Observable Markov Decision Process (POMDP) serves as a mathematical framework encapsulating sequential decision-making under uncertainty within partially observable domains. As an extension of a Markov Decision Process (MDP), it integrates partial observability into the model. The complexity of decision-making in an MDP stems from the agent's uncertainty about the stochastic environment's response to its actions. This complexity escalates in a POMDP, where not only does the environment behave stochastically, but the agent also grapples with limited visibility into various aspects of the environment.

**Definition 3.2.1 — POMDP.** A partially observable Markov decision process (POMDP) is an 11-tuple

$$\langle \mathcal{S}, \mathcal{A}, \mathcal{K}, \mathcal{O}, P, G, G_0, R, \xi_0, \gamma, N \rangle \tag{3.2.1}$$

where

| | |
|---|---|
| $\mathcal{S}$ | is the Borel *state space*, and the elements of $\mathcal{S}$ are called *states* |
| $\mathcal{A}$ | is the Borel *action space*, and the elements of $\mathcal{A}$ are called *actions*. To each $s \in \mathcal{S}$, we associate a nonempty Borel-measurable subset $\mathcal{A}(s) \subseteq \mathcal{A}$, whose elements are the *admissible actions* for the agent when the process is in state $s$ |
| $\mathcal{K}$ | is the set of admissible state-action pairs, and it is assumed to be a Borel subset $\mathcal{K} \subseteq \mathcal{S} \times \mathcal{A}$. In other words, $$\mathcal{K} = \{(s,a) \mid s \in \mathcal{S}, a \in \mathcal{A}(s)\}.$$ |
| $\mathcal{O}$ | is the Borel *observation space*, and the elements of $\mathcal{O}$ are called observations |
| $P(\mathrm{d}s' \mid s,a)$ | is the state (Borel-measurable) stochastic kernel on $\mathcal{S}$ given $\mathcal{S} \times \mathcal{A}$ |
| $G(\mathrm{d}o \mid a,s')$ | is the observation (Borel-measurable) stochastic kernel on $\mathcal{O}$ given $\mathcal{A} \times \mathcal{S}$ |
| $G_0(\mathrm{d}o \mid s')$ | is the initial observation (Borel-measurable) stochastic kernel on $\mathcal{O}$ given $\mathcal{S}$ |
| $R(s,a)$ | is the per-stage (bounded Borel-measurable) reward function given $\mathcal{K}$ |
| $\xi_0$ | is the (*a priori*) initial (belief) state distribution ($\xi_0 \in \mathbf{P}(\mathcal{S})$) |
| $\gamma$ | is the discounting factor where $\gamma \in [0,1]$ |
| $N$ | is the planning horizon. It could be finite, or $N = \infty$ if $\gamma < 1$. |

Kaelbling *et al.* (1996, 1998) introduced the POMDP into the artificial intelligence community, and since then, it has become part of the standard curriculum for introductory courses on artificial intelligence (Russell and Norvig, 2010, Sec. 17.4).

Given the model in Definition 3.2.1, the dynamics of the POMDP proceed according to Algorithm 5. This involves at each decision epoch $n$ choosing an action $a_n$, accruing reward $R(s_n, a_n)$ as the state changes from state $s_n$ to $s_{n+1}$, and partially observing $s_{n+1}$ as $o_{n+1}$. The agent uses all the information available up to decision epoch $n$, namely the observable history $\tilde{h}_n$, to choose action $a_n = \pi_n(\tilde{h}_n)$ using policy $\pi_n$. We denote the sequence of policies that the agent uses from decision epoch 0 to $n-1$ as $\pi = (\pi_0, \pi_1, \ldots, \pi_{n-1})$. The set of all admissible policies is denoted $\Pi$.

Most of the time, the observation is independent of the previous state. If this is not the case, then for each $(s,a,s')$ transition define $G(o \mid s,a,s')$. Then we can write

$$G(o \mid a,s') = \sum_{s \in \mathcal{S}} P(s' \mid s,a) G(o \mid s,a,s'). \tag{3.2.2}$$

This is similar to how we define the expected reward

$$R(s,a) = \sum_{s' \in \mathcal{S}} P(s' \mid s,a) R(s,a,s').$$

## 3.3 History

The *history* refers to the sequence of all past observations, actions, and resulting states up to the current time step in the decision-making process.

In an MDP, the history is entirely observable by the agent;

$$\text{MDP:} \quad \begin{aligned} h_0 &= (s_0) \\ h_n &= (s_0, a_0, \ldots, s_{n-1}, a_{n-1}, s_n) \\ h_{n+1} &= h_n \cup (a_n, s_{n+1}). \end{aligned}$$

While in a POMDP, the history of the model's dynamics is

$$\text{POMDP:} \quad \begin{aligned} h_0 &= (\xi_0, s_0) \\ h_n &= (\xi_0, s_0, a_0, o_1, \ldots, s_{n-1}, a_{n-1}, o_n) \\ h_{n+1} &= h_n \cup (s_n, a_n, o_{n+1}). \end{aligned}$$

However, with a POMDP, the states (listed in red) are not entirely observable by the agent; the state in which the agent is in not known with certainty. Instead, the agent sees the *observable history*, $\tilde{h}_n$, which is

$$\text{POMDP:} \quad \begin{aligned} \tilde{h}_0 &= (\xi_0) \\ \tilde{h}_n &= (\xi_0, a_0, o_1, a_1, o_2, \ldots, a_{n-1}, o_n) \\ \tilde{h}_{n+1} &= \tilde{h}_n \cup (a_n, o_{n+1}). \end{aligned}$$

The observable history is crucial because it's this information that the agent uses to infer the underlying hidden state of the environment, given that the full state is not directly observable.

**Algorithm 5** Dynamics of POMDP

In the beginning $n = 0$, the state $s_0$ is simulated from an initial (belief) state distribution $\xi_0$.
For decision epoch $n = 1, 2, \ldots, N$:

(a) Based on the observable history

$$\begin{aligned} \tilde{h}_0 &= (\xi_0) \\ \tilde{h}_n &= (\xi_0, a_1, o_1, \ldots, a_n, o_n), \end{aligned}$$

the agent performs action

$$a_n = \pi_n(h_n) \in \mathcal{A}, \qquad n = 1, 2, \ldots, N.$$

Here, $\pi_n$ denotes a policy that the agent uses at decision epoch $n$.

(b) The agent receives a reward $R(s_n, a_n)$ for performing action $a_n$ in state $s_n$.

(c) The state evolves randomly with transition probability

$$P(s' \mid s, a) = \mathbf{P}(S_{n+1} = s' \mid S_n = s, A_n = a)$$

to the next state $s_{n+1}$ at decision epoch $n + 1$.

(d) The agent records a noisy observation $O_n \in \mathcal{O}$ of the state $S_{n+1}$ according to

$$G(o \mid a, s') = \mathbf{P}(O_n = o \mid A_n = a, S_{n+1} = s').$$

(e) The agent updates its observable history as

$$\tilde{h}_{n+1} = (\tilde{h}_n, a, o).$$

If $n < N$, then set $n$ to $n + 1$ and go back to step (a).
If $n = N$, then the agent receives the last reward and the process terminates.

## 3.4 Belief State

In Sec. 3.1, we explored the concept of partial observability, where an agent can only infer the state $s \in \mathcal{S}$ from its observable history, as detailed in Sec. 3.3. Therefore, to optimally navigate a partially observable environment and select appropriate actions, an agent necessitates a memory of past observations.

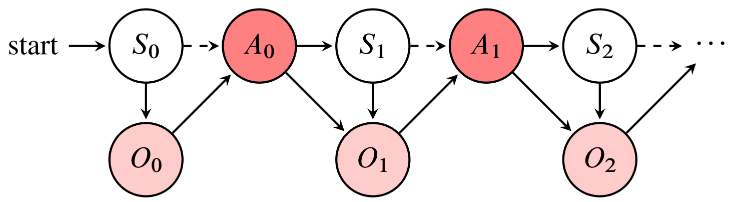


Fig. 3.3 Graphical model of POMDP. The states $S_i$ are hidden from the agent, while the observations $O_i$ are observable. The agent's actions are $A_i$. The solid arrows $\rightarrow$ represent the direct influence from one element to another, while the dashed arrows $\dashrightarrow$ represent indirect influence. For example, action $A_0$ is directly influenced by observation $O_0$ because this is the information that is available to the agent, while state $S_0$ is indirectly influencing $A_0$ since the agent has to infer in which state it is in currently from observation $O_0$.

A naïve approach would be for the agent to remember its sequence of sojourn times, observations, and actions. However, this sequence can grow unbounded over time, which is not practical with finite memory. Instead, we can summarize this information with sufficient statistics (Stratonovich, 1960; Aoki, 1965; Åström, 1965; Dynkin, 1965; Aoki, 1967; Åström, 1969; Sawaragi and Yoshikawa, 1970). A statistic is said to be *sufficient* when no other statistic calculated from the same sample provides any additional information to the parameter value being estimated (Fisher, 1922, p. 310). With these sufficient statistics that summarize the observable history, we can construct a probability distribution of where the agent is in the state space; we call this probability distribution over the state space a belief state.

**Definition 3.4.1 — Belief state.** A belief state $\xi$ is a probability distribution over the state space $\mathcal{S}$.

If the state space $\mathcal{S} = \{s_1, s_2, \ldots, s_{|\mathcal{S}|}\}$ is a finite set, then the belief state $\xi$ is defined by

$$\xi = \begin{bmatrix} \xi(s_1) \\ \xi(s_2) \\ \vdots \\ \xi(s_{|\mathcal{S}|}) \end{bmatrix} \triangleq \begin{bmatrix} \mathbf{P}(S = s_1) \\ \mathbf{P}(S = s_2) \\ \vdots \\ \mathbf{P}(S = s_{|\mathcal{S}|}) \end{bmatrix}.$$

To ensure that $\xi$ is a probability distribution,

$$\xi(s_i) \geq 0 \qquad \forall s_i \in \mathcal{S},$$

and

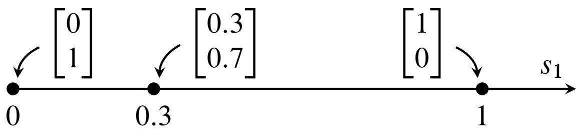
$$\sum_{i=1}^{|\mathcal{S}|} \xi(s_i) = 1.$$

**Definition 3.4.2 — Belief Simplex $\triangle$.** If the state space $\mathcal{S} = \{s_1, s_2, \ldots, s_{|\mathcal{S}|}\}$ is a finite set, then the belief (probability) simplex in $\mathbb{R}^{|\mathcal{S}|}$, denoted by $\triangle$, is the set of belief states $\xi \in \mathbb{R}^{|\mathcal{S}|}$ such that $\xi(s_i) \geq 0$ and $\sum_i \xi(s_i) = 1$ for $i = 1, 2, \ldots, |\mathcal{S}|$.

If the state space $\mathcal{S}$ is a finite set, then the belief simplex $\triangle$ is an $(|\mathcal{S}|-1)$-dimensional manifold in $|\mathcal{S}|$-dimensional space. For example, when there are only two states ($|\mathcal{S}| = 2$), the belief simplex is the set of points

$$\triangle = \left\{ \xi = \begin{bmatrix} \xi(s_1) \\ \xi(s_2) \end{bmatrix} \middle| \, \xi(s_1) \geq 0, \xi(s_2) \geq 0, \xi(s_1) + \xi(s_2) = 1 \right\}. \tag{3.4.1}$$

This can be represented graphically as the interval $[0, 1] \subset \mathbb{R}$ (see Fig. 3.4).



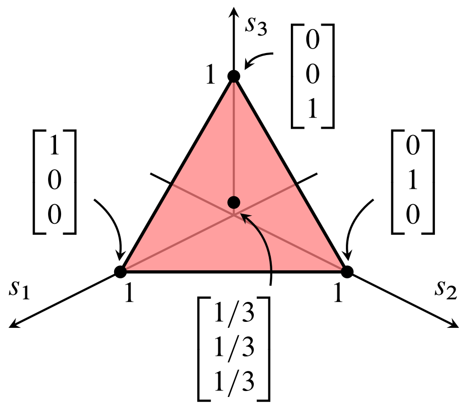
$\begin{bmatrix}0\\1\end{bmatrix}$
$\begin{bmatrix}0.3\\0.7\end{bmatrix}$
$\begin{bmatrix}1\\0\end{bmatrix}$
$s_1$
0
0.3
1

Fig. 3.4 Belief simplex for two states, $\mathcal{S} = \{s_1, s_2\}$, with three belief states shown as examples.

When there are three states ($|\mathcal{S}| = 3$), the belief simplex is the set of points

$$\triangle = \left\{ \xi = \begin{bmatrix}\xi(s_1)\\ \xi(s_2)\\ \xi(s_3)\end{bmatrix} \;\middle|\; \xi(s_1) \geq 0, \xi(s_2) \geq 0, \xi(s_3) \geq 0, \xi(s_1) + \xi(s_2) + \xi(s_3) = 1 \right\}. \tag{3.4.2}$$

Graphically, a triangle represents the simplex for three states (see Fig. 3.5). Since this



$s_3$
$\begin{bmatrix}0\\0\\1\end{bmatrix}$
1
$\begin{bmatrix}1\\0\\0\end{bmatrix}$
$\begin{bmatrix}0\\1\\0\end{bmatrix}$
$s_1$
1
1
$s_2$
$\begin{bmatrix}1/3\\1/3\\1/3\end{bmatrix}$

Fig. 3.5 Belief simplex for three states, $\mathcal{S} = \{s_1, s_2, s_3\}$, with four belief states shown as examples.

is a triangular two-dimensional flat in $\mathbb{R}^3$, we use a triangle to represent the belief simplex in later sections of this chapter.

## 3.5 Belief Update

After taking action $a$ and receiving observation $o$, the agent updates its belief in which state it is now using Bayes' Theorem:

$$\begin{aligned}
\xi(s' \mid a, o) &= \mathbf{P}(S_{n+1} = s' \mid \xi_n = \xi, A_n = a, O_n = o) \\
&= \frac{\mathbf{P}(\xi_n = \xi, A_n = a, O_n = o, S_{n+1} = s')}{\mathbf{P}(\xi_n = \xi, A_n = a, O_n = o)} \\
&= \frac{\mathbf{P}(\xi_n = \xi, A_n = a, O_n = o, S_{n+1} = s')}{\sum_{s'' \in \mathcal{S}} \mathbf{P}(\xi_n = \xi, A_n = a, O_n = o, S_{n+1} = s'')} \\
&= \frac{\sum_{s \in \mathcal{S}} \xi(s)\, \mathbf{P}(S_{n+1} = s' \mid S_n = s, A_n = a)\, \mathbf{P}(O_n = o \mid A_n = a, S_{n+1} = s')}{\sum_{s'' \in \mathcal{S}} \sum_{s \in \mathcal{S}} \xi(s)\, \mathbf{P}(S_{n+1} = s'' \mid S_n = s, A_n = a)\, \mathbf{P}(O_n = o \mid A_n = a, S_{n+1} = s'')} \\
&= \frac{\sum_{s \in \mathcal{S}} \xi(s) P(s' \mid s, a) G(o \mid a, s')}{\sum_{s'' \in \mathcal{S}} \sum_{s \in \mathcal{S}} \xi(s) P(s'' \mid s, a) G(o \mid a, s'')} \\
&= \frac{G(o \mid a, s') \sum_{s \in \mathcal{S}} \xi(s) P(s' \mid s, a)}{\sum_{s'' \in \mathcal{S}} G(o \mid a, s') \sum_{s \in \mathcal{S}} \xi(s) P(s'' \mid s, a)}
\end{aligned}$$

Thus, the belief update is

$$\xi(s' \mid a, o) = \frac{G(o \mid a, s') \sum_{s \in \mathcal{S}} P(s' \mid s, a)\, \xi(s)}{P(o \mid \xi, a)} \qquad \forall s' \in \mathcal{S}, \tag{3.5.1}$$

where the denominator

$$P(o \mid \xi, a) = \sum_{s' \in \mathcal{S}} G(o \mid a, s') \sum_{s \in \mathcal{S}} P(s' \mid s, a)\, \xi(s) \tag{3.5.2}$$

is a normalization factor.

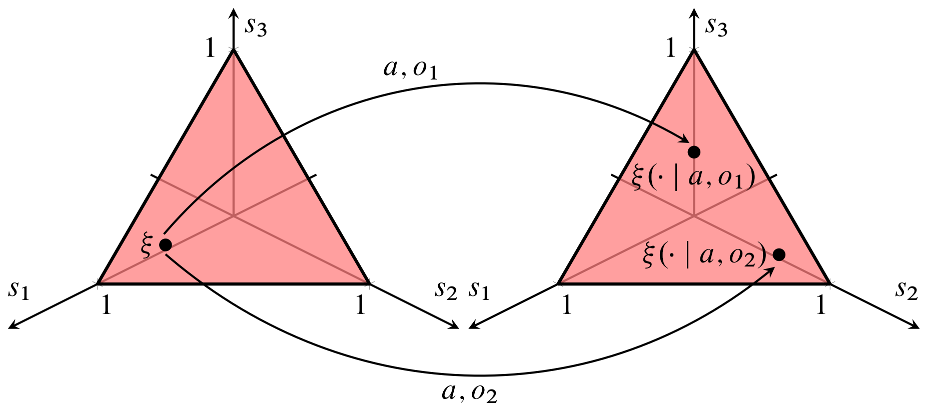


Fig. 3.6 Belief simplex for a three-state $\mathcal{S} = \{s_1, s_2, s_3\}$ two-observation $\mathcal{O} = \{o_1, s_2\}$ POMDP. This shows how the agent's initial belief $\xi$ changes once it selects an action $a$ and observes an observation.

**Algorithm 6** Controller for an Online Agent (POMDP)

In the beginning $n = 0$, the state $s_0$ is simulated from an initial (belief) state distribution $\xi_0$.

For decision epoch $n = 1, 2, \dots, N$:

(a) Based on the current belief state $\xi$, the agent performs action

$$a_n = \pi_n(\xi_n) \in \mathcal{A} \qquad n = 1, 2, \dots, N,$$

according to policy $\pi_n$ that the agent uses at decision epoch $n$.

(b) The agent receives a reward $R(s_n, a_n)$ for performing action $a_n$ in state $s_n$.

(c) The state evolves randomly with transition probability

$$P(s' \mid s, a) = \mathbf{P}(S_{n+1} = s' \mid S_n = s, A_n = a)$$

to the next state $S_{n+1}$ at decision epoch $n + 1$.

(d) The agent records a noisy observation $O_n \in \mathcal{O}$ of the state $S_{n+1}$ according to

$$G(o \mid a, s') = \mathbf{P}(O_n = o \mid A_n = a, S_{n+1} = s').$$

(e) With the selected action $a$ and the observation $o$ received, the agent updates its belief state

$$\xi(s' \mid a, o) = \frac{G(o \mid a, s') \sum_{s \in \mathcal{S}} \xi(s) P(s' \mid s, a)}{\sum_{s'' \in \mathcal{S}} G(o \mid a, s'') \sum_{s \in \mathcal{S}} \xi(s) P(s'' \mid s, a)} \qquad \forall s' \in \mathcal{S},$$

where the initial belief $\xi_0$ is given in the model.

(f) If $n < N$, then set $n$ to $n + 1$ and go back to step (a).
If $n = N$, then the agent receives the last reward and the process terminates.

### 3.5.1 Reward-based POMDP (R-POMDP)

The belief state is a sufficient statistic of the history $h_n = (\xi_0, a_1, o_1, \dots, a_n, o_n)$. However, it is important to note that the history may not encapsulate all available information. The immediate reward $R(s, a, s')$, which depends on the underlying state, may hold significant insights about the state. As argued by Izadi and Precup (2005), rewards may carry valuable information that is not captured in the observation, and this information can aid in differentiating between states.

This point is further underscored by a question-answer exchange recorded in Kaelbling *et al.* (1996). During the discussion, the idea of rewards providing additional information was raised. Kaelbling responded by suggesting that the reward could be integrated into the state space, thereby preserving the original problem's formulation while still leveraging the information contained in the reward.

$$\begin{aligned}
\xi'(s' \mid a, o, r) &= \mathbf{P}(S_{n+1} = s' \mid \xi_n = \xi, A_n = a, O_n = o, R_n = r) \\
&= \frac{\mathbf{P}(\xi_n = \xi, A_n = a, O_n = o, R_n = r, S_{n+1} = s')}{\mathbf{P}(\xi_n = \xi, A_n = a, O_n = o, R_n = r)} \\
&= \frac{\sum_{s \in \mathcal{S}} \xi(s) P(s' \mid s, a) G(o \mid a, s') R(r \mid s, a, s')}{\sum_{s' \in \mathcal{S}} \sum_{s \in \mathcal{S}} \xi(s) P(s' \mid s, a) G(o \mid a, s') R(r \mid s, a, s')}
\end{aligned} \qquad (3.5.3)$$

where

$$R(r \mid s, a, s') = \begin{cases} 1, & \text{if } r = R(s, a, s'); \\ 0, & \text{otherwise.} \end{cases} \tag{3.5.4}$$

When (3.5.3) is used for the belief state update, it is called a reward-based POMDP (R-POMDP) (Izadi and Precup, 2005, p. 595–596).

## 3.6 The Value Function

When dealing with POMDPs, the concept of the value function becomes particularly crucial. Unlike MDPs, where the value function is directly associated with the fully observable state $s \in \mathcal{S}$, POMDPs require a different approach. Given the agent's inability to observe $s$ directly, it relies on a sufficient statistic $\xi(s)$ derived from the observable history to infer the state's probability. Consequently, the agent's planning process becomes more effectively grounded in the belief state.

The value of a belief $\xi$ for a POMDP following a policy $\pi$ is

$$V^\pi(\xi) = R\big(\xi, \pi(\xi)\big) + \gamma \sum_{o \in \mathcal{O}} P\big(o \mid \xi, \pi(\xi)\big) V^\pi\big(\xi(\cdot \mid \pi(\xi), o)\big) \tag{3.6.1}$$

where the next belief vector $\xi\big(\cdot \mid \pi(\xi), o\big)$ is defined by Eq. (3.5.1) and $P\big(o \mid \xi, \pi(\xi)\big)$ is defined by Eq. (3.5.2). The reward function $R\colon \triangle \to \mathbb{R}$ is

$$R(\xi, a) = \sum_{s \in \mathcal{S}} \xi(s) R(s, a). \tag{3.6.2}$$

Figure 3.7 gives the corresponding backup diagram for Eq. (3.6.1). In comparison to the MDP backup diagram in Fig. 2.1 (on p. 11), the state of that the value function changes from $s$ to $\xi$ and we include the observation probability transition $G$.

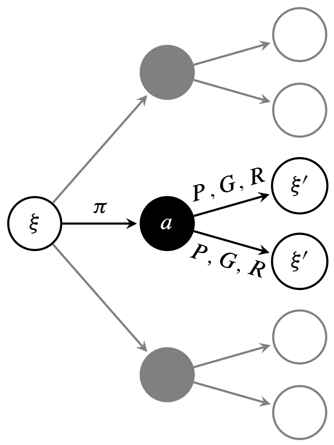


Fig. 3.7 Backup diagram for the value function $V^\pi$ following policy $\pi$.

Thus, Eq. (3.6.1) fully expanded is written as

$$\begin{aligned} V^\pi(\xi) = \sum_{s \in \mathcal{S}} \xi(s) R\big(s, \pi(s)\big) + \gamma \sum_{o \in \mathcal{O}} \sum_{s' \in \mathcal{S}} G\big(o \mid \pi(\xi), s'\big) \\ \sum_{s \in \mathcal{S}} \xi(s) P\big(s' \mid s, \xi(s)\big) V^\pi\big(\xi(\cdot \mid \pi(\xi), o)\big). \end{aligned} \tag{3.6.3}$$

The optimal value function $V^*$ is

$$V^*(\xi) = \max_{a \in \mathcal{A}(s)} \left[ R(\xi, a) + \gamma \sum_{o \in \mathcal{O}} P(o \mid \xi, a) V^*\big(\xi(\cdot \mid a, o)\big) \right] \tag{3.6.4}$$

where the next belief vector $\xi' = \xi(\cdot \mid a, o)$ is defined by the belief update rule in Eq. (3.5.1), $P(o \mid \xi, a)$ is defined by Eq. (3.5.2), and $R(\xi, a)$ is defined by Eq. (3.6.2). Figure 3.7 gives the corresponding backup diagram for Eq. (3.6.1).

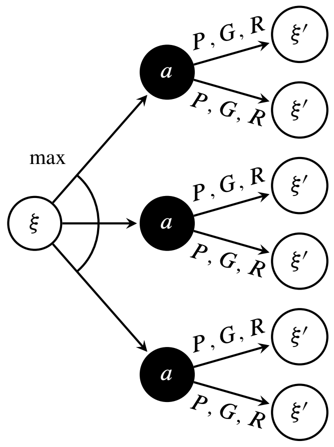


Fig. 3.8 Backup diagram for the value function $V^*$ following policy $\pi^*$.

Thus, Eq. (3.6.4) can be expressed as

$$V^*(\xi) = \max_{a \in \mathcal{A}(s)} \Bigg[ \sum_{s \in \mathcal{S}} \xi(s) R(s, a) + \gamma \sum_{o \in \mathcal{O}} \sum_{s' \in \mathcal{S}} G(o \mid a, s') \sum_{s \in \mathcal{S}} \xi(s) P(s' \mid s, a) V^{\pi}\big(\xi(\cdot \mid a, o)\big) \Bigg] \tag{3.6.5}$$

When Eq. (3.6.5) holds for every belief state $\xi$ in the belief simplex $\triangle$, then the solution is guaranteed to be optimal.

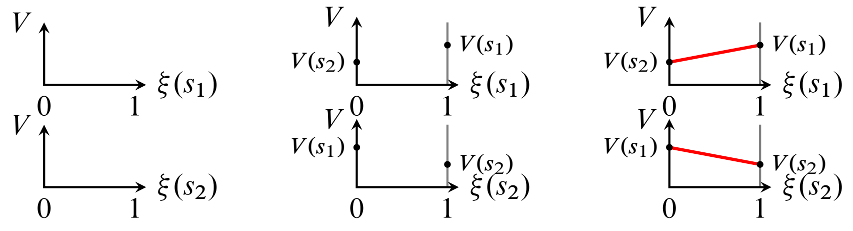
## 3.7 Exact Solution Algorithms

Since the belief simplex $\triangle$ is continuous, the value function $V(\xi)$ in Eq. (3.6.5) is computationally intractable to calculate pointwise for every belief state $\xi \in \triangle$. To circumvent this, Sondik (1971) made the following assumption. Consider a two-state $\mathcal{S} = \{s_1, s_2\}$ POMDP. In Fig. 3.9(a), we represent the belief state on the horizontal axis, while on the vertical axis will be the value function. The agent knows the value with absolute certainty when with absolute certainty it knows in which state it is; see Fig. 3.9(b).

**Assumption 3.7.1 — Sondik (1971).** The value is linearly propositional to how certain the agent believes in which state it is.

In other words, linear interpolation that fills in the values between the two known values; see Fig. 3.9(c). The line segment that connects the two values in Fig. 3.9(c) is an example of what Sondik (1971) called an $\alpha$-vector.

In general, we give the following definition.



(a) Beginning to graph the value function over the belief simplex

(b) The agent knows what the value is when it knows with absolute certainity in which state it is.

(c) The assumption is that the two values are connected by a linear function.

Fig. 3.9 An example of an $\alpha$-vector for a two-state $\mathcal{S} = \{s_1, s_2\}$ POMDP

**Definition 3.7.2 — $\alpha$-vector.** An $\alpha$-vector represents an $|\mathcal{S}|$-dimensional hyperplane that defines the value function for a particular action $a \in \mathcal{A}$ over a bounded region of the belief simplex. In this work, we represent $\alpha$-vectors in matrix form:

$$\alpha = \begin{bmatrix} \alpha(s_1) \\ \alpha(s_2) \\ \vdots \\ \alpha(s_{|\mathcal{S}|}) \end{bmatrix} = \begin{bmatrix} V(s_1) \\ V(s_2) \\ \vdots \\ V(s_{|\mathcal{S}|}) \end{bmatrix}.$$

While we have generally refer to $V$ as the value function, we can also think of $V$ as a set of $\alpha$ vectors:

$$V = \{\alpha_1, \alpha_2, \ldots, \alpha_m\}. \tag{3.7.1}$$

The optimal value function at a particular belief state $\xi$ is the inner product of $\xi$ with the $\alpha$-vector in set $V$ that will yield the highest value, which is

$$V^*(\xi) = \max_{\alpha \in V} \langle \xi, \alpha \rangle \tag{3.7.2}$$

$$= \max_{\alpha \in V} \sum_{s \in \mathcal{S}} \xi(s)\alpha(s), \tag{3.7.3}$$

and the corresponding optimal policy $\pi^*: \xi \to a$ is

$$\pi^*(\xi) = \underset{i \in \{1,2,\ldots,m\}}{\arg\max} \langle \xi, \alpha_i \rangle \tag{3.7.4}$$

$$= \underset{i \in \{1,2,\ldots,m\}}{\arg\max} \sum_{s \in \mathcal{S}} \xi(s)\alpha_i(s), \tag{3.7.5}$$

where each $\alpha_i$ has a corresponding action associated with it.

With the example in Fig. 3.9, we can define the $\alpha$-vector as

$$\alpha = \begin{bmatrix} V(s_1) \\ V(s_2) \end{bmatrix}.$$

Given a belief

$$\xi = \begin{bmatrix} \xi(s_1) \\ \xi(s_2) \end{bmatrix},$$

we can calculate the value by

$$V(\xi) = \langle \xi, \alpha \rangle = \begin{bmatrix} \xi(s_1) & \xi(s_2) \end{bmatrix} \begin{bmatrix} \alpha(s_1) \\ \alpha(s_2) \end{bmatrix} = \xi(s_1)\alpha(s_1) + \xi(s_2)\alpha(s_2).$$

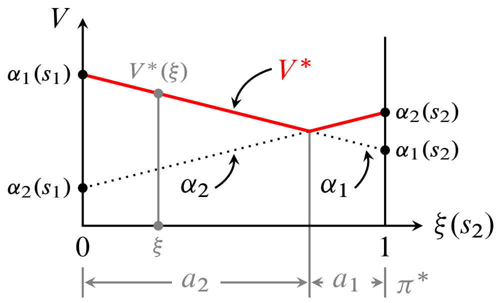


Fig. 3.10 Example of a POMDP with two states $\mathcal{S} = \{s_1, s_2\}$ and two actions $\mathcal{A} = \{a_1, a_2\}$. In this example, there are two $\alpha$-vectors, $\alpha_1$ and $\alpha_2$, which correspond to action $a_1$ and action $a_2$ respectively. The horizontal axis shows only $\xi(s_2)$, but since $|\mathcal{S}| = 2$, the belief simplex $\Delta$ is a one-dimensional interval $[0, 1]$, and so $\xi(s_1) = 1 - \xi(s_2)$. The optimal value function $V^*$ is shown with a thick line, and the optimal policy $\pi^*$ is shown below the graph of $V$ with which regions of the belief space correspond to the optimal action.

Sondik (1971) was the first to realize that the value function of a finite-horizon POMDP is piecewise linear and convex. With this insight, Sondik expressed the value function as a set of linear vectors, which he called $\alpha$-vectors.

Smallwood and Sondik (1973) provided algorithms for solving a POMDP for the finite horizon case, while later, Sondik (1978) considered the infinite horizon case with discounted rewards.

In general, using exact solution methods for long planning horizons with POMDPs is computational intractable (Papadimitriou and Tsitsiklis, 1987).[1] The exact POMDP value iteration algorithm is given in Algorithm 7.[2]

**Algorithm 7** Exact POMDP Value Iteration, Sondik (1971), Monahan (1982)

```
1: function ExactBackup
2:     $\Gamma^{a,*} \leftarrow \alpha^{a,*} = R(s,a)$                                   $\triangleright$ Step 1: Generate initial sets.
3:     $\Gamma^{a,o} \leftarrow \alpha_i^{a,o}(\xi) = \gamma \sum_{s' \in \mathcal{S}} P(s' \mid s,a) G(o \mid a,s') \alpha'_i(\xi') \; \forall \alpha'_i \in V'$
4:     $\Gamma^a = \Gamma^{a,*} \oplus \Gamma^{a,o_1} \oplus \Gamma^{a,o_2} \oplus \cdots$          $\triangleright$ Step 2: Create $\Gamma^a$ for all $a \in \mathcal{A}$
5:     $V = \cup_{a \in \mathcal{A}} \Gamma^a$                                   $\triangleright$ Step 3: Take the union of all $\Gamma^a$ sets.
6: end function
```

In practice, the set of vectors in $V$ may be larger than necessary; many of the vectors may be dominated by other vectors in the set ($\langle \alpha_i, \xi \rangle < \langle \alpha_j, \xi \rangle \; \forall \xi$) or by a combination of other vectors. These redundant vectors can be removed from the set without affecting the quality of the solution, but the removal operation can be computationally expensive requiring the solving of a linear programming problem.

## 3.8 Approximate Solution Algorithms

From Sec. 3.7, we know that solving a POMDP is computationally intractable, and so we turn to approximate solutions to solve POMDPs for real-world problems such as patient care management (Hauskrecht and Fraser, 2000), robotics (Porta *et al.*, 2005), and spoken dialog systems (Young *et al.*, 2013).

[1] The term *exact* means that there is no approximation in the Bellman backup, unlike the approximations we will use in Sec. 3.8. This exact method however is not immune to numerical round off and finite precision effects.

[2] Given two $\alpha$-vectors, the *cross sum* is the pairwise addition of $\alpha$-vectors; that is, given vectors $A = [a_1 \cdots a_n]^\top$ and $B = [b_1 \cdots b_n]^\top$, then $A \oplus B = [a_1 + b_1 \cdots a_n + b_n]^\top$.

### 3.8.1 Point-based value iteration (PBVI)

Instead of planning over the entire belief simplex like exact value iteration in Sec. 3.7, point-based value iteration reduces the computational load by using only a finite set of belief states $B$ from the belief simplex. A naïve approach for selecting the belief states is to sample uniformly from the belief simplex or perhaps to use a regular mesh (Drake, 1962; Lovejoy, 1991); for instance, as depicted by the mesh within the belief simplex shown in Fig. 3.11.

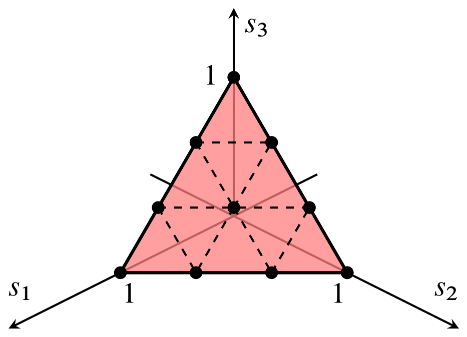


Fig. 3.11 Belief simplex for three states, $\mathcal{S} = \{s_1, s_2, s_3\}$, with a regular mesh of belief states.

Although the naïve approach offers a straightforward strategy for discretizing the belief simplex, it may incorporate some belief states that the agent is unlikely to encounter within the environment. Consequently, calculating the value function with these unlikely belief states is an unnecessary computational burden. An alternative sampling scheme offers a solution by simulating the POMDP using randomly sampled actions and observations. This approach emphasizes the collection of belief states more likely to appear within the belief simplex during a POMDP simulation. These sampled belief states then serve as the basis for executing value iteration to ascertain the optimal value function. This concept proved integral to developing the first Point-Based Value Iteration (PBVI) algorithm, pioneered by (Pineau *et al.*, 2003). The general outline of PBVI can be found in Algorithm 8.

**Algorithm 8** General form of point-based value iteration, Pineau *et al.* (2003)

```
1: while stopping criterion not reached do
2:     Collect belief states $\xi \in \triangle$ and put into set $B$
3:     Update the value function $V$ over the finite belief state set $B$
4: end while
```

In PBVI, it begins with an initial belief, typically the agent's initial state. The algorithm then performs a series of backups and expansions. During the backup process, the value function is updated for the current set of belief points. During the expansion process, the algorithm selects a new belief point that is the furthest from the current set, based on a distance metric. The utilization of this distance metric facilitates the analysis of the value function approximation error, deriving from the sampled belief points. The new belief point is added to the current set, and the process is repeated. This strategy ensures a balanced trade-off between the quality of the approximation and the computational complexity with very few belief points (sometimes less than the number of states) (Pineau *et al.*, 2006).

This exploratory approach allows PBVI to cover a wide range of belief states that the agent could possibly experience, improving the robustness and performance of the policy. However, belief selection in PBVI requires significant computational effort, and the size of the value function is directly tied to the size of the belief set, making it less efficient for larger problems.

### 3.8.2 Perseus

The PERSEUS algorithm is a point-based value iteration algorithm developed by Spaan and Vlassis (2005), and it proceeds in two steps as specified in Algorithm 9: first, it collects the beliefs, and then approximates the value function using the backup operator.

**Algorithm 9** PERSEUS, Spaan and Vlassis (2005)

```
1: function PERSEUS($n, \xi_0, V_0, \epsilon$)
2:     $B \leftarrow$ COLLECTBELIEFS($n, \xi_0$)
3:     $V' \leftarrow V_0$
4:     repeat
5:         $V \leftarrow V'$
6:         $V' \leftarrow$ UPDATE($B, V$)
7:     until $||V - V'||_{\infty,B} < \epsilon$
8:     return $V$
9: end function
```

In the method of belief point selection in Algorithm 10, unlike PBVI that employs a heuristic selection process, PERSEUS follows a more stochastic path. It randomly generates trajectories across the belief simplex, leading to a diverse selection of belief points that could potentially offer a more encompassing approximation of the reachable belief space.

**Algorithm 10** COLLECTBELIEFS

```
1: function COLLECTBELIEFS($n, \xi_0$)
2:     $B \leftarrow \{\xi_0\}$                                          ▷ Initialization
3:     repeat
4:         Randomly select the belief state $\xi \in B$
5:         Randomly select an action $a \in \mathcal{A}$
6:         Randomly select $o \in \mathcal{O}$ according to $P(o \mid \xi, a)$ using Eq. (3.5.2)
7:         Calculate $\xi(\cdot \mid a, o)$ according to Eq. (3.5.1)
8:         $B \leftarrow B \cup \{\xi(\cdot \mid a, o)\}$                   ▷ Add new belief to set $B$
9:     until $|B| = n$
10:    return $B$
11: end function
```

The second distinction emerges in the update process—Algorithm 11—of the value function. PBVI, in its systematic fashion, updates the value at all belief points for every iteration of the value function. Conversely, PERSEUS focuses on a subset of belief points at each epoch, picked out by a randomized sampling process (Line 4). This process iterates until every belief point's value sees an improvement.

Interestingly, an update to the $\alpha$-vector at one belief point (Line 5) often induces value improvements for nearby belief points, leading to their subsequent removal from the sampling set (Line 9).

**Algorithm 11** UPDATE

```
1:  function UPDATE($B, V$)
2:      $B' \leftarrow B, V' \leftarrow \varnothing$                         ▷ Initialization
3:      while $B' \neq \varnothing$ do
4:          Randomly select a belief $\xi \in B'$
5:          $\alpha \leftarrow$ BACKUP($V, \xi$)                ▷ Backup defined by Eq. (3.8.8)
6:          if $\langle \xi, \alpha \rangle < V(\xi)$ then               ▷ $V(\xi) = \max_{\alpha' \in V} \langle \xi, \alpha' \rangle$
7:              $\alpha \leftarrow \arg\max_{\alpha' \in V} \langle \xi, \alpha' \rangle$     ▷ If $\alpha$ is not better, get $\alpha'$ from old set $V$
8:          end if
9:          $B' \leftarrow B' \setminus \{\varsigma \in B' \mid \langle \varsigma, \alpha \rangle \geq V(\varsigma)\}$   ▷ Keep beliefs not improved by $\alpha$.
10:         $V' \leftarrow V' \cup \{\alpha\}$                       ▷ Adds $\alpha$ to set $V'$
11:     end while
12:     $V \leftarrow V'$
13:     return $V$
14: end function
```

These methodological contrasts between PBVI and PERSEUS reflect a shift in strategic focus. PERSEUS leans towards a more random approach, promoting simplicity in concept but demonstrating effectiveness in empirical results (Spaan and Vlassis, 2005; Pineau *et al.*, 2006).

Now, we will go through the creating the BACKUP for PERSEUS as this will be a similar approach that we will follow when creating our CHRONOSPERSEUS solver for POSMDPs in Chap. 5. Recall that the Bellman equation for POMDP is

$$V^*(\xi) = \max_{a \in \mathcal{A}(s)} \left[ R(\xi, a) + \gamma \sum_{o \in \mathcal{O}} P(o \mid \xi, a) V^*\big(\xi(\cdot \mid a, o)\big) \right]. \tag{3.6.4}$$

Using Eq. (3.7.2),

$$V^*\big(\xi(\cdot \mid a, o)\big) = \max_{\alpha \in V} \langle \xi(\cdot \mid a, o), \alpha \rangle\,, \tag{3.8.1}$$

and substituting this into Eq. (3.6.4) becomes

$$\begin{aligned} V^*(\xi) &= \max_{a \in \mathcal{A}(s)} \left[ R(\xi, a) + \gamma \sum_{o \in \mathcal{O}} P(o \mid \xi, a) \max_{\alpha \in V} \langle \xi(\cdot \mid a, o), \alpha \rangle \right] && (3.8.2) \\ &= \max_{a \in \mathcal{A}(s)} \left[ R(\xi, a) + \gamma \sum_{o \in \mathcal{O}} P(o \mid \xi, a) \max_{\alpha \in V} \sum_{s' \in \mathcal{S}} \xi(s' \mid a, o) \alpha(s') \right]. && (3.8.3) \end{aligned}$$

Replacing the belief update $\xi(s' \mid a, o)$ with Eq. (3.5.1) yields

$$
\begin{aligned}
V^*(\xi) &= \max_{a\in\mathcal{A}(s)} \left[ R(\xi,a) + \gamma \sum_{o\in\mathcal{O}} \cancel{P(o \mid \xi, a)} \max_{\alpha\in V} \sum_{s'\in\mathcal{S}} \frac{G(o \mid a, s') \sum_{s\in\mathcal{S}} \xi(s) P(s' \mid s, a)}{\cancel{P(o \mid \xi, a)}} \alpha(s') \right] \\
&= \max_{a\in\mathcal{A}(s)} \left[ R(\xi,a) + \gamma \sum_{o\in\mathcal{O}} \max_{\alpha\in V} \sum_{s'\in\mathcal{S}} G(o \mid a, s') \sum_{s\in\mathcal{S}} \xi(s) P(s' \mid s, a) \alpha(s') \right] \\
&= \max_{a\in\mathcal{A}(s)} \left[ R(\xi,a) + \gamma \sum_{o\in\mathcal{O}} \max_{\alpha\in V} \sum_{s\in\mathcal{S}} \xi(s) \sum_{s'\in\mathcal{S}} \underbrace{G(o \mid a, s') P(s' \mid s, a) \alpha(s')}_{\alpha(s'|a,o)} \right] \\
&= \max_{a\in\mathcal{A}(s)} \left[ R(\xi,a) + \gamma \sum_{o\in\mathcal{O}} \max_{\alpha\in V} \langle \xi, \alpha(\cdot \mid a, o) \rangle \right]. \qquad (3.8.4)
\end{aligned}
$$

The reward $R(\xi, a)$ can be expressed as an inner product

$$R(\xi, a) = \sum_{s\in\mathcal{S}} \xi(s) R(s, a) = \langle \xi, R(\cdot, a) \rangle . \qquad (3.8.5)$$

Then, Eq. (3.8.4) becomes

$$V^*(\xi) = \max_{a\in\mathcal{A}(s)} \left[ \langle \xi, R(\cdot, a) \rangle + \gamma \sum_{o\in\mathcal{O}} \max_{\alpha\in V} \langle \xi, \alpha(\cdot \mid a, o) \rangle \right]$$

and by the distributive property of the inner product

$$\langle \omega, \psi \rangle + \langle \omega, \zeta \rangle = \langle \omega, \psi + \zeta \rangle \qquad (3.8.6)$$

we can factor out $\xi$ to get

$$V^*(\xi) = \max_{a\in\mathcal{A}(s)} \left[ \left\langle \xi, R(\cdot, a) + \gamma \sum_{o\in\mathcal{O}} \max_{\alpha\in V} \alpha(\cdot \mid a, o) \right\rangle \right] \qquad (3.8.7)$$

(Spaan and Vlassis, 2005). The backup can be written as

$$\text{BACKUP}(V, \xi) = \operatorname*{arg\,max}_{\alpha(\cdot|\xi,a):a\in\mathcal{A},\alpha\in V} \langle \xi, \alpha(\cdot \mid \xi, a) \rangle \qquad (3.8.8)$$

where

$$\alpha(s \mid \xi, a) = R(s, a) + \gamma \sum_{o\in\mathcal{O}} \operatorname*{arg\,max}_{\alpha(\cdot|a,o):\alpha\in V} \langle \xi, \alpha(\cdot \mid a, o) \rangle \qquad \forall s \in \mathcal{S}, \qquad (3.8.9)$$

and

$$\alpha(s \mid a, o) = \sum_{s'\in\mathcal{S}} G(o \mid a, s') P(s' \mid s, a) \alpha(s') \qquad \forall s \in \mathcal{S}. \qquad (3.8.10)$$

Let $|V|$ denote the number of $\alpha$ vectors are in the set $V$. The complexity of computing Eq. (3.8.10) is $O(|\mathcal{S}|^2)$ since it needs to be calculated for every $(s, s')$ tuple, and it is done for every $\alpha \in V$, hence computing all $\alpha(\cdot \mid a, o)$ for every $a \in \mathcal{A}$ and $o \in \mathcal{O}$ requires $O(|V| \times |\mathcal{S}|^2 \times |\mathcal{A}| \times |\mathcal{O}|)$. The complexity of computing Eq. (3.8.9) requires the computation of all relevant $\alpha(\cdot \mid a, o)$, but then the summation and inner products require only $O(|\mathcal{S}| \times |\mathcal{O}|)$ operations, and another $O(|\mathcal{S}|)$ operations to add the reward (vector). [This is considering it as a vector operations.] Lastly, the backup (Eq. (3.8.8)) requires for all $\alpha(\cdot \mid \xi, a)$ another $O(|\mathcal{S}|)$ operations for the inner product. Therefore, the full complexity of this point-based backup requires

$$O(|V| \times |\mathcal{S}|^2 \times |\mathcal{A}| \times |\mathcal{O}| + |\mathcal{S}| \times |\mathcal{A}| \times |\mathcal{O}|). \qquad (3.8.11)$$

However, a full backup for the sampled belief state set $B$ will not require $|B|$ times the complexity of a single point-based backup, because $\alpha(\cdot \mid a, o)$ are independent of the current belief state $\xi$ (the $\alpha(\cdot \mid a, o)$ vectors can be cached and reused for backups over other belief states $\xi \in B$). Thus, executing a backup for $|B|$ belief states over a single value function $V$, where we compute every $\alpha(\cdot \mid a, o)$ only once and cache the result, requires only

$$O(|V| \times |\mathcal{S}|^2 \times |\mathcal{A}| \times |\mathcal{O}| + |B| \times |\mathcal{S}| \times |\mathcal{A}| \times |\mathcal{O}|). \tag{3.8.12}$$

The algorithm requires an initial value function $V_0$ upon which to iterate. To conceive of an initial value function $V_0$ guaranteed to be below the optimal value function $V^*$, we assume for each decision epoch $n$ that the agent collects the smallest reward $\min_{(s,a)\in\mathcal{K}} R(s, a)$, even though it is not necessarily possible that the agent can land in state $s$ repeatedly. By making this assumption, we avoid having to find a feasible policy $\pi$ that returns the minimum cumulative discounted reward, which would require extensive computations (equivalent in computation to finding the maximum discounted expected reward). Thus, we set

$$R_{\min} \triangleq \min_{(s,a)\in\mathcal{K}} R(s, a) \tag{3.8.13}$$

and

$$a_{\min} \triangleq \operatorname*{arg\,min}_{(s,a)\in\mathcal{K}} R(s, a). \tag{3.8.14}$$

Then, the sum of discounted minimum reward (following not necessarily a feasible policy) at each decision epoch $n$ is

$$R_{\min} + \gamma R_{\min} + \gamma^2 R_{\min} + \cdots = R_{\min}(1 + \gamma + \gamma^2 + \cdots),$$

which forms a geometric series, so

$$R_{\min} \sum_{n=0}^{\infty} \gamma^n = R_{\min}\left(\frac{1}{1-\gamma}\right),$$

for $\gamma \in (0, 1)$.

We create a single vector $\alpha_{\min}$ in which each component $s \in \mathcal{S}$ is

$$\alpha_{\min}(s) = \frac{R_{\min}}{1-\gamma}, \tag{3.8.15}$$

and $\alpha_{\min}$ has an associated action $a_{\min}$. Then, the initial value function $V_0$ is the set that consists of the single vector $\alpha_{\min}$:

$$V_0 = \{\alpha_{\min}\} \tag{3.8.16}$$

(Zhang and Zhang, 2001, Lemma 3).

## 3.9 Optimal Stopping Problems Revisited

Recall from Sec. 2.7 that an optimal stopping problem—a form of temporal control—is a sequential decision process where an agent must determine the optimal time to halt the ongoing process. We previously studied the elevator problem in Example 2.7.1 (p. 16), which involved an agent waiting for an elevator to arrive for a meeting. This problem, originally formulated as an MDP, showcased temporal control as the agent had to decide optimally when to continue waiting or opt for an alternative action. Having determined an optimal policy for this scenario, we now extend the problem's complexity by introducing partial observability. In this new context, the agent is uncertain about the functioning status of the elevator, thus further highlighting the element of temporal control as the agent navigates the uncertainty in deciding the optimal timing of actions.

**Example 3.9.1 — Elevator Problem (POMDP version).** Suppose that an agent arrives at the lobby (on floor 1) $N$ minutes before its meeting begins. If the elevator arrives before the meeting starts, it earns reward $r$. However, the agent does not enjoy waiting, so to reflect its dislike it incurs a cost $c$ each minute it waits. The agent can see on which floor, $s$, the elevator is on currently by the floor indicator above the elevator door. The agent does not know with certainty if the elevator is working ($w = 1$) or not ($w = 0$); the elevator works 75% of the time.

We can model this problem using the POMDP framework. The state space consists of the floor numbers, the time in minutes (or the decision epoch index), and whether or not the elevator is working,

$$\mathcal{S} = \underbrace{\{1, 2, \dots, S\}}_{\text{floor}} \times \underbrace{\{0, 1, \dots, N-1\}}_{\text{remaining time}} \times \underbrace{\{0, 1\}}_{\text{working}},$$

and the number of states is

$$|\mathcal{S}| = S \times N \times 2.$$

A typical state would be the triple $(s, n, w) \in \mathcal{S}$, where the elevator is on floor $s$, the remaining time $n$ until the meeting, and the functioning state of the elevator $w$. The action space consists of two actions,

$$\mathcal{A} = \{\text{wait, stairs}\}.$$

While the state space has only doubled in size when compared to the MDP version (Example 2.7.1), we must remember that we are now operating within a continuous belief space due to the partial observability of the elevator system. Our agent must maintain a belief—a continuous probability distribution—over all possible states. This shift from a discrete state space to a continuous belief space brings about a significant increase in the complexity of the problem. This phenomenon is commonly referred to as the "curse of dimensionality," which denotes the increase in computational complexity as the dimensionality of the state space (or in our case, the belief space) increases. Despite these challenges, POMDPs provide a more realistic model for many decision-making scenarios, such as our elevator problem, where certain aspects of the system state are not directly observable.

## 3.10 Mixed observability and MOMDPs

Ong *et al.* (2010) introduces the idea of mixed observable Markov decision processes (MOMDPs). Unlike partial observability where the state is not observed but instead an observation is generated by a probability distribution, *mixed observability* occurs when the state space is neither fully hidden nor fully observable by the agent. The state space can be decomposed into finite subspaces in which certain state subspaces are fully observable, while other state subspaces are not observable or hidden from the agent. This means that subspaces of the state space can form part of a subspace or the entirety of the observation space $\mathcal{O}$.

We have already encountered a MOMDP in the Elevator Example 3.9.1. The state space

$$\mathcal{S} = \overbrace{\underbrace{\{1, 2, \dots, S\}}_{\text{floor}}}^{\text{observable}} \times \overbrace{\underbrace{\{0, 1, \dots, N-1\}}_{\text{remaining time}}}^{\text{observable}} \times \overbrace{\underbrace{\{0, 1\}}_{\text{working}}}^{\text{hidden}},$$

is a cross product of three subspaces, of which the floor subspace and the remaining time subspace are fully observable to the agent, while the working subspace is not. The observation space is

$$\mathcal{O} = \underbrace{\{1, 2, \ldots, S\}}_{\text{floor}} \times \underbrace{\{0, 1, \ldots, N-1\}}_{\text{remaining time}},$$

and it is clear that $|\mathcal{O}| < |\mathcal{S}|$. (Note that $\mathcal{O} \not\subset \mathcal{S}$ since $(s, n) \in \mathcal{O}$ while $(s, n, w) \in \mathcal{S}$.)

## 3.11 Bibliographic Remarks

The significance of observation in Bayesian statistics was underscored by Savage (1954, Chap. 6). Savage's advocacy for the Bayesian perspective in statistics was considered contentious at the time. Notably, he extended his exploration to tackle a two-state partially observable problem, referred to as *partition problems* in Chap. 7 in his textbook.

Over the years, POMDPs have been the subject of extensive research under various nomenclatures. The concept of *observable history* has also been termed as the *information vector* by other authors, for instance, Bertsekas (2017, p. 185).

Ross *et al.* (2008b) delved into online planning algorithms specifically designed for POMDPs, contributing further to the literature. For a comprehensive review of the state-of-the-art algorithms for solving and approximating POMDPs up to 2012, we refer the reader to Spaan (2012). We also recommend an excellent recent textbook on POMDPs from the controlled sensing community by Krishnamurthy (2016).

Sutton and Barto (2018) introduce POMDP as a Bayesian approach that uses belief states and a belief update rule (similar to Eq. (3.5.1)). However, they believe that "this approach is popular in theoretical work and has many significant applications, but its assumptions and computational complexity scale poorly and we do not recommend it as an approach to artificial intelligence" (Sutton and Barto, 2018, p. 467). Although, it seems that POMDPs were important enough to be included in the state-of-the-art reinforcement learning monograph (Spaan, 2012). We would like to stress that POMDP is a model and not a computational approach to finding the optimal policies. The assumption that Sutton and Barto refers to is the Markov property. The Markov property requires information that is required to make an optimal decision be encoded in the state. If the decision is dependent on the history, then the state will need to drag the history into the state in order to maintain Markovness, and thus increasing the curse of dimensionality. Sutton's work in relaxing the Markov assumption to just say $k$ previous steps has lead to predictive state representations.

Littman and Sutton (2002) developed the predictive state representation model, or PSR, which is a class of model that represents the state of a dynamical system as a set of predictions about future events based on previously observed evidence. PSRs define the state vector in terms of actions the agent can take and observations that the agent can see. It describes a distribution over observations, from which the agent can get a sample, thus revealing information about the distribution described by the state. Littman and Sutton (2002) had shown that from a finite POMDP could be constructed a linear PSR with a number of tests no larger than the number of states in the minimal POMDP.

In reference to POMDPs computational complexity, Papadimitriou and Tsitsiklis (1987) showed that in the worst case scenario, an optimal exact policy computation for a finite-horizon POMDP is PSPACE-complete[3], and Madani *et al.* (2003) showed for an infinite-horizon POMDP is uncomputable/undecidable. Indeed, this is quite a difficult problem, but this has not deterred the artificial intelligence community. It is difficult to solve POMDPs exactly, but there continues to be a growing tract of research in approximate solutions.

[3]PSPACE-complete is a complexity class representing decision problems that can be solved using an amount of memory that is polynomial in the input length, *i.e.,*requiring polynomial space. It is believed to include problems that are potentially more difficult than NP-complete ones. Importantly, any other problem solvable in polynomial space can be transformed into a PSPACE-complete problem in polynomial time, making this class a robust representative of problems solvable in polynomial space.

# 4
# The Semi-Markov Decision Process (SMDP) Model

The Markov Decision Process (MDP) model and the problems that we have encountered in Chap. 2 assume that decisions happen at regular intervals of time. However, there are situations where actions selected may lead to random amount of time before another action can be taken. In those cases, it may be useful to extend the MDP to include continuous time, by treating the time between decision epochs as a random variable conditional on the agent's state and its action it selects. This type of process is called a *semi-Markov decision process* (SMDP).

$$\begin{tikzcd} \text{MDP} \arrow[d, "\text{continuous time} +"'] \\ \text{SMDP} \end{tikzcd}$$

Fig. 4.1 The relationship between MDP and SMDP

## 4.1 The Framework

We define a semi-Markov decision process (SMDP) in a Borel space, which is a generalization of an MDP, in the following definition.

**Definition 4.1.1 — SMDP.** A semi-Markov decision process (SMDP) is an 7-tuple

$$\langle \mathcal{S}, \mathcal{A}, \mathcal{K}, Q, R, \beta, N \rangle \tag{4.1.1}$$

where

| | |
|---|---|
| $\mathcal{S}$ | is the Borel *state space*, and the elements of $\mathcal{S}$ are called *states* |
| $\mathcal{A}$ | is the Borel *action space*, and the elements of $\mathcal{A}$ are called *actions*. To each $s \in \mathcal{S}$, we associate a nonempty Borel-measurable subset $\mathcal{A}(s) \subseteq \mathcal{A}$, whose elements are the *admissible actions* for the agent when the process is in state $s$ |
| $\mathcal{K}$ | is the set of admissible state-action pairs, and it is assumed to be a Borel subset $\mathcal{K} \subseteq \mathcal{S} \times \mathcal{A}$. In other words, $\mathcal{K} = \{(s,a) \mid s \in \mathcal{S}, a \in \mathcal{A}(s)\}.$ |
| $Q(\tau, s' \mid s, a)$ | is the sojourn time-state (Borel-measurable) stochastic kernel on $\mathbb{R}_{>0}$ given $\mathcal{K}$ |
| $R(s,a)$ | is the per-stage (bounded Borel-measurable) reward function given $\mathcal{K}$ |

| | |
|---|---|
| $\beta$ | is the positive discounting rate |
| $N$ | is the planning horizon. It could be finite, or $N = \infty$. |

**Algorithm 12** Dynamics of SMDP

```
In the beginning $n = 0$, the agent observes it is in state $s_0$.
For each decision epoch $n = 1, 2, \dots, N$:
    (a) Based on the history
            $h_0 = (t_0, s_0)$
            $h_n = (t_0, s_0, a_1, t_1, \dots, s_{n-1}, a_n, t_n, s_n)$,
        the agent performs action
            $a_n = \pi_n(h_n) \in \mathcal{A} \qquad n = 1, 2, \dots, N.$
        Here $\pi_n$ denotes a policy that the agent uses at decision epoch $n$.
    (b) The agent obtains a reward $R(s_n, a_n)$ for choosing action $a_n$ at decision epoch $n$.
    (c) The state evolves randomly with sojourn time-state transition probability
            $Q(\tau, s' \mid s, a) = \mathbf{P}(T_{n+1} - T_n \le \tau, S_{n+1} = s' \mid S_n = s, A_n = a)$
        to the next state $S_{n+1}$ that decision epoch $n + 1$.
    (d) The agent updates its history as
            $h_{n+1} = (h_n, a_{n+1}, t_{n+1}, s_{n+1})$.
        If $n < N$, then set $n$ to $n + 1$, and go back to step (a).
        If $n = N$, then the agent receives the last reward and the process terminates.
```

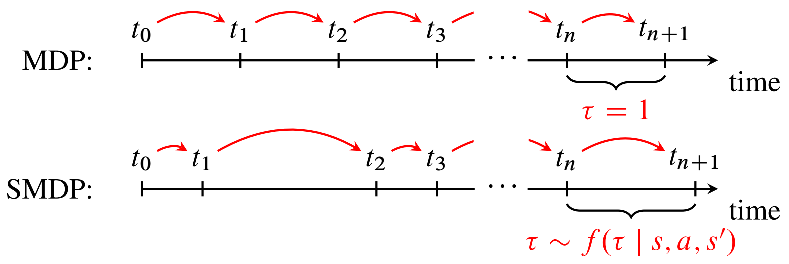


Fig. 4.2 The difference between MDPs and SMDPs for sojourn times. The sojourn time for an MDP is constant whereas the sojourn time for an SMDP is a random amount of time that follows some probability distribution.

We assume that state transitions and actions performed take place at discrete times called *decision epochs*, but unlike MDPs, the *sojourn time*—the random amount of time it takes to transition from one state to the next—follows a non-exponential probability distribution; see Fig. 4.2. We will denote the sojourn time by $\tau$. The state $s \in \mathcal{S}$ and action $a \in \mathcal{A}(s)$ at any time $t$ are denoted by $s(t)$ and $a(t)$, respectively, and stay constant between decision epochs. We will use the following notation:

| | |
|---|---|
| $t_n$ | is the time of the $n^{\text{th}}$ decision epoch. By convention, $t_0 = 0$; |
| $s_n = s(t_n)$ | where $s(t) = s(t_n)$ for $t \in [t_n, t_{n+1})$; and |
| $a_n = a(t_n)$ | where $a(t) = a(t_n)$ for $t \in [t_n, t_{n+1})$. |

We illustrate these terms with Fig. 4.3. The sojourn time-state stochastic kernel specifies the joint probability distribution of the sojourn time and the next state, which

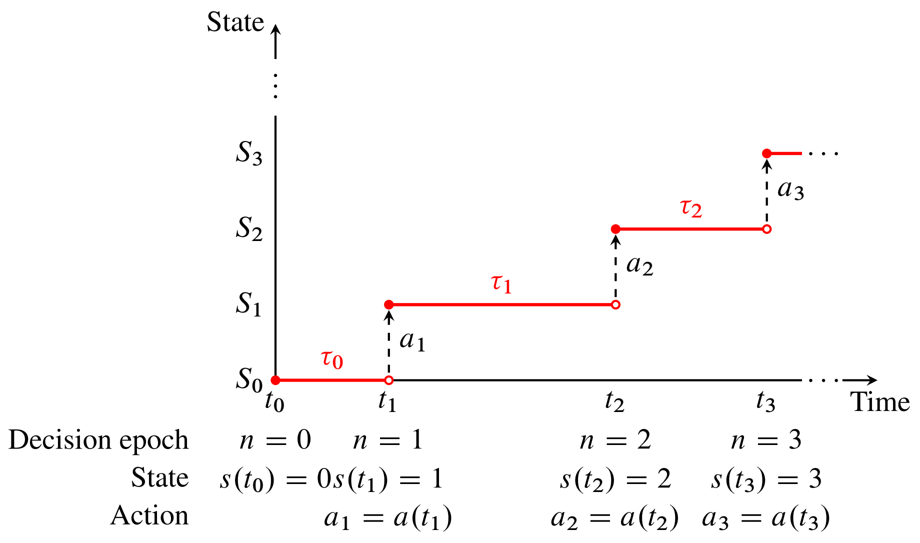


Fig. 4.3 SMDP dynamics

is given by

$$Q(\tau, s' \mid s, a) \triangleq \mathbf{P}(T_{n+1} - T_n \leq \tau, S_{n+1} = s' \mid S_n = s, A_n = a). \tag{4.1.2}$$

By specifying (4.1.2) as a joint probability distribution, the framework does not assume necessarily that the sojourn time $\tau$ and the next state $s'$ are independent.[1] Note that the sojourn time-state stochastic kernel also specifies the state (Borel-measurable) stochastic kernel on $\mathcal{S}$ given $\mathcal{K}$, since

$$\lim_{\tau \to \infty} Q(\tau, s' \mid s, a) = \mathbf{P}(S_{n+1} = s' \mid S_n = s, A_n = a) = P(s' \mid s, a), \tag{4.1.3}$$

which explains its absence from the framework (4.1.1). If $Q(\tau, s' \mid s, a)$ is continuous and piecewise differentiable with respect to the sojourn time $\tau$, then its partial derivative exists, which is

$$\begin{aligned} q(\tau, s' \mid s, a) &\triangleq \frac{\mathrm{d}}{\mathrm{d}\tau} Q(\tau, s' \mid s, a) && (4.1.4)\\ &= \frac{\mathrm{d}}{\mathrm{d}\tau} [P(s' \mid s, a) F(\tau \mid s, a, s')] \\ &= P(s' \mid s, a) \frac{\mathrm{d}}{\mathrm{d}\tau} [F(\tau \mid s, a, s')] \\ &= P(s' \mid s, a) f(\tau \mid s, a, s'). && (4.1.5) \end{aligned}$$

We will generally use the sojourn time-state transition probability $Q(\tau, s' \mid s, a)$ because it is able to model not only discrete distributions for $\tau$, but continuous and mixed distributions as well.

[1]This assumption of $\tau$ and $s'$ independence is made in Puterman (1994, p. 532) in Eq. (11.1.2) where $Q(\tau, s' \mid s, a) = P(s' \mid s, a) F(\tau \mid s, a)$.

The conditional cumulative distribution function of $\tau$ given $s$, $a$, and $s'$ is

$$\begin{aligned} F(\tau \mid s,a,s') &\triangleq \mathbf{P}(T_{n+1}-T_n \leq \tau \mid S_n = s, A_n = a, S_{n+1} = s') \\ &= \frac{\mathbf{P}(T_{n+1}-T_n \leq \tau, S_{n+1} = s', S_n = s, A_n = a)}{\mathbf{P}(S_n = s, A_n = a, S_{n+1} = s')} \\ &= \frac{\mathbf{P}(T_{n+1}-T_n \leq \tau, S_{n+1} = s' \mid S_n = s, A_n = a)\,\mathbf{P}(S_n = s, A_n = a)}{\mathbf{P}(S_{n+1} = s' \mid S_n = s, A_n = a)\,\mathbf{P}(S_n = s, A_n = a)} \\ &= \frac{\mathbf{P}(T_{n+1}-T_n \leq \tau, S_{n+1} = s' \mid S_n = s, A_n = a)}{\mathbf{P}(S_{n+1} = s' \mid S_n = s, A_n = a)} \\ &= \frac{Q(\tau, s' \mid s,a)}{P(s' \mid s,a)} \end{aligned} \qquad (4.1.6)$$

We can similarly show that the conditional probability density function of $\tau$ given $s$, $a$, and $s'$ is

$$f(\tau \mid s,a,s') = \frac{q(\tau, s' \mid s,a)}{P(s' \mid s,a)}. \qquad (4.1.7)$$

Assuming that $P(s' \mid s,a) > 0$, Eq. (4.1.6) implies that

$$Q(\tau, s' \mid s,a) = P(s' \mid s,a)F(\tau \mid s,a,s'). \qquad (4.1.8)$$

Thus, we can view $Q(\tau, s' \mid s,a)$ as a scaled cumulative distribution function (which is scaled/multiplied by $P(s' \mid s,a)$).

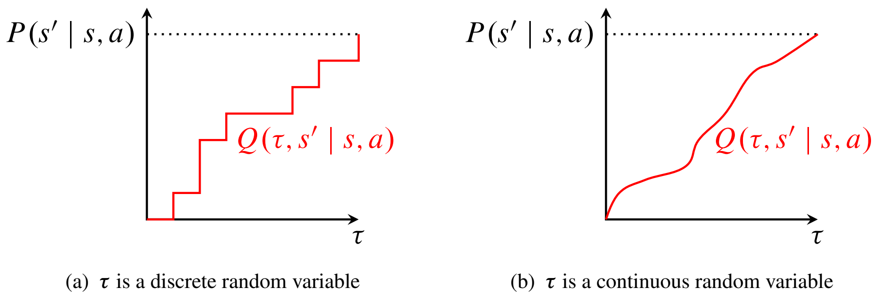


Fig. 4.4 Illustrating $Q(\tau, s' \mid s,a)$ and the conditional cumulative distribution function of $\tau$.

There are two special cases of SMDPs.

(a) *Continuous-time Markov decision process*. This process occurs when the sojourn times between decision decision epochs are exponentially distributed with rate $\beta(s,a,s')$ (or mean $1/\beta(s,a,s')$); that is,

$$F(\tau \mid s,a,s') = 1 - \mathrm{e}^{-\beta(s,a,s')\tau} \qquad (4.1.9)$$

and

$$f(\tau \mid s,a,s') = \beta(s,a,s')\mathrm{e}^{-\beta(s,a,s')\tau}. \qquad (4.1.10)$$

(b) *Discrete-time Markov decision process*. This process has decision epochs occurring at every $\tau'$ time unit. For some fixed $\tau'$ for all $(s,a) \in \mathcal{K}$,

$$F(\tau \mid s,a,s') = \begin{cases} 0, & \tau \leq \tau'; \\ 1, & \tau > \tau'. \end{cases} \qquad (4.1.11)$$

Discrete-time Markov decision processes were discussed in Chap. 2, and arise regularly in practice since many decisions are made on a weekly, monthly, or annual basis. In contrast, it is not common for decisions to be made at exponentially-distributed sojourn times (Howard, 1963, p. 626).

In order to ensure that there are only a finite number decision epochs in a finite amount of time, we make the following assumption:

**Assumption 4.1.2** There exists a sojourn time $\tau > 0$ and $\epsilon > 0$ such that

$$\int_{s' \in \mathcal{S}} Q(\tau, \mathrm{d}s' \mid s, a) \leq 1 - \epsilon \qquad \forall (s, a) \in \mathcal{K}.$$

This assumption was introduced by Ross (1970a, p. 157). Essentially, it says that

**Assumption 4.1.3** Given a state $s' \in \mathcal{S}$,

$$0 < \bar{\tau}(s' \mid s, a) < \infty \qquad \forall (s, a) \in \mathcal{K}.$$

In other words, Assumption 4.1.3 says that the expected sojourn time between state $s$ and state $s'$ under action $a$ is finite.

## 4.2 Probability Distributions for Sojourn Time

This section introduces three probability distributions suitable for modelling sojourn-time: the deterministic, the inverse Gaussian, and the truncated Gaussian distributions. These distributions have been selected for their positive support, making them appropriate candidates for time duration representations in our context.

### 4.2.1 The Deterministic Distribution

The probability mass function of a random variable $X$ that follows the deterministic distribution is

$$f_X(x \mid c_0) = \begin{cases} 1, & \text{if } x = c_0; \\ 0, & \text{otherwise} \end{cases} \tag{4.2.1}$$

where $c_0 \in (-\infty, \infty)$ is a constant. The cumulative distribution function is

$$F_X(x \mid c_0) = \begin{cases} 0, & \text{if } x < c_0; \\ 1, & \text{if } x \geq c_0. \end{cases} \tag{4.2.2}$$

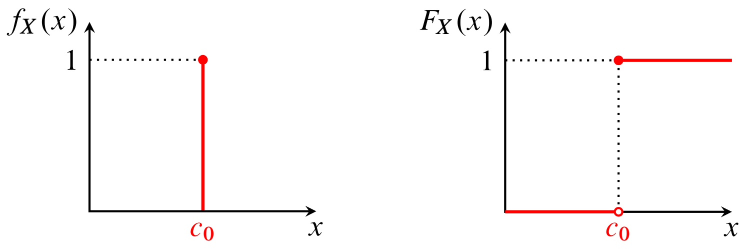
Even though it is deterministic, Eq. (4.2.1) and Eq. (4.2.2) satisfies the definition of being a distribution of a random variable; hence, it is a degenerate case.

### 4.2.2 The Inverse Gaussian Distribution

The inverse Gaussian distribution is a long-tailed distribution that is positively skewed, unimodal, with positive support; see Fig. 4.6. Schrödinger (1915) first encountered it while he was investigating the distribution of when a Brownian motion particle with positive drift first passes a set threshold. It has since been found to be a useful distribution for modelling life-span and reaction-time studies.

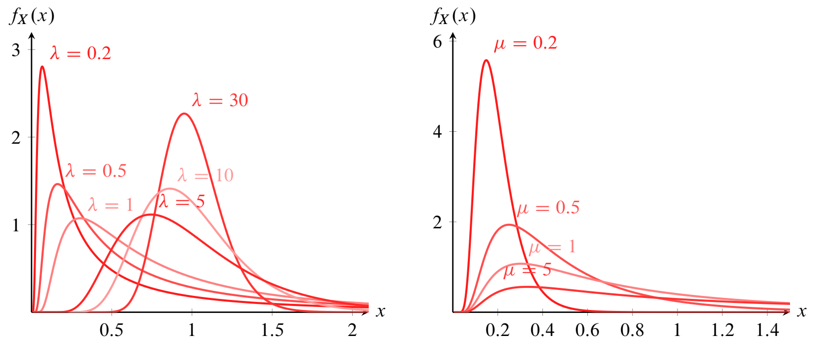
The probability density function of a random variable $X$ that follows the inverse Gaussian distribution is

$$f_X(x \mid \mu, \lambda) = \sqrt{\frac{\lambda}{2\pi x^3}} \exp\left[-\frac{\lambda(x-\mu)^2}{2\mu^2 x}\right], \qquad \forall x > 0, \tag{4.2.3}$$



(a) The probability density function (b) The cumulative probability function

Fig. 4.5 The deterministic distribution



(a) $\mu = 1$ for six values of $\lambda$ (b) $\lambda = 1$ for four values of $\mu$

Fig. 4.6 The inverse Gaussian probability density distribution

where the location parameter $\mu > 0$ is the mean of the distribution and $\lambda > 0$ is the shape parameter[2] (Tweedie, 1957, Eq. (1b)). A random variable $X$ that follows an inverse Gaussian distribution will be denoted by $X \sim \mathcal{IG}(\mu, \lambda)$. The standard inverse Gaussian random variable follows $\mathcal{IG}(1, 1)$.

In terms of the standard Gaussian cumulative distribution function (with expected value of 0 and variance 1) given by

$$\Phi(x) = \frac{1}{\sqrt{2\pi}} \int_{-\infty}^{x} \mathrm{e}^{-\frac{1}{2}u^2}\, \mathrm{d}u, \tag{4.2.4}$$

the cumulative distribution function of a random variable $X$ that follows the inverse Gaussian distribution is given by

$$F_X(x \mid \mu, \lambda) = \Phi\left[\sqrt{\frac{\lambda}{x}}\left(\frac{x}{\mu} - 1\right)\right] + \mathrm{e}^{2\lambda/\mu}\, \Phi\left[-\sqrt{\frac{\lambda}{x}}\left(\frac{x}{\mu} + 1\right)\right] \tag{4.2.5}$$

(Shuster, 1968; Chhikara and Folks, 1974).

For a random sample $X_1, X_2, \ldots, X_n$ from $\mathcal{IG}(\mu, \lambda)$, the maximum likelihood estimators of $\mu$ and $\lambda$ are

$$\hat{\mu} = \bar{X} \tag{4.2.6}$$

and

$$\hat{\lambda} = \frac{n}{\sum_{i=1}^{n} \left(\frac{1}{X_i} - \frac{1}{\bar{X}}\right)} = \frac{n}{\sum_{i=1}^{n} \frac{1}{X_i} - \frac{n}{\bar{X}}}, \tag{4.2.7}$$

[2]A *shape parameter* affects solely the shape of the distribution, in contrast with a *location parameter* that shifts the distribution or a *scale parameter* that shrinks or stretches the distribution.

where

$$\bar{X} = \frac{1}{n}\sum_{i=1}^{n} X_i, \tag{4.2.8}$$

(Tweedie, 1957, Eq. (13) and Eq. (14)). The variance is $\sigma_X^2 = \mu^3/\lambda$. However, the maximum likelihood estimate of the variance $\hat{\mu}^3/\hat{\lambda}$ is biased.

#### 4.2.2.1 Estimation of the inverse Gaussian parameters

The uniformly minimum-variance unbiased estimator[3] (UMVUE) is given by

$$\tilde{\mu} = \hat{\mu} = \bar{X} \tag{4.2.9}$$

and

$$\tilde{\lambda} = \frac{n-3}{\sum_{i=1}^{n}\left(\frac{1}{X_i} - \frac{1}{\bar{X}}\right)} = \frac{n-3}{\sum_{i=1}^{n}\frac{1}{X_i} - \frac{n}{\bar{X}}} \tag{4.2.10}$$

(Iwase and Setô, 1983, Table 1).

#### 4.2.2.2 Generating inverse Gaussian samples

The method of inverse Gaussian variate generation is based on Michael *et al.* (1976) who uses the following $\chi^2_{(1)}$-distribution transformation of Shuster (1968, Theorem 1)

$$V = g(X) = \frac{\lambda(X-\mu)^2}{\mu^2 X} \sim \chi^2_{(1)}. \tag{4.2.11}$$

For each $\chi^2$ variate, $v$, we solve for $x$ and find the two roots of the resulting quadratic equation:

$$x_1 = \mu + \frac{\mu^2 v}{2\lambda} - \frac{\mu}{2\lambda}\sqrt{4\mu\lambda v + \mu^2 v^2} \tag{4.2.12}$$

and

$$x_2 = \frac{\mu^2}{x_1}. \tag{4.2.13}$$

Now, it remains to choose between the two roots for our inverse Gaussian variate. A Bernoulli trial is performed where the final inverse Gaussian variate, $x$, is selected by

$$x = \begin{cases} x_1 & \text{with probability } \frac{\mu}{\mu+x_1} \\ x_2 & \text{with probability } 1 - \frac{\mu}{\mu+x_1}. \end{cases} \tag{4.2.14}$$

Based on Box and Muller (1958), the Gaussian variate requires two uniform variates, and an additional uniform variate for the Bernoulli trial. Thus, the complexity of this algorithm requires three uniform variates for every inverse Gaussian variate generated. We summarize our discussion with Algorithm 13. PyTorch code is provided in Appendix A.2.

### 4.2.3 The Gaussian Distribution

The Gaussian probability density function is

$$f_X(x \mid \mu, \sigma^2) = \frac{1}{\sqrt{2\pi\sigma^2}} \exp\left(-\frac{(x-\mu)^2}{2\sigma^2}\right) \tag{4.2.15}$$

and the cumulative distribution function is

$$\Phi\left(\frac{x-\mu}{\sigma}\right) \triangleq F_X(x \mid \mu, \sigma^2) = \frac{1}{\sqrt{2\pi}} \int_{-\infty}^{x} \mathrm{e}^{-\frac{1}{2}u^2}\, \mathrm{d}u. \tag{4.2.16}$$

[3] A *uniformly minimum-variance unbiased estimator* is an unbiased estimator that has the lower variance than any other unbiased estimator for all possible values of the parameter. It is not necessarily unique.

**Algorithm 13** Generating an inverse Gaussian variate, Michael *et al.* (1976)

```
function RANDOMIG($\mu$, $\lambda$)
    $z \sim \mathcal{N}(0, 1)$                ▷ Generate a standard Gaussian variate.
    $z \leftarrow z^2$                         ▷ Square it so becomes a $\chi^2_{(1)}$ variate.
    $x \leftarrow \mu + \frac{\mu^2 z}{2\lambda} - \frac{\mu}{2\lambda}\sqrt{4\mu\lambda z + \mu^2 z^2}$
    $y \sim \mathcal{U}(0, 1)$                ▷ Generate a uniform variate on [0, 1].
    if $y \geq \frac{\mu}{\mu + x}$ then
        $x \leftarrow \frac{\mu^2}{x}$
    end if
    return $x$
end function
```

## 4.2.4 The Truncated Gaussian Distribution

The truncated Gaussian distribution is a derived Gaussian distribution where the support is bounded below, or bounded above, or both.

The probability density function of a random variable $X$ that follows the truncated Gaussian distribution is

$$f_X(x \mid \mu, \sigma^2, a, b) = \frac{f_X(x \mid \mu, \sigma^2)}{\Phi\left(\frac{b-\mu}{\sigma}\right) - \Phi\left(\frac{a-\mu}{\sigma}\right)} \qquad \forall x \in [a, b], \tag{4.2.17}$$

where $\mu$ is the mean, $\sigma^2$ is the variance, $f_X(\cdot \mid \mu, \sigma^2)$ is Gaussian distribution were it not truncated as defined in Eq. (4.2.15), and $\Phi(\cdot)$ is the standard Gaussian cumulative distribution function defined in Eq. (4.2.16) (Robert, 1995, p. 123). A random variable $X$ that follows a truncated Gaussian distribution will be denoted by $X \sim \mathcal{N}(\mu, \sigma^2, a, b)$. Note that $X \sim \mathcal{N}(\mu, \sigma^2)$ would denote a Gaussian distribution unless the additional truncated points $a$ and $b$ are specified.

The cumulative distribution function of a random variable $X$ that follows the truncated Gaussian distribution is given by

$$F_X(x \mid \mu, \sigma^2, a, b) = \begin{cases} 0, & \text{if } x \in (-\infty, a); \\ \dfrac{\Phi\left(\frac{x-\mu}{\sigma}\right) - \Phi\left(\frac{a-\mu}{\sigma}\right)}{\Phi\left(\frac{b-\mu}{\sigma}\right) - \Phi\left(\frac{a-\mu}{\sigma}\right)}, & \text{if } x \in [a, b]; \\ 1, & \text{if } x \in (b, \infty). \end{cases} \tag{4.2.18}$$

**Example 4.2.1 —** $X \sim \mathcal{N}(\mu, \sigma^2, 0, \infty)$. The truncated Gaussian distribution where the support is $[0, \infty)$.

$$\lim_{b \to \infty} \frac{b-\mu}{\sigma} = \infty$$

which means that

$$\lim_{b \to \infty} \Phi\left(\frac{b-\mu}{\sigma}\right) = 1$$

Then Eq. (4.2.15) becomes

$$f_X(x \mid \mu, \sigma^2, 0, \infty) = \frac{f_X(x \mid \mu, \sigma^2)}{1 - \Phi\left(-\frac{\mu}{\sigma}\right)} \qquad \forall x \in [0, \infty), \tag{4.2.19}$$

and Eq. (4.2.16) becomes

$$F_X(x \mid \mu, \sigma^2, 0, \infty) = \begin{cases} 0, & \text{if } x \in (-\infty, 0); \\ \dfrac{\Phi\left(\frac{x-\mu}{\sigma}\right) - \Phi\left(-\frac{\mu}{\sigma}\right)}{1 - \Phi\left(-\frac{\mu}{\sigma}\right)}, & \text{if } x \in [0, \infty). \end{cases} \tag{4.2.20}$$

◀

## 4.3 Reward Structure

We assume that the reward function $R(s, a)$ is given as part of the SMDP model. Let $R(s, a)$ denote the expected total discounted reward between two decision epochs, given that the agent is in state $s \in \mathcal{S}$ and it chooses to perform action $a \in \mathcal{A}(s)$, then

$$R(s,a) = \underbrace{r_1(s,a)}_{\substack{\text{(a)}\\ \text{immediate}\\ \text{reward}}} + \underbrace{\int_{s' \in \mathcal{S}} \int_0^\infty \left( \int_0^\tau \mathrm{e}^{-\beta t} r_2(t \mid s, a, s')\, \mathrm{d}t \right) Q(\mathrm{d}\tau, \mathrm{d}s' \mid s, a)}_{\substack{\text{(b)}\\ \text{expected discounted reward reward from } s \text{ to } s' \text{ over } \tau \text{ amount of time}}} . \tag{4.3.1}$$

The reward function is comprised of two parts:

(a) the lump sum reward $r_1(s, a)$ that is received immediately upon the agent performing action $a$ in state $s$; and

(b) the continuous reward rate $r_2(\tau \mid s, a, s')$ that is received by the agent continuously over the sojourn time $\tau$ from state $s$ to $s'$ under action $a$.

Note that $R(s, a)$ depends on the discount rate $\beta$ that is also given as part of the model.

If the state space $\mathcal{S}$ is discrete, and $Q$ is differentiable and separable into $P$ and $F$, then

$$R(s,a) = r_1(s,a) + \sum_{s' \in \mathcal{S}} P(s' \mid s,a) \int_0^\infty \left( \int_0^\tau \mathrm{e}^{-\beta t} r_2(t \mid s,a,s')\, \mathrm{d}t \right) F(\mathrm{d}\tau \mid s,a,s'). \tag{4.3.2}$$

It is common to assume that $r_2(t \mid s, a, s') = r_2(s, a, s')$ is a constant rate. If this is the case, then

$$\begin{aligned}
R(s,a) &= r_1(s,a) + \sum_{s' \in \mathcal{S}} P(s' \mid s,a)\, r_2(s,a,s') \int_0^\infty \left( \int_0^\tau \mathrm{e}^{-\beta t}\, \mathrm{d}t \right) F(\mathrm{d}\tau \mid s,a,s') \\
&= r_1(s,a) + \sum_{s' \in \mathcal{S}} P(s' \mid s,a)\, r_2(s,a,s') \int_0^\infty \frac{1 - \mathrm{e}^{-\beta\tau}}{\beta} F(\mathrm{d}\tau \mid s,a,s') \\
&= r_1(s,a) + \frac{1}{\beta} \sum_{s' \in \mathcal{S}} P(s' \mid s,a)\, r_2(s,a,s') \int_0^\infty (1 - \mathrm{e}^{-\beta\tau}) F(\mathrm{d}\tau \mid s,a,s') \\
&= r_1(s,a) + \frac{1}{\beta} \sum_{s' \in \mathcal{S}} P(s' \mid s,a)\, r_2(s,a,s') \Bigg( \int_0^\infty F(\mathrm{d}\tau \mid s,a,s') \\
&\qquad - \int_0^\infty \mathrm{e}^{-\beta\tau} F(\mathrm{d}\tau \mid s,a,s') \Bigg) \\
&= r_1(s,a) + \frac{1}{\beta} \sum_{s' \in \mathcal{S}} P(s' \mid s,a)\, r_2(s,a,s') \left( 1 - \int_0^\infty \mathrm{e}^{-\beta\tau} F(\mathrm{d}\tau \mid s,a,s') \right) && (4.3.3) \\
&= r_1(s,a) + \frac{1}{\beta} \sum_{s' \in \mathcal{S}} P(s' \mid s,a)\, r_2(s,a,s') \left( 1 - \int_0^\infty \mathrm{e}^{-\beta\tau} f(\tau \mid s,a,s')\, \mathrm{d}\tau \right). && (4.3.4)
\end{aligned}$$

## 4.4 Laplace Transform

In Eq. (4.3.4), the integral $\int_0^\infty \mathrm{e}^{-\beta\tau} f(\tau \mid s, a, s')\,\mathrm{d}\tau$ can be calculated using a Laplace tranform formula of the sojourn time probability density function. The next three examples cover distributions that we will use in this thesis.

**Example 4.4.1 — Deterministic distribution.** The moment-generating function of a random variable $X$ that follows a deterministic function with constant parameter $c_0 \in (-\infty, \infty)$ is

$$M_X(t) = \mathbf{E}(\mathrm{e}^{tX}) = \int_{-\infty}^{\infty} \mathrm{e}^{tx} f_X(x)\,\mathrm{d}x = \mathrm{e}^{tc_0}. \tag{4.4.1}$$

Assuming that $c_0$ is positive, we can calculate

$$\int_0^\infty \mathrm{e}^{-\beta\tau} f_X(\tau \mid c_0)\,\mathrm{d}\tau \tag{4.4.2}$$

using the moment-generating function for the deterministic distribution by changing the variable $t$ by $-\beta$ in Eq. (4.4.1) yields

$$\int_0^\infty \mathrm{e}^{-\beta\tau} f_X(\tau \mid c_0)\,\mathrm{d}\tau = \mathrm{e}^{-\beta c_0}. \tag{4.4.3}$$

We can also simplify the reward function $R(s, a)$. If the reward rate $r_2(\tau \mid s, a, s') = r_2(s, a, s')$ is a constant, and the sojourn times for each triple $(s, a, s')$ follows a deterministic distribution with respective parameter $c_0(s, a, s')$, then we can rewrite Eq. (4.3.4) as

$$R(s, a) = r_1(s, a) + \frac{1}{\beta} \sum_{s' \in \mathcal{S}} P(s' \mid s, a) r_2(s, a, s')(1 - \mathrm{e}^{-\beta c_0(s,a,s')}). \tag{4.4.4}$$

◀

**Example 4.4.2 — Inverse Gaussian distribution.** Since we are looking at a nonnegative random variable $X$, the Laplace transform $\mathbf{E}(\mathrm{e}^{-sX})$ for $s \geq 0$ exists. We can view the moment-generating function as a two-sided Laplace transform (Gut, 2009, p. 63). The moment-generating function of a random variable $X$ that follows an inverse Gaussian distribution with parameters $\mu$ and $\lambda$ is

$$M_X(t) = \mathbf{E}(\mathrm{e}^{tX}) = \int_{-\infty}^{\infty} \mathrm{e}^{tx} f_X(x)\,\mathrm{d}x = \exp\left[\frac{\lambda}{\mu}\left(1 - \sqrt{1 - \frac{2\mu^2 t}{\lambda}}\right)\right] \tag{4.4.5}$$

(Tweedie, 1957, Eq. (8b); Chhikara and Folks, 1989, Eq. (3.13), p. 27). We can calculate

$$\int_0^\infty \mathrm{e}^{-\beta\tau} f_X(\tau \mid \mu, \lambda)\,\mathrm{d}\tau \tag{4.4.6}$$

using the moment-generating function for the inverse Gaussian distribution by changing the variable $t$ to $-\beta$ in Eq. (4.4.5) yields

$$\begin{aligned} \int_0^\infty \mathrm{e}^{-\beta\tau} f_X(\tau \mid \mu, \lambda)\,\mathrm{d}\tau &= \int_0^\infty \mathrm{e}^{-\beta\tau} \sqrt{\frac{\lambda}{2\pi\tau^3}} \exp\left[-\frac{\lambda(x-\mu)^2}{2\mu^2\tau}\right] \mathrm{d}\tau \\ &= \exp\left[\frac{\lambda}{\mu}\left(1 - \sqrt{1 + \frac{2\mu^2\beta}{\lambda}}\right)\right]. \end{aligned} \tag{4.4.7}$$

We can also simplify the reward function $R(s, a)$. If the reward rate $r_2(\tau \mid s, a, s') = r_2(s, a, s')$ is a constant, and the sojourn times for each triple $(s, a, s')$ follows an inverse Gaussian distribution $\mathcal{IG}\big(\mu(s,a,s'),\lambda(s,a,s')\big)$ with their respective mean and shape parameters, then we can rewrite Eq. (4.3.4) as

$$R(s,a) = r_1(s,a) + \frac{1}{\beta}\sum_{s'\in\mathcal{S}} P(s' \mid s,a)\, r_2(s,a,s') \left(1 - \exp\left[\frac{\lambda(s,a,s')}{\mu(s,a,s')}\left(1 - \sqrt{1 + \frac{2\mu(s,a,s')^2\beta}{\lambda(s,a,s')}}\right)\right]\right). \tag{4.4.8}$$

◀

**Example 4.4.3 — Truncated Gaussian distribution.** The moment-generating function of a random variable $X$ that follows a truncated Gaussian distribution $\mathcal{N}(\mu,\sigma^2,a,b)$ is

$$\begin{aligned} M_X(t) &= \mathbf{E}(\mathrm{e}^{tX}) \\ &= \int_{-\infty}^{\infty} \mathrm{e}^{tx} f_X(x)\,\mathrm{d}x \\ &= \exp\left(\mu t + \frac{\sigma^2 t^2}{2}\right)\left[\frac{\Phi\Big(\frac{b-\mu}{\sigma} - \sigma t\Big) - \Phi\Big(\frac{a-\mu}{\sigma} - \sigma t\Big)}{\Phi\Big(\frac{b-\mu}{\sigma}\Big) - \Phi\Big(\frac{a-\mu}{\sigma}\Big)}\right] \end{aligned} \tag{4.4.9}$$

Replacing $tx$ with $-\beta\tau$,

$$\int_{-\infty}^{\infty} \mathrm{e}^{-\beta\tau} f_X(\tau \mid \mu,\sigma^2) = \exp\left(\frac{\sigma^2\beta^2}{2} - \mu\beta\right)\left[\frac{\Phi\Big(\frac{b-\mu}{\sigma} + \sigma\beta\Big) - \Phi\Big(\frac{a-\mu}{\sigma} + \sigma\beta\Big)}{\Phi\Big(\frac{b-\mu}{\sigma}\Big) - \Phi\Big(\frac{a-\mu}{\sigma}\Big)}\right] \tag{4.4.10}$$

If we consider the truncated Gaussian distribution $\mathcal{N}(\mu,\sigma^2,0,\infty)$, then

$$\lim_{b\to\infty} \Phi\Big(\frac{b-\mu}{\sigma}\Big) = 1, \tag{4.4.11}$$

and Eq. (4.4.10) becomes

$$\int_{-\infty}^{\infty} \mathrm{e}^{-\beta\tau} f_X(\tau \mid \mu,\sigma^2) = \exp\left(\frac{\sigma^2\beta^2}{2} - \mu\beta\right)\left[\frac{1 - \Phi\Big(-\frac{\mu}{\sigma} + \sigma\beta\Big)}{1 - \Phi\Big(-\frac{\mu}{\sigma}\Big)}\right] \tag{4.4.12}$$

Then we can rewrite Eq. (4.3.4) as

$$R(s,a) = r_1(s,a) + \frac{1}{\beta}\sum_{s'\in\mathcal{S}} P(s' \mid s,a)\, r_2(s,a,s') \left(1 - \exp\left(\frac{\sigma^2\beta^2}{2} - \mu\beta\right)\left[\frac{1 - \Phi\Big(-\frac{\mu}{\sigma} + \sigma\beta\Big)}{1 - \Phi\Big(-\frac{\mu}{\sigma}\Big)}\right]\right). \tag{4.4.13}$$

◀

We assume continuous-time discounting at rate $\beta > 0$. This means that the present value of one reward unit received at time $\tau$ units in the future equals $\mathrm{e}^{-\beta\tau}$ (Puterman, 1994, p. 540). By setting $\mathrm{e}^{-\beta\cdot 1} = \gamma$, where $\gamma$ denotes the discrete-time discount factor, we see that the continuous-time discounting rate of $\beta = -\ln(\gamma)$.

## 4.5 Value Function for SMDP

Recall for discrete-state MDPs, the value of state $s$ evaluated for a given policy $\pi$ is

$$V^\pi(s) = R\big(s, \pi(s)\big) + \sum_{s' \in \mathcal{S}} P\big(s' \mid s, \pi(s)\big) \gamma V^\pi(s') \tag{4.5.1}$$

and Bellman's optimality equation

$$V^*(s) = \max_{a \in \mathcal{A}(s)} \left\{ R(s,a) + \sum_{s' \in \mathcal{S}} P(s' \mid s, a) \gamma V^*(s') \right\}. \tag{4.5.2}$$

We can adapt this for an SMDP by making the following changes: Set $R(s,a)$ to Eq. (4.3.1) and the discount factor as

$$\gamma(s,a,s') = \int_0^\infty \mathrm{e}^{-\beta\tau} F(\mathrm{d}\tau \mid s, a, s').$$

Then, the SMDP version of Eq. (4.5.1) is

$$V^\pi(s) = R\big(s, \pi(s)\big) + \sum_{s' \in \mathcal{S}} P\big(s' \mid s, \pi(s)\big) \int_0^\infty \mathrm{e}^{-\beta\tau} F\big(\mathrm{d}\tau \mid s, \pi(s), s'\big) V^\pi(s') \tag{4.5.3}$$

$$= R\big(s, \pi(s)\big) + \sum_{s' \in \mathcal{S}} \int_0^\infty \mathrm{e}^{-\beta\tau} Q\big(\mathrm{d}\tau, s' \mid s, \pi(s)\big) V^\pi(s') \tag{4.5.4}$$

and Bellman's optimality equation of Eq. (4.5.2) for SMDP is

$$V^*(s) = \max_{a \in \mathcal{A}(s)} \left\{ R(s,a) + \sum_{s' \in \mathcal{S}} P(s' \mid s, a) \int_0^\infty \mathrm{e}^{-\beta\tau} F(\mathrm{d}\tau \mid s, a, s') V^*(s') \right\} \tag{4.5.5}$$

$$= \max_{a \in \mathcal{A}(s)} \left\{ R(s,a) + \sum_{s' \in \mathcal{S}} \int_0^\infty \mathrm{e}^{-\beta\tau} P(s' \mid s, a) F(\mathrm{d}\tau \mid s, a, s') V^*(s') \right\}$$

$$= \max_{a \in \mathcal{A}(s)} \left\{ R(s,a) + \sum_{s' \in \mathcal{S}} \int_0^\infty \mathrm{e}^{-\beta\tau} Q(\mathrm{d}\tau, s' \mid s, a) V^\pi(s') \right\} \tag{4.5.6}$$

where $R(s,a)$ is given by Eq. (4.3.1).

If $\mathcal{S}$ is continuous (non-empty uncountable Borel subset of a complete, separate metric) space then,

$$V^*(s) = \sup_{a \in \mathcal{A}(s)} \left\{ R(s,a) + \int_{s' \in \mathcal{S}} \int_0^\infty \mathrm{e}^{-\beta\tau} Q(\mathrm{d}\tau, \mathrm{d}s' \mid s, a) V^*(s') \right\} \tag{4.5.7}$$

$$= \sup_{a \in \mathcal{A}(s)} \left\{ R(s,a) + \int_{s' \in \mathcal{S}} \int_0^\infty \mathrm{e}^{-\beta\tau} P(\mathrm{d}s' \mid s, a) F(\mathrm{d}\tau \mid s, a, s') V^*(s') \right\} \tag{4.5.8}$$

where

$$R(s,a) = r_1(s,a) + \int_{s' \in \mathcal{S}} \int_0^\infty \left( \int_0^\tau \mathrm{e}^{-\beta t} r_2(t \mid s, a, s')\, \mathrm{d}t \right) Q(\mathrm{d}\tau, \mathrm{d}s' \mid s, a) \tag{4.5.9}$$

$$= r_1(s,a) + \int_{s' \in \mathcal{S}} \int_0^\infty \left( \int_0^\tau \mathrm{e}^{-\beta t} r_2(t \mid s, a, s')\, \mathrm{d}t \right) P(\mathrm{d}s' \mid s, a) F(\mathrm{d}\tau \mid s, a, s'). \tag{4.5.10}$$

**Example 4.5.1 — Two-State SMDP.** Let $\mathcal{S} = \{s_1, s_2\}$, $\mathcal{A}(s_1) = \{a_1, a_2\}$, and $\mathcal{A}(s_2) = \{a_1\}$. We specify the transition probabilities for the embedded MDP as

$$P(\cdot \mid \cdot, a_1) = \begin{array}{c|cc} & s_1 & s_2 \\ \hline s_1 & 0.5 & 0.5 \\ s_2 & 0.1 & 0.9 \end{array} \quad \text{and} \quad P(\cdot \mid \cdot, a_2) = \begin{array}{c|cc} & s_1 & s_2 \\ \hline s_1 & 0 & 1 \\ s_2 & 0 & 0 \end{array}.$$

A graphical representation of the semi-Markov decision process with the transition probabilities labelled is shown in Fig. 4.7.

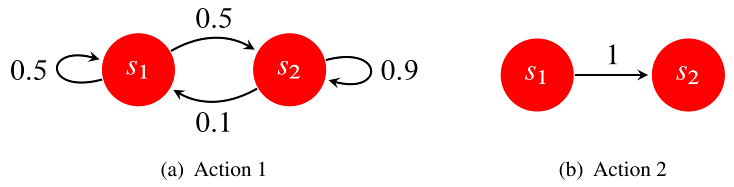


Fig. 4.7 A graphical representation of the semi-Markov decision process with two states and two actions given in Example 4.5.1.

We assume that the sojourn times follow an inverse Gaussian distribution with the mean parameters given by

$$\mu(\cdot, a_1, \cdot) = \begin{array}{c|cc} & s_1 & s_2 \\ \hline s_1 & 2 & 3 \\ s_2 & 4 & 5 \end{array} \quad \text{and} \quad \mu(\cdot, a_2, \cdot) = \begin{array}{c|cc} & s_1 & s_2 \\ \hline s_1 & 6 & 7 \\ s_2 & 8 & 9 \end{array},$$

and the shape parameters given by

$$\lambda(\cdot, a_1, \cdot) = \begin{array}{c|cc} & s_1 & s_2 \\ \hline s_1 & 4 & 9 \\ s_2 & 16 & 25 \end{array} \quad \text{and} \quad \lambda(\cdot, a_2, \cdot) = \begin{array}{c|cc} & s_1 & s_2 \\ \hline s_1 & 36 & 49 \\ s_2 & 64 & 81 \end{array}.$$

The lump sum rewards are given by

$$r_1(\cdot, a_1) = \begin{array}{c|c} s_1 & 0 \\ s_2 & -1 \end{array} \quad \text{and} \quad r_1(\cdot, a_2) = \begin{array}{c|c} s_1 & -1 \\ s_2 & 0 \end{array},$$

and the continuous reward rates are

$$r_2(\cdot, a_1, \cdot) = \begin{array}{c|cc} & s_1 & s_2 \\ \hline s_1 & 5 & -1 \\ s_2 & 1 & 10 \end{array} \quad \text{and} \quad r_2(\cdot, a_2, \cdot) = \begin{array}{c|cc} & s_1 & s_2 \\ \hline s_1 & 6 & 7 \\ s_2 & 9 & 12 \end{array}.$$

We assume a discount rate $\beta = 0.3$, and define

$$m(s, a, s') = \int_0^\infty e^{-\beta\tau} f(\tau \mid s, a, s')\, d\tau,$$

which can be calculated by Eq. (4.4.7). Henceforth, calculations will be carried out to a precision of four decimal places. In matrix form,

$$m(\cdot, a_1, \cdot) = \begin{array}{c|cc} & s_1 & s_2 \\ \hline s_1 & 0.5887 & 0.4517 \\ s_2 & 0.3466 & 0.2659 \end{array} \quad \text{and} \quad m(\cdot, a_2, \cdot) = \begin{array}{c|cc} & s_1 & s_2 \\ \hline s_1 & 0.2040 & 0.1566 \\ s_2 & 0.1201 & 0.0922 \end{array}.$$

We now evaluate $R(s, a)$ using Eq. (4.3.4). For example,

$$\begin{aligned} R(s_1, a_1) &= r_1(s_1, a_1) + \frac{1}{\beta}\big[P(s_1 \mid s_1, a_1) r_2(s_1, a_1, s_1)\big(1 - m(s_1, a_1, s_1)\big) \\ &\qquad + P(s_2 \mid s_1, a_1) r_2(s_1, a_1, s_2)\big(1 - m(s_1, a_1, s_2)\big)\big] \\ &= 0 + \frac{1}{0.3}[0.5(5)(1 - 0.5887) + 0.5(-1)(1 - 0.4517)] \\ &= 2.5136 \end{aligned}$$

The rest of the calculations are similar, and so in matrix form, we have

$$R(\cdot, a_1) = \begin{array}{c} s_1 \\ s_2 \end{array} \begin{bmatrix} 2.5136 \\ 21.2402 \end{bmatrix} \qquad \text{and} \qquad R(\cdot, a_2) = \begin{array}{c} s_1 \\ s_2 \end{array} \begin{bmatrix} 18.6805 \\ 0 \end{bmatrix}.$$

The Bellman optimality equations are

$$\begin{aligned} V^*(s_1) &= \max \begin{cases} R(s_1, a_1) + P(s_1 \mid s_1, a_1) m(s_1, a_1, s_1) V^*(s_1) & (a = a_1) \\ \qquad + P(s_2 \mid s_1, a_1) m(s_1, a_1, s_2) V^*(s_2), \\ R(s_1, a_2) + P(s_1 \mid s_1, a_2) m(s_1, a_2, s_1) V^*(s_1) & (a = a_2) \\ \qquad + P(s_2 \mid s_1, a_2) m(s_1, a_2, s_2) V^*(s_2) \end{cases} \\ &= \max \begin{cases} 2.5136 + 0.5(0.5887) V^*(s_1) & (a = a_1) \\ \qquad + 0.5(0.4517) V^*(s_2) \\ 18.6805 + 0(0.2040) V^*(s_1) & (a = a_2) \\ \qquad + 1(0.1566) V^*(s_2) \end{cases} \\ &= \max \begin{cases} 2.5136 + 0.2944 V^*(s_1) + 0.2259 V^*(s_2) & (a = a_1) \\ 18.6805 + 0.1566 V^*(s_2) & (a = a_2) \end{cases} \end{aligned} \tag{4.5.11}$$

and

$$\begin{aligned} V^*(s_2) &= R(s_2, a_1) + P(s_1 \mid s_2, a_1) m(s_2, a_1, s_1) V^*(s_1) \\ &\qquad + P(s_2 \mid s_2, a_1) m(s_2, a_1, s_2) V^*(s_2) \\ &= 21.2402 + 0.1(0.3466) V^*(s_1) + 0.9(0.2659) V^*(s_2) \\ &= 21.2402 + 0.0347 V^*(s_1) + 0.2393 V^*(s_2). \end{aligned} \tag{4.5.12}$$

We let $\pi_1$ denote the deterministic policy that uses action $a_1$ in $s_1$, and $\pi_2$ that uses action $a_2$ in $s_1$. For $\pi_1$, we can find $V^{\pi_1}$ using the following fixed-point iteration

$$\begin{aligned} s_1 &\leftarrow 0 \\ s_2 &\leftarrow 0 \\ V^{\pi_1}(s_1) &\leftarrow 2.5136 + 0.2944 V^*(s_1) + 0.2259 V^*(s_2) \\ V^{\pi_1}(s_2) &\leftarrow 21.2402 + 0.0347 V^*(s_1) + 0.2393 V^*(s_2), \end{aligned}$$

which yields

$$V^{\pi_1}(\cdot) = \begin{array}{c} s_1 \\ s_2 \end{array} \begin{bmatrix} 12.6869 \\ 28.5006 \end{bmatrix}.$$

For $\pi_2$,

$$\begin{aligned} s_1 &\leftarrow 0 \\ s_2 &\leftarrow 0 \\ V^{\pi_2}(s_1) &\leftarrow 18.6805 + 0.1566 V^*(s_2) \\ V^{\pi_2}(s_2) &\leftarrow 21.2402 + 0.0347 V^*(s_1) + 0.2393 V^*(s_2), \end{aligned}$$

which yields

$$V^{\pi_2}(\cdot) = \begin{array}{c} s_1 \\ s_2 \end{array} \begin{bmatrix} 23.2189 \\ 28.9811 \end{bmatrix}.$$

By substituting the values back into the Bellman optimality equations, Eq. (4.5.11) and Eq. (4.5.12), we see that $\pi_2$ is the optimal policy, and therefore,

$$V^*(\cdot) = \begin{array}{c} s_1 \\ s_2 \end{array} \begin{bmatrix} 23.2189 \\ 28.9811 \end{bmatrix}.$$

◀

## 4.6 Optimal Stopping Problems Revisited

We discussed the elevator problem in Examples 2.7.1 (p. 16) and 3.9.1 (p. 36), described it is as an MDP and POMDP, and found an optimal policy for it.

**Example 4.6.1 — Elevator Problem (SMDP version).** Suppose that when an agent arrives at the lobby (on floor 1), $N$ minutes remain before its meeting begins. If the elevator arrives before the meeting starts, time $\tau < N$, then it earns a lump reward $r\mathrm{e}^{-\beta\tau}$, where $\beta$ is the agent's discounting rate. The agent can see on which floor, $s$, the elevator is on currently by the floor indicator above the elevator door. It also knows the conditional probability $Q(\tau, s' \mid s)$, which represents the probability that the elevator will descend to floor $s'$ in $\tau$ amount of time given that the elevator is currently on floor $s$. The action of taking the stairs incurs a heavy cost of $C$ units. Suppose that when the agent arrives and pushes the elevator button, the descending elevator is on floor $S$. What is the optimal policy that the agent should follow?

**Solution** *Using dynamic programming*. We can model this problem using the SMDP framework. The state space consists of the floor numbers

$$\mathcal{S} = \{1, 2, \ldots, S\},$$

and the number of states is

$$|\mathcal{S}| = S.$$

The action space consists of two actions

$$\mathcal{A} = \{\text{wait}, \text{stairs}\}.$$

The sojourn time-state transition probability matrix is given by

$$Q(\tau, s' \mid s, a) = Q(\tau, s' \mid s) = \mathbf{P}(T_{n+1} - T_n \leq \tau, S_{n+1} = s' \mid S_n = s).$$

There is only one decision to be made which occurs as soon as the agent arrives in the lobby and sees what floor the elevator is on: either it will wait or it will immediately take the stairs. Thus, there is only one decision epoch, and hence it is a finite horizon problem. If the agent decides to take the stairs, there is no discounting of the lump reward $r$ it receives; it only has to pay the heavy cost penalty $C$. Thus, the agent's reward for choosing to take the stairs is

$$r - C. \tag{4.6.1}$$

On the other hand, if the agent decides to wait, there are two possibilities:

(a) the elevator arrives $\tau < N$ before the meeting, which means that the discounted lump reward the agent receives should be $r\mathrm{e}^{-\beta\tau}$. However, we do not know $\tau$, but $\tau$ follows the sojourn-time state transition probability $Q$. Thus, the expected discounted reward if the elevator arrives before the meeting is

$$\int_0^N r\mathrm{e}^{-\beta\tau} Q(\mathrm{d}\tau, 1 \mid s) = r\int_0^N \mathrm{e}^{-\beta\tau} Q(\mathrm{d}\tau, 1 \mid s). \tag{4.6.2}$$

(b) the elevator arrives $\tau \geq N$ after the meeting, the agent will have to take the stairs, which means the reward is $r\mathrm{e}^{-\beta N} - C$; this occurs with probability $\int_N^\infty Q(\mathrm{d}\tau, 1 \mid s)$. Thus, the expected discounted reward if the elevator arrives after the meeting starts is

$$(r\mathrm{e}^{-\beta N} - C)\int_N^\infty Q(\mathrm{d}\tau, 1 \mid s). \tag{4.6.3}$$

Table 4.1 Optimal policy for the numerical example of the elevator problem

| Floor, $s$ | $V^*(s)$ | $\pi^*(s)$ |
|---|---|---|
| 5 | 15 | Stairs |
| 4 | 15 | Stairs |
| 3 | 16.9691 | Wait |
| 2 | 22.9365 | Wait |
| 1 | 24.4998 | Wait |

Combining Eq. (4.6.1)–(4.6.3), the reward function is

$$R(s,a) = \begin{cases} r - C, & (a = \text{stairs}); \\ \underbrace{r \int_0^N \mathrm{e}^{-\beta\tau} Q(\mathrm{d}\tau, 1 \mid s)}_{\text{elevator arrives within } N \text{ minutes}} \\ \quad \underbrace{+ \, (r\mathrm{e}^{-\beta N} - C) \int_N^\infty Q(\mathrm{d}\tau, 1 \mid s)}_{\text{elevator arrives after } N \text{ minutes}}, & (a = \text{wait}), \end{cases}$$

for all floors $s \in \{1, 2, \dots, S\}$. This completes our description of the elevator problem as an SMDP.

Since this is an SMDP, we can use a dynamic programming approach to this problem. This is quite simple since the problem has only one decision epoch, which means that the optimal value function reduces to

$$V^*(s) = \max_{a \in \mathcal{A}} R(s,a).$$

We can now determine the optimal policy $\pi^*(s)$ the agent should follow. Once the agent enters the lobby and sees that the elevator is on floor $s$, it should take the stairs if

$$r - C > r \int_0^N \mathrm{e}^{-\beta\tau} Q(\mathrm{d}\tau, 1 \mid s) + (r\mathrm{e}^{-\beta N} - C) \int_N^\infty Q(\mathrm{d}\tau, 1 \mid s).$$

We give a numerical example by setting $N = 10$, $S = 5$, $r = 25$, $C = 10$ (these values were the same from Example 2.7.1), $\beta = 0.01$ and the sojourn-time state transition probability as $Q(\tau, 1 \mid s) = F(\tau \mid \mu = s, \lambda = s^2)$ where $F$ is the inverse Gaussian cumulative distribution function defined by Eq. (4.2.5). The calculations are provided in Table 4.1.

◀

Contrasting with the MDP version in Example 2.7.1, the number of states is smaller in the SMDP version: $S < S \times N$.

## 4.7 Q-Learning for SMDP

Recall from Sec. 2.6, for a discrete state space $\mathcal{S}$ and a finite action space $\mathcal{A}$, Watkins (1989) defines the state-action value function $\mathcal{Q}$ corresponding to the policy $\pi$ as

$$\mathcal{Q}^\pi(s,a) = R(s,a) + \gamma \sum_{s' \in \mathcal{S}} P(s' \mid s,a) V^\pi(s') \tag{4.7.1}$$

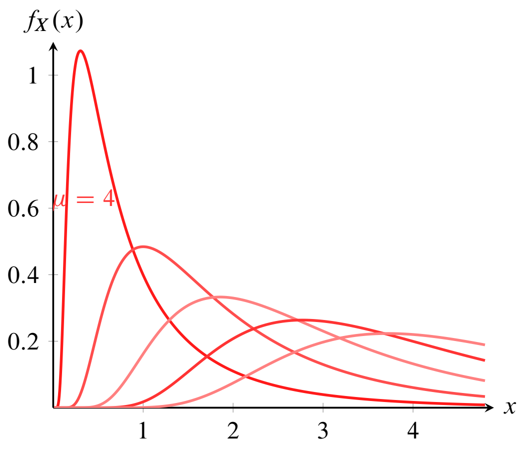


Fig. 4.8 Probability density functions for the numerical example of the elevator problem.

or equivalently,

$$Q^\pi(s,a) = R(s,a) + \gamma \sum_{s' \in \mathcal{S}} P(s' \mid s,a) Q^\pi\big(s', \pi(s')\big). \tag{4.7.2}$$

The corresponding optimal state-action value function $Q^*$ is

$$Q^*(s,a) = R(s,a) + \gamma \sum_{s' \in \mathcal{S}} P(s' \mid s,a) \max_{a' \in \mathcal{A}} Q^*(s',a'). \tag{4.7.3}$$

An online agent can approximate the state-action value function by sampling the reward $R(s,a,s')$ during its transition from state $s$ to $s'$ under action $a$ by

$$Q_{k+1}(s,a) = \underbrace{Q_k(s,a)}_{\text{old value}} + \alpha_k \left[ \overbrace{\underbrace{R(s,a,s') + \gamma \max_{a' \in \mathcal{A}} Q_k(s',a')}_{\text{temporal difference target}} - \underbrace{Q_k(s,a)}_{\text{old value}}}^{\text{temporal difference}} \right], \tag{4.7.4}$$

where $k$ is the number of times the agent has visited state $s$ and performed $a$, and $\alpha_k$ is the learning rate.

Bradtke and Duff (1995) defines the optimal state-action value function $Q^*$ for an SMDP as

$$Q^*(s,a) = R(s,a) + \sum_{s' \in \mathcal{S}} \int_0^\infty e^{-\beta\tau} Q(d\tau, ds' \mid s,a) \max_{a' \in \mathcal{A}} Q^*(s',a'), \tag{4.7.5}$$

or equivalently,

$$Q^*(s,a) = R(s,a) + \sum_{s' \in \mathcal{S}} P(s' \mid s,a) \int_0^\infty e^{-\beta\tau} F(d\tau \mid s,a,s') \max_{a' \in \mathcal{A}} Q^*(s',a'), \tag{4.7.6}$$

where $R(s,a)$ is defined by Eq. (4.3.2). This leads to the $Q$-Learning rule for SMDPs

$$Q_{k+1}(s,a) = Q_k(s,a) + \alpha_k \left[ \frac{1 - e^{-\beta\tau}}{\beta} r_2(s,a,s') + e^{-\beta\tau} \max_{a' \in \mathcal{A}} Q_k(s',a') - Q_k(s,a) \right] \tag{4.7.7}$$

This leads to the following $\mathcal{Q}$-Learning rule for SMDPs

$$\mathcal{Q}_{k+1}(s,a) = \mathcal{Q}_k(s,a) + \alpha_k \left[ R(s,a,s') + \mathrm{e}^{-\beta\tau} \max_{a' \in \mathcal{A}} \mathcal{Q}_k(s',a') - \mathcal{Q}_k(s,a) \right] \tag{4.7.8}$$

where

$$R(s,a,s') = R_1(s,a) + \frac{1-\mathrm{e}^{-\beta\tau}}{\beta} R_2(s,a,s'),$$

$R_1(s,a)$ is the lump sum reward, and $R_2(s,a,s')$ is the total reward accumulated from state $s$ to $s'$. The $\mathcal{Q}$-Learning algorithm of an SMDP is given in Algorithm 4.7.

**Algorithm 14** $\mathcal{Q}$-Learning for SMDP, Bradtke and Duff (1995)

```
1: Select an arbitrary $\mathcal{Q}$-function, arbitrary state $s \in \mathcal{S}$, $\alpha \in (0,1)$, $\beta \in (0,1)$.
2: **repeat**
3:     Take an arbitrary action $a \in \mathcal{A}(s)$
4:     Observe immediate reward $r$, sojourn time $\tau$ and next state $s'$
5:     $\mathcal{Q}(s,a) \leftarrow \mathcal{Q}(s,a) + \alpha \left[ \frac{1-\mathrm{e}^{-\beta\tau}}{\beta} r + \mathrm{e}^{-\beta\tau} \max_{a' \in \mathcal{A}(s')} \mathcal{Q}(s',a') - \mathcal{Q}(s,a) \right]$
6:     $s \leftarrow s'$
7: **until** $s$ is terminal
```

## 4.8 Bibliographic Remarks

Semi-Markov processes, first examined by Lévy (1956) and developed as regenerative stochastic processes by (Smith, 1955), laid the groundwork for subsequent studies. The decision process variant of semi-Markov processes emerged independently through the work of Howard (1963) and Jewell (1963a). In this thesis, our attention is primarily directed towards the discounted reward version of SMDPs. However, it is worth noting that average reward SMDPs, as discussed in Ross (1970b), play a vital role in the analysis of episodic problems.

Another approach that is used in reinforcement learning to extend MDPs to SMDPs is options. Options in reinforcement learning refer to temporally extended actions or sequences of actions that an agent can choose to follow. They allow the agent to plan over longer time horizons by abstracting away lower-level decision-making, thereby facilitating learning and planning in complex environments (Precup, 2000).

# 5

# The Partially Observable Semi-Markov Decision Process (POSMDP) Model

In this chapter, we define a partially observable semi-Markov decision process (POSMDP) in a Borel space, which is an extension of a semi-Markov MDP (SMDP) and a partially observable MDP (POMDP). Once we define the model, one of the significant contributions of this work is presented: we introduce a novel algorithm called ChronosPerseus to solve POSMDPs by combining point-based value iteration and importance sampling.

Two well-known extensions to MDPs are semi-Markov MDP (SMDP) (Howard, 1963; Jewell, 1963a,b; de Cani, 1964) and partially observable MDP (POMDP) (Aoki, 1965; Åström, 1965), which we examined earlier in Chap. 4 and Chap. 3, respectively. The former allows transitions to have sojourn time distributions, while the latter allows for partially observable states (incomplete state information). In reinforcement learning, SMDPs are primarily used with options (Sutton *et al.*, 1999; Precup, 2000), where the underlying MDP of the micro-actions defines the transitions of macro-actions. The SMDP, in that case, represents a temporal compression of the underlying MDP. In contrast, POMDPs are computationally expensive (Papadimitriou and Tsitsiklis, 1987; Madani *et al.*, 1999), as the solvers usually construct a policy over a belief state space that must cover all the combinations of states.

A POSMDP can be considered a generalization of an SMDP or a POMDP. In the case of generalizing the SMDP, partial observability is introduced so that the agent cannot know with certainty in which state it is. Generalizing the POMDP involves introducing random sojourn time between decision epochs. By choosing either the SMDP or POMDP to generalize, we can arrive at the POSMDP; see Fig. 5.1.

$$\begin{tikzcd}
& \text{MDP} \arrow[dl, "\text{continuous time +}"'] \arrow[dr, "\text{+ partial observability}"] & \\
\text{SMDP} \arrow[dr, "\text{partial observability +}"'] & & \text{POMDP} \arrow[dl, "\text{+ continuous time}"] \\
& \text{POSMDP} &
\end{tikzcd}$$

Fig. 5.1 The relationships between MDP, SMDP, POMDP, and POSMDP.

POMDPs are not widely used in practice due to their computational complexity (Papadimitriou and Tsitsiklis, 1987; Madani *et al.*, 1999; Sutton and Barto, 2018). Nevertheless, progress has been made in the development of approximate solutions methods, such as point-based value iteration by Pineau *et al.* (2003) and the subsequent introduction of Perseus by Spaan and Vlassis (2005).

This progress in POMDP solution methods opens new opportunities for addressing more complex problems. Cognitive agents, as well as many scheduling and maintenance problems, could naturally benefit from the ability to solve problems that mix both partial observability and rich temporal transitions. The partially observable semi-Markov decision process (POSMDP) extends POMDP by incorporating stochastic sojourn time between transitions. As demonstrated later, this sojourn time can further be used in the belief state update. White (1976) initially proposed the finite-horizon discrete-time POSMDP model, with his algorithm for computing the optimal cost and optimal policy based on POMDP procedures by Sondik (1971). Half a decade later, Wakuta defined the infinite-horizon continuous-time POSMDP model with the average-cost criterion (Wakuta, 1981) and the discounted-cost criterion (Wakuta, 1982). The definition and notation of POSMDP that we present in Section 5.1 is influenced by Wakuta (1982), Hernández-Lerma (1989), and Yu (2006).

While Zhang and Revie (2017) developed a POSMDP solver based on PERSEUS, their algorithm relied on slow numerical integration techniques to handle the value function, necessitating integration of the sojourn-time distribution for every transition at every iteration. In contrast, we introduce importance sampling to use the collected belief and sojourn-time samples more efficiently. Using this approach reduces computational time from hours to seconds. This leads us to one of the significant contributions of this thesis, which is our importance sampling point-based POSMDP solver called CHRONOSPERSEUS that we will present in Sec. 5.11 with its computational complexity. This solver enables the incorporation of temporal properties into POMDPs without the increased complexity arising from expanding state space dimensions to account for temporal information, allowing us to apply this to more realistic, complex problems.

The remainder of this chapter is organized as follows. First, in Sec. 5.1 we introduce formally the framework, the notation, and the dynamics for the POSMDP model. Then, we discuss how we handle concepts of time in Sec. 5.2, partial observability in Sec. 5.3. In Sec. 5.4, we highlight the difference of the history of the process versus the observable history in POSMDP leading to the construction of the belief state in Sec. 5.5. In Sec. 5.6, we discuss the reward and the assumptions required to ensure the reward function is well-defined. For the value function of a POSMDP in Sec. 5.7, we show that depending on the approach taken, either extending the POMDP model with the addition of continuous time or extending the SMDP model with the addition of partial observability, that we arrive at two versions of the POSMDP value function. In Sec. 5.8, we show how we can rewrite the POSMDP as a POMDP, but this does not require the belief update to know the sojourn time. We believe that the sojourn time is important information that can be incorporated into the belief, which in turn affects the agent's optimal decision. Instead, we show that the POSMDP can be rewritten as a continuous belief state SMDP, where the belief update includes the sojourn time. We cover background mathematical concepts such as the basics of Monte Carlo integration in Sec. 5.9 so that we can use the importance sampling technique. Then, we present how to rewrite the POSMDP to SMDP in Sec. 5.10. Finally, we describe in Sec. 5.11 the main contribution of this chapter—the CHRONOSPERSEUS algorithm. In Sec. 5.12 , we conclude the chapter with some solved POSMDP applications using CHRONOSPERSEUS: an optimal stopping problem involving waiting for a bus, and maintenance of water filters in the real world based on parameters from Zhang and Revie (2017).

## 5.1 The Framework

**Definition 5.1.1 — POSMDP.** A partially observable semi-Markov decision process (POSMDP) is an 11-tuple

$$\langle \mathcal{S}, \mathcal{A}, \mathcal{K}, \mathcal{O}, Q, G, G_0, R, \xi_0, \beta, N \rangle \tag{5.1.1}$$

where

| | |
|---|---|
| $\mathcal{S}$ | is the Borel *state space*, and the elements of $\mathcal{S}$ are called *states* |
| $\mathcal{A}$ | is the Borel *action space*, and the elements of $\mathcal{A}$ are called *actions*. To each $s \in \mathcal{S}$, we associate a nonempty Borel-measurable subset $\mathcal{A}(s) \subseteq \mathcal{A}$, whose elements are the *admissible actions* for the agent when the process is in state $s$ |
| $\mathcal{K}$ | is the set of admissible state-action pairs, and it is assumed to be a Borel subset $\mathcal{K} \subseteq \mathcal{S} \times \mathcal{A}$. In other words, $\mathcal{K} = \{(s,a) \mid s \in \mathcal{S}, a \in \mathcal{A}(s)\}$. |
| $\mathcal{O}$ | is the Borel *observation space*, and the elements of $\mathcal{O}$ are called observations |
| $Q(\mathrm{d}\tau, \mathrm{d}s' \mid s, a)$ | denotes the sojourn time-state (Borel-measurable) stochastic kernel on $\mathbb{R}_{>0} \times \mathcal{S}$ given $\mathcal{S} \times \mathcal{A}$ |
| $G(\mathrm{d}o \mid a, s')$ | is the observation (Borel-measurable) stochastic kernel on $\mathcal{O}$ given $\mathcal{A} \times \mathcal{S}$ |
| $G_0(\mathrm{d}o \mid s')$ | denotes the initial observation (Borel-measurable) stochastic kernel on $\mathcal{O}$ given $\mathcal{S}$ |
| $R(s,a)$ | is the per-stage (bounded Borel-measurable) reward function given $\mathcal{K}$ |
| $\xi_0$ | is the (*a priori*) initial (belief) state distribution ($\xi_0 \in \mathbf{P}(\mathcal{S})$) |
| $\beta$ | is the discounting rate where $\beta \in [0,1]$. |
| $N$ | is the planning horizon. It could be finite, or $N = \infty$ if $\gamma < 1$. |

Given the model in Definition 5.1.1, the dynamics of the POSMDP proceed according to Algorithm 15. This involves at each decision epoch $n$ choosing an action $a_n$, accruing reward $R(s_n, a_n)$ as the state changes from $s_n$ to $s_{n+1}$ in $\tau_n$ amount of time, and partially observing $s_{n+1}$ as $o_{n+1}$. The agent uses all the information available up to decision epoch $n$, namely the observable history $\tilde{h}_n$, to choose action $a_n = \pi_n(\tilde{h}_n)$ using policy $\pi_n$. With the dynamics of the POSMDP specified by Algorithm 15, we denote the sequence of policies that the agent uses from decision epoch 0 to $n-1$ as $\pi = (\pi_0, \pi_1, \ldots, \pi_{n-1})$. The set of all admissible policies is denoted $\Pi$.

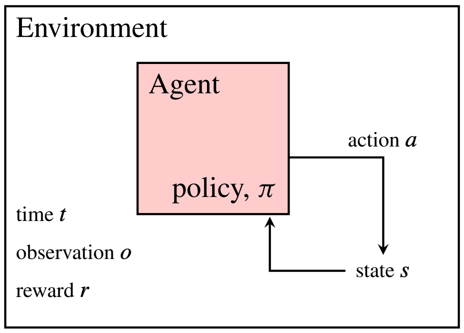


Fig. 5.2 A POSMDP agent interacting with its environment.

**Algorithm 15** Dynamics of POSMDP

```
In the beginning $n = 0$, the state $s_0$ is simulated from an initial (belief) state distribution $\xi_0$.
For each decision epoch $n = 1, 2, \ldots, N$:
    (a) Based on the observable history
            $h_0 = (\xi_0)$
            $h_n = (\xi_0, a_1, t_1, o_1, \ldots, a_n, t_n, o_n)$,
        the agent performs action
            $a_n = \pi_n(h_n) \in \mathcal{A} \qquad n = 1, 2, \ldots, N.$
        Here $\pi_n$ denotes a policy that the agent uses at decision epoch $n$.
    (b) The agent obtains a reward $R(s_n, a_n)$ for choosing action $a_n$ at decision epoch $n$.
    (c) The state evolves randomly with sojourn time-state transition probability
            $Q(\tau, s' \mid s, a) = \mathbf{P}(T_{n+1} - T_n \le \tau, S_{n+1} = s' \mid S_n = s, A_n = a)$   (5.1.2)
        to the next state $s_{n+1}$ that decision epoch $n + 1$.
    (d) The agent records a noisy observation $O_n \in \mathcal{O}$ of the state $S_{n+1}$ according to
            $G(o \mid a, s') = \mathbf{P}(O_n = o \mid A_n = a, S_{n+1} = s')$.
    (e) The agent updates its history as
            $h_{n+1} = (h_n, a_{n+1}, t_{n+1}, s_{n+1})$.
        If $n < N$, then set $n$ to $n + 1$, and go back to step (a).
        If $n = N$, then the agent receives the last reward and the process terminates.
```

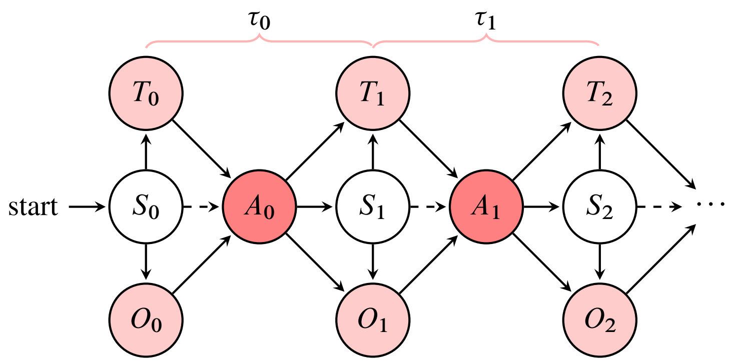


Fig. 5.3 Graphical model of POSMDP. The states $S_i$ are hidden from the agent, while the observations $O_i$ and times $T_i$ are observable. The sojourn times are $\tau_i$ and the agent's actions are $A_i$. The solid arrows $\rightarrow$ represent the direct influence from one element to another, while the dashed arrows $\dashrightarrow$ represent indirect influence. For example, action $A_0$ is directly influenced by time $T_0$ and observation $O_0$ because this is the information that is available to the agent, while state $S_0$ is indirectly influencing action $A_0$ since the agent has to infer in which state it is in currently from time $T_0$ and observation $O_0$.

## 5.2 Time

Unlike POMDPs where the sojourn times are constant, the sojourn times are random in a POSMDP; see Fig. 5.4.

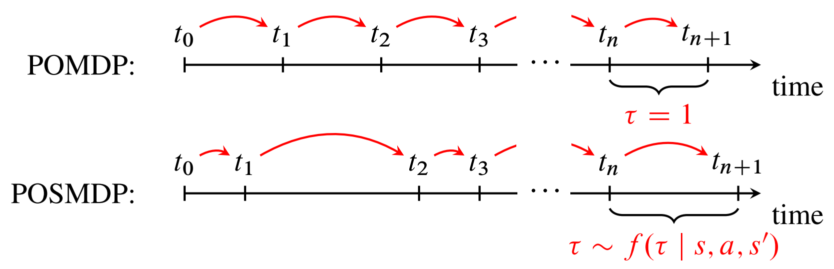

POMDP:
$t_0$ $t_1$ $t_2$ $t_3$ $t_n$ $t_{n+1}$
time
$\tau = 1$
POSMDP:
$t_0$ $t_1$ $t_2$ $t_3$ $t_n$ $t_{n+1}$
time
$\tau \sim f(\tau \mid s, a, s')$

Fig. 5.4 The difference between POMDPs and POSMDPs for sojourn times. The sojourn time for an POMDP is constant whereas the sojourn time for an POSMDP is a random amount of time that follows some probability distribution.

The time of the $n^{\text{th}}$ decision epoch is denoted by the random variable

$$T_n = \begin{cases} 0 & n = 0 \\ T_{n-1} + \tau_{n-1} & n = 1, 2, 3, \dots, \end{cases} \tag{5.2.1}$$

in which the observed time of $T_n$ is denoted by $t_n$. The sojourn time from state $s$ to state $s'$ is a nonnegative real-valued random variable $\tau_n = T_{n+1} - T_n$ that follows

$$F(\tau \mid s, a, s') = \int_{s' \in \mathcal{S}} Q(\tau, \mathrm{d}s' \mid s, a). \tag{5.2.2}$$

From Eq. (5.2.2), we assume implicitly that a sojourn time $\tau_n$ does not affect the next sojourn time $\tau_{n+1}$.

## 5.3 Partial observability

For partial observability, we following the same structure that we introduced in POMDPs in Sec. 3.1. We do not change the notion of partial observability with the introduction of random sojourn time between decision epochs. In Definition 5.1.1, the observation is independent of the previous state. If the observation depends on more information, such as $(s, a, \tau, s')$, then we could define $G(o \mid s, a, \tau, s')$ as

$$G(o \mid a, s') = \sum_{s \in \mathcal{S}} Q(\tau, s' \mid s, a) G(o \mid s, a, \tau, s'). \tag{5.3.1}$$

## 5.4 History

While in a POSMDP, the history of the model's dynamics is

$$\text{POSMDP:} \qquad \begin{aligned} h_0 &= (\xi_0, s_0) \\ h_n &= (\xi_0, s_0, a_0, o_1, \dots, s_{n-1}, a_{n-1}, o_n) \\ h_{n+1} &= h_n \cup (s_n, a_n, o_{n+1}), \end{aligned}$$

it is not entirely observable by the agent; the state in which the agent is in not known with certainty. Instead, the agent sees the *observable history*, $\tilde{h}_n$, which is

$$\text{POMDP:} \qquad \begin{aligned} \tilde{h}_0 &= (\xi_0) \\ \tilde{h}_n &= (\xi_0, a_0, o_1, a_1, o_2, \dots, a_{n-1}, o_n) \\ \tilde{h}_{n+1} &= \tilde{h}_n \cup (a_n, o_{n+1}). \end{aligned}$$

A realization of this process is

$$(s_0, o_0, a_0, \tau_0, s_1, o_1, a_1, \tau_1, s_2, o_2, a_2, \tau_2, \dots) \in (\mathcal{S} \times \mathcal{O} \times \mathcal{A} \times \mathbb{R}_{>0})^{\infty} \triangleq \Omega.$$

However, we are not able to rely on the unobservable states $s_0$, $s_1$, $s_2$, ..., so we use the concept of observable histories. The space of *observable histories* until the $n^{\text{th}}$ decision epoch is defined by

$$\mathcal{H}_n = \begin{cases} \mathbf{P}(\mathcal{S}) \times \mathcal{O}, & n = 0; \\ \mathcal{H}_{n-1} \times \mathcal{A} \times \mathbb{R}_{>0} \times \mathcal{O}, & n = 1, 2, 3, \dots. \end{cases} \tag{5.4.1}$$

Thus, the element $h_0 = (\xi_0, o_0) \in \mathcal{H}_0$ is an initial observed history, and an element $h_n \in \mathcal{H}_n$ is called an *observed history* up to $n$, and it is denoted by

$$h_n = (\underbrace{\xi_0, o_0, a_0, \tau_0, o_1, a_1, \tau_1, \dots, o_{n-1}, \tau_{n-1}}_{h_{n-1}}, a_{n-1}, \tau_n, o_n) \tag{5.4.2}$$

$$= (h_{n-1}, a_{n-1}, \tau_n, o_n). \tag{5.4.3}$$

## 5.5 Belief state

A state variable is the minimally dimensioned function of observable history that is necessary and sufficient to compute the decision function, the transition function, and the contribution function (Powell, 2011, p. 179). The agent can only partially observe the state $s \in \mathcal{S}$ by inferring from its past observations. Thus, in order for an agent to choose optimally its actions in a partially observable environment, memory of its past observations is required.

A naïve approach would allow the agent to remember its sequence of sojourn times, observations, and actions. However, this sequence can grow unbounded over time, which is not practical with finite memory. Instead, this information is summarized using sufficient statistics (Stratonovich, 1960; Aoki, 1965; Åström, 1965; Dynkin, 1965; Aoki, 1967; Åström, 1969; Sawaragi and Yoshikawa, 1970). A statistic is said to be *sufficient* when no other statistic calculated from the same sample provides any additional information to the parameter value being estimated (Fisher, 1922, p. 310). With these sufficient statistics that summarize the observable history, we can construct a probability distribution of where the agent is in the state space; we call this probability distribution over the state space a belief state. We follow a similar structure that for the belief state that was introduced with POMDPs in Sec. 3.4.

**Definition 5.5.1 — Belief state.** A belief state $\xi$ is a probability distribution over the state space $\mathcal{S}$.

If the state space $\mathcal{S} = \{s_1, s_2, \dots, s_{|\mathcal{S}|}\}$ is a finite set, then the belief state $\xi$ is defined by

$$\xi = \begin{bmatrix} \xi(s_1) \\ \xi(s_2) \\ \vdots \\ \xi(s_{|\mathcal{S}|}) \end{bmatrix} \triangleq \begin{bmatrix} \mathbf{P}(S = s_1) \\ \mathbf{P}(S = s_2) \\ \vdots \\ \mathbf{P}(S = s_{|\mathcal{S}|}) \end{bmatrix}.$$

To ensure that $\xi$ is a probability distribution,

$$\xi(s_i) \geq 0 \qquad \forall s_i \in \mathcal{S}, \qquad \text{and} \qquad \sum_{i=1}^{|\mathcal{S}|} \xi(s_i) = 1.$$

This probability distribution encodes the agent's subjective probability of its location in the environment's state space from its history.

$$\xi_n(s_i) = \mathbf{P}(S_n = s_i \mid \xi_{n-1}, a_{n-1}, \tau_n, o_n) \tag{5.5.1}$$

We can update the belief state in two ways: with or without observing the sojourn time $\tau$.

## 5.5.1 The Belief Update without Sojourn Time

We consider the case where the sojourn time $\tau$ is not observed or not part of the information used to update the belief state. By doing this, we will show that the belief update remains the same as the belief update for a POMDP (Eq. (3.5.1)). Recall from Sec. 3.4, the belief update for POMDP is

$$\xi(s' \mid a, o) = \frac{G(o \mid a, s') \sum_{s \in \mathcal{S}} \xi(s) P(s' \mid s, a)}{P(o \mid \xi, a)} \qquad \forall s' \in \mathcal{S}, \tag{3.5.1}$$

where the denominator

$$P(o \mid \xi, a) = \sum_{s' \in \mathcal{S}} G(o \mid a, s') \sum_{s \in \mathcal{S}} P(s' \mid s, a) \xi(s) \tag{3.5.2}$$

is a normalization factor.

By replacing the state transition probability distribution $P$ with the sojourn-time state transition probability distribution $Q$, and if the transition time $\tau$ is not observed/utilized, the belief update for a POSMDP is given by

$$\xi(s' \mid a, o) = \frac{G(o \mid a, s') \sum_{s \in \mathcal{S}} \xi(s) \int_0^\infty Q(\mathrm{d}\tau, s' \mid s, a)}{P(o \mid \xi, a)} \tag{5.5.2}$$

$$= \frac{G(o \mid a, s') \sum_{s \in \mathcal{S}} \xi(s) P(s' \mid s, a) \overbrace{\int_0^\infty F(\mathrm{d}\tau \mid s, a, s')}^{1}}{P(o \mid \xi, a)} \tag{5.5.3}$$

$$= \frac{G(o \mid a, s') \sum_{s \in \mathcal{S}} P(s' \mid s, a)}{P(o \mid \xi, a)} \tag{5.5.4}$$

where the denominator

$$P(o \mid \xi, a) = \sum_{s' \in \mathcal{S}} G(o \mid a, s') \sum_{s \in \mathcal{S}} \xi(s) \int_0^\infty Q(\mathrm{d}\tau, s' \mid s, a) \tag{5.5.5}$$

$$= \sum_{s' \in \mathcal{S}} G(o \mid a, s') \sum_{s \in \mathcal{S}} \xi(s) P(s' \mid s, a) \underbrace{\int_0^\infty F(\mathrm{d}\tau \mid s, a, s')}_{1} \tag{5.5.6}$$

$$= \sum_{s' \in \mathcal{S}} G(o \mid a, s') \sum_{s \in \mathcal{S}} \xi(s) P(s' \mid s, a) \tag{5.5.7}$$

is a normalization factor. Thus, the belief update (including the normalization factor) would remain unchanged from the POMDP belief update. However, incorporating this information into the agent's decision-making process aligns more closely with the Bayesian approach, as different transitions may have distinguishable sojourn time distributions that can aid in discerning the current state.

### 5.5.2 The Belief Update with Sojourn Time

We consider the case now where the sojourn time $\tau$ is observed or is part of the information used to update the belief state.

**Theorem 5.5.2** Given an action $a$ the agent performed, a sojourn time $\tau$, and an observation $o$ seen, the belief update for a particular state $s'$ is

$$\xi(s' \mid a, \tau, o) = \frac{G(o \mid a, s') \sum_{s \in \mathcal{S}} Q(\tau, s' \mid s, a)\xi(s)}{\sum_{s'' \in \mathcal{S}} G(o \mid a, s'') \sum_{s \in \mathcal{S}} Q(\tau, s'' \mid s, a)\xi(s)}.$$

**Proof** By definition,

$$\xi(s' \mid a, \tau, o) \triangleq \mathbf{P}(s' \mid \xi, a, \tau, o).$$

Applying the conditional rule

$$\mathbf{P}(A \mid B) = \frac{\mathbf{P}(A, B)}{\mathbf{P}(B)},$$

we have

$$\xi(s' \mid a, \tau, o) = \mathbf{P}(s' \mid a, \tau, o) = \frac{\mathbf{P}(a, \tau, o, s')}{\mathbf{P}(a, \tau, o)} = \frac{\mathbf{P}(a, \tau, o, s')}{\sum_{s'' \in \mathcal{S}} \mathbf{P}(a, \tau, o, s'')}. \tag{5.5.8}$$

Using the multiplication rule $\mathbf{P}(A, B) = \mathbf{P}(A \mid B)\,\mathbf{P}(B)$, we have

$$\mathbf{P}(s, a, \tau, o, s') = \mathbf{P}(o \mid s, a, \tau, s')\,\mathbf{P}(s, a, \tau, s')$$

and applying the multiplication rule again to the second term,

$$\mathbf{P}(s, a, \tau, o, s') = \underbrace{\mathbf{P}(o \mid s, a, \tau, s')}_{G(o|a,s')}\ \underbrace{\mathbf{P}(\tau, s' \mid s, a)}_{Q(\tau,s'|s,a)}\ \underbrace{\mathbf{P}(s, a)}_{\xi(s)}.$$

Since the observation $o$ only depends on action $a$ and the next state $s'$,

$$\mathbf{P}(o \mid s, a, \tau, s') = G(o \mid a, s'),$$

and by definition $\mathbf{P}(\tau, s' \mid s, a) = Q(\tau, s' \mid s, a)$ (see Eq. (5.1.2)) and $\mathbf{P}(s, a) = \mathbf{P}(a \mid s)\,\mathbf{P}(s) = \xi(s)$ since $\mathbf{P}(a \mid s) = 1$ for the selected action $a$ (the policy is deterministic), and $\mathbf{P}(s) = \xi(s)$ (our Bayesian estimate). Thus,

$$\mathbf{P}(s, a, \tau, o, s') = G(o \mid a, s')Q(\tau, s' \mid s, a)\xi(s)$$

and

$$\mathbf{P}(\xi, a, \tau, o, s') = \sum_{s \in \mathcal{S}} \mathbf{P}(s, a, \tau, o, s') = G(o \mid a, s') \sum_{s \in \mathcal{S}} Q(\tau, s' \mid s, a)\xi(s). \tag{5.5.9}$$

Substituting Eq. (5.5.9) into Eq. (5.5.8) yields the required result. ◀

We can further decomposed the belief update function when $Q(\tau, s' \mid s, a)$ is given as $P(s' \mid s, a)$ and $F(\tau \mid s, a, s')$ by

$$\xi(s' \mid a, \tau, o) = \frac{G(o \mid a, s') \sum_{s \in \mathcal{S}} Q(\tau, s' \mid s, a)\, \xi(s)}{P(o \mid \xi, a, \tau)} \qquad \forall s' \in \mathcal{S} \tag{5.5.10}$$

$$= \frac{G(o \mid a, s') \sum_{s \in \mathcal{S}} P(s' \mid s, a) F(\tau \mid s, a, s')\, \xi(s)}{P(o \mid \xi, a, \tau)} \qquad \forall s' \in \mathcal{S}, \tag{5.5.11}$$

where the denominator

$$P(o \mid \xi, a, \tau) = \sum_{s' \in \mathcal{S}} G(o \mid a, s') \sum_{s \in \mathcal{S}} Q(\tau, s' \mid s, a)\, \xi(s) \tag{5.5.12}$$

$$= \sum_{s' \in \mathcal{S}} G(o \mid a, s') \sum_{s \in \mathcal{S}} P(s' \mid s, a) F(\tau \mid s, a, s')\, \xi(s) \tag{5.5.13}$$

is a normalization factor similar to Wakuta (1982).

Note that this is similar to the belief update for POMDP; for instance, Eq. (5.5.11) is the same as Eq. (3.5.1), with the addition of the sojourn-time cumulative probability distribution $F(t \mid s, a, s')$ in the numerator and the denominator. This is important as Eq. (5.5.11) allows the agent to use the observed transition time to better estimate its real underlying state. Our solver is using this more complete approach.

## 5.6 Reward

We follow the same structure that we introduced with SMDPs in Sec. 4.3, and modify it to allow the reward function to take a belief state instead of a state as one of its parameters. Let $R(s, a)$ denotes the expected total discounted reward between two decision epochs, given that the agent is in state $s$ and it chooses to perform action $a$, which is expressed by

$$R(s, a) = \underbrace{r_1(s, a)}_{\substack{\text{(a)}\\ \text{immediate}\\ \text{reward}}} + \underbrace{\int_{s' \in \mathcal{S}} \int_0^\infty \left( \int_0^\tau e^{-\beta t} r_2(t \mid s, a, s')\, dt \right) Q(d\tau, ds' \mid s, a)}_{\substack{\text{(b)}\\ \text{expected discounted reward reward from } s \text{ to } s' \text{ over } \tau \text{ amount of time}}} . \tag{4.3.1}$$

Note that the discount rate $\beta$ is given as part of the model. The reward function can be thought of as two parts:

(a) the lump sum reward $r_1(s, a)$ that is received immediately upon the agent performing action $a$ in state $s$; and

(b) the continuous reward rate $r_2(\tau \mid s, a, s')$ that is received by the agent continuously over the sojourn time $\tau$ from state $s$ to $s'$ under action $a$.

Since the agent cannot observe the state $s \in \mathcal{S}$ directly, we can rewrite Eq. (4.3.1) for a given belief $\xi \in \triangle$ and action $a \in \mathcal{A}$ as

$$R(\xi, a) = \int_{s \in \mathcal{S}} \xi(ds) R(s, a), \tag{5.6.1}$$

or if it is a discrete set of states, it can be written as

$$R(\xi, a) = \sum_{s \in \mathcal{S}} \xi(s) R(s, a). \tag{5.6.2}$$

To ensure that the expected rewards are well-defined, we need a few assumptions. Time is the safeguard that prevents everything from happening at once. So, we borrow the following assumption for SMDPs by Ross (1970a, p. 157) that allows only a finite number of decision epochs during a finite sojourn time.

**Assumption 5.6.1** There exists a sojourn time $\tau > 0$ and $\epsilon > 0$ such that

$$\int_{s' \in \mathcal{S}} Q(\tau, ds' \mid s, a) \le 1 - \epsilon \qquad \forall (s, a) \in \mathcal{K}.$$

In other words, Assumption 5.6.1 says that for every (admissible) state-action pair $(s, a) \in \mathcal{K}$, there is a positive probability of at least $\epsilon$ that the transition time to state $s'$ will be greater than $\tau$.

We also impose the condition that the per-stage reward $R(s, a)$ is bounded for every state-action pair $(s, a) \in \mathcal{K}$.

**Assumption 5.6.2** There exists a constant $M$ such that for $\beta = 0$,

$$\sup_{(s,a)\in\mathcal{K}} |R(s,a)| < M.$$

This assumption is satisfied if $r_1$ and $r_2$ in Eq. (4.3.1) are bounded, and

$$\mathbf{E}(\tau_n \mid S_n = s, A_n = a) < \infty \qquad \forall (s,a) \in \mathcal{K}.$$

In other words, we would like each expected sojourn time $\mathbf{E}(\tau_n)$ to be bounded. Therefore, we make the following assumption.

**Assumption 5.6.3**

$$\sup_{(s,a)\in\mathcal{K}} \mathbf{E}(\tau_n \mid S_n = s, A_n = a) < \infty.$$

## 5.7 The Value Function for POSMDP

The value function for POSMDP can be derived using previous models we have encountered. We can get the POSMDP value function by taking the

(a) POMDP value function and inserting continuous time; or
(b) SMDP value function and inserting partial observability.

### 5.7.1 From POMDP to POSMDP

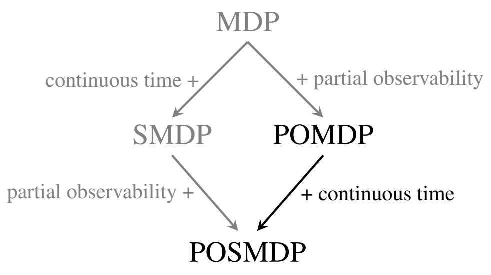


Fig. 5.5 The relationship between POMDP and POSMDP.

Recall from Sec. 3.6, the Bellman equation for a POMDP is

$$V^*(\xi) = \max_{a\in\mathcal{A}(s)} \Bigg[ \sum_{s\in\mathcal{S}} \xi(s) R(s,a) + \sum_{o\in\mathcal{O}} \sum_{s'\in\mathcal{S}} G(o \mid a, s') \\ \sum_{s\in\mathcal{S}} \xi(s) P(s' \mid s, a) \gamma V^{\pi}\big(\xi(\cdot \mid a, o)\big) \Bigg] \tag{3.6.5}$$

for all belief states $\xi$ in the belief simplex $\triangle$. When Eq. (3.6.5) holds for every belief state in the belief simplex, we are ensured the solution is optimal. If we were to compute the value function over the continuous belief space, belief by belief, it may seem intractable. However, Sondik (1971) shows that in this case, the POMDP optimal value function is convex, and that it can be parameterized by a finite set of hyperplanes.

In order to adapt Eq. (3.6.5) for a POSMDP, the discount factor must take the sojourn time probability $F(\mathrm{d}\tau \mid s, a, s')$ into account. It thus becomes

$$\gamma(s,a,s') = \int_0^\infty \mathrm{e}^{-\beta\tau} F(\mathrm{d}\tau \mid s,a,s') \tag{5.7.1}$$

and let $R(s,a)$ be defined by Eq. (4.3.1). Then, the POSMDP version of Eq. (3.6.5) is

$$V^*(\xi) = \max_{a\in\mathcal{A}(\xi)} \Bigg[ \underbrace{\sum_{s\in\mathcal{S}} \xi(s) R(s,a)}_{\text{Reward}} \\ + \underbrace{\sum_{o\in\mathcal{O}} \sum_{s'\in\mathcal{S}} \overbrace{G(o \mid a,s')}^{\text{Observation Probability}} \sum_{s\in\mathcal{S}} \xi(s) \underbrace{P(s' \mid s,a)}_{\text{State Probability}} \int_0^\infty \overbrace{\mathrm{e}^{-\beta\tau}}^{\text{Exponential Discount}} \underbrace{F(\mathrm{d}\tau \mid s,a,s')}_{\text{Sojourn Time Probability}} \overbrace{V^*\big(\xi(\cdot \mid a,\tau,o)\big)}^{\text{Next State Value}}}_{\text{Discounted Expected Future Rewards}} \Bigg] \tag{5.7.2}$$

or equivalently using Eq. (5.1.2),

$$V^*(\xi) = \max_{a\in\mathcal{A}(\xi)} \Bigg[ \underbrace{\sum_{s\in\mathcal{S}} \xi(s) R(s,a)}_{\text{Reward}} \\ + \underbrace{\sum_{o\in\mathcal{O}} \sum_{s'\in\mathcal{S}} \overbrace{G(o \mid a,s')}^{\text{Observation Probability}} \sum_{s\in\mathcal{S}} \xi(s) \int_0^\infty \overbrace{\mathrm{e}^{-\beta\tau}}^{\text{Exponential Discount}} \underbrace{Q(\mathrm{d}\tau, s' \mid s,a)}_{\text{Joint State sojourn-time Probability}} \overbrace{V^*\big(\xi(\cdot \mid a,\tau,o)\big)}^{\text{Next State Value}}}_{\text{Discounted Expected Future Rewards}} \Bigg]. \tag{5.7.3}$$

By including the sojourn time into the belief state, as opposed to including it as part of the state-space, we reduce the state space size at the cost of increasing the complexity of Eq. (5.7.2) or Eq. (5.7.3). In Zhang and Revie (2017), they rely upon numerical integration techniques for the sojourn-time integral at each step. Instead, we use the collected belief states and sampled sojourn times through importance sampling to evaluate the integral. This will be discussed further in Section 5.10.

## 5.7.2 From SMDP to POSMDP

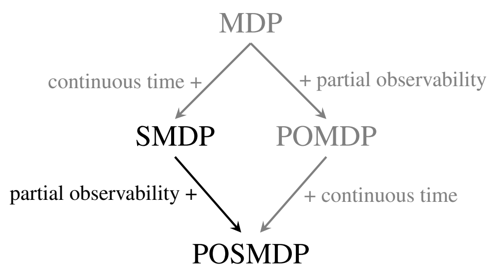


Fig. 5.6 The relationship between SMDP and POSMDP.

Recall from Sec. 4.5, the Bellman equation for a discrete-state infinite-horizon discounted SMDP is

$$V^*(s) = \max_{a\in\mathcal{A}(s)} \left\{ R(s,a) + \sum_{s'\in\mathcal{S}} \int_0^\infty \mathrm{e}^{-\beta\tau} Q(\mathrm{d}\tau, s' \mid s,a) V^\pi(s') \right\} \tag{4.5.6}$$

where $R(s,a)$ is given by Eq. (4.3.1). However, when we transform the state space $\mathcal{S}$ to belief state space $\triangle$, we transform it from a discrete space to a continuous space. Since the belief simplex $\triangle$ is continuous (non-empty uncountable Borel subset of a complete, separate metric) space, then using Eq. (4.5.8) we get

$$V^*(\xi) = \sup_{a\in\mathcal{A}(s)} \left\{ R(\xi,a) + \int_{\xi'\in\triangle} \int_{\tau=0}^{\tau=\infty} \mathrm{e}^{-\beta\tau} Q(\mathrm{d}\tau, \mathrm{d}\xi' \mid \xi, a) V^*(\xi') \right\} \tag{5.7.4}$$

and

$$R(\xi,a) = r_1(\xi,a) + \int_{\xi'\in\triangle} \int_{\tau=0}^{\tau=\infty} \left( \int_{t=0}^{t=\tau} \mathrm{e}^{-\beta t} r_2(t \mid \xi, a, \xi')\,\mathrm{d}t \right) Q(\mathrm{d}\tau, \mathrm{d}\xi' \mid \xi, a) \tag{5.7.5}$$

## 5.8 Reduction of POSMDP to POMDP

We begin with the POSMDP value function

$$\begin{aligned}
V^*(\xi) &= \max_{a\in\mathcal{A}(s)} \left[ R(\xi,a) + \gamma \sum_{o\in\mathcal{O}} P(o\mid\xi,a) V\big(\xi(\cdot\mid a,o)\big) \right] \\
&= \max_{a\in\mathcal{A}(s)} \left[ R(\xi,a) + \gamma \sum_{o\in\mathcal{O}} P(o\mid\xi,a) \max_{\alpha\in V} \langle \xi(\cdot\mid a,o), \alpha\rangle \right] \\
&= \max_{a\in\mathcal{A}(s)} \left[ R(\xi,a) + \gamma \sum_{o\in\mathcal{O}} P(o\mid\xi,a) \max_{\alpha\in V} \sum_{s'\in\mathcal{S}} \alpha(s')\xi(s'\mid a,o) \right] \\
&= \max_{a\in\mathcal{A}(s)} \left[ R(\xi,a) + \gamma \sum_{o\in\mathcal{O}} \cancel{P(o\mid\xi,a)} \max_{\alpha\in V} \sum_{s'\in\mathcal{S}} \alpha(s') \frac{G(o\mid a,s') \sum_{s\in\mathcal{S}} \xi(s) \int_0^\infty \mathrm{e}^{-\beta\tau} Q(\mathrm{d}\tau, s'\mid s,a)}{\cancel{P(o\mid\xi,a)}} \right] \\
&= \max_{a\in\mathcal{A}(s)} \left[ R(\xi,a) + \gamma \sum_{o\in\mathcal{O}} \max_{\alpha\in V} \sum_{s'\in\mathcal{S}} \alpha(s') G(o\mid a,s') \sum_{s\in\mathcal{S}} \xi(s) \int_0^\infty \mathrm{e}^{-\beta\tau} Q(\mathrm{d}\tau, s'\mid s,a) \right] \\
&= \max_{a\in\mathcal{A}(s)} \left[ R(\xi,a) + \gamma \sum_{o\in\mathcal{O}} \max_{\alpha\in V} \sum_{s\in\mathcal{S}} \xi(s) \underbrace{\sum_{s'\in\mathcal{S}} \alpha(s') G(o\mid a,s') \int_0^\infty \mathrm{e}^{-\beta\tau} Q(\mathrm{d}\tau, s'\mid s,a)}_{\alpha(s\mid a,o)} \right] \\
&= \max_{a\in\mathcal{A}(s)} \left[ R(\xi,a) + \gamma \sum_{o\in\mathcal{O}} \max_{\alpha\in V} \langle \xi, \alpha(\cdot\mid a,o)\rangle \right]
\end{aligned}$$

The backup can be written as

$$\textsc{backup}(V,\xi) = \operatorname*{arg\,max}_{\alpha(\cdot\mid\xi,a):a\in\mathcal{A},\alpha\in V} \langle \xi, \alpha(\cdot\mid\xi,a)\rangle \tag{5.8.1}$$

where for all $s\in\mathcal{S}$

$$\alpha(s\mid\xi,a) = R(s,a) + \gamma \sum_{o\in\mathcal{O}} \operatorname*{arg\,max}_{\alpha(\cdot\mid a,o):\alpha\in V} \langle \xi, \alpha(\cdot\mid a,o)\rangle\,, \tag{5.8.2}$$

and

$$\alpha(s \mid a, o) = \sum_{s' \in \mathcal{S}} \alpha(s') G(o \mid a, s') \int_0^\infty \mathrm{e}^{-\beta\tau} Q(\mathrm{d}\tau, s' \mid s, a) \tag{5.8.3}$$

$$= \sum_{s' \in \mathcal{S}} \alpha(s') G(o \mid a, s') \int_0^\infty \mathrm{e}^{-\beta\tau} P(s' \mid s, a) F(\mathrm{d}\tau \mid s, a, s')$$

$$= \sum_{s' \in \mathcal{S}} \alpha(s') G(o \mid a, s') P(s' \mid s, a) \int_0^\infty \mathrm{e}^{-\beta\tau} F(\mathrm{d}\tau \mid s, a, s'). \tag{5.8.4}$$

The algorithm requires an initial value function $V_0$ upon which to iterate. To conceive of an initial value function $V_0$ guaranteed to be below the optimal value function $V^*$, we assume for each decision epoch $n$ that the agent collects the smallest reward $\min_{(s,a)\in\mathcal{K}} R(s, a)$, even though it is not necessarily possible that the agent can land in state $s$ repeatedly. By making this assumption, we avoid having to find a feasible policy $\pi$ that returns the minimum cumulative discounted reward, which would require extensive computations (equivalent in computation to finding the maximum discounted expected reward). Thus, we set

$$R_{\min} \triangleq \min_{(s,a)\in\mathcal{K}} R(s, a) \tag{5.8.5}$$

and

$$\gamma_{\min} \triangleq \min_{(s,a)\in\mathcal{K}} \gamma(s, a) \tag{5.8.6}$$

Then, the sum of discounted minimum reward (following not necessarily a feasible policy) at each decision epoch $n$ is

$$R_{\min} + \gamma_{\min} R_{\min} + \gamma_{\min}^2 R_{\min} + \cdots = R_{\min}(1 + \gamma_{\min} + \gamma_{\min}^2 + \cdots),$$

which forms a geometric series, so

$$R_{\min} \sum_{n=0}^{\infty} \gamma_{\min}^n = \frac{R_{\min}}{1 - \gamma_{\min}},$$

for $\gamma \in (0, 1)$. The initial value function $V_0$ would then be defined by

$$\alpha_{\min}(s) = \frac{R_{\min}}{1 - \gamma_{\min}}, \qquad \forall s \in \mathcal{S}, \tag{5.8.7}$$

and

$$V_0 = \{\alpha_{\min}\}. \tag{5.8.8}$$

For a POSMDP, the discount factor depends on the triple $(s, a, s')$, which is given by

$$\gamma(s, a, s') = \int_0^\infty \mathrm{e}^{-\beta\tau} F(\mathrm{d}\tau \mid s, a, s'). \tag{5.8.9}$$

If $F$ is an inverse Gaussian distribution, then

$$\int_0^\infty \mathrm{e}^{-\beta\tau} F(\mathrm{d}\tau \mid \mu, \lambda) = \exp\left[\frac{\lambda}{\mu}\left(1 - \sqrt{1 + \frac{2\mu^2\beta}{\lambda}}\right)\right]. \tag{5.8.10}$$

From Eq. (5.8.7),

$$\alpha_{\min}(s) = \frac{R_{\min}}{1 - \gamma(s, a)}$$

$$= \frac{R_{\min}}{1 - \sum_{s' \in \mathcal{S}} P(s' \mid s, a)\gamma(s, a, s')}$$

Then,

$$\begin{aligned}
R_{\min} &= \sum_{n=0}^{\infty} \gamma^n \min_{(s,a)\in\mathcal{K}} R(s,a) \\
&= \sum_{n=0}^{\infty} \min_{(s,a)\in\mathcal{K}} (\gamma(s,a))^n R(s,a) \\
&= \min_{(s,a)\in\mathcal{K}} R(s,a) \sum_{n=0}^{\infty} (\gamma(s,a))^n \\
&= \min_{(s,a)\in\mathcal{K}} \frac{R(s,a)}{1-\gamma(s,a)} \\
&= \min_{(s,a)\in\mathcal{K}} \frac{R(s,a)}{1-\sum_{s'\in\mathcal{S}} P(s' \mid s,a)\gamma(s,a,s')}
\end{aligned}$$

To ensure that the denominator of $R_{\min}$ is not undefined, we have the following lemma.

**Lemma 5.8.1**

$$0 \le \sum_{s'\in\mathcal{S}} P(s' \mid s,a)\gamma(s,a,s') < 1.$$

**Proof** By Assumption 5.6.1, $\tau > 0$. Since we are considering the discounted POSMDP case, then $\beta > 0$ and

$$0 < \mathrm{e}^{-\beta\tau} < 1.$$

Multiplying through by the likelihood function $f(\tau \mid s,a,s')$,

$$0 < \mathrm{e}^{-\beta\tau} f(\tau \mid s,a,s') < f(\tau \mid s,a,s'),$$

and integrating over $(0,\infty)$ with respect to time yields

$$0 < \int_0^{\infty} \mathrm{e}^{-\beta\tau} f(\tau \mid s,a,s')\,\mathrm{d}\tau < \underbrace{\int_0^{\infty} f(\tau \mid s,a,s')\,\mathrm{d}\tau}_{1}$$

$$0 < \int_0^{\infty} \mathrm{e}^{-\beta\tau} f(\tau \mid s,a,s')\,\mathrm{d}\tau < 1$$

$$0 < \underbrace{\int_0^{\infty} \mathrm{e}^{-\beta\tau} F(\mathrm{d}\tau \mid s,a,s')}_{\gamma(s,a,s')} < 1.$$

Thus, $0 < \gamma(s,a,s') < 1$. Continuing, we have

$$0 < \gamma(s,a,s') < 1,$$

and multiplying through with $P(s' \mid s,a)$,

$$0 \le P(s' \mid s,a)\gamma(s,a,s') < P(s' \mid s,a). \tag{5.8.11}$$

The (nonstrict) inequality on the left hand side of Eq. (5.8.11) is due to the case when $P(s' \mid s,a) = 0$. Then, summing over all landing states $s' \in \mathcal{S}$,

$$0 \le \sum_{s'\in\mathcal{S}} P(s' \mid s,a)\gamma(s,a,s') < \sum_{s'\in\mathcal{S}} P(s' \mid s,a),$$

and since $\sum_{s' \in \mathcal{S}} P(s' \mid s, a) = 1$ by Eq. (2.2.3),

$$0 \leq \sum_{s' \in \mathcal{S}} P(s' \mid s, a) \gamma(s, a, s') < 1,$$

as required. ◀

## 5.9 Monte Carlo Integration

Before we move on to the reduction of POSMDP to SMDP, we will introduce a little bit of background knowledge about the Monte Carlo method and importance sampling. While Monte Carlo integration can apply to higher-dimensional integrals, we will concern ourselves with one-dimensional integrals since this is what we will need in Sec. 5.10. We follow the presentation given in Hammersley and Handscomb (1964, Sec. 5.2).

### 5.9.1 Naïve Monte Carlo

Consider the integral

$$\theta = \int_a^b f(x) \, \mathrm{d}x, \tag{5.9.1}$$

where the function $f\colon [a, b] \to \mathbb{R}$ is square integrable; that is, $\int_a^b |f(x)|^2 \, \mathrm{d}x < \infty$ and therefore, $\theta$ exists.

If $x_1, x_2, \ldots, x_N$ are random numbers that are independent and uniformly sampled along the interval $(a, b)$, then the quantities $f(x_n)$ are independent random variates with expected value $\theta$. An unbiased estimator of $\theta$ is

$$\hat{\theta} = \frac{1}{N} \sum_{n=1}^{N} f(x_n), \tag{5.9.2}$$

which we will refer to as the *naïve Monte Carlo estimator* of $\theta$, and its variance is

$$\frac{1}{N} \int_a^b \big(f(x) - \theta\big)^2 \, \mathrm{d}x = \frac{\sigma^2}{N}. \tag{5.9.3}$$

The standard error of $\hat{\theta}$ is

$$\sigma_{\hat{\theta}} = \frac{\sigma}{\sqrt{N}}. \tag{5.9.4}$$

The factor $\sqrt{N}$ in the denominator implies that in order to halve the error we must take four times as many samples. In practice, the standard error would be unknown, and so we can estimate $\sigma$ by using the sample standard deviation $s$, which is given by

$$s = \sqrt{\frac{\sum_{n=1}^{N} \big(f(x_n) - \hat{\theta}\big)^2}{N - 1}} = \sqrt{\frac{\sum_{n=1}^{N} f(x_n)^2 - \frac{\left(\sum_{n=1}^{N} f(x_n)\right)^2}{N}}{N - 1}}. \tag{5.9.5}$$

### 5.9.2 Importance Sampling

The objective of importance sampling is to concentrate the distribution of the sample points in the parts of the interval that have some importance or area under the curve instead of spreading them out uniformly, thus reducing the variance on the estimated value.

We have

$$\theta = \int_a^b f(x)\,\mathrm{d}x = \int_a^b \frac{f(x)}{g(x)} g(x)\,\mathrm{d}x = \int_a^b \frac{f(x)}{g(x)}\,\mathrm{d}G(x), \tag{5.9.6}$$

for any functions $g$ and $G$ satisfying

$$G(x) = \int_a^x g(y)\,\mathrm{d}y. \tag{5.9.7}$$

Let us restrict $g$ to be a positive-valued function such that

$$G(b) = \int_a^b g(y)\,\mathrm{d}y = 1. \tag{5.9.8}$$

Then, $G(x)$ is a distribution function for $x \in [a, b]$. Based on generating a sample $x_1$, $x_2, \ldots, x_N$ from the probability density function $g$, Eq. (5.9.6) can be approximated by

$$\hat{\theta} = \frac{1}{N} \sum_{n=1}^{N} \frac{f(x_n)}{g(x_n)}, \tag{5.9.9}$$

and the variance is

$$\sigma_{f/g}^2 = \int_a^b \left( \frac{f(x)}{g(x)} - \theta \right)^2 \mathrm{d}G(x). \tag{5.9.10}$$

**Example 5.9.1** We will numerically integrate the Laplace transform of the inverse Gaussian distribution (from Example 4.4.2, on p. 48). Let

$$I = \int_0^\infty \mathrm{e}^{-\beta x} f_X(x \mid \mu, \lambda)\,\mathrm{d}x, \tag{5.9.11}$$

where $f_X(x \mid \mu, \lambda)$ is the inverse Gaussian probability density function defined by Eq. (4.2.3). We can rewrite Eq. (5.9.11) as

$$I = \int_0^\infty \frac{\mathrm{e}^{-\beta x} f_X(x \mid \mu, \lambda)}{f_X(x \mid \mu, \lambda)} f_X(x \mid \mu, \lambda)\ \mathrm{d}x = \int_0^\infty \mathrm{e}^{-\beta x}\ \mathrm{d}F_X(x \mid \mu, \lambda). \tag{5.9.12}$$

From Eq. (5.9.12), we see that if we observe $x_1, x_2, \ldots, x_n$ from the probability density function $f_X(x \mid \mu, \lambda)$, then we can approximate Eq. (5.9.11) by

$$\hat{I} = \frac{1}{N} \sum_{n=1}^{N} \mathrm{e}^{-\beta x_n}. \tag{5.9.13}$$

To demonstrate this numerically, we set $\beta = 0.3$, $\mu = 3$, and $\lambda = 9$. We take 16 random inverse Gaussian variates,[1] and evaluate Eq. (5.9.13). The calculation is set out in Table 5.1: we find that $\hat{I} = 0.4739$ by Eq. (5.9.13) and $I = 0.4517$ by Eq. (4.4.7) so that $|\hat{I} - I| = 0.0222$, whilst the standard error is

$$\frac{s}{\sqrt{N}} = \frac{\sqrt{\dfrac{3.9736 - \frac{(7.5821)^2}{16}}{15}}}{\sqrt{16}} = \frac{0.1593}{4} = 0.0398,$$

in good agreement. ◀

[1] Actually, we extracted one Gaussian variate and one uniform variate from a printed table of randomly generated numbers (RAND Corporation, 1955) to create one inverse Gaussian variate in accordance with Algorithm 13.

Table 5.1 Monte Carlo integration of $\int_0^\infty e^{-\beta x} f(x \mid \mu, \lambda)\,dx$, where $\beta = 0.3$, $\mu = 3$, and $\lambda = 9$.

| $n$ | $z_n \sim \mathcal{N}(0,1)$ | $y_n \sim \mathcal{U}(0,1)$ | $x_n \sim \mathcal{IG}(3,9)$ | $e^{-\beta x_n}$ |
|---|---|---|---|---|
| 1 | −1.276 | 0.1009 | 1.4588 | 0.6456 |
| 2 | −1.218 | 0.7325 | 5.9780 | 0.1664 |
| 3 | −0.453 | 0.3376 | 2.3113 | 0.4999 |
| 4 | −0.350 | 0.5201 | 2.4519 | 0.4792 |
| 5 | 0.723 | 0.3586 | 1.9821 | 0.5518 |
| 6 | 0.676 | 0.3467 | 2.0355 | 0.5430 |
| 7 | −1.099 | 0.3548 | 1.6069 | 0.6175 |
| 8 | −0.314 | 0.7680 | 3.5954 | 0.3401 |
| 9 | −0.394 | 0.9590 | 3.7644 | 0.3233 |
| 10 | −0.633 | 0.9117 | 4.3149 | 0.2740 |
| 11 | −0.318 | 0.3929 | 2.4975 | 0.4727 |
| 12 | −0.799 | 0.2749 | 1.8990 | 0.5657 |
| 13 | −1.664 | 0.4537 | 1.1870 | 0.7004 |
| 14 | 1.391 | 0.5420 | 1.3712 | 0.6627 |
| 15 | 0.382 | 0.4805 | 2.4073 | 0.4857 |
| 16 | 0.733 | 0.6489 | 4.5663 | 0.2541 |
| Average | | | | 0.4739 |
| True Value | | | | 0.4517 |

## 5.10 Reduction of POSMDP to SMDP

In this case, the partially observable nature of the POSMDP is reduced to a fully observable SMDP model.

Recall the Bellman equation for a POSMDP,

$$V^*(\xi) = \max_{a \in \mathcal{A}(\xi)} \Bigg[ R(\xi, a) + \sum_{o \in \mathcal{O}} \sum_{s' \in \mathcal{S}} G(o \mid a, s') \sum_{s \in \mathcal{S}} \xi(s) P(s' \mid s, a) \int_0^\infty e^{-\beta\tau} F(d\tau \mid s, a, s') V^*\big(\xi(\cdot \mid a, \tau, o)\big) \Bigg].$$

The difficulty with calculating the integral

$$\int_0^\infty e^{-\beta\tau} f(\tau \mid s, a, s') V^*\big(\xi(\cdot \mid a, \tau, o)\big)\,d\tau$$

in the Bellman equation for a POSMDP is that the value of the next belief $V^*\big(\xi(\cdot \mid a, \tau, o)\big)$ is dependent on the sojourn time $\tau$. This means that unlike the integral in Eq. (5.7.2) where

$$\int_0^\infty e^{-\beta\tau} f(\tau \mid s, a, s') V^*\big(\xi(\cdot \mid a, o)\big)\,d\tau = V^*\big(\xi(\cdot \mid a, o)\big) \int_0^\infty e^{-\beta\tau} f(\tau \mid s, a, s')\,d\tau,$$

we cannot use the Laplace transform like we did in Sec. 4.4.

To approximate the integral

$$\mu(\xi, s, o, a, s') = \int_0^\infty \mathrm{e}^{-\beta\tau} f(\tau \mid s, a, s') V^*\big(\xi(\cdot \mid a, \tau, o)\big)\,\mathrm{d}\tau, \tag{5.10.1}$$

we use the Monte Carlo method with importance sampling as described in Sec. 5.9.

Suppose we have a set $C$ whose elements $\tau_1, \tau_2, \ldots, \tau_{|C|}$ are independent sojourn time samples from the probability density function $D$. An unbiased estimator of $\mu(s, o, a, s')$ is

$$\hat{\mu}(\xi, s, o, a, s') = \frac{1}{|C|}\sum_{n=1}^{|C|} \mathrm{e}^{-\beta\tau_n} \frac{f(\tau_n \mid s, a, s')}{D(\tau_n)} V^*\big(\xi(\cdot \mid a, \tau_n, o)\big) \tag{5.10.2}$$

where the function $D(\tau_n)$ given by

$$D(\tau_n) = \sum_{s\in\mathcal{S}}\sum_{a\in\mathcal{A}}\sum_{s'\in\mathcal{S}} w(s, a, s') f(\tau_n \mid s, a, s') \tag{5.10.3}$$

is a mixture distribution of each sojourn-time distribution for each $s, a, s'$ and the weights $w(s, a, s')$ reflect the proportion of samples that came from each distribution. Note that the weights must sum to one; that is,

$$\sum_{s\in\mathcal{S}}\sum_{a\in\mathcal{A}}\sum_{s'\in\mathcal{S}} w(s, a, s') = 1. \tag{5.10.4}$$

**Example 5.10.1 — Mixture of Two Inverse Gaussian Distributions.** Suppose we sample $\tau_1, \ldots, \tau_{12}$ from an inverse Gaussian distribution $\mathcal{IG}(3, 9)$, and $\tau_{13}, \ldots, \tau_{16}$ from another inverse Gaussian distribution $\mathcal{IG}(5, 25)$. The probability density function of the mixture of these two inverse Gaussian distributions is $D(\tau_n) = w_1 f(\tau_n \mid \mu = 3, \lambda = 9) + w_2 f(\tau_n \mid \mu = 5, \lambda = 25)$, where $w_1 = \frac{12}{16} = 75\%$ and $w_2 = \frac{4}{16} = 25\%$.

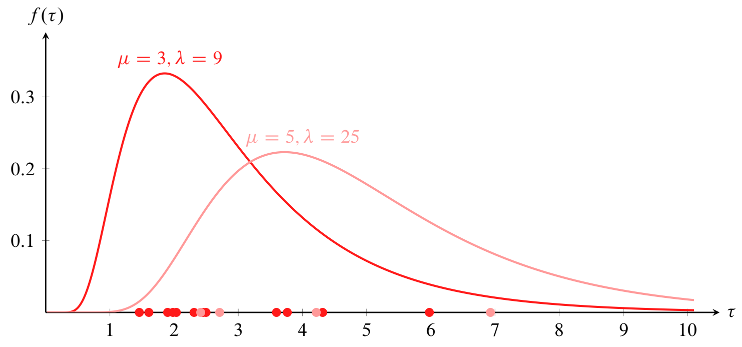


Fig. 5.7 Two inverse Gaussian probability density distributions $\mathcal{IG}(3, 9)$ and $\mathcal{IG}(5, 25)$ with the sampled points $\tau_1, \ldots, \tau_{16}$ along the $\tau$-axis in Example 5.10.1.

If we wish to estimate the integrals

$$\int_0^\infty \mathrm{e}^{-\beta\tau} f(\tau \mid \mu = 3, \lambda = 9)\,\mathrm{d}\tau \tag{5.10.5}$$

and

$$\int_0^\infty \mathrm{e}^{-\beta\tau} f(\tau \mid \mu = 5, \lambda = 25)\,\mathrm{d}\tau, \tag{5.10.6}$$

based on our sample $\tau_1, \ldots, \tau_{16}$, for integral (5.10.5) the estimator is

$$\frac{1}{16}\sum_{n=1}^{16} \mathrm{e}^{-\beta\tau_n} \frac{f(\tau_n \mid \mu = 3, \lambda = 9)}{D(\tau_n)}, \tag{5.10.7}$$

and for integral (5.10.6) the estimator is

$$\frac{1}{16}\sum_{n=1}^{16} \mathrm{e}^{-\beta\tau_n}\frac{f(\tau_n \mid \mu = 5, \lambda = 25)}{D(\tau_n)}. \tag{5.10.8}$$

The calculation is set out in Table 5.2 (on p. 93). ◀

By replacing the integral (5.10.1) with the approximation (5.10.2) in the Bellman equation for POSMDP, we have

$$V^*(\xi) = \max_{a\in\mathcal{A}(\xi)}\Bigg[R(\xi,a) + \sum_{o\in\mathcal{O}}\sum_{s'\in\mathcal{S}} G(o \mid a,s')\sum_{s\in\mathcal{S}}\xi(s)P(s' \mid s,a) \\ \frac{1}{|C|}\sum_{n=1}^{|C|}\mathrm{e}^{-\beta\tau_n}\frac{f(\tau_n \mid s,a,s')}{D(\tau_n)}V^*\big(\xi(\cdot \mid a,\tau_n,o)\big)\Bigg].$$

We bring $\frac{1}{|C|}\sum_{n=1}^{|C|}\frac{\mathrm{e}^{-\beta\tau_n}}{D(\tau_n)}$ to the beginning of the second term since it does not depend on state $s$ or observation $o$, yielding

$$V^*(\xi) = \max_{a\in\mathcal{A}(\xi)}\Bigg[R(\xi,a) + \frac{1}{|C|}\sum_{n=1}^{|C|}\frac{\mathrm{e}^{-\beta\tau_n}}{D(\tau_n)}\sum_{o\in\mathcal{O}}\sum_{s'\in\mathcal{S}} G(o \mid a,s') \\ \sum_{s\in\mathcal{S}}\xi(s)P(s' \mid s,a)f(\tau_n \mid s,a,s')V^*\big(\xi(\cdot \mid a,\tau_n,o)\big)\Bigg],$$

and by Eq. (5.5.13),

$$V^*(\xi) = \max_{a\in\mathcal{A}(\xi)}\left[R(\xi,a) + \frac{1}{|C|}\sum_{n=1}^{|C|}\frac{\mathrm{e}^{-\beta\tau_n}}{D(\tau_n)}\sum_{o\in\mathcal{O}}P(o \mid \xi,a,\tau_n)V^*\big(\xi(\cdot \mid a,\tau_n,o)\big)\right]. \tag{5.10.9}$$

Next, we will write a concise backup operation by following the procedure similar to POMDPs in Sec. 3.8.2 to express the value function as a finite set of $\alpha$ vectors. Using Eq. (3.7.2) in Eq. (5.10.9), we can express the value of the next belief as an inner product as

$$V^*(\xi) = \max_{a\in\mathcal{A}(\xi)}\left[R(\xi,a) + \frac{1}{|C|}\sum_{n=1}^{|C|}\frac{\mathrm{e}^{-\beta\tau_n}}{D(\tau_n)}\sum_{o\in\mathcal{O}}P(o \mid \xi,a,\tau_n)\max_{\alpha\in V}\langle\xi(\cdot \mid a,\tau_n,o),\alpha\rangle\right]. \tag{5.10.10}$$

Expressing the inner product as a sum by Eq. (3.7.3), we have

$$V^*(\xi) = \max_{a\in\mathcal{A}(\xi)}\Bigg[R(\xi,a) + \frac{1}{|C|}\sum_{n=1}^{|C|}\frac{\mathrm{e}^{-\beta\tau_n}}{D(\tau_n)}\sum_{o\in\mathcal{O}}P(o \mid \xi,a,\tau_n) \\ \max_{\alpha\in V}\sum_{s'\in\mathcal{S}}\alpha(s')\xi(s' \mid a,\tau_n,o)\Bigg] \tag{5.10.11}$$

and using Eq. (5.5.11) for the value of the next belief allows us to cancel out $P(o \mid \xi, a, \tau_n)$ so that

$$V^*(\xi) = \max_{a \in \mathcal{A}(\xi)} \left[ R(\xi, a) + \frac{1}{|C|} \sum_{n=1}^{|C|} \frac{e^{-\beta\tau_n}}{D(\tau_n)} \sum_{o \in \mathcal{O}} \cancel{P(o \mid \xi, a, \tau_n)} \right.$$
$$\left. \max_{\alpha \in V} \sum_{s' \in \mathcal{S}} \alpha(s') \frac{G(o \mid a, s') \sum_{s \in \mathcal{S}} \xi(s) P(s' \mid s, a) f(\tau_n \mid s, a, s')}{\cancel{P(o \mid \xi, a, \tau_n)}} \right]$$
$$= \max_{a \in \mathcal{A}(\xi)} \left[ R(\xi, a) + \frac{1}{|C|} \sum_{n=1}^{|C|} \frac{e^{-\beta\tau_n}}{D(\tau_n)} \sum_{o \in \mathcal{O}} \max_{\alpha \in V} \sum_{s' \in \mathcal{S}} \alpha(s') G(o \mid a, s') \right.$$
$$\left. \sum_{s \in \mathcal{S}} \xi(s) P(s' \mid s, a) f(\tau_n \mid s, a, s') \right].$$

Observe that this approach allows the elimination of the computationally expensive term $P(o \mid \xi, a, \tau_n)$. Rearranging the order of the summations and the terms so that we have an $\alpha$-vector yields,

$$V^*(\xi) = \max_{a \in \mathcal{A}(\xi)} \left[ R(\xi, a) + \frac{1}{|C|} \sum_{n=1}^{|C|} \frac{e^{-\beta\tau_n}}{D(\tau_n)} \sum_{o \in \mathcal{O}} \max_{\alpha \in V} \sum_{s \in \mathcal{S}} \xi(s) \right.$$
$$\left. \underbrace{\sum_{s' \in \mathcal{S}} \alpha(s') G(o \mid a, s') P(s' \mid s, a) f(\tau_n \mid s, a, s')}_{\alpha(s \mid a, \tau_n, o)} \right]$$
$$= \max_{a \in \mathcal{A}(\xi)} \left[ R(\xi, a) + \frac{1}{|C|} \sum_{n=1}^{|C|} \frac{e^{-\beta\tau_n}}{D(\tau_n)} \sum_{o \in \mathcal{O}} \max_{\alpha \in V} \sum_{s \in \mathcal{S}} \xi(s) \alpha(s \mid a, \tau_n, o) \right] \tag{5.10.12}$$

and ultimately, by Eq. (3.7.3), we arrive at the final optimal value function equation

$$\boxed{V^*(\xi) = \max_{a \in \mathcal{A}(\xi)} \left[ R(\xi, a) + \frac{1}{|C|} \sum_{n=1}^{|C|} \frac{e^{-\beta\tau_n}}{D(\tau_n)} \sum_{o \in \mathcal{O}} \max_{\alpha \in V} \langle \xi, \alpha(\cdot \mid a, \tau_n, o) \rangle \right]} \tag{5.10.13}$$

This expression represents the approximate form of the POSMDP value function that we will employ in the ChronosPerseus algorithm. It is noteworthy that the integral of the sojourn time, which was previously situated in one of the inner summation loops in Eq. (5.7.2) or Eq. (5.7.3), now resides in the outer summation loop in Eq. (5.10.12) as a result of importance sampling.

## 5.11 ChronosPerseus

We now introduce one of the main contributions of the thesis—the ChronosPerseus algorithm. This algorithm represents an extension of point-based value iteration, enhanced with importance sampling, specifically designed for POSMDPs. As delineated in Algorithm 16, ChronosPerseus operates in two stages: initially, it collects belief and sojourn time samples; subsequently, it approximates the value function using the backup operator. Distinctions between ChronosPerseus and the original Perseus algorithm (see Algorithm 9 on p. 32), used for solving POMDPs, are highlighted in red. These adaptations incorporated into ChronosPerseus, specifically addressing the importance sampling related to sojourn times, have effectively broadened its applicability to POSMDPs. By doing so, it has contributed to making these complex processes more computationally manageable.

**Algorithm 16** ChronosPerseus

```
1: function ChronosPerseus($n$, $\xi_0$, $V_0$, $\epsilon$)
2:     $(B, C, w) \leftarrow$ collectBeliefs$(n, \xi_0)$
3:     $V' \leftarrow V_0$
4:     repeat
5:         $V \leftarrow V'$
6:         $V' \leftarrow$ update$(B, C, w, V)$
7:     until $||V - V'||_{\infty,B} < \epsilon$
8:     return $V$
9: end function
```

The first step, as outlined in Algorithm 17, is to let the agent explore the environment and collect a finite set of beliefs $B$, a finite set of sojourn times $C$, and the weights $w(s, a, s')$ that reflect the proportion of samples that came from each state-action-state transition; after these are collected, they remain fixed for the rest of the algorithm.

**Algorithm 17** CollectBeliefs

```
1: function collectBeliefs($n$, $\xi_0$)
2:     $B \leftarrow \{\xi_0\}$, $C \leftarrow \varnothing$, $w \leftarrow 0$                ▷ Initialization
3:     repeat
4:         Randomly select the belief state $\xi \in B$
5:         Generate state $s$ from the multinomial distribution with weights $\xi$
6:         Randomly select an action $a \in \mathcal{A}$
7:         Randomly select state $s'$ according to $P(\cdot \mid s, a)$
8:         Generate a random sojourn-time $\tau$ according to $f(\tau \mid s, a, s')$
9:         $C \leftarrow C \cup \{\tau\}$                                  ▷ Add $\tau$ to set $C$
10:        $w(s, a, s') \leftarrow w(s, a, s') + 1$                     ▷ Update counts
11:        Randomly select $o \in \mathcal{O}$ according to $P(o \mid \xi, a, \tau)$ using Eq. (5.5.13)
12:        Update $\xi(\cdot \mid a, \tau, o)$ according to Eq. (5.5.11)
13:        $B \leftarrow B \cup \{\xi(\cdot \mid a, \tau, o)\}$                 ▷ Add new belief to set $B$
14:    until $|B| = n$
15:    $w \leftarrow \frac{w}{||w||}$               ▷ Normalize $w$; $||w|| = \sum_{s \in \mathcal{S}} \sum_{a \in \mathcal{A}} \sum_{s' \in \mathcal{S}} w(s, a, s')$.
16:    return $B$, $C$, and $w$
17: end function
```

In the second step, as outlined in Algorithm 18, from the initial value function $V_0$, the algorithm will continue to perform backups until a convergence criterion is achieved such as

$$||V - V'||_{\infty,B} \triangleq \max_{\xi \in B} |V(\xi) - V'(\xi)| < \epsilon. \tag{5.11.1}$$

**Algorithm 18** UPDATE

```
1: function UPDATE($B, C, w, V$)
2:     $B' \leftarrow B$, $V' \leftarrow \varnothing$                    ▷ Initialization
3:     while $B' \neq \varnothing$ do
4:         Randomly select a belief $\xi \in B'$
5:         $\alpha \leftarrow$ BACKUP($V, C, w, \xi$)          ▷ Backup defined by Eq. (5.11.2)
6:         if $\langle \xi, \alpha \rangle < V(\xi)$ then             ▷ $V(\xi) = \max_{\alpha' \in V} \langle \xi, \alpha' \rangle$
7:             $\alpha \leftarrow \arg\max_{\alpha' \in V} \langle \xi, \alpha' \rangle$    ▷ If $\alpha$ is not better, get $\alpha'$ from old set $V$
8:         end if
9:         $B' \leftarrow B' \setminus \{\varsigma \in B' \mid \langle \varsigma, \alpha \rangle \geq V(\varsigma)\}$    ▷ Keep beliefs not improved by $\alpha$.
10:        $V' \leftarrow V' \cup \{\alpha\}$                        ▷ Adds $\alpha$ to set $V'$
11:    end while
12:    $V \leftarrow V'$
13:    return $V$
14: end function
```

The initial value function, $V_0$, must be chosen carefully to ensure that it is lower than the optimal value function $V^*$. In PERSEUS, $V_0$ is initialized by setting all of its components each to $\frac{1}{1-\gamma} \min_{(s,a)\in\mathcal{K}} R(s,a)$, where $\gamma$ represents the discount factor. By following a similar argument by Zhang and Zhang (2001), we can modify the discount factor $\gamma$ to $\lambda$ that represents the exponential discount in a POSMDP. This leads us to our next theorem.

**Theorem 5.11.1** Let $M = \min_{(s,a)\in\mathcal{K}} R(s,a)$. The initial value function $V_0$ is a single $\alpha$-vector with

$$V_0(s) = \frac{M}{1-\lambda} \qquad \forall s \in \mathcal{S},$$

where

$$\lambda = \begin{cases} \min_{1\leq n\leq |C|} \dfrac{\mathrm{e}^{-\beta\tau_n}}{D(\tau_n)}, & \text{if } M \geq 0; \\[2ex] \max_{1\leq n\leq |C|} \dfrac{\mathrm{e}^{-\beta\tau_n}}{D(\tau_n)}, & \text{if } M < 0. \end{cases}$$

**Proof** Since $\mathcal{S}$ and $\mathcal{A}$ are finite sets, the per-stage minimum reward $M$ exists by Assumption 5.6.2.

If $M = 0$, then this is trivial: set $\alpha(s) = 0$ for all $s \in \mathcal{S}$.

For $M > 0$, we start with Eq. (5.10.9), for any belief state $\xi$. Let $\lambda_n = \dfrac{\mathrm{e}^{-\beta\tau_n}}{D(\tau_n)}$, and $0 < \lambda < 1$. Since $M$ is positive, we would like the discount to be as small as possible. So, we define

$$\lambda = \min_{1\leq n\leq |C|} \lambda_n, \qquad n^* = \arg\min_{1\leq n\leq |C|} \lambda_n, \qquad \text{and} \qquad \tau = \tau_{n*},$$

so that

$$\begin{aligned}
V(\xi) &\geq \max_{a \in \mathcal{A}(\xi)} \left[ R(\xi, a) + \frac{1}{|C|} \sum_{n=1}^{|C|} \lambda \sum_{o \in \mathcal{O}} P(o \mid \xi, a, \tau) V\big(\xi(\cdot \mid a, \tau, o)\big) \right] \\
&= \max_{a \in \mathcal{A}(\xi)} \left[ R(\xi, a) + \lambda \sum_{o \in \mathcal{O}} P(o \mid \xi, a, \tau) V\big(\xi(\cdot \mid a, \tau, o)\big) \right] \\
&\geq \max_{a \in \mathcal{A}(\xi)} \left[ M + \lambda \left( \frac{M}{1-\lambda} \right) \right] \\
&= M + \frac{\lambda M}{1-\lambda} \\
&= \frac{M}{1-\lambda} = V_0.
\end{aligned}$$

If $M < 0$, then it is a negative reward, and so the discount should be as large as possible, so

$$\lambda = \max_{1 \leq n \leq |C|} \lambda_n, \qquad n^* = \arg\max_{1 \leq n \leq |C|} \lambda_n, \qquad \text{and} \qquad \tau = \tau_{n*}.$$

◀

We can now write a concise backup operation that generates a new $\alpha$ vector for a specific belief $\xi$; that is,

$$\text{BACKUP}(V, C, w, \xi) = \underset{\alpha(\cdot \mid \xi, a): a \in \mathcal{A}, \alpha \in V}{\arg\max} \langle \xi, \alpha(\cdot \mid \xi, a) \rangle \tag{5.11.2}$$

where for all $s \in \mathcal{S}$,

$$\alpha(s \mid \xi, a) = R(s, a) + \frac{1}{|C|} \sum_{n=1}^{|C|} \frac{\mathrm{e}^{-\beta \tau_n}}{D(\tau_n)} \sum_{o \in \mathcal{O}} \underset{\alpha(\cdot \mid a, \tau_n, o): \alpha \in V}{\arg\max} \langle \xi, \alpha(\cdot \mid a, \tau_n, o) \rangle, \tag{5.11.3}$$

and

$$\alpha(s \mid a, \tau_n, o) = \sum_{s' \in \mathcal{S}} G(o \mid a, s') P(s' \mid s, a) f(\tau_n \mid s, a, s') \alpha(s'). \tag{5.11.4}$$

The complexity of computing Eq. (5.11.4) is $O(|\mathcal{S}|^2)$ since it needs to be calculated for every $(s, s')$ tuple, and it is done for every $\alpha \in V$, hence computing all $\alpha(\cdot \mid a, \tau, o)$ for every $a \in \mathcal{A}$, $\tau \in C$, and $o \in \mathcal{O}$, requires $O(|V| \times |\mathcal{S}|^2 \times |\mathcal{A}| \times |C| \times |\mathcal{O}|)$. The complexity of computing Eq. (5.11.3) requires the computation of all relevant $\alpha(\cdot \mid a, \tau, o)$, but then the summation and inner products require only $O(|\mathcal{S}| \times |C| \times |\mathcal{O}|)$ operations and another $O(|\mathcal{S}|)$ operations to add the reward (vector). Lastly, the BACKUP (Eq. (5.11.2)) requires for all $\alpha(\cdot \mid \xi, a)$ another $O(|\mathcal{S}|)$ operations for the inner product. Therefore, the complexity of the BACKUP operation requires

$$O(|V| \times |\mathcal{S}|^2 \times |\mathcal{A}| \times |C| \times |\mathcal{O}|). \tag{5.11.5}$$

Given there are no more beliefs than observed sojourns time, $|B| \leq |C|$, the complexity of an UPDATE iteration is therefore

$$O(|V| \times |\mathcal{S}|^2 \times |\mathcal{A}| \times |C|^2 \times |\mathcal{O}|). \tag{5.11.6}$$

**Algorithm 19** Controller for an Online Agent (POSMDP)

In the beginning $n = 0$, the state $s_0$ is simulated from an initial (belief) state distribution $\xi_0$.

For each decision epoch $n = 1, 2, \dots, N$:

(a) Based on the current belief state $\xi$, the agent performs action

$$a_n = \pi_n(\xi_n) \in \mathcal{A}, \qquad n = 1, 2, \dots, N,$$

according to policy $\pi_n$ that the agent uses at decision epoch $n$.

(b) The agent obtains a reward $R(s_n, a_n)$ for choosing action $a_n$ at decision epoch $n$.

(c) The state evolves randomly with sojourn time-state transition probability

$$Q(\tau, s' \mid s, a) = \mathbf{P}(T_{n+1} - T_n \le \tau, S_{n+1} = s' \mid S_n = s, A_n = a)$$

to the next state $S_{n+1}$ that decision epoch $n+1$.

(d) The agent records an exact time $\tau$ according to

$$F(\tau \mid s, a, s')$$

(e) The agent records a noisy observation $O_n \in \mathcal{O}$ of the state $S_{n+1}$ according to

$$G(o \mid a, s') = \mathbf{P}(O_n = o \mid A_n = a, S_{n+1} = s').$$

(f) With the selected action $a$, and the observation $o$ and time $\tau$ observed, the agent updates its belief state as

$$\xi(s' \mid a, \tau, o) = \frac{G(o \mid a, s') \sum_{s \in \mathcal{S}} \xi(s) Q(\tau, s' \mid s, a)}{\sum_{s'' \in \mathcal{S}} G(o \mid a, s'') \sum_{s \in \mathcal{S}} \xi(s) Q(\tau, s'' \mid s, a)} \qquad \forall s' \in \mathcal{S},$$

or

$$\xi(s' \mid a, \tau, o) = \frac{G(o \mid a, s') \sum_{s \in \mathcal{S}} \xi(s) P(s' \mid s, a) f(\tau \mid s, a, s')}{\sum_{s'' \in \mathcal{S}} G(o \mid a, s'') \sum_{s \in \mathcal{S}} \xi(s) P(s'' \mid s, a) f(\tau \mid s, a, s')} \qquad \forall s' \in \mathcal{S},$$

where the initial belief $\xi_0$ is given in the model.

(g) If $n < N$, then set $n$ to $n + 1$, and go back to step (a).
If $n = N$, then the agent receives the last reward and the process terminates.

## 5.12 Applications

We now showcase the application of ChronosPerseus to two compelling examples. The first example is an episodic, relatable scenario we frequently experience in our daily life, where waiting time influences an individual's belief and subsequent decision-making. In this simplified scenario, an agent travels on a bus with an unknown traffic intensity and has the option to abandon the bus and ride their bicycle at any bus stop. The second example is a real-world application concerning routine maintenance of a water filtration system, initially presented with a continuous observation space by Zhang and Revie (2017).

These two examples will illustrate how ChronosPerseus can effectively solve real POSMDP problems with mixed observability, a combination of observable continuous and discrete random sojourn times, discrete or continuous observation spaces, and episodic or non-episodic tasks.

**Example 5.12.1 — Bus Problem.** Suppose an agent wishes to travel by bus with a bicycle attached to the bus rack. The agent faces four bus stops before reaching the final destination and encounters three levels of traffic intensity $i$ (low $i = 1$, medium $i = 2$, and high $i = 3$) that remain constant throughout the journey. The traffic intensity $i$ is unknown to the agent. At each bus stop $s$, the agent has the option to either remain on the bus or disembark and ride its bicycle directly to the destination (the last stop); there is no option to reboard the bus at subsequent stops. Reaching the destination leads to a (lump sum) reward, but the longer it takes, the more discounted it will be. Suppose the agent decides to continue. In that case, the next state will be $s + 1$ with sojourn time $\tau$ according to the probability density function $f\big(\tau, (s+1, i) \mid (s, i)\big)$. If the agent chooses to take the bike, it takes a fixed time to reach the destination depending on the current stop $s$. What is the optimal policy that will maximize the agent's discounted reward while minimizing travel time, taking into account their belief about traffic intensity?

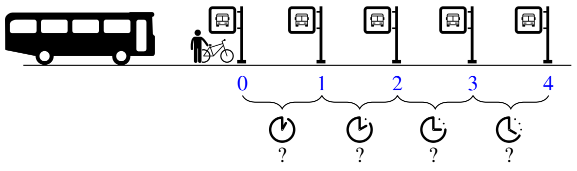


Fig. 5.8 The agent waiting for the bus in Example 5.12.1.

**Solution** Intuitively, we would expect that the agent should ride the bus long enough to deduce which traffic intensity level:

(a) if the traffic intensity is low, then the agent would ride the bus to the end;
(b) if the traffic intensity is high, then the agent would disembark at the next stop and ride the bike to the end; and
(c) if the traffic intensity is medium, then the agent would have to balance the advantage of riding the bus versus riding the bike.

With this intuition in mind, we can model this problem formally using the POSMDP framework. The state space consists of the bus stops and the three levels of traffic intensity,

$$\mathcal{S} = \overbrace{\underbrace{\{0, 1, 2, 3, 4\}}_{\text{bus stops}}}^{\text{observable}} \times \overbrace{\underbrace{\{1, 2, 3\}}_{\text{traffic}}}^{\text{hidden}},$$

or

$$\begin{aligned}\mathcal{S} = \{&(0, 1), (0, 2), (0, 3),\\ &(1, 1), (1, 2), (1, 3),\\ &(2, 1), (2, 2), (2, 3),\\ &(3, 1), (3, 2), (3, 3),\\ &(4, 1), (4, 2), (4, 3)\},\end{aligned}$$

which means the number of states is

$$|\mathcal{S}| = 5 \times 3 = 15.$$

A typical state would be the pair $(s, i) \in \mathcal{S}$, where the bus is at stop $s$ and the traffic intensity is $i$. We will assume that the traffic intensity remains $i$ throughout the entire bus ride; that is,

$$(0, i) \to (1, i) \to (2, i) \to (3, i) \to (4, i).$$

The action space consists of two actions

$$\mathcal{A} = \{\text{bus}, \text{bike}\}.$$

The observation space consists of the bus stops that the agent observes,

$$\mathcal{O} = \underbrace{\{0, 1, 2, 3, 4\}}_{\text{bus stops}}.$$

Now that we have defined the state and action spaces, we turn our attention to the observation space. In this problem, the observation space differs from the state space, $\mathcal{O} \neq \mathcal{S}$, which leads to the issue of *mixed observability* (Ong *et al.*, 2010). This means that while the agent can perfectly observe its location $s$, it cannot directly observe the traffic intensity $i$. We will address this issue as we develop the observation transition function.

If the agent decides to continue riding the bus, the traffic intensity $i$ does not change, but the bus stop changes from $s$ to $s + 1$. If the agent decides to ride the bus to the last bus stop, then the problem resets probabilistically by placing the agent at bus stop 0 and randomly selecting the traffic intensity with equal probability; this is reflected in the probability $P\big((0, i') \mid (4, i), \text{bus}\big) = \frac{1}{3}$. To illustrate this visually, a state transition diagram for the bus action is given in Fig. 5.9 on p. 84. Thus, the transition probability matrices for continuing to ride the bus are

$$P\big((\cdot, i) \mid (\cdot, i), \text{bus}\big) = \begin{array}{c|ccccc} & s'{=}0 & s'{=}1 & s'{=}2 & s'{=}3 & s'{=}4 \\ \hline s{=}0 & 0 & 1 & 0 & \cdots & 0 \\ s{=}1 & \vdots & \ddots & \ddots & \ddots & \vdots \\ s{=}2 & \vdots & & \ddots & \ddots & 0 \\ s{=}3 & 0 & \cdots & \cdots & 0 & 1 \\ s{=}4 & \frac{1}{3} & 0 & \cdots & \cdots & 0 \end{array}$$

and

$$P\big((\cdot, i' \neq i) \mid (\cdot, i), \text{bus}\big) = \begin{array}{c|ccccc} & s'{=}0 & s'{=}1 & s'{=}2 & s'{=}3 & s'{=}4 \\ \hline s{=}0 & 0 & \cdots & \cdots & \cdots & 0 \\ s{=}1 & \vdots & & & & \vdots \\ s{=}2 & \vdots & & & & \vdots \\ s{=}3 & 0 & \cdots & \cdots & \cdots & 0 \\ s{=}4 & \frac{1}{3} & 0 & \cdots & \cdots & 0 \end{array}$$

for all traffic intensities $i \in \{1, 2, 3\}$.

If at stop $s \neq 4$, the agent decides to disembark from the bus and ride the bike to the final stop, the traffic intensity $i$ does not change, but the bus stop changes directly from $s$ to $s = 4$. Again, when the last stop is reached, the problem resets probabilistically by placing the agent at bus stop 0 independently of the selected action, and randomly selecting a new traffic intensity $i$ with equal probability. Visually, this can be illustrated with a state diagram for the bike action, which is given in Fig. 5.10 on p. 85. Thus, the transition probability matrices for disembarking from the bus and riding the bike are

$$P\big((\cdot, i) \mid (\cdot, i), \text{bike}\big) = \begin{array}{c|ccccc|} & s'{=}0 & s'{=}1 & s'{=}2 & s'{=}3 & s'{=}4 \\ s{=}0 & 0 & \cdots & \cdots & 0 & 1 \\ s{=}1 & \vdots & & & \vdots & \vdots \\ s{=}2 & \vdots & & & \vdots & \vdots \\ s{=}3 & 0 & \cdots & \cdots & 0 & 1 \\ s{=}4 & \frac{1}{3} & 0 & \cdots & \cdots & 0 \end{array}$$

and

$$P\big((\cdot, i' \neq i) \mid (\cdot, i), \text{bike}\big) = \begin{array}{c|ccccc|} & s'{=}0 & s'{=}1 & s'{=}2 & s'{=}3 & s'{=}4 \\ s{=}0 & 0 & \cdots & \cdots & \cdots & 0 \\ s{=}1 & \vdots & & & & \vdots \\ s{=}2 & \vdots & & & & \vdots \\ s{=}3 & 0 & \cdots & \cdots & \cdots & 0 \\ s{=}4 & \frac{1}{3} & 0 & \cdots & \cdots & 0 \end{array}$$

for all traffic intensities $i \in \{1, 2, 3\}$.

If the agent continues to ride the bus, the sojourn time distributions $f\big(\tau \mid (s, i), \text{bus}, (s', i)\big)$ follow an inverse Gaussian distribution (Eq. (4.2.3)) with parameters conditional on the bus stops and traffic intensity given by

$$\mu\big((\cdot, 1), \text{bus}, (\cdot, 1)\big) = \begin{array}{c|ccccc|} & s'{=}0 & s'{=}1 & s'{=}2 & s'{=}3 & s'{=}4 \\ s{=}0 & 0 & 1 & 0 & 0 & 0 \\ s{=}1 & 0 & 0 & 2 & 0 & 0 \\ s{=}2 & 0 & 0 & 0 & 1 & 0 \\ s{=}3 & 0 & 0 & 0 & 0 & 2 \\ s{=}4 & 455 & 0 & 0 & 0 & 0 \end{array} \qquad \begin{array}{c}(i = 1) \\ \text{low intensity}\end{array} \tag{5.12.1}$$

$$\mu\big((\cdot, 2), \text{bus}, (\cdot, 2)\big) = \begin{array}{c|ccccc|} & s'{=}0 & s'{=}1 & s'{=}2 & s'{=}3 & s'{=}4 \\ s{=}0 & 0 & 3 & 0 & 0 & 0 \\ s{=}1 & 0 & 0 & 3 & 0 & 0 \\ s{=}2 & 0 & 0 & 0 & 4 & 0 \\ s{=}3 & 0 & 0 & 0 & 0 & 5 \\ s{=}4 & 455 & 0 & 0 & 0 & 0 \end{array} \qquad \begin{array}{c}(i = 2) \\ \text{medium intensity}\end{array} \tag{5.12.2}$$

$$\mu\big((\cdot, 3), \text{bus}, (\cdot, 3)\big) = \begin{array}{c|ccccc|} & s'{=}0 & s'{=}1 & s'{=}2 & s'{=}3 & s'{=}4 \\ s{=}0 & 0 & 5 & 0 & 0 & 0 \\ s{=}1 & 0 & 0 & 5 & 0 & 0 \\ s{=}2 & 0 & 0 & 0 & 15 & 0 \\ s{=}3 & 0 & 0 & 0 & 0 & 20 \\ s{=}4 & 455 & 0 & 0 & 0 & 0 \end{array} \qquad \begin{array}{c}(i = 3) \\ \text{high intensity}\end{array} \tag{5.12.3}$$

$$\mu\big((s, i), \text{bus}, (s', i' \neq i)\big) = 0 \tag{5.12.4}$$

and

$$\lambda\big((s, i), \text{bus}, (s', i)\big) = \mu\big((s, i), \text{bus}, (s', i)\big)^2.$$

The graphs of the sojourn time distributions for riding the bus from stop to stop at each traffic intensity are shown in Fig. 5.11. We can see that the time it takes from stop to stop in low traffic is relatively short compared to the long times between stops

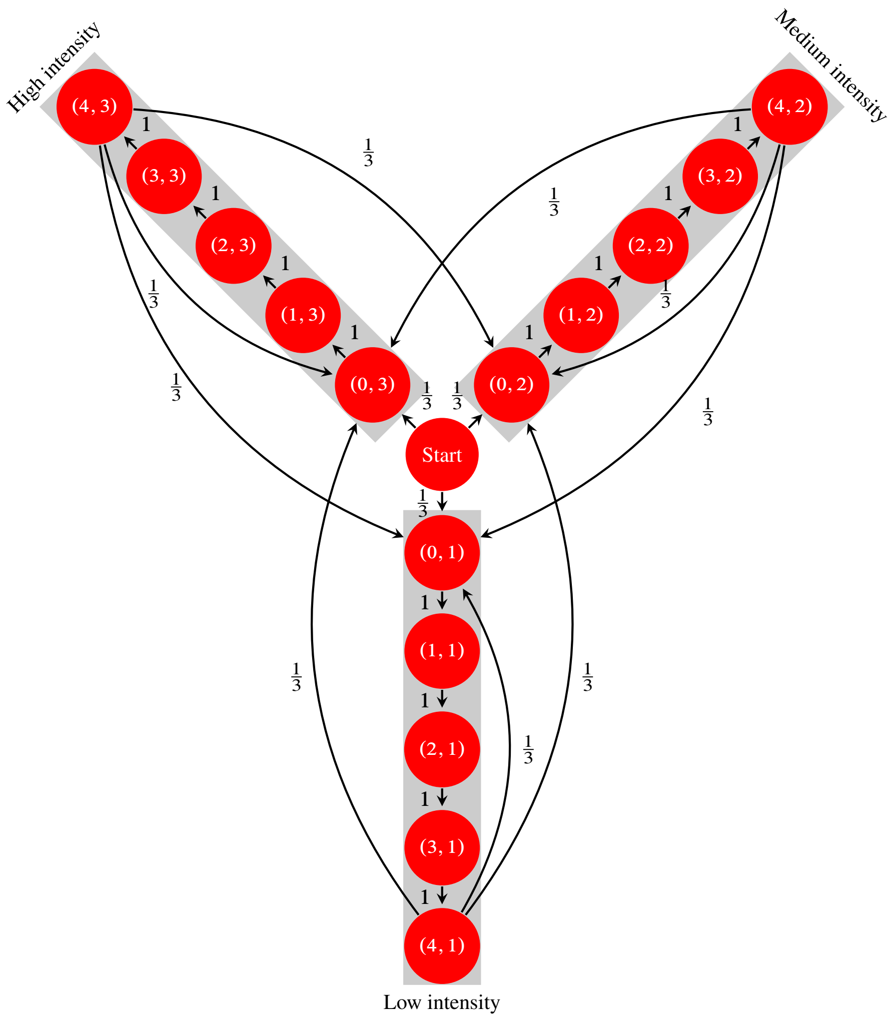


Fig. 5.9 State transition diagram for the agent's selection of action $a$ = bus in Example 5.12.1. The problem begins at stop $s = 0$ with a randomly selected initial intensity $i$. The bus action allows the agent to travel from one bus stop to the next while the traffic intensity remains constant until it reaches the end of the bus line. At this point, the agent is returned to the beginning of the bus line for a new randomly selected intensity.

in high traffic; the mean total time is 20 minutes in low traffic whereas the mean total time in high traffic is 105 minutes. Note that technically we cannot have $\mu = 0$ in the matrices because the inverse Gaussian distribution is defined for $\tau > 0$. This is not an issue since the corresponding probability $P(s' \mid s, a) = 0$, hence, $Q(\tau, s' \mid s, a) = 0$.

If the agent decides to disembark from the bus and use its bicycle, then the sojourn time distribution follows a deterministic function $f\left(\tau \mid (s,i), \text{bike}, (s',i)\right)$. If $c_0$ is a constant, then the probability mass function is

$$f\left(\tau \mid c_0\big((s,i), \text{bike}, (s',i)\big)\right) = \begin{cases} 1, & \text{if } \tau = c_0\big((s,i), \text{bike}, (s',i)\big); \\ 0, & \text{otherwise,} \end{cases}$$

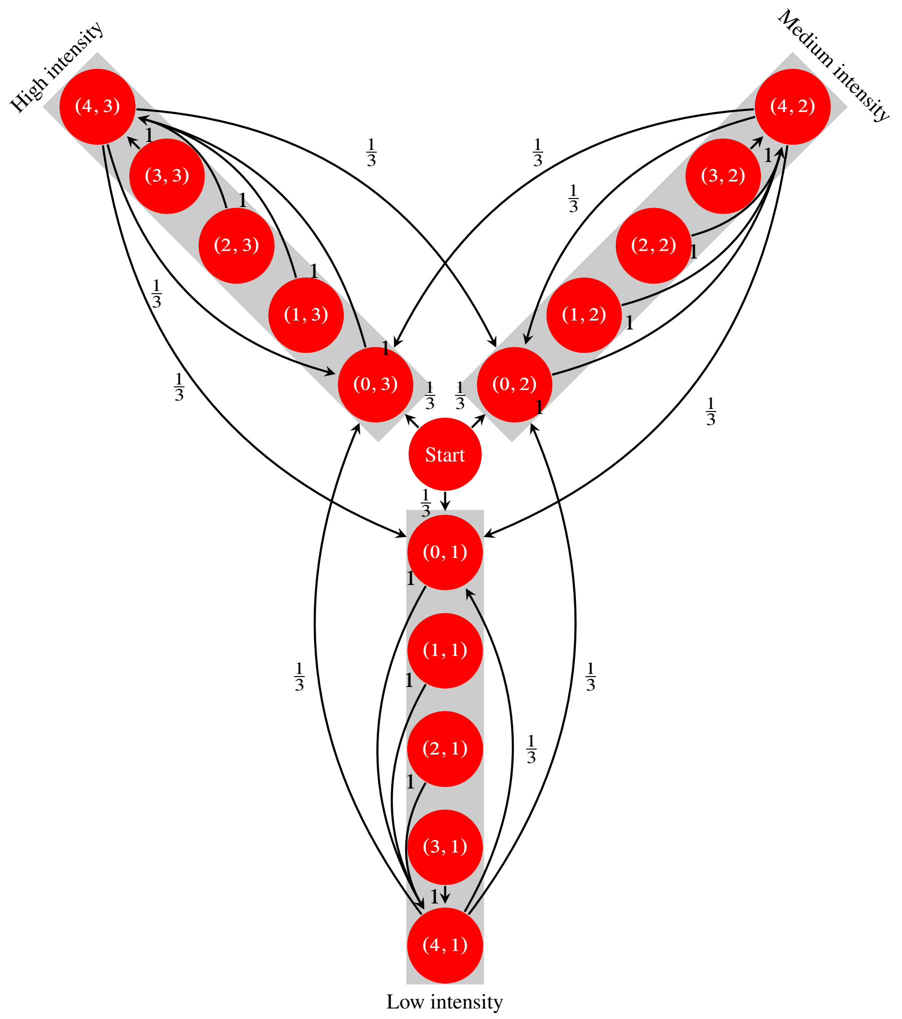


Fig. 5.10 State transition diagram for the agent's selection of action $a$ = bike in Example 5.12.1. Riding the bike leads directly to the last stop, at which point, it is returned to the beginning of the bus line for a new randomly selected intensity.

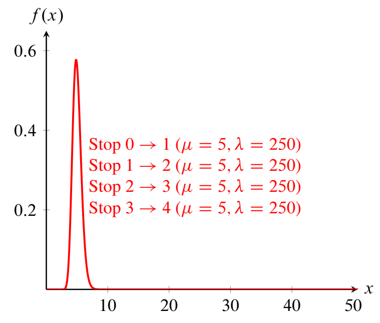
where

$$c_0\big((\cdot, i), \text{bike}, (\cdot, i)\big) = \begin{array}{c|ccccc|} & s'{=}0 & s'{=}1 & s'{=}2 & s'{=}3 & s'{=}4 \\ s{=}0 & 0 & \cdots & \cdots & 0 & 30 \\ s{=}1 & \vdots & & & \vdots & 25 \\ s{=}2 & \vdots & & & \vdots & 20 \\ s{=}3 & 0 & \cdots & \cdots & 0 & 12 \\ s{=}4 & 455 & 0 & \cdots & \cdots & 0 \end{array} \qquad (5.12.5)$$

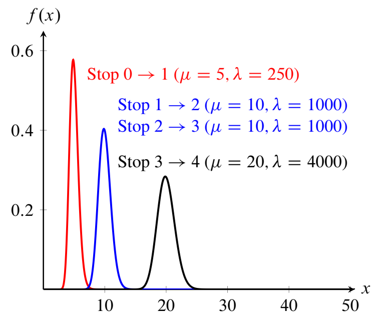
and

$$c_0\big((s, i), \text{bike}, (s', i' \neq i)\big) = 0.$$

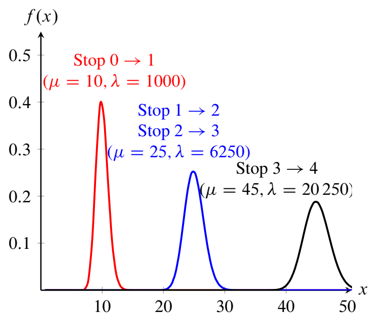
Since CHRONOSPERSEUS uses an infinite planning horizon, we need to ensure rewards from one trip (episode) from $s = 0$ to $s = 4$ does not the affect the value for the next trip (episode), which creates effectively a large discount when the environment resets probabilistically. This technique turns this episodic task into a non-episodic task, allowing CHRONOSPERSEUS to work seamlessly for episodic and non-episodic tasks. We assume continuous-time discounting at a rate of $\beta = 0.02$. This means



(a) Low traffic intensity



(b) Medium traffic intensity



(c) High traffic intensity

Fig. 5.11 The sojourn time distributions for the action $a$ = bus.

that the present value of one reward unit received at time $\tau$ units in the future equals $\mathrm{e}^{-0.02\tau} = \gamma$. To get a discount factor of $\gamma = 0.0001$, we set $\tau = 455$ between $s = 4$ to $s = 0$; this is the entry for $\mu\big((4, \cdot), \text{bus}, (0, \cdot)\big)$ in (5.12.1)–(5.12.3) and $c_0\big((4, \cdot), \text{bike}, (0, \cdot)\big)$ in Eq. (5.12.5).

The cumulative distribution function is

$$F\left(\tau \mid c_0\big((s,i), \text{bike}, (s',i)\big)\right) = \begin{cases} 0, & \text{if } \tau < c_0\big((s,i), \text{bike}, (s',i)\big); \\ 1, & \text{if } \tau \geq c_0\big((s,i), \text{bike}, (s',i)\big). \end{cases}$$

The sojourn time-state transition function is defined by

$$\begin{aligned} Q\big(\tau, (s',i) \mid (s,i), a\big) &= P\big((s',i) \mid (s,i), a\big) F\big(\tau \mid (s,i), a, (s',i)\big) \\ &= P\big((s',i) \mid (s,i), a\big) \int_0^\tau f\big(t \mid (s,i), a, (s',i)\big)\, \mathrm{d}t \end{aligned}$$

Given the landing state $(s', i)$, regardless of which action $a$ was selected, the agent with certainty would observe bus stop $s'$. Remember, it is the traffic intensity $i$ that is hidden from the agent, but this does not influence what bus stop $s'$ is observed; it is deterministic what bus stop is observed depending on whether the agent decides to continue to ride the bus or take the bicycle. This leads to the observation probability matrix given by

$$G\big(\cdot \mid a, (\cdot, i)\big) = \begin{array}{c} \\ s'{=}0 \\ s'{=}1 \\ s'{=}2 \\ s'{=}3 \\ s'{=}4 \end{array} \begin{array}{c} \begin{array}{ccccc} o{=}0 & o{=}1 & o{=}2 & o{=}3 & o{=}4 \end{array} \\ \begin{bmatrix} 1 & 0 & \cdots & \cdots & 0 \\ 0 & \ddots & \ddots & & \vdots \\ \vdots & \ddots & \ddots & \ddots & \vdots \\ \vdots & & \ddots & \ddots & 0 \\ 0 & \cdots & \cdots & 0 & 1 \end{bmatrix} \end{array} \qquad \forall i \in \{1, 2, 3\}.$$

Now, we will setup the reward function. Set the lump sum reward $r_1\big((s,i),a\big) = 0$ and continuous reward rate as $r_2\big((s,i),a,(s',i')\big) = -1$ in Eq. (4.3.4) (on p. 47) . Using Examples 4.4.1 (on p. 48) and 4.4.2 (on p. 48) for the Laplace transform of the inverse Gaussian distribution

$$\int_0^\infty \mathrm{e}^{-\beta\tau} f\big(\tau \mid (s,i),a,(s',i')\big)\,\mathrm{d}\tau,$$

the reward function is

$$R\big((s,i),a\big) = -\frac{1}{\beta}\sum_{(s',\,i')\in\mathcal{S}} P\big((s',i') \mid (s,i),a\big)\Big(1 - M\big((s,i),a,(s',i')\big)\Big)$$

where

$$M\big((s,i),a,(s',i')\big) = \begin{cases} \exp\left[-\beta c_0\big((s,i),\text{bike},(s',i)\big)\right], & \text{if } a = \text{bike};\\ \exp\left[\frac{\lambda(s,\text{bus},s')}{\mu(s,\text{bus},s')}\left(1-\sqrt{1+\frac{2\mu(s,\text{bus},s')^2\beta}{\lambda(s,\text{bus},s')}}\right)\right], & \text{if } a = \text{bus}. \end{cases} \tag{5.12.6}$$

The lump sum reward is

$$r_1\big((4,i),a\big) = \begin{cases} 100, & \forall i \in \{1,2,3\}, a \in \mathcal{A};\\ 0, & \text{otherwise,} \end{cases} \tag{5.12.7}$$

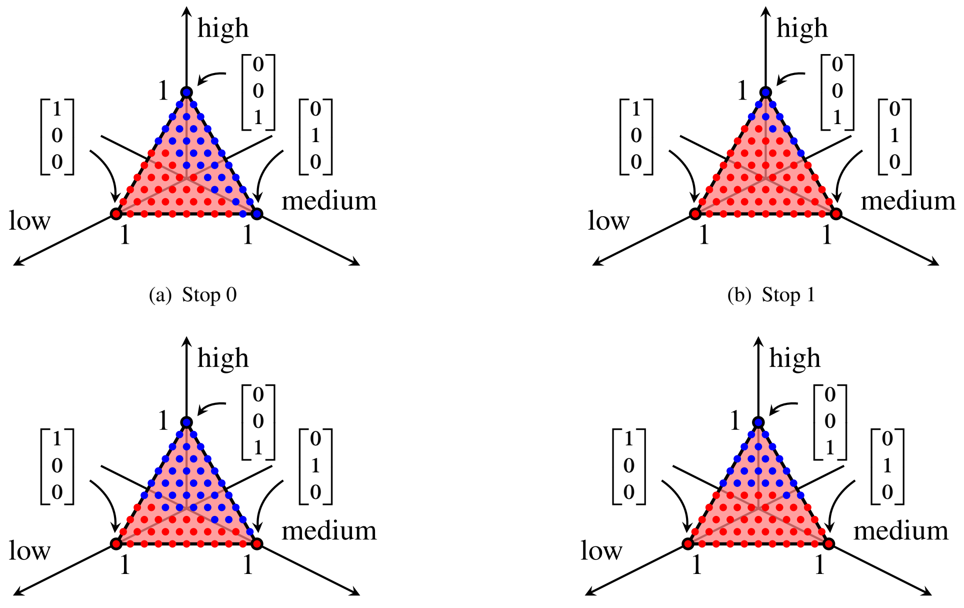
and the continuous reward rate is

$$r_2(\cdot,\cdot,\cdot) = 0. \tag{5.12.8}$$

The agent begins at bus stop 0, and since this is fully observable, but the traffic level intensity is not, the agent assumes that the three traffic intensities are equally likely; that is,

$$\xi_0\big((\cdot,i)\big) = \begin{array}{c} s{=}0\\ s{=}1\\ s{=}2\\ s{=}3\\ s{=}4 \end{array}\begin{bmatrix} \frac{1}{3}\\ 0\\ 0\\ 0\\ 0 \end{bmatrix} \qquad \forall i \in \{1,2,3\}.$$

The resulting policy using ChronosPerseus is depicted in Fig. 5.12. Figure 5.12(a) reveals that if the agent possesses sufficient evidence to infer the traffic was medium or high, it is more advantageous to ride the bicycle (with an expected time of 30 minutes) than to remain on the bus (expected times of 45 and 105 minutes, respectively). In the absence of initial information, the optimal strategy is to stay on the bus for at least one stop (expected time of 5 minutes) before considering disembarking (the exact centre of the simplex would be a red dot for the bus action). Upon reaching the first stop (Fig. 5.12(b)), medium and high traffic intensities diverge significantly in remaining expected time, making the bicycle a more appealing option solely under high traffic. The expected remaining time is 25 minutes for the bicycle, and 15, 40, and 95 minutes for the low, medium, and high traffic, respectively. However, the bicycle action must be weighed against the average of the three traffic intensity's expected discounting weighted by the agent's belief. Figures 5.12(c) and 5.12(d) present the optimal action at stops 2 and 3, given the current belief about the traffic intensities. It is important to note that the action decision boundary may appear different due to the consideration of belief-weighted exponential discounting over stochastic inverse-Gaussian distributed sojourn time instead of assuming a negative reward per time unit with no discounting. ◀



(a) Stop 0 (b) Stop 1

(c) Stop 2 (d) Stop 3

Fig. 5.12 Optimal policy for the bus problem at each bus stop, illustrated using a regular mesh of belief states representing the agent's confidence in traffic intensity. A red dot ● indicates that the optimal action for the corresponding belief is to continue riding the bus, while a blue dot ● signifies that the optimal action for the respective belief is to disembark from the bus and ride the bicycle.

In the previous example, we demonstrated the capability of CHRONOSPERSEUS to efficiently solve a problem involving time without expanding the state space to account for elapsed time. Such an expansion would have required a state space featuring an elapsed time dimension of no less than $5 \times 3 \times T$, where $T \geq 125$, for a single state per time unit. Instead, we have seamlessly integrated various durations into sojourn time distributions, permitting the belief to be updated directly in accordance with the distribution's likelihood given the observed transitions.

Using CHRONOSPERSEUS, we see that POSMDPs are not more complicated to solve than their POMDP counterparts. Instead, like using SMDPs in options (Sutton *et al.*, 1999; Precup, 2000), it allows for the compression of the temporal properties of the problem. This simplifies not only the creation of the model, but also its evaluation by avoiding the explosion of the state space. This approach also allows us to easily mix various temporal distributions and solve episodic problems as efficiently as non-episodic problems.

The following example underscores the adaptability of CHRONOSPERSEUS in managing continuous observations. As examined by Zhang and Revie (2017), an industrial application of POSMDPs was employed to maintain rapid gravity filters within municipal drinking water treatment plants. These filters are crucial in purifying water and providing communities with safe and clean drinking water. The maintenance process of these filters encounters multiple challenges, such as the inability to directly observe a filter's condition and the necessity for efficient maintenance planning. Accurate determination of the condition demands thorough inspections, which are time-consuming and temporarily disrupt the functionality of the filters.

**Example 5.12.2 — Filter Maintenance.** Filters can be classified into four states

$$\mathcal{S} = \{1 = \text{good}, 2 = \text{acceptable}, 3 = \text{poor}, 4 = \text{awful}\}.$$

Unlike in Example 5.12.1, the state of the filter is (fully) hidden and it must be inferred with uncertain observations. To extend the longevity and reliability of the filters, there are four maintenance actions, which are

$$\mathcal{A} = \{1 = \text{do nothing}, 2 = \text{backwash}, 3 = \text{dose chemicals}, 4 = \text{replace}\}.$$

An observation to infer the state of the filter is to measure turbidity. Turbidity is the amount of cloudiness or haziness of a fluid caused by suspended particles: this is measured by a nephelometer, which shines a light beam through the water sample and measures how much light is reflected into a detector (often located at 90° from the source beam). The turbidity of the incoming water into the filter and the turbidity of the outgoing water from the filter are measured. If the outgoing to incoming water turbidity ratio is close to zero, then the filter is in a good state. However, if the ratio is closer to one, the filter is likely to be in a poor or awful state. Hence, the observation space is given by the continuous interval $\mathcal{O} = [0, 1]$.

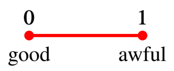


Fig. 5.13 The observation space serves as a scale of the current state of the filter.

The probability transition matrices are

$$P(\cdot \mid \cdot, 1) = P(\cdot \mid \cdot, 2) = \begin{array}{c} \\ s{=}1 \\ s{=}2 \\ s{=}3 \\ s{=}4 \end{array} \begin{array}{c} \begin{array}{cccc} s'{=}1 & s'{=}2 & s'{=}3 & s'{=}4 \end{array} \\ \begin{bmatrix} 0.1043 & 0.7413 & 0.1493 & 0.0051 \\ 0 & 0.1043 & 0.7413 & 0.1544 \\ 0 & 0 & 0.1043 & 0.8957 \\ 0 & 0 & 0 & 1 \end{bmatrix} \end{array},$$

$$P(\cdot \mid \cdot, 3) = \begin{array}{c} \\ s{=}1 \\ s{=}2 \\ s{=}3 \\ s{=}4 \end{array} \begin{array}{c} \begin{array}{cccc} s'{=}1 & s'{=}2 & s'{=}3 & s'{=}4 \end{array} \\ \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0.50 & 0.50 & 0 & 0 \\ 0.25 & 0.70 & 0.05 & 0 \\ 0.20 & 0.55 & 0.20 & 0.05 \end{bmatrix} \end{array},$$

and

$$P(\cdot \mid \cdot, 4) = \begin{array}{c} \\ s{=}1 \\ s{=}2 \\ s{=}3 \\ s{=}4 \end{array} \begin{array}{c} \begin{array}{cccc} s'{=}1 & s'{=}2 & s'{=}3 & s'{=}4 \end{array} \\ \begin{bmatrix} 1 & 0 & 0 & 0 \\ 1 & 0 & 0 & 0 \\ 1 & 0 & 0 & 0 \\ 1 & 0 & 0 & 0 \end{bmatrix} \end{array}.$$

The sojourn times for actions 1, 2, and 3 are fixed, whereas $\tau(s' \mid s, 1) = 78.7433$, $\tau(s' \mid s, 2) = 85.3052$, and $\tau(s' \mid s, 3) = 3$. The sojourn time for action 4 follows a truncated Gaussian distribution with mean $\mu = 10$, standard deviation $\sigma = 1.5$, and $\tau(s' \mid s, 4) > 0$; in other words,

$$f(\tau \mid s, a = 4, s') = \begin{cases} \dfrac{1}{1.5} \cdot \dfrac{\varphi\left(\dfrac{\tau - 10}{1.5}\right)}{1 - \Phi\left(-\dfrac{10}{1.5}\right)}, & \text{if } \tau > 0; \\ 0, & \text{otherwise,} \end{cases} \tag{5.12.9}$$

where

$$\varphi(x) = \frac{1}{\sqrt{2\pi}} \exp\left(-\frac{1}{2}x^2\right) \tag{5.12.10}$$

is the standard normal probability density function, and $\Phi$ is its corresponding cumulative distribution function defined by Eq. (4.2.16).

To deal with continuous observations, we discretize the interval $[0, 1]$ into $j$ evenly spaced numbers, which yields a finite set of observations $O$ ($|O| = j$). The observation transition probability is

$$G(o \mid a, s') = G(o \mid a) = \frac{f_B\big(o \mid \phi(a), \eta(a)\big)}{\sum_{o' \in O} f_B\big(o' \mid \phi(a), \eta(a)\big)}$$

where $f_B\big(o \mid \phi(a), \eta(a)\big)$ is the beta probability density function defined by

$$f_B\big(o \mid \phi(a), \eta(a)\big) = \frac{\Gamma\big(\phi(a) + \eta(a)\big)}{\Gamma\big(\phi(a)\big)\Gamma\big(\eta(a)\big)} o^{\phi(a)-1}(1-o)^{\eta(a)-1}, \tag{5.12.11}$$

$\Gamma(\cdot)$ is the gamma function, and the parameters are

$$\phi(\cdot) = \begin{array}{c} a{=}1 \\ a{=}2 \\ a{=}3 \\ a{=}4 \end{array} \begin{bmatrix} 2 \\ 6 \\ 18 \\ 18 \end{bmatrix} \qquad \text{and} \qquad \eta(\cdot) = \begin{array}{c} a{=}1 \\ a{=}2 \\ a{=}3 \\ a{=}4 \end{array} \begin{bmatrix} 18 \\ 18 \\ 18 \\ 6 \end{bmatrix}.$$

The beta probability density functions are shown in Fig. 5.14.

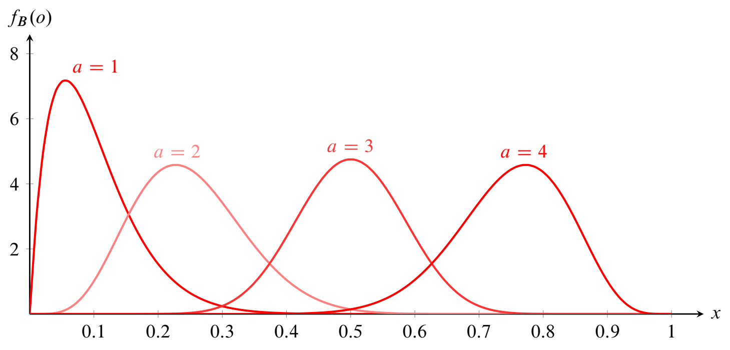


Fig. 5.14 The beta probability density functions used for the observation transition probabilities in Example 5.12.2.

The lump sum reward is

$$r_1(s, \cdot) = \begin{array}{c} a{=}1 \\ a{=}2 \\ a{=}3 \\ a{=}4 \end{array} \begin{bmatrix} 0 \\ -100 \\ -200 \\ -500 \end{bmatrix} \qquad \forall s \in \mathcal{S},$$

and the continuous reward rate is

$$r_2(\cdot, \cdot, s') = \begin{array}{c|cccc} & a{=}1 & a{=}2 & a{=}3 & a{=}4 \\ \hline s{=}1 & 500 & 500 & -100 & -100 \\ s{=}2 & 250 & 250 & -100 & -100 \\ s{=}3 & -300 & -300 & -100 & -100 \\ s{=}4 & -500 & -500 & -100 & -100 \end{array} \qquad \forall s' \in \mathcal{S}.$$

The initial $\alpha$-vector is

$$\alpha(\cdot) = \begin{array}{c} s{=}1 \\ s{=}2 \\ s{=}3 \\ s{=}4 \end{array} \begin{bmatrix} -10^6 \\ -10^6 \\ -10^6 \\ -10^6 \end{bmatrix}.$$

The initial belief $\xi_0$ is that the filter is good; that is,

$$\xi_0(\cdot) = \begin{array}{c} s{=}1 \\ s{=}2 \\ s{=}3 \\ s{=}4 \end{array} \begin{bmatrix} 1 \\ 0 \\ 0 \\ 0 \end{bmatrix}.$$

Running ChronosPerseus on the problem with discount rate $\beta = 0.01$ generates the set

$$V = \left\{ \begin{array}{ccccc} & a{=}\text{backwash} & a{=}\text{dose} & a{=}\text{dose} & a{=}\text{dose} \\ \begin{array}{r} V(\text{good}) \\ V(\text{acceptable}) \\ V(\text{poor}) \\ V(\text{awful}) \end{array} & \begin{bmatrix} 46351.3242 \\ 31551.5723 \\ -71.0176 \\ -11550.9883 \end{bmatrix}, & \begin{bmatrix} 43257.2109 \\ 41467.2891 \\ 40529.1953 \\ 40172.0430 \end{bmatrix}, & \begin{bmatrix} 43404.0859 \\ 41505.2148 \\ 40516.1875 \\ 40159.6211 \end{bmatrix}, & \begin{bmatrix} 43690.0625 \\ 41519.4531 \\ 40405.4766 \\ 40057.0312 \end{bmatrix}, \end{array} \right.$$

$$\begin{array}{ccccc} & a{=}\text{dose} & a{=}\text{dose} & a{=}\text{dose} & a{=}\text{dose} \\ \begin{array}{r} V(\text{good}) \\ V(\text{acceptable}) \\ V(\text{poor}) \\ V(\text{awful}) \end{array} & \begin{bmatrix} 43987.1328 \\ 41352.2109 \\ 40036.6328 \\ 39764.9141 \end{bmatrix}, & \begin{bmatrix} 43567.4844 \\ 41526.0000 \\ 40470.0547 \\ 40112.9766 \end{bmatrix}, & \begin{bmatrix} 44235.8906 \\ 40804.3320 \\ 39155.7461 \\ 39067.2930 \end{bmatrix}, & \begin{bmatrix} 43890.7227 \\ 41438.7500 \\ 40201.9414 \\ 39896.9844 \end{bmatrix}, \end{array}$$

$$\begin{array}{ccccc} & a{=}\text{dose} & a{=}\text{dose} & a{=}\text{dose} & a{=}\text{dose} \\ \begin{array}{r} V(\text{good}) \\ V(\text{acceptable}) \\ V(\text{poor}) \\ V(\text{awful}) \end{array} & \begin{bmatrix} 44127.5703 \\ 41123.7617 \\ 39652.8359 \\ 39461.3516 \end{bmatrix}, & \begin{bmatrix} 44061.6992 \\ 41249.8320 \\ 39859.4453 \\ 39625.1172 \end{bmatrix}, & \begin{bmatrix} 43800.3984 \\ 41489.0391 \\ 40313.0742 \\ 39984.2852 \end{bmatrix}, & \begin{bmatrix} 44187.7969 \\ 40967.9141 \\ 39406.7695 \\ 39265.6484 \end{bmatrix}, \end{array}$$

$$\left. \begin{array}{cc} & a{=}\text{replace} \\ \begin{array}{r} V(\text{good}) \\ V(\text{acceptable}) \\ V(\text{poor}) \\ V(\text{awful}) \end{array} & \begin{bmatrix} 40504.4414 \\ 40504.4414 \\ 40504.4414 \\ 40504.4414 \end{bmatrix} \end{array} \right\},$$

where $V$ comprises 13 $\alpha$-vectors with their corresponding actions labelled at the top, while the states are labelled along the rows. Zhang and Revie (2017) utilized $|B| = 5000$ sampled belief points and 40 iterations, taking approximately 16 hours to complete the calculations on an Intel Core i5-4590 CPU at 3.30 GHz. In contrast, ChronosPerseus, using the same number of belief points and iterations, finished the computation in under 40 seconds on an Intel Core Xeon Silver 4210 CPU at 2.20 GHz and Nvidia GeForce RTX 2080 Super 8 GB. The original paper did not specify the number of observation bins, so we chose 100 for our simulation.

Table 5.3 (on p. 94) presents a selection of belief points and their corresponding optimal values and actions, comparing the results from ChronosPerseus to those of Zhang and Revie. The beliefs in Table 5.3 were chosen by Zhang and Revie and were also included in the sampled belief set $B$ used by ChronosPerseus. The column begins with beliefs corresponding to a filter believed to be in good condition and proceeds to acceptable, poor, and finally, awful condition. We observe that ChronosPerseus opts to backwash for a filter believed to be in good condition, while Zhang and Revie recommends do-nothing action. The accumulated reward for backwashing is greater than that for doing nothing: $R(1,1) = 27249.43$, while $R(1,2) = 28594.38$. The additional time during the backwash allows for the accumulation of continuous rewards, offsetting the initial negative reward of $-100$ from backwashing compared to the time allowed for the do-nothing action. The accumulated reward must account for the risk of producing poor water quality, such as reaching the poor or awful states. As a result, we believe that CHRONOSPERSEUS has selected the appropriate action. ◀

It appears that Zhang and Revie (2017) may have selected the truncated Gaussian distribution due to its support on $(0, \infty)$. The inverse Gaussian distribution could be used as an alternative to the truncated Gaussian distribution for modelling the lifetime of the rapid gravity filter. The advantage of using the inverse Gaussian is that the distribution addresses a broader class of lifetime distributions, and the physical interpretation of the first passage time distribution leads to its natural application to studying lifetime (Chhikara and Folks, 1977, p. 467).

We showed with these examples that CHRONOSPERSEUS can address real-world POSMDP problems with continuous observation spaces. This algorithm, when coupled with importance sampling and GPU implementation, significantly accelerates computations by several orders of magnitude. Such enhancements permit more iterations, extending the planning horizon. The adjustment of CHRONOSPERSEUS to handle continuous observations, as exemplified by the turbidity measurements in filter maintenance scenarios, substantiates its potential in optimizing maintenance planning and managing associated challenges. The incorporation of importance sampling for time variables in CHRONOSPERSEUS further streamlines the computation process, minimizing the duration required to determine an optimal policy from hours to just a few seconds.

## 5.13 Conclusion

The operations research community has extensively utilized the partially observable semi-Markov decision process (POSMDP) for decades to facilitate planning under uncertainty, as demonstrated in diverse tasks such as maintenance scheduling. However, the potential of POSMDP extends beyond traditional operations research and it is equally valuable in the field of artificial intelligence. The POSMDP model can capture not only the agent's interaction with the environment but also the temporal aspect of the system, specifically addressing the challenge of how long it takes for the agent to transition between hidden states.

To this end, we developed a novel POSMDP solver—CHRONOSPERSEUS—that combines PERSEUS and importance sampling to efficiently solve a POSMDP where the transition time is observable. The solver can handle various problem types, such as episodic and non-episodic, with mixed-observable, discrete, or continuous observation space, and a mixture of fixed and stochastic continuous sojourn times.

Furthermore, we demonstrated the effectiveness of the solver by showing how it can learn a policy on a POSMDP where time is the only available information to resolve the partially observable state. This scenario is prevalent in many real-world problems, and our proposed solver provides significant advantages in decision-making and planning under uncertainty.

Table 5.2 Monte Carlo integration of $\int_0^\infty \mathrm{e}^{-\beta x} f(\tau \mid \mu, \lambda)\, \mathrm{d}\tau$, where (1) $\beta = 0.3$, $\mu = 3$, $\lambda = 9$ and (2) $\beta = 0.3$, $\mu = 5$, and $\lambda = 25$

| $n$ | $z_n \sim \mathcal{N}(0,1)$ | $y_n \sim \mathcal{U}(0,1)$ | $\tau_n \sim \mathcal{IG}(3,9)$ | $\tau_n \sim \mathcal{IG}(5,25)$ | $f(\tau_n \mid 3,9)$ | $f(\tau_n \mid 5,25)$ | $D(\tau_n)$ | $\frac{\mathrm{e}^{-\beta\tau_n} f(\tau_n \mid 3,9)}{D(\tau_n)}$ | $\frac{\mathrm{e}^{-\beta\tau_n} f(\tau_n \mid 5,25)}{D(\tau_n)}$ |
|---|---|---|---|---|---|---|---|---|---|
| 1 | −1.276 | 0.1009 | 1.4588 | | 0.3009 | 0.0154 | 0.2295 | 0.8463 | 0.0433 |
| 2 | −1.218 | 0.7325 | 5.9780 | | 0.0390 | 0.1260 | 0.0607 | 0.1068 | 0.3451 |
| 3 | −0.453 | 0.3376 | 2.3113 | | 0.3074 | 0.1188 | 0.2602 | 0.5904 | 0.2282 |
| 4 | −0.350 | 0.5201 | 2.4519 | | 0.2932 | 0.1382 | 0.2545 | 0.5522 | 0.2603 |
| 5 | 0.723 | 0.3586 | 1.9821 | | 0.3302 | 0.0718 | 0.2656 | 0.6859 | 0.1492 |
| 6 | 0.676 | 0.3467 | 2.0355 | | 0.3279 | 0.0793 | 0.2658 | 0.6700 | 0.1620 |
| 7 | −1.099 | 0.3548 | 1.6069 | | 0.3212 | 0.0272 | 0.2477 | 0.8007 | 0.0679 |
| 8 | −0.314 | 0.7680 | 3.5954 | | 0.1671 | 0.2224 | 0.1809 | 0.3141 | 0.4180 |
| 9 | −0.394 | 0.9590 | 3.7644 | | 0.1516 | 0.2230 | 0.1695 | 0.2892 | 0.4253 |
| 10 | −0.633 | 0.9117 | 4.3149 | | 0.1093 | 0.2108 | 0.1347 | 0.2224 | 0.4289 |
| 11 | −0.318 | 0.3929 | 2.4975 | | 0.2883 | 0.1442 | 0.2523 | 0.5402 | 0.2703 |
| 12 | −0.799 | 0.2749 | 1.8990 | | 0.3324 | 0.0606 | 0.2644 | 0.7111 | 0.1296 |
| 13 | −1.664 | 0.4537 | | 2.4144 | 0.2971 | 0.1332 | 0.2562 | 0.5622 | 0.2520 |
| 14 | 1.391 | 0.5420 | | 2.7101 | 0.2641 | 0.1699 | 0.2406 | 0.4869 | 0.3133 |
| 15 | 0.382 | 0.4805 | | 4.2157 | 0.1160 | 0.2142 | 0.1406 | 0.2330 | 0.4302 |
| 16 | 0.733 | 0.6489 | | 6.9296 | 0.0215 | 0.0836 | 0.0370 | 0.0727 | 0.2822 |
| Average | −0.288 | 0.5114 | 2.8246 | 4.0675 | 0.2292 | 0.1274 | 0.2038 | 0.4803 | 0.2629 |
| True Value | 0.000 | 0.5000 | 3.0000 | 5.0000 | 0.2303 | 0.1784 | 0.2173 | 0.4517 | 0.2659 |

Table 5.3 Sample of belief points, their corresponding optimal values and actions

| | ChronosPerseus | | Zhang and Revie (2017) | |
|---|---|---|---|---|
| Belief state, $\xi$ | Optimal Value | Optimal Action | Optimal Value | Optimal Action |
| $[0.9972\ 0.0028\ 0.0000\ 0.0000]^\top$ | 46309.8867 | 2 | 46316.40 | 1 |
| $[0.9965\ 0.0035\ 0.0000\ 0.0000]^\top$ | 46299.5234 | 2 | 46306.04 | 1 |
| $[0.8714\ 0.1286\ 0.0000\ 0.0000]^\top$ | 44448.0742 | 2 | 44454.43 | 2 |
| $[0.8160\ 0.1840\ 0.0000\ 0.0000]^\top$ | 43628.1680 | 2 | 43634.51 | 2 |
| $[0.0031\ 0.6803\ 0.3165\ 0.0001]^\top$ | 41197.9805 | 3 | 41215.31 | 3 |
| $[0.0001\ 0.0390\ 0.9457\ 0.0152]^\top$ | 40560.6250 | 3 | 40574.81 | 3 |
| $[0.0000\ 0.0003\ 0.8488\ 0.1509]^\top$ | 40504.4453 | 4 | 40498.43 | 4 |
| $[0.0000\ 0.0000\ 0.0000\ 1.0000]^\top$ | 40504.4414 | 4 | 40385.84 | 4 |

# 6
# Bayes-Adaptive SMDPs

Decisions made in the face of uncertainty pervade the life of every individual and organization. Even animals might be said continually to make such decisions, and the psychological mechanisms by which men decide may have much in common with those by which animals do so.

—Savage (1954, p. 6)

This chapter explores the Bayesian approach to model-based reinforcement learning, which maintains probability distributions over unknown model parameters and updates them as new observations are received. The primary focus in this approach is the exploration versus exploitation trade-off: should the agent *explore* the environment in search of larger rewards or *exploit* its current knowledge about the location of the largest reward? The concept of *optimal learning* involves finding an equilibrium between these two crucial aspects.

Bayesian reinforcement learning (BRL) allows modelling the uncertainty of parameters in a Markov decision process (MDP) (Duff, 2002) and a partially observable Markov decision process (POMDP) (Ross *et al.*, 2008a). To achieve this, BRL models uncertainty by explicitly maintaining a probability distribution over quantities such as model parameters, the value function, or its gradient (Duff, 2002). The uncertainty is expressed through a prior distribution over unknown model parameters, and learning occurs by updating the posterior distribution based on the agent's experiences in the environment (Martin, 1967; DeGroot, 1970; Duff, 2002). For optimal learning in MDPs, the strategy considers all potential state-action-state transitions and their associated rewards. This necessitates augmenting the original state space of the MDP with counts of experienced transitions. The application of dynamic programming on this augmented state space enables the consideration of all possible trajectories and states of knowledge throughout the environment. This approach yields an optimal learning policy that balances reward acquisition and knowledge collection.

Time is critical to cognitive agents making decisions, as planning not only involves determining *what* to do, but also *when* to do it (Maniadakis and Trahanias, 2011; Rivest and Kohar, 2020). For instance, agents need a time model to decide when to give up waiting for a bus, when to perform maintenance on machinery, or when an autonomous robotic vacuum cleaner should return to its base before running out of battery life. However, in MDPs or POMDPs, these models are not concerned with how long it takes to perform an action. While the learning of sojourn times in traditional reinforcement learning has already been studied (Bradtke and Duff, 1995), BRL has overlooked this problem.

While the agent moves discretely in the state space within an MDP framework—facilitating the consideration of all possible state transitions—the scenario becomes more complex in semi-Markov decision processes (SMDPs). In SMDPs, the agent does not necessarily move discretely in time. This raises questions like, does it move in 1 second, 1.5 seconds, or 1.2 seconds? How do we consider all the possible times like we did with the state transitions?

Our goal is to extend the BRL framework from MDP to the more general SMDP, where the sojourn time is not necessarily exponentially distributed. We introduce a new mathematical model, the Bayes-adaptive semi-Markov decision process (BA-SMDP), which improves the SMDP domain knowledge through interaction with the environment and learns an optimal policy that balances the trade-off between model improvement and reward gain. The BRL approach for SMDPs offers three potential advantages:

(a) Prior knowledge can be encoded into the appropriate prior distributions to speed up learning;
(b) The risk of low reward return can be modelled and incorporated to obtain robust policies (perform well across a range of situations, even in the presence of uncertainties, variations, or changes in the environment or the underlying model); and
(c) An online-learning policy for an reinforcement learning agent that has an optimal tradeoff between exploration and exploitation of the agent's model.

The last point of solving the exploration versus exploitation problem for SMDPs in a reinforcement learning setting forms the focus of this chapter.

First, we review the principles of Bayesian learning in Sec. 6.3. In Sec. 6.6, we look at how the reinforcement community solved the exploration versus exploitation issue for case of the MDP with unknown model parameters (the state transition probabilities). Then, a contribution of this thesis, we extend Sec. 6.6 in Sec. 6.7 for an SMDP with unknown model parameters (the state transition probabilities and the sojourn time distribution). The difference between the MDP and the SMDP approach is that the SMDP requires continuous model parameters. We look at four scenarios where we can use approximations or knowledge about the model to avoid the curse of dimensionality.

## 6.1 Tracing the Roots: The Evolution of Bayesian Reinforcement Learning and Bayes-Adaptive Models

The Bayesian approach to dealing with uncertainty gained traction in the 1940s and 1950s, thanks to key contributions by Wald (1945, 1947), Arrow *et al.* (1949), and Savage (1954). Wald introduced the sequential probability ratio test, which laid the groundwork for sequential hypothesis testing, and lead to his 1947 seminal work on a general theory of statistical decisions. Next, Arrow *et al.* extended the minimax work of von Neumann (1928) in game theory to sequential decision problems, demonstrating that minimax solutions in this context aim to minimize the maximum possible cost or risk associated with a sequence of decisions, thereby applying the minimax concept to decision-making under uncertainty. While we may take the Bayesian approach for granted today, Savage's textbook was considered controversial and ignited the "neo-Bayesian revival" (Fienberg, 2006) when it was published. In *The Foundations of Statistics*, he developed a comprehensive axiomatic system for subjective probability and decision-making under uncertainty, providing a rigorous basis for Bayesian methods. This work was later expanded upon by Raiffa and Schlaifer (1961) to present a unified theory of statistical decision-making under uncertainty suitable for applications. The Kalman (1960) filter, another early example of a Bayesian method, tracks and predicts the state of a system over time. The filter's recursive nature and ability to handle uncertainty is a forerunner to the sequential decision-making framework in BRL.

While the groundbreaking work of these pioneers significantly advanced the field of Bayesian statistics, the global conflict that spurred the race for nuclear weapons and the development of modern digital computers emerged as another crucial factor. These computational advancements enabled the implementation of resource-intensive Bayesian techniques, such as the Monte Carlo Method (Metropolis and Ulam, 1949; Metropolis *et al.*, 1953), further propelling the evolution and adoption of Bayesian approaches.

The foundations of BRL can be traced back to the mid-1950s, well before the formalization of Reinforcement Learning by Sutton (1984). Bellman (1956) applied a dynamic programming approach, using what is now known as a likelihood function to quantify the uncertainty of an unknown parameter. While his focus may have been on the "sequential design of experiments," the early ideas of Markov decision processes (MDPs) were already emerging. Bellman aimed to find an optimal policy that maximized the expected (undiscounted) reward in a finite horizon while incorporating parameter information into the state space. He later formally introduced MDPs in 1957.

At the time, Bellman (1954) considered states to be physical states of the system, but later developments expanded this view. Bellman (1961b) introduced the notion of an information pattern, which in contemporary terms is known as an *information state* (which we denote by $\theta$). This concept aids in determining initially unknown parameters. By merging the physical state $s$ with the information state $\theta$, Bellman created the construct $(s, \theta)$, which has been referred to as the *hyperstate* by Duff (2002, p. 3). Bellman acknowledged that incorporating the entire history of the process as an information state was an obvious choice, but "one can do much better than this and substantially compress the vast amount of data" (Bellman, 1961b, p. 41). At this time, due to Bellman, reinforcement learning was known as adaptive control processes.

In their work on partially observable Markov decision processes (POMDPs), Aoki (1965) and Åström (1965) demonstrated that it was possible to construct a belief state space using recursive Bayesian updates to track information about an unknown physical state. This approach effectively incorporated observations to infer state probabilities without the need to include the entire history of the system in the state representation. A great summary of the work up until the 1960s is Martin (1967).

The genesis of reinforcement learning can be traced back to the work of Sutton (1988) on temporal difference learning and the development of $Q$-Learning by Watkins (1989). Both these foundational techniques were primarily concerned with learning point estimates of parameters. The introduction of Bayesian thinking into the realm of reinforcement learning, however, did not occur until Dearden *et al.* (1998) presented Bayesian $Q$-Learning. This adaptation addressed the conundrum of exploration versus exploitation by maintaining a probability distribution over $Q$-values to guide action selection. For a comprehensive review of Bayesian Reinforcement Learning (BRL), we direct the reader to Ghavamzadeh *et al.* (2015).

In the present work, our primary interest lies in the contributions of Martin (1967) and Duff (2002). These researchers focused on MDPs with unknown state transition probabilities and incorporated probability distributions on the parameters to record the uncertainty as evidence accumulated. This conceptualization was subsequently expanded to the POMDP context by Ross *et al.* (2008a, 2011).

## 6.2 The Exploration versus Exploitation Dilemma

As mentioned in the opening of the chapter, the exploration versus exploitation dilemma revolves around the agent deciding if it should be greedy and collect the reward it knows about already, or search the environment for an even greater reward. The $\epsilon$-greedy schedule is a prevalent heuristic employed by the reinforcement learning community to address the exploration versus exploitation dilemma. The $\epsilon$-schedule incorporates the following two components:

(a) *Exploration*: With probability $\epsilon$, where $\epsilon \in (0, 1)$, the agent chooses a random action. This exploratory approach enables the agent to examine the environment and gather information about less-explored state-action pairs, which may lead to the discovery of better long-term policies.

(b) *Exploitation*: With probability $1 - \epsilon$, the agent opts for the action associated with the highest reward according to its current knowledge of the environment. This is the greedy action, which aims to maximize the agent's immediate reward.

The parameter $\epsilon$ regulates the trade-off between exploration and exploitation: a larger value of $\epsilon$ fosters exploration, while a smaller value emphasizes exploitation. In practice, the value of $\epsilon$ can be held constant or permitted to decay over time according to a schedule. The schedule will dictate how quickly the value $\epsilon$ will decrease, including linear decay, exponential decay, or more sophisticated adaptation schemes based on the agent's performance (Thrun, 1992). By carefully designing the schedule, the reinforcement learning agent can achieve more efficient learning and better long-term performance.

In standard reinforcement learning, such as $Q$-Learning (see Sec. 2.6), agents can effectively learn about their environment when they can sample actions and rewards repeatedly with minimal cost. For example, agents can try various moves and strategies in computer games like Go or chess (Silver *et al.*, 2018) without incurring high costs. However, when exploration becomes too costly or time-consuming, sampling efficiency is essential. In high-cost environments, such as autonomous vehicle control or medical treatment plans, mistakes can lead to severe consequences, including serious injuries or human fatalities. In these cases, learning from fewer samples and minimizing the risks associated with exploration become imperative. Addressing the exploration versus exploitation dilemma is one approach to improving sampling efficiency.

Considering all possibilities for learning about the environment and collecting rewards is necessary to achieve an optimal balance between exploration and exploitation. As the planning horizon stretches, the number of possibilities increases exponentially, necessitating clever approximations in the state space to circumvent Bellman's curse of dimensionality.

## 6.3 Bayesian Learning

We learn by changing our probability distribution on the basis of experience.

—Bellman (1978, p. 85)

Bayesian Learning is an approach to how we quantify our uncertainty about a random variable, and how to update our uncertainty in light of new information or evidence.

Suppose we have a partially observable random variable $S$ in which we can infer by observing a related random variable $O$. We proceed in the following manner:

(a) We begin with some probability distribution over the random variable $S$, $\mathbf{P}(S)$, called the *prior distribution*. This prior distribution provides an expressive mechanism to encode knowledge about $S$ before observing any data.
(b) We select a likelihood $\mathbf{P}(S \mid O)$ that represents the dependence between $S$ and $O$.
(c) We observe the observation $O = o$.
(d) Using Bayes' Theorem, we update the probability distribution over $S$ given that we observed $O = o$ by

$$\mathbf{P}(S \mid O = o) = \frac{\mathbf{P}(o \mid S)\,\mathbf{P}(S)}{\int \mathbf{P}(o \mid S')\,\mathbf{P}(S)\,\mathrm{d}S'}, \tag{6.3.1}$$

or in other words,

$$\text{posterior} = \frac{\text{likelihood} \times \text{prior}}{\text{evidence}}. \tag{6.3.2}$$

This new updated probability distribution over $S$ is called the *posterior distribution*.
(e) When there is another observation, the posterior distribution becomes the prior distribution, and step (d) is repeated to find the new posterior distribution.

## 6.4 The Dirichlet Distribution

The Dirichlet distribution is used in Bayesian statistics to model uncertainty in probabilities or proportions, serving as conjugate priors for multinomial distributions. This enables computationally efficient updates when dealing with count data. Given $\phi_i$, the number of times event $e_i$ has occurred over $n$ trials, the probabilities $p_i$ of each event follows a Dirichlet distribution; that is, $(p_1, \ldots, p_k) \sim \text{Dir}(\phi_1, \ldots, \phi_k)$. The Dirichlet distribution represents the probability that a discrete random variable follows some discrete probability distribution $(p_1, \ldots, p_k)$ given the counts $(\phi_1, \ldots, \phi_k)$ observed over $n = \sum_{i=1}^{k} \phi_i$ trials.

A random variable $X$ that follows a Dirichlet distribution with parameters

$$\phi_1, \phi_2, \ldots, \phi_k > 0 \tag{6.4.1}$$

has a probability density function given by

$$f_X(p_1, \ldots, p_k \mid \phi_1, \ldots, \phi_k) = \frac{\Gamma(\phi_1 + \phi_2 + \cdots + \phi_k)}{\Gamma(\phi_1)\Gamma(\phi_2)\cdots\Gamma(\phi_k)} \prod_{i=1}^{k} p_i^{\phi_i - 1} \tag{6.4.2}$$

where $\Gamma(\cdot)$ is the gamma function, and the support is

$$p_1 + p_2 + \cdots + p_k = 1, \tag{6.4.3}$$

and

$$p_i \geq 0, \qquad \text{for } i \in \{1, \ldots, k\}. \tag{6.4.4}$$

The expected value of $p_i$ is

$$\mathbf{E}(p_i) = \frac{\phi_i}{\phi_1 + \phi_2 + \cdots + \phi_k}, \tag{6.4.5}$$

and the variance of $p_i$ is

$$\text{var}(p_i) = \frac{\mathbf{E}(p_i)\big(1 - \mathbf{E}(p_i)\big)}{1 + \sum_{i=1}^{k} \phi_k}. \tag{6.4.6}$$

**Example 6.4.1** Let $p_1 = 0.4$, $p_2 = 0.4$, and $p_3 = 0.2$. Let $\phi_1 = 4$, $\phi_2 = 4$, and $\phi_3 = 4$. Then,

$$\frac{\Gamma(\phi_1 + \phi_2 + \phi_3)}{\Gamma(\phi_1) \times \Gamma(\phi_2) \times \Gamma(\phi_3)} = \frac{\Gamma(4+4+4)}{\Gamma(4) \times \Gamma(4) \times \Gamma(4)} = \frac{11!}{3! \times 3! \times 3!} = \frac{39\,916\,800}{216},$$

and

$$\prod_{i=1}^{3} p_i^{\phi_i - 1} = 0.4^3 \times 0.4^3 \times 0.2^3 = 0.000\,032\,768.$$

Thus,

$$f(p_1, p_2, p_3 \mid \phi_1, \phi_2, \phi_3) \approx 6.055\,53.$$

The expected values are

$$\mathbf{E}(p_i) = \frac{4}{12} = \frac{1}{3} \qquad \text{for } i = 1, 2, 3.$$

◀

## 6.5 Online versus Offline Learning

The terms *online* and *offline* refer to the agent's process of learning. *Online learning* is the process that occurs interactively and in real-time, as the agent interacts with the environment. As the agent performs actions and receives reward, it updates its knowledge and adjusts its strategy accordingly. This repeating loop of action and adaptation allows the agent to learn and evolve in response to changing conditions.

On the other hand, *offline learning* or *batch learning* is characterized by a separation of the learning process from the agent's interaction with the environment. Here, the agent collects information from the environment first, typically through a series of actions and rewards, before undergoing a learning phase or batch processing phase where it updates its knowledge and strategies based on the collected data.

In the Bayes-adaptive approach, planning is conducted offline, resulting in an optimal learning policy. This policy represents the best course of action the agent should take given its current knowledge. It is important to clarify, however, that while the policy is constructed offline, it is executed online. The agent follows the predetermined policy while it is actively interacting with the environment. Through this interaction, the agent gathers data and learns about the environment, which is an online process. Therefore, even though the policy itself was determined offline, its implementation and the ensuing learning occur in an online manner. In other words, the agent is learning about the environment in real-time, while following a strategy that was determined offline. This approach capitalizes on the benefits of both online and offline learning: the computational efficiency of offline planning and the real-time adaptability of online learning.

## 6.6 Bayes Adaptive Markov Decision Process (BA-MDP)

In this section, we briefly summarize the work of Martin (1967) and Duff (2002) on the formulation of an MDP with unknown state transition probabilities, which Duff refers to this as *Bayes-Adaptive Markov Decision Process* (BA-MDP).

In a BA-MDP, the state space $\mathcal{S}$, action space $\mathcal{A}$, reward function $R$, discount factor $\gamma$, and planning horizon $N$ are known, except for the state transition probabilities $P$.

In a finite-state, finite-action MDP, the Markov property

$$\begin{aligned}&\mathbf{P}(S_{n+1} = s_{n+1} \mid S_n = s_n, A_n = a_n)\\ &\quad = \mathbf{P}(S_{n+1} = s_{n+1} \mid S_0 = s_0, A_0 = a_0, \dots, S_n = s_n, A_n = a_n) \qquad (6.6.1)\end{aligned}$$

states that the probability of the next state depends on the current state and action, and does not require any past history. This is to say that the state variable $s \in \mathcal{S}$ provides a sufficient statistic of the agent's past to make a decision. In an MDP, the state transition probability is known, $P(s' \mid s, a)$, and so with this, it is sufficient to calculate a probability for the next state $s'$ just given the current state $s$ and action $a$.

When dealing with an unknown state transition probability $P(s' \mid s, a)$, as in the case of BA-MDP, the state variable $s \in \mathcal{S}$ alone is insufficient and requires incorporating additional information from the past. To account for this, a new variable $M$ called the information variable is introduced. This information variable captures the relevant data from the agent's history, enabling it to inform future actions. Consequently, the hyperstate $(s, M)$ takes the place of the original state variable $s \in \mathcal{S}$.

If the information variable $M$ is an an array that records the number of occurrences of a particular $(s, a, s')$ transition, then the state space $\mathcal{S}_{\text{BA}} = \mathcal{S} \times \mathcal{M}$, where

$$\mathcal{M} = \left\{ M \in \mathbb{W}^{|\mathcal{S}|\times|\mathcal{A}|\times|\mathcal{S}|} \mid \forall(s, a) \in \mathcal{K}, \sum_{s' \in \mathcal{S}} M(s, a, s') > 0 \right\}. \qquad (6.6.2)$$

In this context, we will refer to $M$ as a count array. It is important to note that $\mathcal{M}$ represents a countably infinite space; Bellman's curse of dimensionality has arrived.

The state transition probabilities $P(s' \mid s, a)$ can be composed into a block matrix that Duff (2002, p. 25) calls the *generalized transition matrix* (or Martin (1967, p. 12, Eq. (1.3.24)) calls it a *generalized stochastic matrix*) which is defined as

$$P \triangleq \begin{bmatrix} P(\cdot \mid s_1, \cdot) \\ P(\cdot \mid s_2, \cdot) \\ \vdots \\ P(\cdot \mid s_{|\mathcal{S}|}, \cdot) \end{bmatrix} \qquad (6.6.3)$$

$$= \begin{bmatrix}
P(s_1 \mid s_1, a_1) & P(s_2 \mid s_1, a_1) & \cdots & P(s_{|\mathcal{S}|} \mid s_1, a_1) \\
P(s_1 \mid s_1, a_2) & P(s_2 \mid s_1, a_2) & \cdots & P(s_{|\mathcal{S}|} \mid s_1, a_2) \\
\vdots & \vdots & & \vdots \\
P(s_1 \mid s_1, a_{|\mathcal{A}_{s_1}|}) & P(s_2 \mid s_1, a_{|\mathcal{A}_{s_1}|}) & \cdots & P(s_{|\mathcal{S}|} \mid s_1, a_{|\mathcal{A}_{s_1}|}) \\
 & & & \\
P(s_1 \mid s_2, a_1) & P(s_2 \mid s_2, a_1) & \cdots & P(s_{|\mathcal{S}|} \mid s_2, a_1) \\
P(s_1 \mid s_2, a_2) & P(s_2 \mid s_2, a_2) & \cdots & P(s_{|\mathcal{S}|} \mid s_2, a_2) \\
\vdots & \vdots & & \vdots \\
P(s_1 \mid s_2, a_{|\mathcal{A}_{s_2}|}) & P(s_2 \mid s_2, a_{|\mathcal{A}_{s_2}|}) & \cdots & P(s_{|\mathcal{S}|} \mid s_2, a_{|\mathcal{A}_{s_2}|}) \\
 & & & \\
\vdots & \vdots & & \vdots \\
 & & & \\
P(s_1 \mid s_{|\mathcal{S}|}, a_1) & P(s_2 \mid s_{|\mathcal{S}|}, a_1) & \cdots & P(s_{|\mathcal{S}|} \mid s_{|\mathcal{S}|}, a_1) \\
P(s_1 \mid s_{|\mathcal{S}|}, a_2) & P(s_2 \mid s_{|\mathcal{S}|}, a_2) & \cdots & P(s_{|\mathcal{S}|} \mid s_{|\mathcal{S}|}, a_2) \\
\vdots & \vdots & & \vdots \\
P(s_1 \mid s_{|\mathcal{S}|}, a_{|\mathcal{A}_{s_{|\mathcal{S}|}}|}) & P(s_2 \mid s_{|\mathcal{S}|}, a_{|\mathcal{A}_{s_{|\mathcal{S}|}}|}) & \cdots & P(s_{|\mathcal{S}|} \mid s_{|\mathcal{S}|}, a_{|\mathcal{A}_{s_{|\mathcal{S}|}}|})
\end{bmatrix} \qquad (6.6.4)$$

where $|\mathcal{A}_{s_i}|$ is the number of admissible actions when in state $s_i$, and let $|\mathcal{A}| = |\cup_{s_i \in \mathcal{S}} \mathcal{A}_{s_i}|$ be the number of rows of $P$. Moreover, the conditions for MDPs still hold from Eq. (2.2.2) and Eq. (2.2.3).

Now, consider the matrix $P$ as a random variable with a prior distribution $H(P \mid M)$, where the information variable $M$ is a point in multi-dimensional Euclidean space. Mathematically,

$$\bar{P}(s' \mid s, a, M) \triangleq \int_P P(s' \mid s, a) \, \mathrm{d}H(P \mid M). \tag{6.6.5}$$

The expected immediate reward given state $s$ and action $a$ with respect to the prior distribution is

$$\bar{R}(s, a, M) = \sum_{s' \in \mathcal{S}} \bar{P}(s' \mid s, a, M) R(s, a, s'). \tag{6.6.6}$$

If the agent observes an $(s, a, s')$ transition, then $H$ is updated using Bayes' Rule:

$$\mathrm{d}H(P \mid M, s, a, s') = \frac{P(s' \mid s, a) \, \mathrm{d}H(P \mid M)}{\displaystyle\int_P P(s' \mid s, a) \, \mathrm{d}H(P \mid M)}. \tag{6.6.7}$$

If $H$ is taken to be a Dirichlet distribution (see Sec. 6.4) with parameter $M \in \mathbb{N}^{|\mathcal{S}| \times |\mathcal{A}| \times |\mathcal{S}|}$, then the joint density along one row of the generalized transition matrix (which means that the initial state $s \in \mathcal{S}$ and action $a \in \mathcal{A}(s)$ is fixed) is

$$\mathrm{d}H\big(P(\cdot \mid s, a) \mid M(s, a, \cdot)\big) = \frac{1}{\mathrm{B}\big(M(s, a, \cdot)\big)} \prod_{s' \in \mathcal{S}} \big(P(s' \mid s, a)\big)^{M(s,a,s')-1} \tag{6.6.8}$$

where the normalizing constant is expressed in terms of the gamma function as

$$\mathrm{B}\big(M(s, a, \cdot)\big) = \frac{\displaystyle\prod_{s' \in \mathcal{S}} \Gamma\big(M(s, a, s')\big)}{\Gamma\Big(\displaystyle\sum_{s' \in \mathcal{S}} M(s, a, s')\Big)}. \tag{6.6.9}$$

The support constraint (6.4.3) of the Dirichlet distribution is met since

$$\sum_{s' \in \mathcal{S}} P(s' \mid s, a) = 1, \qquad \forall (s, a) \in \mathcal{K}, \tag{6.6.10}$$

and the parameter restriction (6.4.1) is met if

$$M(s, a, s') > 0, \qquad \forall (s, a) \in \mathcal{K}, \forall s' \in \mathcal{S}. \tag{6.6.11}$$

**Remark 6.6.1** We give two possible interpretations of (6.6.11), which are the following:

(a) Given that $M(s, a, s') > 0$ represents the number of occurrences for a transition, it implies the agent must have observed every $(s, a, s')$ transition at least once. As $M(s, a, s')$ is non-zero, the transition probability is also non-zero. With one transition in the row of the generalized transition matrix having non-zero probability, according to Eq. (6.6.10), no transition in that row can have a probability of one. Thus, for $(s, a) \in \mathcal{K}$, $0 < P(s' \mid s, a) < 1$.

(b) The agent possesses a non-zero prior belief for all $(s, a, s')$ transitions, enabling it to learn from experience even when the true transition probability is zero. If a particular transition $(s, a, s')$ is never observed, the agent will update its belief accordingly, causing the probability associated with that transition to tend towards zero.

By Eq. (6.4.5), the expected state transition probability for $P(s' \mid s,a)$ is

$$\bar{P}(s' \mid s,a,M) = \mathbf{E}\left(P(s' \mid s,a)\right) = \frac{M(s,a,s')}{\sum_{s'' \in \mathcal{S}} M(s,a,s'')}, \tag{6.6.12}$$

and by Eq. (6.4.6), the variance is

$$\operatorname{var}\left(P(s' \mid s,a)\right) = \frac{\bar{P}(s' \mid s,a,M)\left(1 - \bar{P}(s' \mid s,a,M)\right)}{1 + \sum_{s'' \in \mathcal{S}} M(s,a,s'')} \tag{6.6.13}$$

The MDP version of the Bellman equation (2.4.2) can be rewritten by replacing the state $s \in \mathcal{S}$ with the hyperstate $(s, M) \in \mathcal{S} \times \mathcal{M}$, $P$ with Eq. (6.6.12), and $R$ with Eq. (6.6.6), and thus the Bellman equation for BA-MDP is

$$V^*(s,M) = \max_{a \in \mathcal{A}(s)} \left[\bar{R}(s,a,M) + \gamma \sum_{s' \in \mathcal{S}} \bar{P}(s' \mid s,a,M) V^*(s', \underbrace{M'}_{M+\delta_{s,a,s'}})\right] \tag{6.6.14}$$

In the array $M \in \mathbb{W}^{|\mathcal{S}| \times |\mathcal{A}| \times |\mathcal{S}|}$, each entry is the number of observed transitions $(s,a,s')$. For an observed transition $(s,a,s')$, the update to $M$ is

$$M' = M + \delta_{s,a,s'} \tag{6.6.15}$$

where $\delta_{s,a,s'}$ is an array of the same size as $M$ with a 1 in the $(s,a,s')$ position, and 0 otherwise. In Eq. (6.6.15), the addition between the arrays is element by element. It follows from Eq. (6.6.14) that the optimal policy $\pi^*$ is defined by actions that maximize the hyperstate value

$$\pi^*(s,M) = \operatorname*{arg\,max}_{a \in \mathcal{A}(s)} \left[\bar{R}(s,a,M) + \gamma \sum_{s' \in \mathcal{S}} \bar{P}(s' \mid s,a,M) V^*(s', M')\right]. \tag{6.6.16}$$

## 6.7 Bayes-Adaptive Semi-Markov Decision Processes (BA-SMDP)

In this section, we introduce one of the main contributions of the thesis. We create the Bayes-Adaptive Semi-Markov Decision Process (BA-SMDP), and look at four approaches to resolve how to learn or find a representation of the sojourn-time distribution.

We begin with an SMDP model $\langle \mathcal{S}, \mathcal{A}, Q, R, \gamma, N \rangle$ (see Sec. 4.1) that is known except for the sojourn-time state probability transition

$$\begin{aligned} Q(\tau, s' \mid s,a) &= P(s' \mid s,a) F(\tau \mid s,a,s') \\ &= P(s' \mid s,a) \int_0^\tau f(t \mid s,a,s')\, \mathrm{d}t. \end{aligned}$$

The four approaches that we consider are the following:

(a) learning the sojourn time distribution parameters using a count array to record the number of each sojourn time occurrence for a particular $(s,a,s')$ transition, where sojourn times come from a finite set (Sec. 6.8);
(b) learning the mixture of known sojourn time distributions with unknown proportions (Sec. 6.9);
(c) learning the mixture of known SMDPs with unknown proportions (Sec. 6.10); and
(d) learning the unknown continuous sojourn-time distribution parameters (Sec. 6.11).

## 6.8 Learning the Sojourn Time Distribution Parameters Using a Count Array with a Finite Set of Sojourn Times

For the state transition probability, we will follow the same as the BA-MDP in Sec. 6.6. Let $H$ be a Dirichlet distribution given by Eq. (6.6.5) with parameter $M$, then the expected state transition probability for $P(s' \mid s, a)$ is given by Eq. (6.6.12).

In this section, to deal with how the agent will learn the sojourn time, we suppose that the agent knows that there is a finite set of possible sojourn times

$$C = \{\tau_1, \tau_2, \ldots, \tau_{|C|}\},$$

however, it does not know with certainty which sojourn time will be selected for a particular $(s, a, s')$ transition. Then, we will let $W \in \mathbb{W}^{|\mathcal{S}|\times|\mathcal{A}|\times|C|\times|\mathcal{S}|}$ be the count array where $W(s, a, \tau, s')$ is the number of $\tau$ occurrences for transition $(s, a, s')$.

In order to use the sojourn time of transition $(s, a, s')$ as a parameter of a Dirichlet distribution, we need to ensure that each $\tau \in C$ is in the interval [0, 1], and that they sum to one. To find the rescaled and normalized sojourn time $\tilde{\tau}$, we do the following:

(a) Calculate the minimum and maximum of the sojourn times so that

$$\tau_{\min} = \min\{\tau_1, \tau_2, \ldots, \tau_{|C|}\} \tag{6.8.1}$$

and

$$\tau_{\max} = \max\{\tau_1, \tau_2, \ldots, \tau_{|C|}\}. \tag{6.8.2}$$

(b) Normalize each sojourn time by

$$\tau_i' = \frac{\tau_i - \tau_{\min}}{\tau_{\max} - \tau_{\min}}. \tag{6.8.3}$$

(c) Rescale the normalized sojourn times to make their sum equal to one; that is,

$$\tilde{\tau}_i = \frac{\tau_i'}{\sum\limits_{\tau \in C} \tau}. \tag{6.8.4}$$

We denote the rescaled and normalized set as

$$\tilde{C} = \{\tilde{\tau}_1, \tilde{\tau}_2, \ldots, \tilde{\tau}_{|\tilde{C}|}\}$$

and $W(s, a, \tau, s') = W(s, a, \tilde{\tau}, s')$.

If $U$ is taken to be a Dirichlet distribution with parameter $W \in \mathbb{N}^{|\mathcal{S}|\times|\mathcal{A}|\times|C|\times|\mathcal{S}|}$, then the joint density on a particular $(s, a, s')$ transition is

$$\mathrm{d}U\big(\tilde{\tau}_1, \ldots, \tilde{\tau}_{|\tilde{C}|} \mid W(s, a, \cdot, s')\big) = \frac{1}{B\big(W(s, a, \cdot, s')\big)} \prod_{i=1}^{|\tilde{C}|} \tilde{\tau}_i^{W(s,a,\tilde{\tau}_i,s')-1} \tag{6.8.5}$$

where the normalizing constant is expressed in terms of the gamma function as

$$B\big(W(s, a, \cdot, s')\big) = \frac{\prod\limits_{i=1}^{|\tilde{C}|} \Gamma\big(W(s, a, \tilde{\tau}_i, s')\big)}{\Gamma\Big(\sum\limits_{i=1}^{|\tilde{C}|} W(s, a, \tilde{\tau}_i, s')\Big)} \tag{6.8.6}$$

The expected value of the scaled and normalized mean sojourn time of a particular transition $(s, a, s')$ is

$$\mathbf{E}(\tilde{\tau}_i) = \frac{W(s, a, \tilde{\tau}_i, s')}{\sum\limits_{j=1}^{|\tilde{C}|} W(s, a, \tilde{\tau}_j, s')} = \frac{W(s, a, \tau_i, s')}{\sum\limits_{j=1}^{|C|} W(s, a, \tau_j, s')} \tag{6.8.7}$$

and its variance is

$$\operatorname{var}(\tilde{\tau}_i) = \frac{\mathbf{E}(\tilde{\tau}_i)\big(1 - \mathbf{E}(\tilde{\tau}_i)\big)}{1 + \sum\limits_{\tilde{\tau}\in\tilde{C}} W(s, a, \tilde{\tau}, s')}. \tag{6.8.8}$$

Expressing the mean sojourn time using the original sojourn times from set $C$, we have

$$\bar{\tau}(s, a, s') = \frac{\sum\limits_{\tau\in C} \big(\tau W(s, a, \tau, s')\big)}{\sum\limits_{\tau\in C} W(s, a, \tau, s')}. \tag{6.8.9}$$

We can express $f$ as a generalized sojourn-time distribution matrix such that

$$f_\theta \triangleq \begin{bmatrix} \theta(s_1, \cdot, \cdot) \\ \theta(s_2, \cdot, \cdot) \\ \vdots \\ \theta(s_{|\mathcal{S}|}, \cdot, \cdot) \end{bmatrix} \tag{6.8.10}$$

where the entries correspond to parameters that define distributions for each $(s, a, s')$ transition. We suppose now that the matrix $f_\theta$ is a random variable with a prior distribution $U(f_\theta \mid W)$, where $W$ is a point in multi-dimensional Euclidean space. Mathematically,

$$\bar{f}(\tau \mid s, a, s', W) \triangleq \int_{f_\theta} f_\theta(\tau \mid s, a, s')\,\mathrm{d}U(f_\theta \mid W). \tag{6.8.11}$$

If $U$ is taken to be a Dirichlet distribution with parameter $W \in \mathbb{N}^{|\mathcal{S}|\times|\mathcal{A}|\times|\mathcal{S}|}$, then

$$\mathrm{d}U(f_\theta \mid W) = \frac{1}{\mathrm{B}(W)} \prod_{s\in\mathcal{S}} \prod_{a\in\mathcal{A}(s)} \prod_{s'\in\mathcal{S}} \big(\theta(s, a, s')\big)^{W(s,a,s')-1} \tag{6.8.12}$$

where the normalizing constant is expressed in terms of the gamma function as

$$\mathrm{B}(W) = \frac{\prod\limits_{s\in\mathcal{S}} \prod\limits_{a\in\mathcal{A}(s)} \prod\limits_{s'\in\mathcal{S}} \Gamma\big(\theta(s, a, s')\big)^{W(s,a,s')-1}}{\Gamma\Big(\sum\limits_{s\in\mathcal{S}} \sum\limits_{a\in\mathcal{A}(s)} \sum\limits_{s'\in\mathcal{S}} W(s, a, s')\Big)} \tag{6.8.13}$$

The expected sojourn-time transition probability for $f(\tau \mid s, a, s')$ is

$$\bar{f}(\tau \mid s, a, s', W) = \mathbf{E}\left(f(\tau \mid s, a, s')\right) = \frac{\tau W(s, a, \tau', s')}{\sum\limits_{\tau'\in C} W(s, a, \tau, s')} \tag{6.8.14}$$

where $W(s, a, \tau, s')$ is the number of occurrences sojourn time $\tau$ for $(s, a, s')$ transition. Combining both together to get an estimate of $Q$, we have

$$\begin{aligned} \bar{Q}(\tau, s' \mid s, a, M, W) &= \bar{P}(s' \mid s, a, M)\bar{F}(\tau \mid s, a, s', W) \\ &= \frac{M(s, a, s')W(s, a, \tau, s')}{\sum\limits_{s''\in\mathcal{S}} M(s, a, s'') \sum\limits_{\tau'\in C} W(s, a, \tau', s'')}. \end{aligned}$$

The reward function is

$$\bar{R}(s,a,M,U) = r_1(s,a) + \sum_{s' \in \mathcal{S}} \bar{P}(s' \mid s,a,M) \frac{1}{|C|} \sum_{\tau \in C} \frac{W(s,a,\tau,s')}{\sum_{\tau' \in C} W(s,a,\tau',s')} \int_0^{\tau} \mathrm{e}^{-\beta t} r_2(t \mid s,a,s')\, \mathrm{d}t, \tag{6.8.15}$$

where $r_1(s,a)$ is the lump sum and $r_2(t \mid s,a,s')$ is the continuous reward rate.

The Bellman equation is

$$V^*(s,M,U) = \max_{a \in \mathcal{A}(s)} \Bigg[ \bar{R}(s,a,M,U) + \sum_{s' \in \mathcal{S}} \bar{P}(s' \mid s,a,M) \frac{1}{|C|} \sum_{\tau \in C} \mathrm{e}^{-\beta\tau} \frac{W(s,a,\tau,s')}{\sum_{\tau' \in C} W(s,a,\tau',s')} V^*(s', \underbrace{M'}_{M+\delta_{s,a,s'}}, \overbrace{U'}^{U+\delta_{s,a,\tau,s'}}) \Bigg] \tag{6.8.16}$$

In the array $M \in \mathbb{W}^{|\mathcal{S}|\times|\mathcal{A}|\times|\mathcal{S}|}$, each entry $M(s,a,s')$ is the number of observed transitions $(s,a,s')$. For an observed transition $(s,a,s')$, the update to $M$ in

$$M' = M + \delta_{s,a,s'}, \tag{6.8.17}$$

where $\delta_{s,a,s'}$ is an array of the same size as $M$ with a 1 in the $(s,a,s')$ position, and 0 otherwise. For the array $U \in \mathbb{W}^{|\mathcal{S}|\times|\mathcal{A}|\times|C|\times|\mathcal{S}|}$, each entry $W(s,a,\tau,s')$ is the number of times a particular $\tau \in C$ was observed for a particular $(s,a,s')$ transition. For an observed transition with sojourn time $(s,a,\tau,s')$, the update to $U$ is

$$U' = U + \delta_{s,a,\tau,s'}, \tag{6.8.18}$$

where $\delta_{s,a,\tau,s'}$ is an array of the same size as $U$ with a 1 in the $(s,a,\tau,s')$ position, and 0 otherwise.

It follows from Eq. (6.8.16) that the optimal policy $\pi^*$ is defined by actions that maximize the hyperstate

$$\pi^*(s,M,U) = \arg\max_{a \in \mathcal{A}} \Bigg[ \bar{R}(s,a,M,U) + \sum_{s' \in \mathcal{S}} \bar{P}(s' \mid s,a,M) \frac{1}{|C|} \sum_{\tau \in C} \mathrm{e}^{-\beta\tau} \frac{W(s,a,\tau,s')}{\sum_{\tau' \in C} W(s,a,\tau',s')} V^*(s',M',U') \Bigg] \tag{6.8.19}$$

## 6.9 Learning the Mixture of Sojourn-Time Distributions

For a BA-SMDP $\langle \mathcal{S}, \mathcal{A}, Q, R, \beta, N \rangle$, we consider that everything about the SMDP model is known, except for $Q$. It does not seem fair in a Bayes-adaptive setting that the agent knows the state space $\mathcal{S}$ or the action space $\mathcal{A}$, and yet the agent is not allowed to know the space concerning the sojourn-time distributions.

In this approach, we consider allowing the agent to know the space of sojourn-time distributions, but the agent upon observing a sojourn time $\tau$ does not know from which distribution $\tau$ came.

**Example 6.9.1 — Two Inverse Gaussian Distributions.** For a particular $(s, a, s')$ transition in an SMDP, the sojourn-time distribution is a mixture of two inverse Gaussians with respective probability density functions $f_1(\tau \mid \mu_1, \lambda_1)$ and $f_2(\tau \mid \mu_2, \lambda_2)$. The mixture of these two probability density functions is

$$f(\tau \mid s, a, s') = \eta f_1(\tau \mid \mu_1, \lambda_1) + (1 - \eta) f_2(\tau \mid \mu_2, \lambda_2), \tag{6.9.1}$$

where $\eta \in [0, 1]$ is the mixing weight. The agent knows the probability density functions $f_1$ and $f_2$ (and their respective parameters), but does not know the mixing weight $\eta$.

Initially, the agent's estimate of the mixing weight could be $\hat{\eta} = 0.5$. When the agent observes a sojourn time $\tau$ from the $(s, a, s')$ transition, the estimate update of the mixing weight is

$$\hat{\eta} \leftarrow \frac{\hat{\eta} f_1(\tau \mid s, a, s')}{\hat{\eta} f_1(\tau \mid s, a, s') + (1 - \hat{\eta}) f_2(\tau \mid s, a, s')}. \tag{6.9.2}$$

◀

Now, a critic may say in Example 6.9.1 that it may be fine to know the space of the sojourn-time distributions by specifying which family (for example, inverse Gaussian), but we go too far by specifying the parameters of those distributions which is something that should be learned by the agent. To soothe this criticism, we can increase the number of sojourn-time distributions with specified parameters so that the mixture of these distributions will have more expressive power for an arbitrary distribution. For $k$ many sojourn-time distributions with respective probability density functions $f_1(\tau \mid \theta_1)$, $f_2(\tau \mid \theta_2)$, …, $f_k(\tau \mid \theta_k)$, the mixture of these functions is

$$f(\tau \mid s, a, s') = \eta_1 f_1(\tau \mid \theta_1) + \eta_2 f_2(\tau \mid \theta_2) + \cdots + \eta_k f_k(\tau \mid \theta_k) \tag{6.9.3}$$

where $\eta_1 + \eta_2 + \cdots + \eta_k = 1$. The estimate update of the mixing weight would be

$$\hat{\eta}_i \leftarrow \frac{\hat{\eta}_i f_i(\tau \mid \theta_i)}{\eta_1 f_1(\tau \mid \theta_1) + \eta_2 f_2(\tau \mid \theta_2) + \cdots + \eta_k f_k(\tau \mid \theta_k)}. \tag{6.9.4}$$

Let $\varphi(s, a, s', \ell)$ is the number of observances that the sojourn time came from distribution $\ell$ in $(s, a, s')$ transition. Then, we can rewrite the Bellman equation as

$$V(s, M, \varphi) = \max_{a \in \mathcal{A}(s)} \Bigg[ \bar{R}(s, a, M, \varphi) + \sum_{s' \in \mathcal{S}} \bar{P}(s' \mid s, a, M) \\ \sum_{\ell \in C} \frac{\varphi(s, a, s', \ell)}{\sum_{\ell' \in C} \varphi(s, a, s', \ell')} \int_0^\infty e^{-\beta \tau} f_\ell(\tau) V(s, M', \varphi') \, d\tau \Bigg]. \tag{6.9.5}$$

## 6.10 Learning the Mixture of SMDPs

Let us consider SMDPs where the parameters are known, and through experience, the agent's belief over time will select the appropriate SMDP that closely resembles the environment. The policy mapping beliefs to actions should balance between attempting to achieve the maximum discounted expected reward and performing sufficient probing actions to determine the SMDP in which the agent operates.

Suppose that there are $K$ many SMDP models that could be the SMDP model of the underlying environment. For an arbitrary SMDP model with index $k$, we will denote this by the tuple

$$\langle \mathcal{S}, \mathcal{A}, Q_k, R, \beta, N \rangle, \tag{6.10.1}$$

where the state space $\mathcal{S}$, action space $\mathcal{A}$, reward function $R$, discount rate $\beta$, and planning horizon $N$ are the same for each SMDP model.

We will wrap these $K$ SMDP model into a single partially observable semi-Markov decision process (POSMDP)

$$\langle \mathcal{S}_{\text{BA}}, \mathcal{A}_{\text{BA}}, \mathcal{O}_{\text{BA}}, Q_{\text{BA}}, G_{\text{BA}}, G_0, R_{\text{BA}}, \beta, N \rangle .$$

The state space is

$$\mathcal{S}_{\text{BA}} = \mathcal{S} \times \{k\}_{k=1}^{K} \tag{6.10.2}$$

where $k$ is the index of the SMDP model. A hyperstate $(s, k) \in \mathcal{S}_{\text{BA}}$ would represent a state $s \in \mathcal{S}$ from the original state space and an SMDP model index $k$. The action space is

$$\mathcal{A}_{\text{BA}} = \mathcal{A} \tag{6.10.3}$$

is the same as the action space for all of the SMDP model. The observation space is

$$\mathcal{O}_{\text{BA}} = \mathcal{S}, \tag{6.10.4}$$

since the states $s \in \mathcal{S}$ are fully observable and form the observations as well.

We assume that if the agent is in a particular SMDP model $k$ that the state transition stays within that SMDP; in other words, the transition would be $(s, k) \to (s', k)$. This leads us to the sojourn-time state transition probability as

$$Q_{\text{BA}}\big(\tau, (s', k') \mid (s, k), a\big) = \begin{cases} Q_k(\tau, s' \mid s, a), & \text{if } k' = k; \\ 0, & \text{otherwise.} \end{cases} \tag{6.10.5}$$

or equivalently,

$$Q_{\text{BA}}\big(\tau, (s', k') \mid (s, k), a\big) = \begin{cases} P_k(s' \mid s, a) F_k(\tau \mid s, a, s'), & \text{if } k' = k; \\ 0, & \text{otherwise.} \end{cases} \tag{6.10.6}$$

The observation transition probability is

$$G_{\text{BA}}\big(o \mid a, (s', k')\big) = \begin{cases} 1, & \text{if } s' = o; \\ 0, & \text{otherwise,} \end{cases} \tag{6.10.7}$$

since the landing state $s' \in \mathcal{S}$ is the observation and it is fully observable, the observation transition probability becomes deterministic. Similarly, we define the initial observation probability by

$$G_0\big(o \mid (s', k')\big) = \begin{cases} 1, & \text{if } s' = o; \\ 0, & \text{otherwise.} \end{cases} \tag{6.10.8}$$

The reward function is

$$R_{\text{BA}}\big((s, k), a\big) = R(s, a). \tag{6.10.9}$$

The discount rate $\beta$ and the planning horizon $N$ for the POSMDP is the same as in all of the SMDP models.

To fully define a POSMDP, we have to specify an initial belief $\xi_0$. A prior distribution can be used to incorporate expert knowledge about the problem, but if no information is available, we suggest

$$\xi_0(s, k) = \frac{1}{|\mathcal{S}|} \cdot \frac{1}{K}, \qquad \forall s \in \mathcal{S}, \forall k \in \{1, 2, \ldots, K\}, \tag{6.10.10}$$

or

$$\xi_0(k) = \frac{1}{K}, \qquad \forall k \in \{1, 2, \ldots, K\}, \tag{6.10.11}$$

which means that agent believes it is equally likely to be in any of the $k$ possible SMDP models.

To update the belief, we contribute the following theorem.

**Theorem 6.10.1** Given a belief $\xi(s,k)$, and the agent takes action $a$, waits $\tau$ sojourn time, and observes $o$, the belief update is

$$\xi((s',k') \mid a,\tau,o) = \begin{cases} \dfrac{\sum_{s\in\mathcal{S}}\sum_{k=1}^{K} Q_k(\tau,s' \mid s,a)\xi(s,k)}{\sum_{s''\in\mathcal{S}}\sum_{s\in\mathcal{S}}\sum_{k=1}^{K} Q_k(\tau,s'' \mid s,a)\xi(s,k)}, & \text{if } k'=k \text{ and } s'=o; \\ 0, & \text{otherwise,} \end{cases}$$

where $Q_k$ denotes the $Q$ from SMDP model $k$.

**Proof** From Theorem 5.5.2 for the belief update of a POSMDP, given an action $a$ the agent performed, a sojourn time $\tau$, and an observation $o$ seen, the belief update for a particular hyperstate $(s,k) \in \mathcal{S}_{\text{BA}}$ is

$$\xi((s',k') \mid a,\tau,o) = \frac{G_{\text{BA}}(o \mid a,(s',k'))\sum_{s\in\mathcal{S}}\sum_{k=1}^{K} Q_{\text{BA}}(\tau,(s',k') \mid (s,k),a)\xi(s,k)}{\sum_{s''\in\mathcal{S}} G_{\text{BA}}(o \mid a,(s'',k''))\sum_{s\in\mathcal{S}}\sum_{k=1}^{K} Q_{\text{BA}}(\tau,(s'',k'') \mid (s,k),a)\xi(s,k)}.$$

Using Eq. (6.10.5), we can substitute $Q_{\text{BA}}$ with $Q_k$ under the condition that $k'=k$. This means that the state $s$ can transition to state $s'$, but it must remain within the SMDP model $k$. This leads to

$$\xi((s',k') \mid a,\tau,o) = \begin{cases} \dfrac{G_{\text{BA}}(o \mid a,(s',k'))\sum_{s\in\mathcal{S}}\sum_{k=1}^{K} Q_k(\tau,s' \mid s,a)\xi(s,k)}{\sum_{s''\in\mathcal{S}} G_{\text{BA}}(o \mid a,(s'',k''))\sum_{s\in\mathcal{S}}\sum_{k=1}^{K} Q_k(\tau,s'' \mid s,a)\xi(s,k)}, & \text{if } k'=k; \\ 0, & \text{otherwise.} \end{cases}$$

Lastly, using Eq. (6.10.7), we can remove $G_{\text{BA}}$ by adding the condition $s'=o$, which gives us

$$\xi\big((s',k') \mid a,\tau,o\big) = \begin{cases} \dfrac{\sum_{s\in\mathcal{S}}\sum_{k=1}^{K} Q_k(\tau,s' \mid s,a)\xi(s,k)}{\sum_{s''\in\mathcal{S}}\sum_{s\in\mathcal{S}}\sum_{k=1}^{K} Q_k(\tau,s'' \mid s,a)\xi(s,k)}, & \text{if } k'=k \text{ and } s'=o; \\ 0, & \text{otherwise.} \end{cases}$$

◀

We can marginalize $s'$ so that the belief is expressed in terms of SMDP model index $k$

$$\xi(k \mid a,\tau,o) = \sum_{s'\in\mathcal{S}} \xi\big((s',k') \mid a,\tau,o\big). \tag{6.10.12}$$

With the belief expressed this way, $\xi(k)$ represents the agent's subjective probability that it believes it is in SMDP model $k$, such as in Fig. 6.1.

This completes our transformation of wrapping $K$ SMDPs into a single POSMDP.

Now that we have fully defined the BA-SMDP as an equivalent POSMDP, we can use a POSMDP solver such as CHRONOSPERSEUS that we developed in Sec. 5.11. We demonstrate this in the following example.

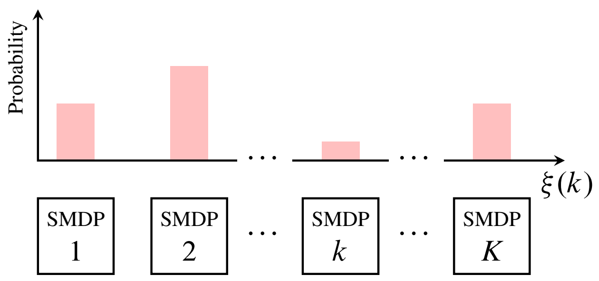


Fig. 6.1 An example of a belief distribution over the possible SMDP models. The belief state with the highest probability, in this example, SMDP 2, is what the agent believes to be the most likely model of the environment.

**Example 6.10.2 — Discerning between two SMDPs.** We will consider two distinct SMDP models that are possible for an environment. Essentially, the agent will observe and interact with the environment, and will update it's belief of which of the two SMDPs the agent is most likely residing.

For both SMDPs, the state space is

$$\mathcal{S} = \{s_1, s_2\},$$

and the action space is

$$\mathcal{A} = \{a_1, a_2\}.$$

The POSMDP state space consists of hyperstates

$$\begin{aligned}\mathcal{S}_{\text{BA}} &= \underbrace{\{s_1, s_2\}}_{\mathcal{S}} \times \underbrace{\{1, 2\}}_{\text{model index}} \\ &= \{(s_1, 1), (s_2, 1), (s_1, 2), (s_2, 2)\},\end{aligned}$$

and the action space stays the same

$$\mathcal{A}_{\text{BA}} = \mathcal{A}.$$

For the state transition probabilities $P$, we will assign a subscript 1 or 2 to SMDP 1 and SMDP 2, respectively. For SMDP 1:

$$P_1(\cdot \mid \cdot, a_1) = \begin{array}{c} \\ s_1 \\ s_2 \end{array} \begin{array}{c} \begin{array}{cc} s_1 & s_2 \end{array} \\ \begin{bmatrix} P_1(s_1 \mid s_1, a_1) & P_1(s_2 \mid s_1, a_1) \\ P_1(s_1 \mid s_2, a_1) & P_1(s_2 \mid s_2, a_1) \end{bmatrix} \end{array}$$

and

$$P_1(\cdot \mid \cdot, a_2) = \begin{array}{c} \\ s_1 \\ s_2 \end{array} \begin{array}{c} \begin{array}{cc} s_1 & s_2 \end{array} \\ \begin{bmatrix} P_1(s_1 \mid s_1, a_2) & P_1(s_2 \mid s_1, a_2) \\ P_1(s_1 \mid s_2, a_2) & P_1(s_2 \mid s_2, a_2) \end{bmatrix} \end{array}$$

For SMDP 2:

$$P_2(\cdot \mid \cdot, a_1) = \begin{array}{c} \\ s_1 \\ s_2 \end{array} \begin{array}{c} \begin{array}{cc} s_1 & s_2 \end{array} \\ \begin{bmatrix} P_2(s_1 \mid s_1, a_1) & P_2(s_2 \mid s_1, a_1) \\ P_2(s_1 \mid s_2, a_1) & P_2(s_2 \mid s_2, a_1) \end{bmatrix} \end{array}$$

and

$$P_2(\cdot \mid \cdot, a_2) = \begin{array}{c} \\ s_1 \\ s_2 \end{array} \begin{array}{c} \begin{array}{cc} s_1 & s_2 \end{array} \\ \begin{bmatrix} P_2(s_1 \mid s_1, a_2) & P_2(s_2 \mid s_1, a_2) \\ P_2(s_1 \mid s_2, a_2) & P_2(s_2 \mid s_2, a_2) \end{bmatrix} \end{array}.$$

We wrap the two SMDP state transition probability functions into a single function for the POSMDP, and for action $a_1$ this yields

$$P_{\mathrm{BA}}\big((\cdot,\cdot) \mid (\cdot,\cdot),a_1\big) = \begin{array}{c|cc} & (s_1,1)\ (s_2,1) & (s_1,2)\ (s_2,2) \\ \hline (s_1,1),\ (s_2,1) & P_1(\cdot \mid \cdot, a_1) & 0 \\ (s_1,2),\ (s_2,2) & 0 & P_2(\cdot \mid \cdot, a_1) \end{array}$$

and similarly for action $a_2$ is

$$P_{\mathrm{BA}}\big((\cdot,\cdot) \mid (\cdot,\cdot),a_1\big) = \begin{array}{c|cc} & (s_1,1)\ (s_2,1) & (s_1,2)\ (s_2,2) \\ \hline (s_1,1),\ (s_2,1) & P_1(\cdot \mid \cdot, a_2) & 0 \\ (s_1,2),\ (s_2,2) & 0 & P_2(\cdot \mid \cdot, a_2) \end{array}.$$

Similarly, for the sojourn-time distribution parameters, we have for SMDP 1:

$$\mu_1(\cdot,a_1,\cdot) = \begin{array}{c|cc} & s_1 & s_2 \\ \hline s_1 & \mu_1(s_1,a_1,s_1) & \mu_1(s_1,a_1,s_2) \\ s_2 & \mu_1(s_2,a_1,s_1) & \mu_1(s_2,a_1,s_2) \end{array}$$

and

$$\mu_1(\cdot,a_2,\cdot) = \begin{array}{c|cc} & s_1 & s_2 \\ \hline s_1 & \mu_1(s_1,a_2,s_1) & \mu_1(s_1,a_2,s_2) \\ s_2 & \mu_1(s_2,a_2,s_1) & \mu_1(s_2,a_2,s_2) \end{array}.$$

We define the matrices in a similar manner for the shape parameter $\lambda$. This holds for SMDP 2.

Then, the sojourn-time distribution parameters for the POSMDP,

$$\mu_{\mathrm{BA}}\big((\cdot,\cdot),a_1,(\cdot,\cdot)\big) = \begin{array}{c|cc} & (s_1,1)\ (s_2,1) & (s_1,2)\ (s_2,2) \\ \hline (s_1,1),\ (s_2,1) & \mu_1(\cdot,a_1,\cdot) & 0 \\ (s_1,2),\ (s_2,2) & 0 & \mu_2(\cdot,a_1,\cdot) \end{array}$$

and

$$\mu_{\mathrm{BA}}\big((\cdot,\cdot),a_2,(\cdot,\cdot)\big) = \begin{array}{c|cc} & (s_1,1)\ (s_2,1) & (s_1,2)\ (s_2,2) \\ \hline (s_1,1),\ (s_2,1) & \mu_1(\cdot,a_2,\cdot) & 0 \\ (s_1,2),\ (s_2,2) & 0 & \mu_2(\cdot,a_2,\cdot) \end{array}$$

We define $\lambda_{\mathrm{BA}}\big((\cdot,\cdot),a_1,(\cdot,\cdot)\big)$ and $\lambda_{\mathrm{BA}}\big((\cdot,\cdot),a_2,(\cdot,\cdot)\big)$ similarly.

The observation transition function for the POSMDP is

$$G_{\mathrm{BA}}\big(\cdot \mid a_1,(\cdot,\cdot)\big) = G_{\mathrm{BA}}\big(\cdot \mid a_2,(\cdot,\cdot)\big) = \begin{array}{c|cc} & o{=}s_1 & o{=}s_2 \\ \hline (s_1,1) & 1 & 0 \\ (s_2,1) & 0 & 1 \\ (s_1,2) & 1 & 0 \\ (s_2,2) & 0 & 1 \end{array}$$

For the reward function, since the reward function is the same for each SMDP,

$$R_{\mathrm{BA}}\big((s,k),a\big) = R(s,a).$$

This concludes how we wrap two SMDP models into a single POSMDP.

We give a numerical example using CHRONOSPERSEUS. Let the state transition probabilities be for action $a_1$

$$P_{\text{BA}}\big((\cdot,\cdot) \mid (\cdot,\cdot), a_1\big) = \begin{array}{c|cccc} & (s_1,1) & (s_2,1) & (s_1,2) & (s_2,2) \\ \hline (s_1,1) & 0.5 & 0.5 & 0 & 0 \\ (s_2,1) & 0.5 & 0.5 & 0 & 0 \\ (s_1,2) & 0 & 0 & 0.6 & 0.4 \\ (s_2,2) & 0 & 0 & 0.3 & 0.7 \end{array}$$

and for action $a_2$

$$P_{\text{BA}}\big((\cdot,\cdot) \mid (\cdot,\cdot), a_2\big) = \begin{array}{c|cccc} & (s_1,1) & (s_2,1) & (s_1,2) & (s_2,2) \\ \hline (s_1,1) & 0.5 & 0.5 & 0 & 0 \\ (s_2,1) & 0.6 & 0.4 & 0 & 0 \\ (s_1,2) & 0 & 0 & 0.6 & 0.4 \\ (s_2,2) & 0 & 0 & 0.5 & 0.5 \end{array}$$

Using the two-parameter inverse Gaussian probability density function, Eq. (4.2.3), the mean sojourn-time distribution parameters for action $a_1$ are

$$\mu_{\text{BA}}\big((\cdot,\cdot), a_1, (\cdot,\cdot)\big) = \begin{array}{c|cccc} & (s_1,1) & (s_2,1) & (s_1,2) & (s_2,2) \\ \hline (s_1,1) & 2 & 2 & 0 & 0 \\ (s_2,1) & 3 & 3 & 0 & 0 \\ (s_1,2) & 0 & 0 & 3 & 3 \\ (s_2,2) & 0 & 0 & 2 & 2 \end{array}$$

and for action $a_2$

$$\mu_{\text{BA}}\big((\cdot,\cdot), a_2, (\cdot,\cdot)\big) = \begin{array}{c|cccc} & (s_1,1) & (s_2,1) & (s_1,2) & (s_2,2) \\ \hline (s_1,1) & 2 & 3 & 0 & 0 \\ (s_2,1) & 3 & 2 & 0 & 0 \\ (s_1,2) & 0 & 0 & 2 & 3 \\ (s_2,2) & 0 & 0 & 3 & 2 \end{array}$$

The shape sojourn-time distribution parameters for action $a_1$ are

$$\lambda_{\text{BA}}\big((\cdot,\cdot), a_1, (\cdot,\cdot)\big) = \begin{array}{c|cccc} & (s_1,1) & (s_2,1) & (s_1,2) & (s_2,2) \\ \hline (s_1,1) & 4 & 4 & 0 & 0 \\ (s_2,1) & 9 & 9 & 0 & 0 \\ (s_1,2) & 0 & 0 & 9 & 9 \\ (s_2,2) & 0 & 0 & 4 & 4 \end{array}$$

and for action $a_2$

$$\lambda_{\text{BA}}\big((\cdot,\cdot), a_2, (\cdot,\cdot)\big) = \begin{array}{c|cccc} & (s_1,1) & (s_2,1) & (s_1,2) & (s_2,2) \\ \hline (s_1,1) & 4 & 9 & 0 & 0 \\ (s_2,1) & 9 & 4 & 0 & 0 \\ (s_1,2) & 0 & 0 & 4 & 9 \\ (s_2,2) & 0 & 0 & 9 & 4 \end{array}$$

For the reward function, we set

$$r_1(\cdot,\cdot) = \begin{array}{c|cc} & a_1 & a_2 \\ \hline s_1 & 100 & 125 \\ s_2 & 125 & 100 \end{array}$$

and

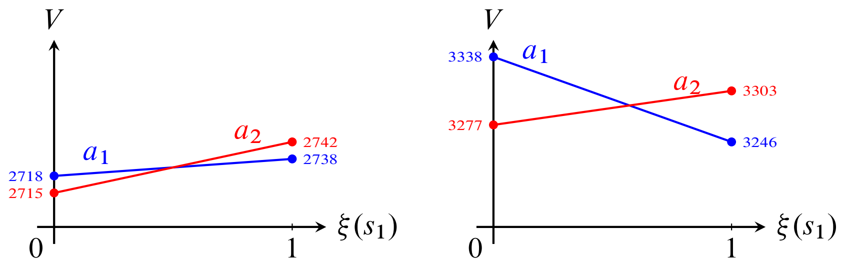
$$r_2(\cdot, a_1, \cdot) = \begin{array}{c c} & \begin{array}{cc} s_1 & s_2 \end{array} \\ \begin{array}{c} s_1 \\ s_2 \end{array} & \begin{bmatrix} 10 & 5 \\ 5 & 10 \end{bmatrix} \end{array} \quad \text{and} \quad r_2(\cdot, a_2, \cdot) = \begin{array}{c c} & \begin{array}{cc} s_1 & s_2 \end{array} \\ \begin{array}{c} s_1 \\ s_2 \end{array} & \begin{bmatrix} 5 & 10 \\ 10 & 5 \end{bmatrix} \end{array}$$

At the beginning of the problem $n = 0$, we will assume that the agent believes it is equally likely to be in either of the SMDP models, but starts in $s_1$:

$$\xi_0(\cdot, \cdot) = \begin{array}{c c} & \begin{array}{cc} k{=}1 & k{=}2 \end{array} \\ \begin{array}{c} s_1 \\ s_2 \end{array} & \begin{bmatrix} 0.5 & 0.5 \\ 0 & 0 \end{bmatrix} \end{array}$$

After running ChronosPerseus on 10 000 beliefs for 100 iterations, we have a value function where

$$V^* = \left\{ \begin{array}{c c} & \begin{array}{c} a_1 \end{array} \\ \begin{array}{c} V(s_1,k{=}1) \\ V(s_2,k{=}1) \\ V(s_1,k{=}2) \\ V(s_2,k{=}2) \end{array} & \begin{bmatrix} 2738.1836 \\ 2717.5034 \\ 3245.6313 \\ 3338.2234 \end{bmatrix} \end{array}, \begin{array}{c} a_2 \\ \begin{bmatrix} 2741.6379 \\ 2715.3396 \\ 3303.1980 \\ 3277.3660 \end{bmatrix} \end{array} \right\}$$



(a) SMDP, $k = 1$ (b) SMDP, $k = 2$

Fig. 6.2 Graphs of the $\alpha$-vectors for the value function in Example 6.10.2

In Fig. 6.2, we are able to display the $\alpha$-vectors for each of the SMDP models. Due to the structure of the problem, the agent knows with certainty which state they are in, but not the SMDP model. Thus, the belief is distributed over one state over two models. For instance, suppose that

$$\xi(\cdot, \cdot) = \begin{array}{c c} & \begin{array}{cc} k{=}1 & k{=}2 \end{array} \\ \begin{array}{c} s_1 \\ s_2 \end{array} & \begin{bmatrix} 0.9 & 0.1 \\ 0 & 0 \end{bmatrix} \end{array}$$

then,

$$\begin{aligned} V^*(\xi) &= \max \left\{ \begin{bmatrix} 0.9 & 0 & 0.1 & 0 \end{bmatrix} \begin{bmatrix} 2738.1836 \\ 2717.5034 \\ 3245.6313 \\ 3338.2234 \end{bmatrix}, \begin{bmatrix} 0.9 & 0 & 0.1 & 0 \end{bmatrix} \begin{bmatrix} 2741.6379 \\ 2715.3396 \\ 3303.1980 \\ 3277.3660 \end{bmatrix} \right\} \\ &= \max\{2788.9284, 2797.7939\} \\ &= 2797.7939. \end{aligned}$$

In this case, with the agent's belief of 90% in SMDP 1, the optimal decision is to do action 2.

◀

## 6.11 Learning the Continuous Parameter of the Sojourn-Time Distribution

In this section, we will propose a method for learning the continuous parameter of the sojourn-time distribution for each $(s, a, s')$ transition.

The Bellman equation is

$$V^*(s, M, \theta) = \max_{a \in \mathcal{A}} \left[ \bar{R}(s, a, M, \theta) + \sum_{s' \in \mathcal{S}} \bar{P}(s' \mid s, a, M) \int_0^\infty \mathrm{e}^{-\beta\tau} \bar{f}(\tau \mid s, a, s', \theta) V^*(s, \underbrace{M'}_{M+\delta_{s,a,s'}}, \overbrace{\theta'}^{\theta+\delta_{s,a,\tau,s'}}) \,\mathrm{d}\tau \right] \tag{6.11.1}$$

The reward function would be

$$\bar{R}(s, a, M, \theta) = r_1(s, a) + \sum_{s' \in \mathcal{S}} \bar{P}(s' \mid s, a, M) \int_0^\infty \left( \int_0^\tau \mathrm{e}^{-\beta\tau} r_2(t \mid s, a, s') \,\mathrm{d}\tau \right) \bar{F}(\mathrm{d}\tau \mid s, a, s', \theta). \tag{6.11.2}$$

Notice that while we assumed that the reward function is given as part of the model, we are building estimates for many of the terms. In this case, we only require to know what the lump sum $r_1(s, a)$ and the continuous reward rate $r_2(s, a, s')$.

For the state transition probability, we will follow the same as the BA-MDP in Sec. 6.6. Let $H$ be a Dirichlet distribution given by Eq. (6.6.5) with parameter $M$, then the expected state transition probability for $P(s' \mid s, a)$ is given by Eq. (6.6.12). In the array $\theta \in \mathbb{W}^{|\mathcal{S}|\times|\mathcal{A}|\times|\mathcal{S}|}$, each entry is the current estimate of the expected sojourn time $\hat{\mu}(s, a, s')$ for the transition $(s, a, s')$. Suppose that the agent observes a sequence of $m = M(s, a, s')$-many sojourn times $\tau_1, \tau_2, \ldots, \tau_m$ for transition $(s, a, s')$. As the agent observes each sojourn time, it will update a probability distribution on the mean sojourn time. To do this, we will assume that the sojourn times follow an inverse Gaussian distribution.

### 6.11.1 Our Conjugate Prior: The Gamma-Inverse Gaussian Conjugate Prior

In this section, we introduce our own conjugate prior on the parameter $\theta$ using the gamma and the inverse Gaussian distribution. The following definition is originally due to Barnard (1954).

**Definition 6.11.1 — Conjugate.** A family $\mathcal{F}$ of probability distributions is said to conjugate (or closed under sampling) for a likelihood function $f(x \mid \theta)$ if, for every $f \in \mathcal{F}$, the posterior distribution $f(\theta \mid x)$ also belongs to $\mathcal{F}$ (Robert, 2007, p. 114).

Informally, a family of probability distributions being conjugate or closed under sampling implies that when updating the posterior, the resulting distribution remains within the same family of distributions. This is accomplished by merely adjusting the distribution parameters with newly observed information. Their computational tractability has made conjugate families a popular choice for modelling (Robert, 2007, p. 115). This substantial simplification, which involves considering families of distributions closed under sampling, reduces the problem of sequential decision processes from a random walk in infinitely-many dimensions to a random walk in finitely-many dimensions (Wetherill, 1961, p. 283). The idea of parametric families closed under sampling has its origins in Bayesian statistical decision making (Barnard, 1954; Wetherill, 1961; Martin, 1965; Martin, 1967; DeGroot, 1970, p. 159), adaptive control processes (Bellman and Kalaba, 1959; Bellman, 1961a), and BRL (Duff, 2002; Ross *et al.*, 2008a).

We begin by using the gamma distribution as a prior to get the form of the posterior. Then, we will use that form as the prior candidate, and see if that prior candidate is closed under sampling.

Recall that the gamma distribution is

$$f(\theta \mid a, b) = \frac{b^a}{\Gamma(a)} \theta^{a-1} \mathrm{e}^{-b\theta}, \qquad \theta \in (0, \infty) \tag{6.11.3}$$

where $a > 0$ is the shape parameter, $b > 0$ is the rate parameter, and $\Gamma(\cdot)$ is the gamma function. The one-parameter form of the inverse Gaussian distribution is

$$f(\tau \mid \theta, \theta^2) = f(\tau \mid \theta) = \frac{\theta}{\sqrt{2\pi\tau^3}} \exp\left[-\frac{(\tau - \theta)^2}{2\tau}\right], \qquad \tau \in (0, \infty). \tag{6.11.4}$$

The support of the gamma function is $\theta \in (0, \infty)$, which is the same interval for the mean parameter of the inverse Gaussian; this makes the gamma distribution a good prior candidate.

The one-parameter inverse Gaussian distribution exhibits a key property where the expected value equals the variance; in other words, $\mathbf{E}(\tau) = \mathrm{var}(\tau)$. This characteristic is important in the context of timing perception in animals and humans, as it mimics Weber's Law for timing (Gibbon, 1977). According to *Weber's Law for timing*, the difference required to discern the perception of one time interval from another must be of a significantly greater order of magnitude. For example, when operating on a seconds-level time scale, the distinction between a bus arriving in 5 or 6 seconds would be imperceptible. However, the difference between a bus arriving in 5 seconds versus 500 seconds would be discernible. As the time scale progresses from seconds to minutes to hours, the precision of time perception grows proportionally. In probabilistic terms, this implies that as the expected sojourn time $\mathbf{E}(\tau)$ increases to longer durations, the error (standard deviation) must proportionally expand, and consequently impacts the variance $\mathrm{var}(\tau)$ as well. Remarkably, studies have shown that fish, birds, bees, rats, and even humans exhibit a linear scaling of error (standard deviation) with respect to the mean (Buhusi *et al.*, 2009).

Suppose that we observed a sojourn time $\tau_1$. The posterior distribution is proportional to the (sampling) likelihood distribution multiplied by prior distribution:

$$\begin{aligned}
&\text{posterior} \propto \text{likelihood} \times \text{prior} \\
&f(\theta \mid \tau_1) \propto f(\tau_1 \mid \theta) f(\theta) \\
&\qquad = \frac{\theta}{\sqrt{2\pi\tau_1^3}} \exp\left[-\frac{(\tau_1 - \theta)^2}{2\tau_1}\right] \frac{b^a}{\Gamma(a)} \theta^{a-1} \mathrm{e}^{-b\theta} \\
&\qquad = \frac{1}{\sqrt{2\pi\tau_1^3}} \frac{b^a}{\Gamma(a)} \theta^a \exp\left[-\frac{(\tau_1 - \theta)^2}{2\tau_1} - b\theta\right].
\end{aligned}$$

Now, let us use this posterior as the prior. We observe another sojourn time $\tau_2$, and we use this to update.

$$\begin{aligned} f(\theta \mid \tau_2) &\propto f(\tau_2 \mid \theta) f(\theta \mid \tau_1) \\ &= \frac{\theta}{\sqrt{2\pi\tau_2^3}} \exp\left[-\frac{(\tau_2-\theta)^2}{2\tau_2}\right] \frac{1}{\sqrt{2\pi\tau_1^3}} \frac{b^a}{\Gamma(a)} \theta^a \exp\left[-\frac{(\tau_1-\theta)^2}{2\tau_1} - b\theta\right] \\ &= \frac{1}{\sqrt{2\pi\tau_1^3}\sqrt{2\pi\tau_2^3}} \frac{b^a}{\Gamma(a)} \theta^{a+1} \exp\left[-\frac{(\tau_1-\theta)^2}{2\tau_1} - \frac{(\tau_2-\theta)^2}{2\tau_2} - b\theta\right] \end{aligned}$$

By induction,

$$\begin{aligned} f(\theta \mid \tau_1, \tau_2, \ldots, \tau_n) &\propto f(\tau_n \mid \theta) f(\theta \mid \tau_1, \tau_2, \ldots, \tau_{n-1}) \\ &= (2\pi)^{-\frac{n}{2}} \prod_{i=1}^{n} \tau_i^{-\frac{3}{2}} \frac{b^a}{\Gamma(a)} \theta^{a+n-1} \exp\left[-\frac{1}{2}\sum_{i=1}^{n} \frac{(\tau_i-\theta)^2}{\tau_i} - b\theta\right]. \end{aligned} \tag{6.11.5}$$

We can now find the maximum likelihood estimator of $\theta$ using our new likelihood function $f(\theta \mid \tau_1, \tau_2, \ldots, \tau_n)$. It is often more convenient to work with the (natural) logarithm of the likelihood function (Hogg *et al.*, 2013, p. 205), as it simplifies the differentiation process by dealing with additive terms rather than multiplicative terms. The (natural) logarithm function is a strictly increasing transformation on the interval $(0, \infty)$, which preserves the order of values as they are transformed. As a result, the value of $\theta$ that maximizes the transformed function $\ln f(\theta)$ will still maximize the original function $f(\theta)$. Given that the likelihood function always yields positive values, and using $K$ to denote the normalizing constant, we can take the (natural) logarithm of both sides so that

$$\begin{aligned} &\ln f(\theta \mid \tau_1, \tau_2, \ldots, \tau_n) \\ &= \ln\left(K(2\pi)^{-\frac{n}{2}} \prod_{i=1}^{n} \tau_i^{-\frac{3}{2}} \frac{b^a}{\Gamma(a)} \theta^{a+n-1} \exp\left[-\frac{1}{2}\sum_{i=1}^{n} \frac{(\tau_i-\theta)^2}{\tau_i} - b\theta\right]\right) \\ &= \ln\left(K(2\pi)^{-\frac{n}{2}} \prod_{i=1}^{n} \tau_i^{-\frac{3}{2}} \frac{b^a}{\Gamma(a)}\right) + (a+n-1)\ln\theta - \frac{1}{2}\sum_{i=1}^{n} \frac{(\tau_i-\theta)^2}{\tau_i} - b\theta. \end{aligned}$$

Now, we determine its critical value; that is, we will solve the equation

$$\frac{\partial \ln f}{\partial \theta} = 0.$$

So, we have

$$\begin{aligned} \frac{\partial \ln f}{\partial \theta} &= \frac{a+n-1}{\theta} + \frac{\tau_1-\theta}{\tau_1} + \frac{\tau_2-\theta}{\tau_2} + \cdots + \frac{\tau_n-\theta}{\tau_n} - b \\ &= \frac{a+n-1}{\theta} - b + \sum_{i=1}^{n} \frac{\tau_i-\theta}{\tau_i} \\ &= \frac{a+n-1}{\theta} - b + \sum_{i=1}^{n} \left(1 - \frac{\theta}{\tau_i}\right) \\ &= \frac{a+n-1}{\theta} - b + n - \theta \sum_{i=1}^{n} \frac{1}{\tau_i}. \end{aligned}$$

Setting it to zero and multiplying by $\theta$, we have

$$-\left(\sum_{i=1}^{n} \frac{1}{\tau_i}\right)\theta^2 + (n-b)\theta + (a+n-1) = 0. \tag{6.11.6}$$

Since the first term is negative, this is a concave downward quadratic function, which means that its critical point is a global maximum. Using the quadratic formula, we have

$$\hat{\theta} = \frac{n-b \pm \sqrt{(n-b)^2 + 4(a+n-1)\sum_{i=1}^{n} \frac{1}{\tau_i}}}{2\sum_{i=1}^{n} \frac{1}{\tau_i}} \tag{6.11.7}$$

Let us now write the maximum likelihood estimator $\hat{\theta}$ in a recursive form so that the agent does not need to store all the previous sojourn times; in other words, we would like an update rule that only requires $a$, $b$, $n$, the recent observed sojourn time $\tau_n$, and the estimate of $\hat{\theta}$ at $n-1$. Beginning with Eq. (6.11.6) at $n-1$, we have

$$-\left(\sum_{i=1}^{n-1} \frac{1}{\tau_i}\right)\theta_{n-1}^2 + (n-1-b)\theta_{n-1} + (a+n-2) = 0,$$

and solving for the sum of the reciprocal sojourn times, we have

$$\sum_{i=1}^{n-1} \frac{1}{\tau_i} = \frac{(n-1-b)\theta_{n-1} + a + n - 2}{\theta_{n-1}^2}. \tag{6.11.8}$$

Then, with the fact that

$$\sum_{i=1}^{n} \frac{1}{\tau_i} = \frac{1}{\tau_n} + \sum_{i=1}^{n-1} \frac{1}{\tau_i} \tag{6.11.9}$$

and using Eq. (6.11.7) at $n$,

$$\begin{aligned}\hat{\theta}_n &= \frac{n-b \pm \sqrt{(n-b)^2 + 4(a+n-1)\sum_{i=1}^{n} \frac{1}{\tau_i}}}{2\sum_{i=1}^{n} \frac{1}{\tau_i}} \\ &= \frac{n-b \pm \sqrt{(n-b)^2 + 4(a+n-1)\left(\frac{1}{\tau_n} + \sum_{i=1}^{n-1} \frac{1}{\tau_i}\right)}}{2\left(\frac{1}{\tau_n} + \sum_{i=1}^{n-1} \frac{1}{\tau_i}\right)}.\end{aligned}$$

Then, we can substitute Eq. (6.11.8) into $\hat{\theta}_n$, which yields

$$\hat{\theta}_n = \frac{n-b \pm \sqrt{(n-b)^2 + 4(a+n-1)\left(\frac{1}{\tau_n} + \frac{(n-1-b)\theta_{n-1} + a + n - 2}{\theta_{n-1}^2}\right)}}{2\left(\frac{1}{\tau_n} + \frac{(n-1-b)\theta_{n-1} + a + n - 2}{\theta_{n-1}^2}\right)}. \tag{6.11.10}$$

To ensure that $\hat{\theta}_n$ is nonnegative (since $\theta \in (0, \infty)$), and to avoid a complex radical, we require the following two conditions:

(a) $n + a - 2 > 0$; and
(b) $n - b - 1 > 0$.

For the initial value $\theta_0$, we start just with the gamma distribution with parameters $a > 0$ and $b > 0$. So,

$$f(\theta_0) = \frac{b^a}{\Gamma(a)}\theta_0^{a-1}\mathrm{e}^{-b\theta_0}$$

Then, we take the (natural) logarithm

$$\ln f(\theta_0) = a\ln(b\theta_0) - \ln(\theta_0\Gamma(a)) - b\theta_0,$$

and by taking the derivative with respect to $\theta_0$, we have

$$\frac{\partial \ln f}{\partial\theta_0} = \frac{a - b\theta_0 - 1}{\theta_0},$$

and setting it to zero to find the critical point gives

$$\frac{a - b\theta_0 - 1}{\theta_0} = 0$$

$$\theta_0 = \frac{a-1}{b}. \tag{6.11.11}$$

Modelling priors is crucial when working with limited data, but as the sample size increases, their impact on the inference process diminishes. As sample size grows, the likelihood function becomes the primary source of information for inference. (Robert, 2007, p. 124).

**Example 6.11.2** In this example, we use 16 random sojourn times generated by the (one-parameter) inverse Gaussian distribution $\mathcal{IG}(\mu = 3, \mu^2 = 3^2)$ as listed in Table 5.1. We begin with initial gamma parameters of $a = 3$ and $b = 2$, and calculate the estimate $\hat{\theta}$ for each new sample observed, as shown in Table 6.1.

Table 6.1 Calculations for $\hat{\theta}$.

| $n$ | $\tau_i$ | $\frac{1}{\tau_i}$ | $\sum_{i=1}^{n}\frac{1}{\tau_i}$ | $\hat{\theta_n}$ |
|---|---|---|---|---|
| 0 | | | | 1.0000 |
| 1 | 1.4588 | 0.6855 | 0.6855 | 1.4861 |
| 2 | 5.9780 | 0.1673 | 0.8528 | 2.1658 |
| 3 | 2.3113 | 0.4327 | 1.2854 | 2.3992 |
| 4 | 2.4519 | 0.4078 | 1.6933 | 2.5634 |
| 5 | 1.9821 | 0.5045 | 2.1978 | 2.5932 |
| 6 | 2.0355 | 0.4913 | 2.6891 | 2.6221 |
| 7 | 1.6069 | 0.6223 | 3.3114 | 2.5682 |
| 8 | 3.5954 | 0.2781 | 3.5895 | 2.7024 |
| 9 | 3.7644 | 0.2656 | 3.8552 | 2.8256 |
| 10 | 4.3149 | 0.2318 | 4.0869 | 2.9521 |
| 11 | 2.4975 | 0.4004 | 4.4873 | 2.9784 |
| 12 | 1.8990 | 0.5266 | 5.0139 | 2.9432 |
| 13 | 1.1870 | 0.8425 | 5.8564 | 2.7948 |
| 14 | 1.3712 | 0.7293 | 6.5857 | 2.7165 |
| 15 | 2.4073 | 0.4154 | 7.0011 | 2.7423 |
| 16 | 4.5663 | 0.2190 | 7.2201 | 2.8224 |

Using either Eq. (6.11.7) or the recursive form in Eq. (6.11.10) yields the same result. As we can see in Fig. 6.3, the conjugate prior distribution (Eq. (6.11.5)) evolves and converges around $\theta = 3$ as more sojourn times are observed, with deeper red

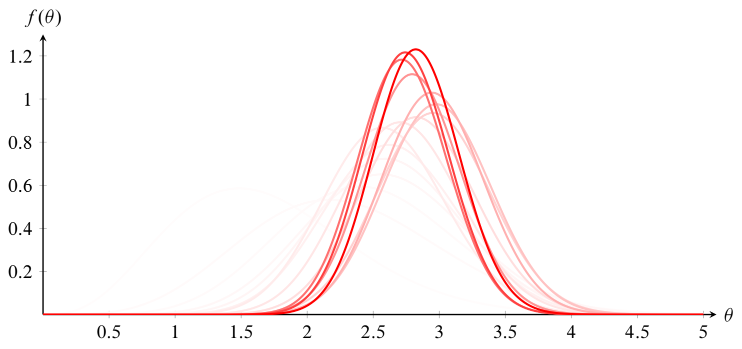


Fig. 6.3 The conjugate prior distribution on the $\mu$ mean parameter of the inverse Gaussian distribution $\mathcal{IG}(3,9)$. Initial parameters were $a = 3$ and $b = 2$. The darker red indicates higher sample counts.

indicating a higher number of observations. The Matlab code to produce all the curves in Figure 6.3 is available in Appendix A.4.

◀

The Bellman equation is

$$V^*(s, M, \theta) = \max_{a\in\mathcal{A}} \Big\{\bar{R}(s, a, M, \theta) +$$
$$\sum_{s'\in\mathcal{S}} \bar{P}(s' \mid s, a, M) \int_0^\infty e^{-\beta\tau} \bar{f}(\tau \mid s, a, s', \theta) \int_0^\infty f(\theta' \mid \theta, \tau) V^*(s', M', \theta')\, d\tau\, d\theta \Big\} \tag{6.11.12}$$

Now, we can use the maximum likelihood estimator for the next $\theta'$. This means that we do not need to calculate the integral for the weighting of every possible $\theta$.

$$V^*(s, M, \theta) = \max_{a\in\mathcal{A}} \Big\{\bar{R}(s, a, M, \theta) +$$
$$\sum_{s'\in\mathcal{S}} \bar{P}(s' \mid s, a, M) \int_0^\infty e^{-\beta\tau} \bar{f}(\tau \mid s, a, s', \theta) V^*(s', M', \theta')\, d\tau \Big\}$$

Next, we apply importance sampling to deal with the integral. This integral is quite difficult since it requires the next state value. To approximate the integral,

$$\mu(s, a, s', M, \theta) = \int_0^\infty e^{-\beta\tau} \bar{f}(\tau \mid s, a, s', \theta) V^*(s', M', \theta')\, d\tau \tag{6.11.13}$$

we use

$$\hat{\mu}(s, a, s', M, \theta) = \frac{1}{|C|} \sum_{n=1}^{|C|} e^{-\beta\tau_n} \frac{\bar{f}(\tau_n \mid s, a, s', \theta)}{D(\tau_n)} V^*(s', M', \theta') \tag{6.11.14}$$

where the function $D(\tau_n)$ is given by

$$\begin{aligned} D(\tau_n) &= \sum_{s\in\mathcal{S}} \sum_{a\in\mathcal{A}} \sum_{s'\in\mathcal{S}} w(s, a, s') \bar{f}(\tau \mid s, a, s', \theta) \\ &= \sum_{s\in\mathcal{S}} \sum_{a\in\mathcal{A}} \sum_{s'\in\mathcal{S}} \frac{M(s, a, s')}{\sum_{s''\in\mathcal{S}} M(s, a, s'')} \bar{f}(\tau \mid s, a, s', \theta) \end{aligned}$$

Replacing the integral with the approximation yields

$$V^*(s, M, \theta) = \max_{a \in \mathcal{A}} \Bigg\{ \bar{R}(s, a, M, \theta) + \sum_{s' \in \mathcal{S}} \bar{P}(s' \mid s, a, M) \frac{1}{|C|} \sum_{n=1}^{|C|} \mathrm{e}^{-\beta\tau_n} \frac{\bar{f}(\tau_n \mid s, a, s', \theta)}{D(\tau_n)} V^*(s', M', \theta') \Bigg\} \tag{6.11.15}$$

and since $\frac{1}{|C|} \sum_{n=1}^{|C|} \mathrm{e}^{-\beta\tau_n}$ does not depend on $s'$, we can move this in front of the summation of $s'$

$$V^*(s, M, \theta) = \max_{a \in \mathcal{A}} \Bigg\{ \bar{R}(s, a, M, \theta) + \frac{1}{|C|} \sum_{n=1}^{|C|} \mathrm{e}^{-\beta\tau_n} \sum_{s' \in \mathcal{S}} \bar{P}(s' \mid s, a, M) \frac{\bar{f}(\tau_n \mid s, a, s', \theta)}{D(\tau_n)} V^*(s', M', \theta') \Bigg\} \tag{6.11.16}$$

After seeing that in Sec. 6.10 it is possible to formulate a BA-SMDP as a POSMDP, we can answer in the affirmative that we can express the BA-SMDP as a POSMDP, but this is a special case. In that instance, we presupposed the existence of a finite set of known SMDPs and then aggregated these into a single POSMDP, in which the agent builds a belief state over these SMDPs. However, in this section, we shall examine the general scenario in which our uncertainty is described through distributions on state transition probabilities and distributions on parameters of the sojourn-time distribution, as opposed to a finite set of known parameters. We will carry out a similar analysis of BA-SMDPs as we did with POSMDPs in Sec. 5.10, but we will have to generalize the notion of $\alpha$-vectors to $\alpha$-functions. Generalizing from $\alpha$-vectors to $\alpha$-functions was described in Duff (2002) and extended to continuous POMDPs in Porta *et al.* (2005). Porta *et al.* (2006) describes how to deal with continuous observations, and while time maybe considered a form of continuous observations, Porta *et al.* use linear combinations of Gaussian distributions for their models which is not a great family of distributions to use for sojourn times.

For POSMDPs, the state $s \in \mathcal{S}$ is partially observable where the state space is a finite set. While with BA-SMDPs, the unknown are the parameters for the transition probabilities and the sojourn-time distributions; the space of unknown parameters in a point in a compact subset of $\mathbb{R}^{|\mathcal{S}|\times|\mathcal{A}|\times|\mathcal{S}|\times|\Theta|}$.

For POSMDPs, the observations are generated by $G(o \mid a, s')$, a distribution over observations given by action $a$ and state $s'$. However, BA-SMDP observations are state transitions $s, a, \tau, s'$ (state $s$, action $a$, and sojourn time $\tau$ resulting in transition to state $s'$) which is generated by

$$P(s' \mid s, a) F(\tau \mid s, a, s').$$

For POSMDPs, the belief state $\xi$ is a distribution over $|\mathcal{S}|$ possible states. For BA-SMDPs, the information state is $\langle \mathrm{d}H(P \mid M), \mathrm{d}U(f \mid M, \theta) \rangle$ is a distribution (density) over possible probability transition arrays and a distribution over the possible sojourn-time distribution parameters.

For POSMDPs, the belief state update in light of an action $a$, sojourn time $\tau$, and observation $o$ is

$$\xi(s' \mid a, \tau, o) = \frac{G(o \mid a, s') \sum_{s \in \mathcal{S}} P(s' \mid s, a) F(\tau \mid s, a, s') \xi(s)}{\sum_{s'' \in \mathcal{S}} G(o \mid a, s'') \sum_{s \in \mathcal{S}} P(s'' \mid s, a) F(\tau \mid s, a, s'') \xi(s)}, \tag{6.11.17}$$

which we write abstractly as

$$\xi' = T(\xi \mid a, \tau, o). \tag{6.11.18}$$

This is a Bayes' update that works on a (discrete) probability mass function and changes the probabilities of each state accordingly.

For BA-SMDPs, the update in light of

(a) an observed state transition $(s, a, s')$

$$\mathrm{d}H(P \mid s, a, s', M) = \frac{P(s' \mid s, a)\,\mathrm{d}H(P \mid M)}{\displaystyle\int_P P(s' \mid s, a)\,\mathrm{d}H(P \mid M)}. \tag{6.11.19}$$

This is a Bayes' update that works on a continuous probability distribution and adjusts the curve accordingly.

If $H$ is closed under sampling, then we write abstractly as

$$M' = T(M \mid s, a, s'). \tag{6.11.20}$$

and

$$\mathrm{d}H(P \mid M') = \mathrm{d}H(P \mid s, a, s', M) \tag{6.11.21}$$

(b) an observed sojourn time $\tau$ for a particular state transition $(s, a, s')$

$$\mathrm{d}U(f \mid s, a, \tau, s', M, \theta) = \frac{f(\tau \mid s, a, s')\,\mathrm{d}U(f \mid \theta)}{\displaystyle\int_f f(\tau \mid s, a, s')\,\mathrm{d}U(f \mid \theta)}. \tag{6.11.22}$$

Again, this is a Bayes' update that works on a continuous probability distribution and adjusts the sojourn-time distribution curve accordingly.

If $U$ is closed under sampling, then we write abstractly as

$$\theta' = T(\theta \mid s, a, \tau, s', M). \tag{6.11.23}$$

and

$$\mathrm{d}U(f \mid \theta') = \mathrm{d}U(f \mid s, a, \tau, s', M, \theta) \tag{6.11.24}$$

(We have shown that the gamma inverse-Gaussian prior is closed, and to get from $\theta$ to $\theta'$, we can use the maximum likelihood estimator.)

The probability of transitioning from state $s$ to state $s'$ after performing action $a$ with respect to the prior is

$$\bar{P}(s' \mid s, a, M) = \int_P P(s' \mid s, a)\,\mathrm{d}H(P \mid M) \tag{6.11.25}$$

If $H$ is a Dirichlet distribution with parameter $M$, then

$$\bar{P}(s' \mid s, a, M) = \frac{M(s, a, s')}{\displaystyle\sum_{s'' \in \mathcal{S}} M(s, a, s'')}. \tag{6.11.26}$$

The sojourn-time $\tau$ for transitioning from state $s$ to state $s'$ after performing action $a$ with respect to the prior is

$$\bar{f}(\tau \mid s, a, s', \theta) = \int_f f(\tau \mid s, a, s')\,\mathrm{d}U(f \mid \theta) \tag{6.11.27}$$

If $U$ is the gamma inverse-Gaussian conjugate prior (derived in Sec. 6.11.1) with parameter $\theta$, then

$$\bar{f}(\tau \mid s, a, s', \theta) = \theta(s, a, s'). \tag{6.11.28}$$

The value function of a POSMDP is a piecewise-linear convex function for a finite planning horizon. To achieve a similar outcome for the BA-SMDP, we will use $\alpha$-functions instead of $\alpha$-vectors. These $\alpha$-functions serve the same purpose as sufficient statistics, which captures the necessary information to calculate the value function.

The $\alpha$-function, denoted by

$$\alpha(s, P, f)$$

where $s$ is a state, $P$ is an array of state transition probabilities for all $(s, a, s')$, and $f$ is all sojourn-time distributions for every $(s, a, s')$ transition. The set of all $\alpha$-functions is represented as $V$, where

$$V = \{\alpha_1, \alpha_2, \alpha_3, \ldots\}.$$

As the value function changes at each decision epoch $n$ due to new observations, we will denote $V_n$ as the set of $\alpha$-functions that describe the value function at that stage.

**Theorem 6.11.3** The value function of a BA-SMDP at decision epoch $n$ for hyperstate $\big(s, \mathrm{d}H(P \mid M), \mathrm{d}U(f \mid \theta)\big)$ is

$$\begin{aligned} &V_n\big(s, \mathrm{d}H(P \mid M), \mathrm{d}U(f \mid \theta)\big) \\ &\qquad = \max_{\alpha \in V_n} \left\{ \int_f \int_P \alpha(s, P, f)\, \mathrm{d}H(P \mid M)\, \mathrm{d}U(f \mid \theta) \right\}. \end{aligned} \tag{6.11.29}$$

where $\alpha$-functions on hyperstate $(s, P, f)$ are defined by

$$\begin{aligned} \alpha(s, P, f) = \sum_{s' \in \mathcal{S}} P(s' \mid s, a) \int_0^\infty & \mathrm{e}^{-\beta\tau} f(\tau \mid s, a, s')\big[R(s, a, s') \\ &+ \alpha^*_{n+1}(s, a, \tau, s', M, \theta)\big]\, \mathrm{d}\tau \end{aligned} \tag{6.11.30}$$

**Proof** We begin with the definition of the value function being the maximum inner product between an $\alpha$-function and a hyperstate. At decision epoch $n$, the value function for a particular state is defined by

$$V_n\big(s, \mathrm{d}H(P \mid M), \mathrm{d}U(f \mid \theta)\big) = \max_{\alpha \in V_n} \int_f \int_P \alpha(s, P, f)\, \mathrm{d}H(P \mid M)\, \mathrm{d}U(f \mid \theta). \tag{6.11.31}$$

The value function is based on a double integral (instead of a summation with POSMDPs using $\alpha$-vectors). Now, what remains to be shown is what set of appropriate $\alpha$-functions are required to satisfy Eq. (6.11.31).

The initial value function $V_0$ is the set with a single $\alpha$-function; that is,

$$V_0 = \{\alpha_0\}. \tag{6.11.32}$$

We will define the initial $\alpha_0$-function to be

$$\alpha_0 \triangleq R_0(s, P, f) \tag{6.11.33}$$

where $R_0(s, P, f)$ is the initial reward or value assigned to the state $(s, P, f)$. Then using Eq. (6.11.31),

$$\begin{aligned} V_0\big(s, \mathrm{d}H(P \mid M), \mathrm{d}U(f \mid \theta)\big) &= \max_{\{\alpha_0\}} \int_f \int_P \alpha_0(s, P, f)\, \mathrm{d}H(P \mid M)\, \mathrm{d}U(f \mid \theta) \\ &= \int_f \int_P R_0(s, P, f)\, \mathrm{d}H(P \mid M)\, \mathrm{d}U(f \mid \theta). \end{aligned} \tag{6.11.34}$$

For our inductive hypothesis, using Eq. (6.11.31), suppose that we can write the value function for hyperstate $\big(s', \mathrm{d}H(P \mid M'), \mathrm{d}U(f \mid \theta')\big)$ at decision epoch $n+1$ as

$$V_{n+1}\big(s', \mathrm{d}H(P \mid M'), \mathrm{d}U(f \mid \theta')\big) = \max_{\alpha \in V_{n+1}} \int_f \int_P \alpha(s', P, f)\, \mathrm{d}H(P \mid M')\, \mathrm{d}U(f \mid \theta') \quad (6.11.35)$$

As we vary $M'$ and $\theta'$, different $\alpha$-functions will have the largest value with the densities $\mathrm{d}H$ and $\mathrm{d}U$. Now, we substitute Eq. (6.11.19) and Eq. (6.11.22) into Eq. (6.11.35) to get

$$\begin{aligned} &V_{n+1}\big(s', \mathrm{d}H(P \mid M'), \mathrm{d}U(f \mid \theta')\big) \\ &= \max_{\alpha \in V_{n+1}} \Bigg\{ \int_f \int_P \alpha(s', P, f) \\ &\qquad \frac{P(s' \mid s, a)\, \mathrm{d}H(P \mid M)}{\int_P P(s' \mid s, a)\, \mathrm{d}H(P \mid M)} \frac{f(\tau \mid s, a, s')\, \mathrm{d}U(f \mid \theta)}{\int_f f(\tau \mid s, a, s')\, \mathrm{d}U(f \mid \theta)} \Bigg\} \end{aligned} \quad (6.11.36)$$

We define the optimal $\alpha$-function at decision epoch $n+1$ for a particular transition $(s, a, \tau, s')$ given the (sufficient statistic) parameters $M$ and $\theta$ by

$$\begin{aligned} &\alpha^*_{n+1}(s, a, \tau, s', M, \theta) \\ &= \arg\max_{\alpha \in V_{n+1}} \Bigg\{ \int_f \int_P \alpha(s', P, f) \\ &\qquad \frac{P(s' \mid s, a)\, \mathrm{d}H(P \mid M)}{\int_P P(s' \mid s, a)\, \mathrm{d}H(P \mid M)} \frac{f(\tau \mid s, a, s')\, \mathrm{d}U(f \mid \theta)}{\int_f f(\tau \mid s, a, s')\, \mathrm{d}U(f \mid \theta)} \Bigg\} \end{aligned} \quad (6.11.37)$$

so that

$$V_{n+1}\big(s', \mathrm{d}H(P \mid M'), \mathrm{d}U(f \mid \theta')\big) \quad (6.11.38)$$

$$= \frac{\int_f \int_P \alpha^*_{n+1}(s, a, \tau, s', M, \theta) P(s' \mid s, a) f(\tau \mid s, a, s')\, \mathrm{d}H(P \mid M)\, \mathrm{d}U(f \mid \theta)}{\int_f \int_P P(s' \mid s, a) f(\tau \mid s, a, s')\, \mathrm{d}H(P \mid M)\, \mathrm{d}U(f \mid \theta)} \quad (6.11.39)$$

$$= \frac{\int_f \int_P \alpha^*_{n+1}(s, a, \tau, s', M, \theta) P(s' \mid s, a) f(\tau \mid s, a, s')\, \mathrm{d}H(P \mid M)\, \mathrm{d}U(f \mid \theta)}{\bar{P}(s' \mid s, a, M) \bar{f}(\tau \mid s, a, s', \theta)} \quad (6.11.40)$$

At this moment, we have presented $V_n$ and $V_{n+1}$ in terms of $\alpha$-functions, but we have yet to define the $\alpha$-function formally. To accomplish this, we will revisit the original form of the Bellman equation, substitute in $V_{n+1}$, and simplify the expression to derive a representation of the $\alpha$-function.

The Bellman equation for decision epoch $n$ is

$$\begin{aligned}
&V_n\big(s, \mathrm{d}H(P \mid M), \mathrm{d}U(f \mid \theta)\big) \\
&= \max_{a \in \mathcal{A}(s)} \Bigg\{ \sum_{s' \in \mathcal{S}} \bar{P}(s' \mid s, a, M) \int_0^\infty \mathrm{e}^{-\beta\tau} \bar{f}(\tau \mid s, a, s', \theta) \\
&\qquad \Big[ R(s, a, s') + V_{n+1}\big(s', \mathrm{d}H(P \mid M'), \mathrm{d}U(f \mid \theta')\big) \Big] \mathrm{d}\tau \Bigg\} \qquad (6.11.41) \\
&= \max_{a \in \mathcal{A}(s)} \Bigg\{ \sum_{s' \in \mathcal{S}} \bar{P}(s' \mid s, a, M) \int_0^\infty \mathrm{e}^{-\beta\tau} \bar{f}(\tau \mid s, a, s', \theta) \, \mathrm{d}\tau \; R(s, a, s') + \\
&\qquad \sum_{s' \in \mathcal{S}} \bar{P}(s' \mid s, a, M) \int_0^\infty \mathrm{e}^{-\beta\tau} \bar{f}(\tau \mid s, a, s', \theta) \\
&\qquad\qquad V_{n+1}\big(s', \mathrm{d}H(P \mid M'), \mathrm{d}U(f \mid \theta')\big) \, \mathrm{d}\tau \Bigg\} \qquad (6.11.42)
\end{aligned}$$

Substituting Eq. (6.11.40) for $V_{n+1}$ in the above leads to

$$\begin{aligned}
&V_n\big(s, \mathrm{d}H(P \mid M), \mathrm{d}U(f \mid \theta)\big) \\
&= \max_{a \in \mathcal{A}(s)} \Bigg\{ \sum_{s' \in \mathcal{S}} \bar{P}(s' \mid s, a, M) \int_0^\infty \mathrm{e}^{-\beta\tau} \bar{f}(\tau \mid s, a, s', \theta) \, \mathrm{d}\tau \; R(s, a, s') + \\
&\qquad \sum_{s' \in \mathcal{S}} \cancel{\bar{P}(s' \mid s, a, M)} \int_0^\infty \mathrm{e}^{-\beta\tau} \cancel{\bar{f}(\tau \mid s, a, s', \theta)} \\
&\frac{\int_f \int_P \alpha^*_{n+1}(s, a, \tau, s', M, \theta) P(s' \mid s, a) f(\tau \mid s, a, s') \, \mathrm{d}H(P \mid M) \, \mathrm{d}U(f \mid \theta)}{\cancel{\bar{P}(s' \mid s, a, M)} \cancel{\bar{f}(\tau \mid s, a, s', \theta)}} \, \mathrm{d}\tau \Bigg\} \\
&\qquad\qquad (6.11.43) \\
&= \max_{a \in \mathcal{A}(s)} \Bigg\{ \sum_{s' \in \mathcal{S}} \bar{P}(s' \mid s, a, M) \int_0^\infty \mathrm{e}^{-\beta\tau} \bar{f}(\tau \mid s, a, s', \theta) \, \mathrm{d}\tau \; R(s, a, s') + \\
&\qquad \int_f \int_P \sum_{s' \in \mathcal{S}} P(s' \mid s, a) \int_0^\infty \mathrm{e}^{-\beta\tau} f(\tau \mid s, a, s') \alpha^*_{n+1}(s, a, \tau, s', M, \theta) \, \mathrm{d}\tau \\
&\qquad\qquad \mathrm{d}H(P \mid M) \, \mathrm{d}U(f \mid \theta) \Bigg\} \\
&\qquad\qquad (6.11.44)
\end{aligned}$$

We expand $\bar{P}(s' \mid s, a, M)$ and $\bar{f}(\tau \mid s, a, s', \theta)$ and factor to get

$$\begin{aligned}
&V_n\big(s, \mathrm{d}H(P \mid M), \mathrm{d}U(f \mid \theta)\big) \\
&= \max_{a \in \mathcal{A}(s)} \Bigg\{ \int_f \int_P \sum_{s' \in \mathcal{S}} P(s' \mid s, a) \int_0^\infty \mathrm{e}^{-\beta\tau} f(\tau \mid s, a, s')\, \mathrm{d}\tau \\
&\qquad R(s, a, s')\, \mathrm{d}H(P \mid M)\, \mathrm{d}U(f \mid \theta) + \int_f \int_P \sum_{s' \in \mathcal{S}} P(s' \mid s, a) \\
&\qquad \int_0^\infty \mathrm{e}^{-\beta\tau} f(\tau \mid s, a, s') \alpha^*_{n+1}(s, a, \tau, s', M, \theta)\, \mathrm{d}\tau\, \mathrm{d}H(P \mid M)\, \mathrm{d}U(f \mid \theta) \Bigg\}
\end{aligned} \tag{6.11.45}$$

$$\begin{aligned}
&= \max_{a \in \mathcal{A}(s)} \Bigg\{ \int_f \int_P \sum_{s' \in \mathcal{S}} P(s' \mid s, a) \int_0^\infty \mathrm{e}^{-\beta\tau} f(\tau \mid s, a, s') \\
&\qquad \big[R(s, a, s') + \alpha^*_{n+1}(s, a, \tau, s', M, \theta)\big]\, \mathrm{d}\tau\, \mathrm{d}H(P \mid M)\, \mathrm{d}U(f \mid \theta) \Bigg\}
\end{aligned} \tag{6.11.46}$$

If we define $\alpha(s, P, f)$ as

$$\begin{aligned}
\alpha(s, P, f) = \sum_{s' \in \mathcal{S}} P(s' \mid s, a) \int_0^\infty \mathrm{e}^{-\beta\tau} f(\tau \mid s, a, s') \big[R(s, a, s') \\
+ \alpha^*_{n+1}(s, a, \tau, s', M, \theta)\big]\, \mathrm{d}\tau
\end{aligned} \tag{6.11.47}$$

then we have our appropriate set of $\alpha$-functions, and

$$\begin{aligned}
&V_n\big(s, \mathrm{d}H(P \mid M), \mathrm{d}U(f \mid \theta)\big) \\
&\qquad = \max_{\alpha \in V_n} \Bigg\{ \int_f \int_P \alpha(s, P, f)\, \mathrm{d}H(P \mid M)\, \mathrm{d}U(f \mid \theta) \Bigg\},
\end{aligned} \tag{6.11.48}$$

which completes our proof. ◀

Equation (6.11.48) along with Eq. (6.11.47) and Eq. (6.11.37) defines the value function $V_n\big(s, \mathrm{d}H(P \mid M), (f \mid \theta)\big)$ around some local neighbourhood of some selected information state.

To summarize, for a hyperstate $(s, M, \theta)$ and a set of $\alpha$-functions that define $V_{n+1}$, we are doing the following:

(a) Find the optimal $\alpha^*_{n+1}(s, a, \tau, s', M, \theta)$ in the set $V_{n+1}$ using Eq. (6.11.37) for all possible action $a \in \mathcal{A}$ and $\tau \in (0, \infty)$.

(b) The summation over $s' \in \mathcal{S}$ and the integral over $\tau \in (0, \infty)$ in Eq. (6.11.46) marginalizes the red term's dependence on $s'$ and $\tau$, and for a fixed initial state $s$ what remains is a function of $P$ and $f$ for each choice of action $a$.

(c) Selecting the action $a$ that maximizes the inner product of the term with the densities $\mathrm{d}H(P \mid M)$ and $\mathrm{d}U(f \mid \theta)$ fixes $a$—for fixed state $s$, $M$ and $\theta$, the newly constructed $\alpha_n(s, P, M)$ is a mixture (over $s'$ and $\tau$) of functions of the form $R(s, a, s') + \alpha^*_{n+1}(s', P, f)$ where $a$ is the optimal action as selected in Eq. (6.11.47). This action $a$ is associated with the newly constructed $\alpha_n$-function.

## 6.12 Convergence

In this section, we show that the BA-SMDP value iteration converges to the optimal BA-SMDP value function by using a recent result by Ren and Stachurski (2021).

Ren and Stachurski (2021) introduces the idea of value convexity. To begin with, we need to define the following. Let $\mathbb{R}(\mathcal{S}_{\text{BA}})$ be the set of all functions from the metric space $\mathcal{S}_{\text{BA}}$ to $\mathbb{R}$, and let

$$J : \mathcal{S}_{\text{BA}} \times \mathcal{A} \times \mathbb{R}(\mathcal{S}_{\text{BA}}) \to \mathbb{R} \tag{6.12.1}$$

be a given mapping.

We need to show that the BA-SMDP hyperstate is a metric space. A hyperstate for a BA-SMDP is

$$(s, \mathrm{d}H(P \mid M), \mathrm{d}U(f \mid \theta)) \triangleq s_{\text{BA}},$$

where

| | |
|---|---|
| $s$ | is a physical state from the state space $\mathcal{S}$; |
| $\mathrm{d}H$ | is the density over $[0,1]^{\lvert\mathcal{S}\rvert\times\lvert\mathcal{A}\rvert\times\lvert\mathcal{S}\rvert}$ that describes the uncertainty of the state transition probability for each $(s, a, s')$ transition; and |
| $\mathrm{d}U$ | is the density over $(0,\infty)^{\lvert\mathcal{S}\rvert\times\lvert\mathcal{A}\rvert\times\lvert\mathcal{S}\rvert}$ that describes the uncertainty of the sojourn-time distribution for each $(s, a, s')$ transition. |

This means that the hyperstate space is

$$\mathcal{S} \times [0,1]^{\lvert\mathcal{S}\rvert\times\lvert\mathcal{A}\rvert\times\lvert\mathcal{S}\rvert} \times (0,\infty)^{\lvert\mathcal{S}\rvert\times\lvert\mathcal{A}\rvert\times\lvert\mathcal{S}\rvert} \triangleq \mathcal{S}_{\text{BA}}. \tag{6.12.2}$$

**Lemma 6.12.1** If $\mathcal{S}$ with metric $d_{\mathcal{S}}$ is a metric space, then the hyperstate space $\mathcal{S}_{\text{BA}}$ is a metric space with metric $d_{\mathcal{S}_{\text{BA}}}$.

**Proof** We will consider each component of the hyperstate:

(a) If $\mathcal{S}$ is discrete or finite, then $(\mathcal{S}, d_{\mathcal{S}})$ is a metric space, where $d_{\mathcal{S}}$ is the discrete metric defined by

$$d_{\mathcal{S}}(s, s') = \begin{cases} 1, & \text{if } s \neq s' \\ 0, & \text{if } s = s'. \end{cases}$$

(b) The closed interval $[0,1]$ is a complete metric space with the absolute-value metric defined by

$$d_{[0,1]}(x, y) = \lvert x - y \rvert.$$

For the space underlying $h$, it is comprised of the $[0,1]^{\lvert\mathcal{S}\rvert\times\lvert\mathcal{A}\rvert\times\lvert\mathcal{S}\rvert}$, which is a Cartesian product of finitely many metric spaces, so it is a product metric space, where the metric would be defined by

$$\begin{aligned} d_{[0,1]^{\lvert\mathcal{S}\rvert\times\lvert\mathcal{A}\rvert\times\lvert\mathcal{S}\rvert}}(h, h') &= \sqrt{\sum_{s\in\mathcal{S}}\sum_{a\in\mathcal{A}}\sum_{s'\in\mathcal{S}} d_{[0,1]}\big(h(s,a,s'), h'(s,a,s')\big)^2} \\ &= \sqrt{\sum_{s\in\mathcal{S}}\sum_{a\in\mathcal{A}}\sum_{s'\in\mathcal{S}} \lvert h(s,a,s') - h'(s,a,s')\rvert^2}. \end{aligned}$$

The open interval $(0,\infty)$ is an incomplete[1] metric space where the metric would be defined by

$$d_{(0,\infty)}(x, y) = \lvert x - y \rvert,$$

[1] It is not complete since not all Cauchy sequences converge inside of the interval. If we consider $(0,\infty)$ with the absolute metric, then the Cauchy sequence $1/n \to 0$ as $n \to \infty$, but $0 \notin (0,\infty)$.

and

$$d_{(0,\infty)^{|\mathcal{S}|\times|\mathcal{A}|\times|\mathcal{S}|}}(u,u') = \sqrt{\sum_{s\in\mathcal{S}}\sum_{a\in\mathcal{A}}\sum_{s'\in\mathcal{S}} d_{(0,\infty)}\big(u(s,a,s'),u'(s,a,s')\big)^2}$$
$$= \sqrt{\sum_{s\in\mathcal{S}}\sum_{a\in\mathcal{A}}\sum_{s'\in\mathcal{S}} |u(s,a,s') - u'(s,a,s')|^2}.$$

We introduce $s_{\mathrm{BA}} = (s,h,u)$ and $s'_{\mathrm{BA}} = (s',h',u')$. Then, $\mathcal{S}_{\mathrm{BA}}$ is a metric space since it is a Cartesian product of finitely many metric spaces, and the metric for $\mathcal{S}_{\mathrm{BA}}$ is

$$d_{\mathcal{S}_{\mathrm{BA}}}(s_{\mathrm{BA}},s'_{\mathrm{BA}}) = \sqrt{d_{\mathcal{S}}(s,s')^2 + d_{[0,1]^{|\mathcal{S}|\times|\mathcal{A}|\times|\mathcal{S}|}}(h,h')^2 + d_{(0,\infty)^{|\mathcal{S}|\times|\mathcal{A}|\times|\mathcal{S}|}}(u,u')^2}.$$

◀

Ren and Stachurski (2021) then introduce the following definition.

**Definition 6.12.2 — Value-Convex.** A mapping $J$ is called value-convex if for all $(s,a) \in \mathbb{K}$, $\lambda \in [0,1]$, and $V_1$ and $V_2$ are value functions, then

$$J\big(s,a,\lambda V_1 + (1-\lambda)V_2\big) \leq \lambda J(s,a,V_1) + (1-\lambda)J(s,a,V_2).$$

For a BA-SMDP, consider the mapping

$$J\big(s, \mathrm{d}H(P \mid M), \mathrm{d}U(f \mid \theta), a, V\big)$$
$$= R(s,a,M,\theta) + \int_f\int_P \sum_{s'\in\mathcal{S}} \bar{P}(s' \mid s,a,M)\int_0^\infty \mathrm{e}^{-\beta\tau}\bar{f}(\tau \mid s,a,s',\theta)$$
$$V\big(s', \mathrm{d}H(P \mid H'), \mathrm{d}U(f \mid \theta')\big).$$

If $J$ is assumed to be value-convex, and has an upper bound, then we can use the following result.

**Theorem 6.12.3 — Ren and Stachurski (2021).** If the mapping $J$ is value-convex, and there exists an $\epsilon > 0$ such that

$$J(s,a,\check{V}) \leq \check{V}(s) - \epsilon \qquad \forall (s,a) \in \mathcal{K}$$

where $\check{V}$ is an upper bound, then
(a) The Bellman operator is geometrically stable on the set of all possible value functions;
(b) The Bellman equation has a unique solution in the set of all possible value functions, and that solution is $V^*$; and
(c) Bellman's principle of optimality holds and at least one optimal policy exists.

## 6.13 Discussion and Further Work

In Sec. 6.6, we summarized the work of Martin and Duff on the framework of BA-MDPs. Duff (2002, Chap. 5) showed that a BA-MDP could be rewritten as a POMDP, and so we followed a similar analysis of transforming a BA-SMDP into POSMDP so that we could apply ChronosPerseus. At the time, Duff did not use Perseus to solve his reformed POMDP. Instead, there were two main reinforcement learning algorithms that Duff developed for BA-MDPs:

(a) Actor-Critic Policy Gradient: This method focuses on policy iteration for improving stochastic policies in MDPs and creates a sample-based variant.

Duff combines parameterized functions for representing policies and value functions into the policy iteration process. The algorithm utilizes a Monte Carlo technique for estimating the gradient of the value function concerning policy parameters and applies the resulting "actor-critic, policy-gradient" method to BA-MDPs.

(b) State-action-reward-state-action SARSA($\lambda$) for BA-MDPs: For each combination of physical state and action, a linearly-parameterized function approximator is used to generalize the evolving $Q$-value estimates over the information state components.

Later, the approximate offline BEETLE algorithm (Bayesian Exploration Exploitation Tradeoff in LEarning) by Poupart *et al.* (2006) extended PERSEUS (Spaan and Vlassis, 2005) for BA-MDPs. BEETLE first samples reachable state-belief pairs using a default or random policy simulation. Then, it performs approximate value iteration at these pairs using the derived equations. However, BEETLE encounters a challenge due to the increasing $\alpha$-functions in the resulting value function polynomial, growing exponentially with the planning horizon. This intractability issue is a crucial aspect to address for the algorithm's efficiency and scalability.

In our current focus on developing conjugate priors for state transition probabilities and sojourn-time distribution parameters, an alternative *model-free* approach could be considered. Dearden *et al.* (1998) introduced one of the first model-based exploration methods for BRL—Bayesian $Q$-Learning—which extends Watkins (1989) $Q$-Learning algorithm by constructing and updating probability distributions over the $Q$-values. As a potential avenue for future work, this model-free approach could be extended to the BA-SMDP, exploring its applicability and potential benefits within this framework.

From the very beginning of BRL, Bellman (1956) assumes that the reward was known and focuses on learning the unknown state transition probabilities. The majority of BRL research still continues with this tradition of assuming a known reward function (Ghavamzadeh *et al.*, 2015, p. 417). For example, with BA-POMDP, Ross *et al.* (2007) mentions that their approach can be generalized to learn the reward function as well: this can be accomplished by adding posterior parameters for the reward in the hyperstate (Ross *et al.*, 2011, p. 1737). The Dirichlet distribution could be used if the reward function is drawn from a discrete set of values. If the reward is continuous, then Ghavamzadeh *et al.* (2015, p. 417) suggests following Strens (2000) and assume that the rewards are Gaussian distributed. Unfortunately, we are not able to use this Gaussian conjugate prior approach for a parameter on the sojourn-time distribution, which lead to our creation of the gamma-inverse Gaussian conjugate prior in Sec. 6.11.1.

In BRL, the reward function is known typically, which could inadvertently provide the agent with substantial information. For instance, when an agent attempts to discern between two SMDPs, like in Example 6.10.2, the reward function may contain sufficient information to indicate which SMDP the agent currently inhabits. Notably, even the sojourn time distribution is incorporated into the reward function of an SMDP (4.3.1). In POMDPs, utilizing reward functions for belief updates has enhanced performance (Izadi and Precup, 2005). Furthermore, Silver, Singh, Precup and Sutton (2021) recently argued that even a seemingly innocuous reward signal in a complex environment could lead to the emergence of general intelligence, emphasizing the importance of reward maximization for driving intelligent behaviour. However, it is important to acknowledge that the article does not explicitly discuss the information aspect of the rewards, which might still be an open question. Future work in this domain may need to reconsider the assumption of known rewards or to leverage the known rewards to extract information about the sojourn-time parameters.

In Sec. 6.10, we discuss an agent learning a mixture of SMDPs by approximating the BA-SMDP with a finite number of SMDPs. We then employ ChronosPerseus, which further approximates the solution using importance sampling methods. A natural question arises: How much error is introduced in total and at each step of this process? Can we quantify the impact of these approximations on the overall numerical stability and precision of the solution? It would be valuable to investigate this issue from a numerical analysis perspective, aiming to derive error bounds that would provide insights into the trade-offs between approximation techniques and the quality of the resulting policies.

In Sec. 6.11.1, we devised a conjugate prior for the mean $\mu$ of an inverse Gaussian distribution, assuming *exchangeability*—a standard premise in Bayesian analysis. This implies that the joint distribution of the mean sojourn time parameter $\theta$ remains unchanged regardless of the observed permutation of sojourn times $\tau_1, \tau_2, \ldots, \tau_n$ (owing to the commutative properties of multiplication and addition in the joint density (6.11.5)). Nevertheless, recent research indicates that both rats and humans can recall event sequences in episodic tasks, possibly utilizing similar cognitive processes and mnemonic representations (Allen *et al.*, 2014). This finding suggests the potential need for more sophisticated time representations. For instance, consider music: knowing the names of four notes and the number of occurrences may not be sufficient to identify the title of the piece without knowing the order in which they were played. Consequently, our focus may extend beyond merely examining *what* and *when* an event transpired to include *how* it was sequenced.

## 6.14 Conclusion

In this chapter, one of the major contributions from this thesis, the novel framework and strategies of a Bayes-adaptive semi-Markov decision process (BA-SMDP), was introduced. The essence of this framework lies in demonstrating how an agent could optimally learn the parameters of an SMDP, while concurrently addressing the exploration versus exploitation quandary. We considered four distinct strategies within the context of the BA-SMDP:

(a) learning the sojourn time distribution parameters using a count array to record the number of each sojourn time occurrence for a particular $(s, a, s')$ transition, where sojourn times come from a finite set;
(b) learning the mixture of known sojourn time distributions with unknown proportions;
(c) learning the mixture of known SMDPs with unknown proportions; and
(d) learning the unknown continuous sojourn-time distribution parameters.

In the fourth strategy, we conceptualized a conjugate prior specifically designed to monitor the uncertainty associated with the mean parameter of the inverse Gaussian distribution. This technique, although developed specifically for the inverse Gaussian sojourn time distribution, could potentially provide a foundation for formulating conjugate priors for parameters of other sojourn time distributions. Furthermore, we demonstrated that the value function of this BA-SMDP can be represented by a set of $\alpha$-functions. These functions can serve as the basis for a solution using a point-based value iteration approach.

# 7
# Conclusion

What do mathematicians do? This is not an easy question to answer. However, one good reply is that mathematicians makes natural questions precise.

—Bellman (1984, p. 114)

As we approach the end of this work, it is fitting to circle back to the bus problem introduced in the first chapter. Recall in this scenario, you are embarking on a journey involving four bus stops, with a bicycle at your disposal should you decide to switch modes of transport. The duration of your bus ride is uncertain, primarily due to the variability of traffic conditions. However, with repeated daily experiences, you would gradually learn to estimate how long it might take to reach your final destination under different circumstances. This seemingly simple example aptly illustrates our inherent human ability to learn over time and make decisions under uncertainty. Our rich tapestry of experiences allows us to form reasonable predictions about traffic patterns and to estimate travel times with a degree of accuracy.

## 7.1 Summary

Building on these insights, the primary aim of this thesis was to equip reinforcement learning with the capacity to tackle more intricate, real-world problems where timing plays a pivotal role. Futhermore, we want this framework and its methods to be able to tackle uncertainty and deal with the exploration versus exploitation tradeoff.

We first review Markov decision processes (MDPs), a basic model that mirrors understanding the fixed number of bus stops during a journey (Chap. 2). Venturing further, we described MDPs with an added layer of complexity, partial observability (POMDPs). In the bus problem, this would be the uncertainty of traffic conditions during the bus ride (Chap. 3). Next, we reviewed MDPs that take into account time, known as semi-Markov decision processes (SMDPs). This aligns with finding the optimal policy where the sojourn time distribution is known, but stochastic (Chap. 4).

Despite these advances, the intersection of time and partial observability, encapsulated by partially observable semi-Markov decision processes (POSMDPs) has been largely unexplored (Chap. 5). In particular, scenarios where time is the key observation, akin to deciding the best moment to switch from bus to bicycle, had yet to be extensively probed.

In Chap. 5, we developed the partially observable semi-Markov decision process (POSMDP) solver—CHRONOSPERSEUS—combining PERSEUS and importance sampling for efficient POSMDP solutions where transition time is observable. The versatility of the solver spans various problem types, including episodic, non-episodic, mixed-observable, discrete, continuous observation space, and a mixture of fixed and stochastic continuous sojourn times. When the agent knows the entire model, like in the bus problem, we saw that CHRONOSPERSEUS that is similar to human intuition in such scenarios.

The effectiveness of CHRONOSPERSEUS was demonstrated in learning a policy on a POSMDP where time is the only available information to resolve the partially observable state. This scenario is prevalent in many real-world problems, and the proposed solver provides significant advantages in decision-making and planning under uncertainty.

The critical interplay between exploration and exploitation, akin to our daily decision to either try a potentially faster but uncertain route (exploration) or stick to the usual, known path (exploitation), has been thoroughly investigated within the context of MDPs. However, this key equilibrium has not been extensively explored for SMDPs. This is comparable to learning whether to switch modes of transport optimally, given the uncertain duration of the bus journey, while trying to reduce transit time. Recognizing this gap, this thesis introduces novel mathematical models and algorithmic advancements.

In bridging these gaps, the focus of Chap. 6 was on the development of an innovative Bayesian paradigm known as Bayes-adaptive semi-Markov decision processes (BA-SMDPs). This framework and its strategies incorporates the crucial role of timing in decision-making processes under uncertainty, all the while finding an optimal balance between exploration and exploitation during the learning of environmental dynamics. Through this endeavour, the work not only broadens the horizons of reinforcement learning but also equips it to tackle real-world problems of greater complexity where timing is integral and allows for optimal exploration versus exploitation policy in such an environment.

Four distinct learning approaches were developed, including learning of sojourn time distribution parameters using a count array, learning the mixture of known sojourn time distributions with unknown proportions, learning the mixture of known semi-Markov decision processes (SMDPs) with unknown proportions, and learning unknown continuous sojourn-time distribution parameters. The final approach necessitated the development of a novel conjugate prior for the mean parameter of the inverse Gaussian distribution, which allows tracking of the uncertainty about the parameter. These methodologies will afford researchers and modellers the capacity to optimally learn an SMDP, thereby enhancing sample efficiency and reward maximization.

## 7.2 Future Work

While this research has made significant strides, it has also illuminated avenues for future exploration. For instance, an extension of the model-free approach of Bayesian $Q$-Learning to BA-SMDP could be explored to evaluate its applicability and potential benefits. Moreover, the underlying reward function, treated as known in this study, warrants further scrutiny.

It is important to acknowledge the limitations inherent in this study. The exchangeability assumption—a cornerstone of Bayesian analysis—was employed throughout the BA-SMDP. However, recent studies suggest that both rats and humans can recall the sequence of events in episodic tasks, potentially employing similar cognitive processes and mnemonic representations. This opens up a compelling perspective that goes beyond merely examining the *what* and *when* of an event to include *how* it unfolds in a sequence. Hierarchical models could offer a richer depiction of temporal dynamics in such contexts.

Moreover, the stationary environment assumption, where transition probabilities or sojourn time distributions within the environment remain unchanged over time, confines the types of problems that can be addressed using this approach. This constraint points to a future research direction exploring methods to accommodate non-stationary environments, thus broadening the method's applicability.

Despite these constraints, there are potential extensions of the techniques developed here. Specifically, the method designed for the inverse Gaussian sojourn time distribution could serve as a blueprint for creating conjugate priors for parameters of other sojourn time distributions.

Another consideration lies in the realm of temporal control, which refers to the timing of an agent's action. While our work focused on temporal learning—the tracking of time when an event occurs—addressing the issue of temporal control remains to be studied further in depth.

Finally, this thesis introduced the concept of a multi-SMDP encapsulated within a POSMDP. This model may be further generalized into a basis set, potentially allowing a linear combination of SMDPs to express the true underlying SMDP, even if it falls outside the original set of SMDPs. Such an advancement would significantly enhance the expressiveness and flexibility of our models, providing a robust platform for future investigations.

In this way, the research presented herein serves not only as a testament to the progress made but also as a signpost for future developments in the field.

## 7.3 Final Thoughts

Ultimately, this thesis stands as a testament to the potential that lies in the use of more expressive models, such as POSMDP or the BA-SMDP. By employing these models, we can circumvent the computational complexity that arises from expanding the state space of simpler MDPs, effectively avoiding Bellman's Curse of Dimensionality. This work lays a solid foundation and opens new avenues for future research in the realm of sequential decision-making under uncertainty involving time.

# Appendix A

# Code

## A.1 Elevator Problem (MDP version)

```
import numpy as np
import matplotlib.pyplot as plt

#Parameters
N = 9       #time in minutes
r = 25.0    #reward
c = 3.0     #cost
C = 10.0    #penalty
S = 5       #number of floors

V = np.zeros((S,N+1))
optimalPolicy = np.zeros((S,N+1))

#Probability transition matrix
#P(i,j) = P(floor at
#                 1    2    3    4    5
P = np.array([[1.0, 0.0, 0.0, 0.0, 0.0], # 1
              [1/2, 1/2, 0.0, 0.0, 0.0], # 2
              [0.0, 1/2, 1/2, 0.0, 0.0], # 3
              [0.0, 0.0, 1/2, 1/2, 0.0], # 4
              [0.0, 0.0, 0.0, 1/2, 1/2]])# 5

#Calculating the last decision epoch, n = N-1
for s in np.arange(S):
    V[s,N] = np.max([r - C, r*P[s,0] - c])
    optimalPolicy[s,N] = np.argmax([r - C, r*P[s,0] - c])

#Calculating the decision epochs, n < N-1
for n in np.arange(N-1, -1, -1):
    for s in np.arange(S):
        tempSum = 0
        for i in np.arange(S):
            tempSum = tempSum + P[s,i]*V[i,n+1]
        V[s,n] = np.max([r-C, r*P[s,0]-c + tempSum])
        optimalPolicy[s,n] = np.argmax([r - C, r*P[s,0]-c + tempSum])

plt.imshow(V);
plt.colorbar()
print(optimalPolicy) #Optimal actions
print(0.5 * (V/np.max(V)))  #What I used to calculate the percentage of
                            #colour for the figure
print(V) #The values of the value function
```

## A.2 Inverse Gaussian Pseudo Random Number Generator

```
# Inverse Gaussian distribution
# 28 March 2019: Richard Kohar

import math
import torch as pt
from torch.distributions import Normal
from torch.distributions import Uniform

def pdfIG(x,mu,Lambda):
    # Input:
    # x is a tensor
    # mu is a scalar
    # Lambda is a scalar
    #
    # Output:
    # pdf is a tensor
    pdf = pt.sqrt(Lambda/(2*math.pi*x.float()**3))*pt.exp(-(Lambda*
                 (x.float()-mu)**2)/(2*mu**2*x.float()))
    return pdf

def cdfIG(x, mu, Lambda):
    # Input:
    # x is a tensor
    # mu is a scalar
    # Lambda is a scalar
    #
    # Output:
    # cdf is a tensor
    m = Normal(pt.tensor([0.0]), pt.tensor([1.0]))

    cdf = m.cdf(pt.sqrt(Lambda/x.float())*(x/mu - 1)) + math.exp(2*Lambda
               /mu)*m.cdf(-pt.sqrt(Lambda/x.float())*(x/mu + 1))
    return cdf

def randomIG(size, mu, Lambda):
    # Input:
    # size is a tensor [m,n] that gives the size of the random IG sample.
    # mu is a scalar
    # Lambda is a scalar

    # Generate a normal distribution
    m = Normal(pt.tensor([0.0]), pt.tensor([1.0]))
    nu = m.sample(size)

    y = nu**2

    x = mu + (mu**2 * y / (2*Lambda)) - (mu/(2*Lambda))*pt.sqrt(4*mu*
             Lambda*y.float() + mu**2 * (y.float())**2)

    u = Uniform(pt.tensor([0.0]), pt.tensor([1.0]))
    z = u.sample(size)

    ind1 = (z <= (mu/(mu + x)))
    ind2 = 1 - ind1

    sampleIG = (ind1.float()*x) + (ind2.float()*((mu**2)/x))

    return sampleIG
```

## A.3 Truncated Gaussian Distribution

```
# Truncated Gaussian distribution when b = infty (for PyTorch)
# 28 Feb 2021: Richard Kohar

import math
import torch as pt
from torch.distributions import Normal

pt.set_grad_enabled(False)

def pdfGaussian(x, mu, sigma, device):
    # Input:
    # x is the tensor data
    # mu is the mean
    # sigma is the standard deviation
    # device is the GPU or CPU
    #
    # Output:
    # pdf is the pdf of the data x.

    pdf = 1.0/pt.sqrt(2.0*math.pi*(sigma**2.0))*
           pt.exp(-((x - mu)**2.0)/(2.0*(sigma**2.0)))

    return pdf

def laplaceTruncGaussLeft(mu, sigma, beta, a, device):
    # Input:
    # mu is a scalar
    # sigma is a scalar
    # beta is a scalar
    # a is a scalar
    # device is if it's on the GPU or CPU
    m = Normal(pt.tensor(0.0, device=device), pt.tensor(1.0, device=device))

    I = pt.exp(((sigma**2.0 * beta**2.0) / 2.0) - (mu * beta)) *
             ((1 - m.cdf(((a-mu)/sigma) +
  (sigma*beta)))/(1 - m.cdf((a - mu)/sigma)))

    return I

def pdfTruncGauss(x, mu, sigma, a, b, device):
    # Input:
    # x is a tensor
    # mu is a scalar/tensor
    # sigma is a scalar/tensor
    # a is the left truncation point
    # b is the right truncation point
    # device is if it is a GPU or CPU
    #
    # Output:
    # pdf is the pdf of the data x.

    assert a < b, 'a is not less than b'

    m = Normal(pt.tensor(0.0, device=device), pt.tensor(1.0, device=device))

    pdf = pdfGaussian(x, mu, sigma, device) / (m.cdf((b - mu)/sigma) -
                  m.cdf((a - mu)/sigma))

    xind = x > a
    xind2 = x < b
    pdf = pdf * (xind*xind2)

    return pdf

# This is for a Gaussian distribution that is truncated on the
# left hand side and allow the right hand side to go to
# infinity.
def pdfTruncGaussLeft(x, mu, sigma, a, device):
    # Input:
    # x is a tensor
    # mu is a scalar/tensor
    # sigma is a scalar/tensor
    # a is the left truncation point
    # device is if it is a GPU or CPU
    #
    # Output:
    # pdf is a tensor

    m = Normal(pt.tensor(0.0, device=device), pt.tensor(1.0, device=device))
    pdf = pdfGaussian(x, mu, sigma, device) / (1.0 - m.cdf((a - mu)/sigma))

    xind = x > a

    pdf = pdf * xind

    return pdf

# This is for a Gaussian distribution that is truncated on the
# right hand side and allow
# the left hand side to go to infinity.
def pdfTruncGaussRight(x, mu, sigma, b, device):
    # Input:
    # x is a tensor
    # mu is a scalar/tensor
    # sigma is a scalar/tensor
    # b is the right truncation point
    # device is if it is a GPU or CPU
    #
    # Output:
    # pdf is a tensor

    m = Normal(pt.tensor(0.0, device=device), pt.tensor(1.0, device=device))

    pdf = pdfGaussian(x, mu, sigma, device) / (m.cdf((b - mu)/sigma) - 1.0)

    xind = x < b

    pdf = pdf * xind


    return pdf

def randomTruncGaussLeft(mu, sigma, a, max_rejections=10000):
    # Input:
    # mu is a scalar/tensor
    # sigma is a scalar/tensor
    # a is the left truncation point
    #
    # Output:
    # sampleTruncGaussLeft is a tensor

    # Sample X ~ N(x | mu, sigma^2, )

    rejections = 0

    m = Normal(mu, sigma)

    while True:
        sampleTruncGaussLeft = m.sample()
        if a <= sampleTruncGaussLeft:
            return sampleTruncGaussLeft
        rejections = rejections + 1
        if rejections > max_rejections:
            assert False, 'Too many rejections'

def randomTruncGaussRight(mu, sigma, b, max_rejections=10000):
    # Input:
    # mu is a scalar/tensor
    # sigma is a scalar/tensor

    # b is the right truncated point
    #
    # Output:
    # sampleTruncGaussRight is a tensor

    # Sample X ~ N(x | mu, sigma^2, )

    rejections = 0

    m = Normal(mu, sigma)

    while True:
        sampleTruncGaussRight = m.sample()
        if sampleTruncGaussRight <= b:
            return sampleTruncGaussRight
        rejections = rejections + 1
        if rejections > max_rejections:
            assert False, 'Too many rejections'

def randomTruncGauss(mu, sigma, a, b, device, max_rejections=10000):
    # Input:
    # mu is a scalar/tensor
    # sigma is a scalar/tensor
    # a is the left truncation point
    # b is the left truncation point
    #
    # Output:
    # sampleTruncGaussLeft is a tensor

    # Sample X ~ N(x | mu, sigma^2)

    rejections = 0

    m = Normal(mu, sigma)

    while True:
        sampleTruncGauss = m.sample()
        if a <= sampleTruncGauss <= b:
            return sampleTruncGauss
        rejections = rejections + 1
        if rejections > max_rejections:
            assert False, 'Too many rejections'
```

## A.4 Calculating the conjugate prior distribution

```
% Calculating the distribution on the mean parameter of the inverse
% Gaussian distribution

% 2023 Jan 21: Richard Kohar

%These times are sampled from an Inverse Gaussian distribution
%of mu = 3, and lambda = 9.
times = [1.4588, 5.9780, 2.3113, 2.4519, 1.9821, 2.0355,...
    1.6069, 3.5954, 3.7644, 4.3149, 2.4975, 1.8990, 1.1870,...
1.3712, 2.4073, 4.5663];

%These are the prior parameters for the gamma distribution.
a = 3;
b = 2;

x = linspace(0, 5, 500);
maxval = zeros(1, length(times));
maxTheta = zeros(1, length(times));
hold on
for i = 1:length(times)
    taus = times(1:i);

    priorlikelihood = @(theta) 1/sqrt(2*pi)*prod(taus.^(-3/2)) ...
    * (b^a)/ gamma(a) ...
        .* theta.^(a + length(taus) - 1) ...
        .* exp(-1/2 .* sum( ((repmat(taus, length(theta), 1) - ...
        repmat(theta', 1, length(taus))).^2) ...
        ./repmat(taus, length(theta), 1), 2 )' - b*theta);

    normalized= @(theta) priorlikelihood(theta) ./
                    integral(priorlikelihood, 0, Inf);
    [maxval(i), maxTheta(i)] = max(normalized(x));

    maxTheta(i) = maxTheta(i)/length(x) * x(end);

plot(x, normalized(x), "Color", [1 (44-(2*i + 12))/44 (44-(2*i + 12))/44])
    ylim([0, 1.3])
end
hold off
```

## A.5 ChronosPerseus

### A.5.1 Solver

```
# -*- coding: utf-8 -*-
```

```
"""
Created on Thu Sep 24 15:31:29 2020

Chronos Perseus: A point-based POSMDP solver

@author: Richard Kohar
"""

import torch as pt  # pt for PyTorch
from inverseGaussian import *    # imports my IG functions

pt.utils.backcompat.broadcast_warning.enabled=True

pt.backends.cudnn.deterministic=True

pt.set_grad_enabled(False)

# CollectBeliefs
def collectBeliefs(P, G, xi0, sojournTime, numStates, numActions,
                                                numBeliefs, device):
    # Input:
    # P[a,s,s] or [numActions, numStates, numStates]
    # G[a,s,o] or [numActions, numStates, numObs]
    # xi0[s] or [numStates]
    # sojournTime is a class that represents the time distribution
    #   methods: sampleTime, pdf
    # numStates: number of states
    # numActions: number of actions
    # numBeliefs: number of beliefs to generate
    # device: cpu or gpu

    # Output:
    # B set of beliefs
    # C set of sampled sojourn times
    # w the proportion of sojourn times that came from each s,a,s' transition
    # f is the likelihood of each sampled sojourn times

    B = pt.zeros(numBeliefs, numStates, device=device)
    C = pt.zeros(numBeliefs, device=device)
    B[0] = xi0
    w = pt.zeros(numActions,numStates,numStates, device=device)
    f = pt.zeros(numBeliefs, numActions, numStates, numStates, device=device)
#f(tau_n | s, a, s')  -- [xi, a, s, s]
    a = pt.randint(numActions,(numBeliefs,1), device=device)
#Randomly select an action a (preselect all actions)
    for b in pt.arange(1,numBeliefs):
        xiOld = B[pt.randint(b,(1,))]
#Randomly select a belief from set B
        s1 = pt.multinomial(xiOld, 1).squeeze()
#Generate state s from belief distribution xi
```

```
        #s1 is a scalar tensor / torch.Size([])
        #Action is already randomly generated
        s2 = pt.multinomial(P[a[b], s1, :], 1).squeeze()
#Generate state s’ according to probability transition matrix for P(.|s,a)
        #s2 is a scalar tensor / torch.Size([])
        C[b] = sojournTime.sampleTime(a[b],s1,s2)
#Generate sojourn time
        f[b,:,:,:] = sojournTime.pdf(C[b])
#[tau, a, s, s’]  It is [b,a,s,s’] but at the end,
#we reduce to unique beliefs, but the number of times will remain the same.
        w[a[b], s1, s2] = w[a[b], s1, s2] + 1

        #Randomly select an observation o according to P(o | xi, a, tau)
        myG = G[a[b]]  #[1,s’,o]  slice along action (don’t worry
               #about dim 0, because we’ll use that for s)
        myXi = xiOld.t().unsqueeze(dim=1) #[s,1,1]
        myP = P[a[b]].squeeze(dim=0).unsqueeze(dim=2)
#[1,s,s’] -> [s,s’] -> [s,s’,1]
        myf = f[b,a[b]].squeeze(dim=0).unsqueeze(dim=2)
#[1,s,s’] -> [s,s’] -> [s,s’,1]
        temp = myG * myXi * myP * myf
#[1,s’,o] * [s,1,1] * [s,s’,1] * [s,s’,1] = [s,s’,o]
        temp = pt.sum(temp, dim=0)
#Sum along s, [s,s’,o] -> [s’,o]
        Pxiaotau = pt.sum(temp,dim=0)
#Sum along s’ [s’,o] -> [o] (row vector)
        o = pt.multinomial(Pxiaotau, 1)

        #Calculate the new belief
        xiNew = temp[:,o] #[s’] (in column vector)
        B[b] = xiNew.t()/Pxiaotau[o] #[s’].t()/Normalization constant  [1,s]
    w = w/pt.sum(w)     #normalize w
    B = B.unique(dim=0) #reduces the number of beliefs if redundant
    C = C[1:]           #removes the zeroth term (which is not generated)
    f = f[1:]           #removes the zeroth term (which is not generated)

    return B,C,w,f

# This returns the calculation for alpha(s | a, tau_n, o)
def computeAlphaATauO(P,G,V,f):
    # Input:
    # P[a,s,s] or [numActions, numStates, numStates]
    # G[a,s,o] or [numActions, numStates, numObs]
    # V[i,s] or [numVec, numState]
    # f[tau,a,s,s’] -- f(tau_n | s, a, s’)

    # Output:
    # alphaATauO = G * P * f * V
    # alphaATauO [i,f,a,o,s] or [numInV, numInf, numActions, numObs, numStates]
    #Temporary format [i, f, a, s, s', o]
    myG = G.unsqueeze(dim=0).unsqueeze(dim=1).unsqueeze(dim=3)
#[a,s',o] -> [1,1,a,1,s',o]
    myP = P.unsqueeze(dim=0).unsqueeze(dim=1).unsqueeze(dim=5)
#[a,s,s'] -> [1,1,a,s,s',1]
    myf = f.unsqueeze(dim=0).unsqueeze(dim=5) #[f,a,s,s'] -> [1,f,a,s,s',1]
    myV = V.unsqueeze(dim=1).unsqueeze(dim=2).unsqueeze(dim=3).
          unsqueeze(dim=5) #[i,s'] -> [i,1,1,1,s',1]

    # Essentially, we are doing: G * P * f * V

    # This has myG(1,1,a,1,s',o).*myP(1,1,a,s,s',1).*
     myF(1,f,a,s,s',1).*myV(i,1,1,1,s',1)
    myM =  myG * myP * myf * myV #[i, f, a, s, s', o]

    #Summing over all s', thus alphaAO (i,c,a,s,o) and then
#transpose to get (i,c,a,o,s)
    alphaATauO = myM.sum(dim=4).transpose(3,4)
    return alphaATauO #[i,f,a,o,s] or [numInV, numInf, numActions,
     numObs, numStates]

# Backing up to an alpha vector.
def backup(V,P,G,R,C,w,f,D,xi,flag):
    # Input:
    # V[i,s] or [numVec, numState]
    # P[a,s,s] or [numActions, numStates, numStates]
    # G[a,s,o] or [numActions, numStates, numObs]
    # R[a, s] or [numActions, numStates]
    # C[tau] (the sampled sojourn times)
    # w[a,s,s'] or [numActions, numStates, numStates]
#     (the proportion that (s,a,s') was sampled)
    # f[tau,a,s,s'] -- f(tau_n | s, a, s')
    # D is the discount factor that is precalculated for every time.
    # xi[1,s] or [1, numState] -- a single belief used for the update

    # Output:
    # alpha[numStates]

    numStates = V.size(1)
#This will get the number of states from the size of the V set.

    alphaATauO = computeAlphaATauO(P,G,V,f)
#[i,f,a,o,s] alpha(s | a, tau, o)

    # Computing the argmax using beliefs for all s,a,tau,o
    myXi = xi.unsqueeze(0).unsqueeze(0).unsqueeze(0)
#[1,s] -> [1,1,1,1,s]
    XiDotAlphaATauO = (alphaATauO * myXi).sum(dim=4)
#[i,f,a,o,s] -> [i,f,a,o]  basically xi \cdot alpha, for every alpha in V
```

```
    # Doing the argmax
    bestalphaind_forallATauO = XiDotAlphaATauO.argmax(dim=0,
          keepdim=True).unsqueeze(4).repeat(1,1,1,1,numStates)
    # unsqueeze(4) expands the dimension for s
    # repeat [1,1,1,1,numStates] because [i,f,a,o]

    bestalphareduce_forallATauO = pt.gather(alphaATauO,
index=bestalphaind_forallATauO, dim=0).squeeze(0)
    # Now, we have those best alpha vectors for each time, action, and
# observation, thus [f,a,o,s].
    # The singleton dimension along alpha vectors is to be removed.

    # Computing backup
    bestalpha_forTau = bestalphareduce_forallATauO.sum(dim=2) #[f,a,o,s] -> [f,a,s]
    myD = D.unsqueeze(1).unsqueeze(1) #[f] -> [f,1,1]
    alphaAXi = R + pt.sum(myD * bestalpha_forTau, dim=0) #[f,a,s] -> [a,s]

    # Computing backup
    alphaAction = pt.argmax((alphaAXi* xi).sum(dim=1)) #[]
    alpha = alphaAXi[alphaAction].unsqueeze(0) #[alphaAction, :] -> [1,s]

    # Why is the action not selected?
    # alphaAXi = R + pt.sum(myD * bestalpha_forTau, dim=0)
    # You will want to print the alpha and alphaAction because
    # check

    if flag:
        print("alpha", alpha)
        print("alphaAction", alphaAction)
        print("xi", xi)
        print("alphaAXi*xi", alphaAXi*xi) #see why we are not getting action 0?
        print("sum of alphaAXi*xi", (alphaAXi* xi).sum(dim=1))

    # Assumption: Zhang could be wrong. we need to break down the backup value
    # calculation.

        print("R*xi (immediate)", pt.sum(R*xi, dim=1))
        print("exp future reward", pt.sum(pt.sum(myD *
                   bestalpha_forTau, dim=0)*xi, dim=1))

    # These are the two components that are used for determining which action
    # is to be selected. We should see why action 1 is better from this by seeing
    # that the value for action 1 is better than the value for action 0.

    return alpha, alphaAction
#alpha[numStates]   alphaAction is scalar with 0 to numActions

# Calculates the optimal value function at a particular belief state
# and it's corresponding optimal action.
def valuefunc(V,Vactions,B):
    # Input:
    # V[i,s] or [numVec, numState]
    # Vactions[i] or [numVec]
    # B[b,s] or [numBelief, numState]

    # Output:
    # VxB.values, VxBcorrespondingActions   Return the max V*xi [value, action]

    # B * V' = [b,i] -> max [b]
    # b * V' = [1,i] -> max [1]

    myV = V.unsqueeze(0) #[i,s] -> [1,i,s]
    myB = B.unsqueeze(1) #[b,s] -> [b,1,s]
    VxB = myV * myB #[b,i,s]
    VxB = pt.sum(VxB, dim=2) #[b,i,s] -> [b,i].  <V,xi> for every xi in B.
    VxB = VxB.max(dim=1)
    VxBcorrespondingActions = Vactions[VxB.indices]
    return VxB.values, VxBcorrespondingActions #Return the max V*xi [value, action]


# Update function
def update(V,Vactions,B,P,G,R,C,w,f,D,device):
    # Input:
    # V is the number of vectors i by number of states s   V[i,s]
    # Vactions
    # B

    # Output:

    numStates = V.size(1)
    numBeliefs = B.size(0)
    V2 = pt.zeros_like(B) #[b,s] There cannot be more alpha vectors than beliefs.
    V2actions = pt.zeros(numBeliefs, dtype=pt.int16, device=device)
#corresponding action
    B2ind = pt.ones(numBeliefs, device=device)
#1 if belief is not improved; otherwise, 0 if belief is improved.

    (Vb, VbActions) = valuefunc(V,Vactions,B)
    k = pt.tensor(0, dtype=pt.int16, device=device)

    alphaATauO = computeAlphaATauO(P,G,V,f) #[i,f,a,o,s] alpha(s | a, tau_n, o)

    while pt.sum(B2ind) > 0:
        xiInd = pt.multinomial(B2ind, 1)
        xi = B[xiInd]  #Randomly select a belief xi from B2.

        #Creates the new alpha vector candidate that could be added to the set V.
        (alpha, alphaAction) = backup(V,P,G,R,C,w,f,D,xi,xiInd==(B.size(0)-9))

        XiAlpha = (xi*alpha).sum(dim=1) #[1,s]*[1,s] -> [s]
        #No improvement
        if XiAlpha < Vb[xiInd]:
            #Keep the old one
            alpha = Vb[xiInd]
            alphaAction = VbActions[xiInd]
            B2ind[xiInd] = 0
#Enforce the belief removal (numerical rounding instability)

        #Improvement
        VAlphaB = (B*alpha).sum(dim=1)  #[b] compute alpha value for all beliefs
        B2ind = B2ind * (VAlphaB < Vb)  #1 if belief is not improved;
                                #otherwise, 0 if belief is improved.
        V2[k] = alpha
        V2actions[k] = alphaAction
        k = k + 1
    V = V2[0:k]
    Vactions = V2actions[0:k]
    return V, Vactions

#MAIN SCRIPT
def solvePOSMDP(problem, numBeliefs, numIter):
    """
    Solve a given POMDP assuming pytorch tensors on CUDA device (or CPU)

    :param problem: Dictionnnary containing
        'P' : Tensor of dimensions (a,s,s') of transition probabilities
        'G' : Tensor of dimensions (a,s',o) of observation probabilities
        'r1' : Tensor of dimensions (a,s) of lump sum rewards
        'r2' : Tensor of dimensions (a,s,s') of continuous reward rate
        'mu' : Tensor of dimensions (a,s,s') of mean parameter for
                sojourn time distribution
        'Lambda' : Tensor of dimensions (a,s,s') of shape parameter
                for sojourn time distribution
        'xi0' : Tensor of dimensions (1,s) of initial belief
        'beta' : Scalar discount rate

    :param numBeliefs: Number of beliefs to use
    :param numIter: Number of iteration of value iteration

    :return: V, VActions
       'V' : Tensor of dimensions (alpha_vector, state) (all the alpha vectors)
       'VActions' : Tensor of dimensions (alpha_vector) (corresponding actions)
    """

    # Extract problem
    P = problem["P"]
    G = problem["G"]
    R = problem["R"]
    V = problem["V0"]
    Vactions = problem["V0actions"]
    sojournTime = problem["sojournTime"]
    xi0 = problem["xi0"]
    beta = problem["beta"]

    #Extract other information
    device = P.device
    numActions, numStates, numObs = G.size()

    #MAIN SCRIPT
    (B, C, w, f) = collectBeliefs(P, G, xi0, sojournTime, numStates,
numActions, numBeliefs, device)

    #Precomputing the discount factor for each tau.
    D = (1/C.size(dim=0))*pt.exp(-beta*C)/((w.unsqueeze(dim=0)*f).
         sum(dim=1).sum(dim=1).sum(dim=1))

    minValue = pt.zeros(numIter)
    maxValue = pt.zeros(numIter)

    for j in pt.arange(1,numIter):
        print("Iteration", j)
        (V, Vactions) = update(V,Vactions,B,P,G,R,C,w,f,D,device)
        minValue[j] = V.min()
        maxValue[j] = V.max()
        # print(j, V.size())
    minValue = minValue[1:]
    maxValue = maxValue[1:]
    return B, C, V, Vactions, minValue, maxValue
```

# Bibliography

Åström, K. J. (1965). Optimal control of Markov processes with incomplete state information, *Journal of Mathematical Analysis and Applications* **10**, 1, pp. 174–205, doi:10.1016/0022-247X(65)90154-X.

Åström, K. J. (1969). Optimal control of Markov processes with incomplete state-information II. The convexity of the lossfunction, *Journal of Mathematical Analysis and Applications* **26**, 2, pp. 403–406, doi:10.1016/0022-247x(69)90163-2.

Allen, T. A., Morris, A. M., Mattfeld, A. T., Stark, C. E., and Fortin, N. J. (2014). A sequence of events model of episodic memory shows parallels in rats and humans, *Hippocampus* **24**, 10, pp. 1178–1188, doi:10.1002/hipo.22301.

Aoki, M. (1965). Optimal control of partially observable Markovian systems, *Journal of the Franklin Institute* **280**, 5, pp. 367–386, doi:10.1016/0016-0032(65)90528-4.

Aoki, M. (1967). *Optimization of Stochastic Systems: Topics in Discrete-Time Systems*, *Mathematics in Science and Engineering*, Vol. 32 (Academic Press, New York), doi:10.1016/S0076-5392(08)62031-7.

Arrow, K. J., Blackwell, D., and Girshick, M. A. (1949). Bayes and minimax solutions of sequential decision problems, *Econometrica* **17**, 3/4, pp. 213–244, doi:10.2307/1905525.

Balci, F., Gallistel, C. R., Allen, B. D., Frank, K. M., Gibson, J. M., and Brunner, D. (2009). Acquisition of peak responding: What is learned? *Behavioural Processes* **80**, 1, pp. 67–75, doi:10.1016/j.beproc.2008.09.010.

Balsam, P. D., Drew, M. R., and Yang, C. (2002). Timing at the start of associative learning, *Learning and Motivation* **33**, 1, pp. 141–155, doi:10.1006/lmot.2001.1104.

Balsam, P. D. and Gallistel, C. R. (2009). Temporal maps and informativeness in associative learning, *Trends in Neurosciences* **32**, 2, pp. 73–78, doi:10.1016/j.tins.2008.10.004.

Barnard, G. A. (1954). Sampling inspection and statistical decisions, *Journal of the Royal Statistical Society: Series B (Methodological)* **16**, 2, pp. 151–174, doi:10.1111/j.2517-6161.1954.tb00157.x.

Bellman, R. (1954). The theory of dynamic programming, *Bulletin of the American Mathematical Society* **60**, 6, pp. 503–515, doi:10.1090/s0002-9904-1954-09848-8.

Bellman, R. (1956). A problem in the sequential design of experiments, *Sankhyā: The Indian Journal of Statistics (1933-1960)* **16**, 3/4, pp. 221–229.

Bellman, R. (1957a). *Dynamic Programming* (Princeton University Press).

Bellman, R. (1957b). A Markovian decision process, *Indiana University Mathematics Journal* **6**, 4, pp. 679–684, doi:10.1512/iumj.1957.6.56038.

Bellman, R. (1961a). *Adaptive Control Processes: A Guided Tour* (Princeton University Press).

Bellman, R. (1961b). A mathematical formulation of varational processes of adaptive type, in *Proceedings of the Fourth Berkeley Symposium on Mathematical Statistics and Probability, Volume 1: Contributions to the Theory of Statistics*, pp. 37–48.

Bellman, R. (1978). *An Introduction to Artificial Intelligence: Can Computers Think?* (Boyd & Fraser Publishing Company, San Francisco).

Bellman, R. and Kalaba, R. (1959). On adaptive control processes, *IRE Transactions on Automatic Control* **4**, 2, pp. 1–9, doi:10.1109/tac.1959.1104847.

Bellman, R. E. (1984). *Eye of the Hurricane: an autobiography* (World Scientific).

Bertsekas, D. P. (2017). *Dynamic Programming and Optimal Control: Volume I*, 4th edn. (Athena Scientific).

Bertsekas, D. P. (2022). *Abstract Dynamic Programming* (Athena Scientific).

Bertsekas, D. P. and Tsitsiklis, J. N. (1996). *Neuro-dynamic programming* (Athena Scientific).

Blum, A. L. and Rivest, R. L. (1992). Training a 3-node neural network is NP-complete, *Neural Networks* **5**, 1, pp. 117–127, doi:10.1016/s0893-6080(05)80010-3.

Box, G. E. P. and Muller, M. E. (1958). A note on the generation of random normal deviates, *The Annals of Mathematical Statistics* **29**, 2, pp. 610–611, doi:10.1214/aoms/1177706645.

Bradtke, S. J. and Duff, M. O. (1995). Reinforcement learning methods for continuous-time Markov decision problems, in G. Tesauro, D. S. Touretzky, and T. K. Leen (eds.), *Advances in Neural Information Processing Systems 7* (MIT Press), pp. 393–400.

Buhusi, C. V., Aziz, D., Winslow, D., Carter, R. E., Swearingen, J. E., and Buhusi, M. C. (2009). Interval timing accuracy and scalar timing in c57bl/6 mice. *Behavioral Neuroscience* **123**, 5, pp. 1102–1113, doi:10.1037/a0017106.

Chhikara, R. S. and Folks, J. L. (1974). Estimation of the inverse Gaussian distribution function, *Journal of the American Statistical Association* **69**, 345, pp. 250–254, doi:10.1080/01621459.1974.10480165.

Chhikara, R. S. and Folks, J. L. (1977). The inverse gaussian distribution as a lifetime model, *Technometrics* **19**, 4, pp. 461–468, doi:10.1080/00401706.1977.10489586.

Chhikara, R. S. and Folks, J. L. (1989). *The Inverse Gaussian Distribution: Theory, Methodology, and Applications* (Marcel Dekker, New York and Basel).

Church, R. M. (2012). Temporal learning in humans and other animals, in N. M. Seel (ed.), *Encyclopedia of the Sciences of Learning* (Springer US), pp. 3301–3303, doi:10.1007/978-1-4419-1428-6_66.

de Cani, J. S. (1964). A dynamic programming algorithm for embedded Markov chains when the planning horizon is at infinity, *Management Science* **10**, 4, pp. 716–733, doi:10.1287/mnsc.10.4.716.

Dearden, R., Friedman, N., and Russell, S. (1998). Bayesian Q-learning, in *Proceedings of the Fifteenth National/Tenth Conference on Artificial Intelligence/Innovative Applications of Artificial Intelligence*, AAAI '98/IAAI '98 (American Association for Artificial Intelligence), ISBN 0-262-51098-7, pp. 761–768.

DeGroot, M. H. (1970). *Optimal Statistical Decisions*, McGraw-Hill Series in Probability and Statistics (McGraw-Hill Book Company, New York), doi:10.1002/0471729000.

d'Epenoux, F. (1960). Sur un problème de production et de stockage dans l'aléatoire, *Revue Française de Recherche Opérationnelle* , 14, for a revised and expanded English translation see d'Epenoux (1963).

d'Epenoux, F. (1963). A probabilistic production and inventory problem, *Management Science* **10**, 1, pp. 98–108, doi:10.1287/mnsc.10.1.98.

Drake, A. W. (1962). *Observation of a Markov Process through a Noisy Channel*, Ph.D. thesis, Massachusetts Institute of Technology.

Duff, M. O. (2002). *Optimal Learning: Computational Procedures for Bayes-adaptive Markov Decision Processes*, Ph.D. thesis, University of Massachusetts Amherst.

Dynkin, E. B. (1965). Controlled random sequences, *Theory of Probability & Its Applications* **10**, 1, pp. 1–14, doi:10.1137/1110001.

Fienberg, S. E. (2006). When did Bayesian inference become "Bayesian"? *Bayesian Analysis* **1**, 1, doi:10.1214/06-ba101.

Fisher, R. A. (1922). On the mathematical foundations of theoretical statistics, *Philosophical Transactions of the Royal Society of London A: Mathematical, Physical and Engineering Sciences* **222**, pp. 309–368, doi:10.1098/rsta.1922.0009.

Gallistel, C. R. and Gibbon, J. (2000). Time, rate, and conditioning, *Psychological Review* **107**, 2, pp. 289–344, doi:10.1037/0033-295X.107.2.289.

Gers, F. A., Schraudolph, N. N., and Schmidhuber, J. (2002). Learning precise timing with LSTM recurrent networks, *Journal of Machine Learning Research* **3**, 1, pp. 115–143.

Ghavamzadeh, M., Mannor, S., Pineau, J., and Tamar, A. (2015). Bayesian reinforcement learning: A survey, *Foundations and Trends in Machine Learning* **8**, 5-6, pp. 359–492, doi:10.1561/2200000049.

Gibbon, J. (1977). Scalar expectancy theory and weber's law in animal timing, *Psychological Review* **84**, 3, pp. 279–325, doi:10.1037/0033-295X.84.3.279.

Good, I. J. (1952). Rational decisions, *Journal of the Royal Statistical Society: Series B (Methodological)* **14**, 1, pp. 107–114, doi:10.1111/j.2517-6161.1952.tb00104.x.

Gut, A. (2009). *An Intermediate Course in Probability*, Springer Texts in Statistics (Springer), doi:10.1007/978-1-4419-0162-0.

Hammersley, J. M. and Handscomb, D. C. (1964). *Monte Carlo Methods*, Monographs on Applied Probability and Statistics (John Wiley and Sons, New York), doi:10.1007/978-94-009-5819-7.

Hauskrecht, M. and Fraser, H. (2000). Planning treatment of ischemic heart disease with partially observable markov decision processes, *Artificial Intelligence in Medicine* **18**, 3, pp. 221–244, doi:10.1016/s0933-3657(99)00042-1.

Hernández-Lerma, O. (1989). *Adaptive Markov Control Processes*, *Applied Mathematical Sciences*, Vol. 79 (Springer-Verlag, New York), doi:10.1007/978-1-4419-8714-3.

Hogg, R. V., McKean, J. W., and Craig, A. T. (2013). *Introduction to Mathematical Statistics*, 7th edn. (Pearson).

Howard, R. A. (1958). *Studies in Discrete Dynamic Programming*, Ph.D. thesis, Massachusetts Institute of Technology.

Howard, R. A. (1963). Semi-Markovian decision processes, *Bulletin of the International Statistical Institute* **40**, 2, pp. 625–652, from the Proceedings of the 34th Session of the International Statistical Institute, Ottawa, Canada, August 21-29, 1963.

Howard, R. A. (1971). *Dynamic Probabilistic Systems—Volume II: Semi-Markov and Decision Processes* (John Wiley and Sons).

Iwase, K. and Setô, N. (1983). Uniformly minimum variance unbiased estimation for the inverse Gaussian distribution, *Journal of the American Statistical Association* **78**, 383, pp. 660–663, doi:10.1080/01621459.1983.10478026.

Izadi, M. T. and Precup, D. (2005). Using rewards for belief state updates in partially observable Markov decision processes, in J. Gama, R. Camacho, P. B. Brazdil, A. Jorge, and L. Torgo (eds.), *Machine Learning: ECML 2005*, *Lecture Notes in Computer Science*, Vol. 3720 (Springer), pp. 593–600, doi:10.1007/11564096_58.

Jewell, W. S. (1963a). Markov-renewal programming: I. Formulation, finite return models, *Operations Research* **11**, 6, pp. 938–948, doi:10.1287/opre.11.6.938.

Jewell, W. S. (1963b). Markov-renewal programming: II. Infinite return models, example, *Operations Research* **11**, 6, pp. 949–971, doi:10.1287/opre.11.6.949.

Kaelbling, L. P., Littman, M. L., and Cassandra, A. R. (1996). Partially observable Markov decision processes for artificial intelligence, in L. Dorst, M. van Lambalgen, and F. Voorbraak (eds.), *Reasoning with Uncertainty in Robotics: International Workshop, RUR '95 Amsterdam, The Netherlands December 4–6, 1995 Proceedings*, *Lecture Notes in Computer Science*, Vol. 1093 (Springer Berlin Heidelberg), pp. 146–163, doi:10.1007/BFb0013957.

Kaelbling, L. P., Littman, M. L., and Cassandra, A. R. (1998). Planning and acting in partially observable stochastic domains, *Artificial Intelligence* **101**, 1-2, pp. 99–134, doi:10.1016/s0004-3702(98)00023-x.

Kalman, R. E. (1960). A new approach to linear filtering and prediction problems, *Transactions of the ASME–Journal of Basic Engineering* **82**, Series D, pp. 35–45.

Krishnamurthy, V. (2016). *Partially Observed Markov Decision Processes: From Filtering to Controlled Sensing* (Cambridge University Press).

Lévy, P. (1956). Processus semi-markoviens, in *Proceedings of the International Congress of Mathematics 1954*, Vol. 3 (Erven P. Noordhoff N.V., Groningen and North-Holland Publishing Co., Amsterdam), pp. 416–426.

Littman, M. L. and Sutton, R. S. (2002). Predictive representations of state, in T. G. Dietterich, S. Becker, and Z. Ghahramani (eds.), *Advances in Neural Information Processing Systems 14* (MIT Press), pp. 1555–1561.

Lovejoy, W. S. (1991). Computationally feasible bounds for partially observed Markov decision processes, *Operations Research* **39**, 1, pp. 162–175, doi:10.1287/opre.39.1.162.

Luzardo, A., Ludvig, E. A., and Rivest, F. (2013). An adaptive drift-diffusion model of interval timing dynamics, *Behavioural Processes* **95**, pp. 90–99, doi:10.1016/j.beproc.2013.02.003.

Luzardo, A., Rivest, F., Alonso, E., and Ludvig, E. A. (2017). A drift-diffusion model of interval timing in the peak procedure, *Journal of Mathematical Psychology* **77**, pp. 111–123, doi:10.1016/j.jmp.2016.10.002.

Madani, O., Hanks, S., and Condon, A. (1999). On the undecidability of probabilistic planning and infinite-horizon partially observable Markov decision problems, in *Proceedings of the Sixteenth National Conference on Artificial Intelligence*, pp. 541–548.

Madani, O., Hanks, S., and Condon, A. (2003). On the undecidability of probabilistic planning and related stochastic optimization problems, *Artificial Intelligence* **147**, 1-2, pp. 5–34, doi:10.1016/s0004-3702(02)00378-8.

Maniadakis, M. and Trahanias, P. (2011). Temporal cognition: A key ingredient of intelligent systems, *Frontiers in Neurorobotics* **5**, p. 2, doi:10.3389/fnbot.2011.00002.

Martin, J. J. (1965). *Some Bayesian Decision Problems in a Markov Chain*, Ph.D. thesis, Massachusetts Institute of Technology, supervised by Ronald A. Howard.

Martin, J. J. (1967). *Bayesian Decision Problems and Markov Chains*, *Publications in Operations Research*, Vol. 13 (John Wiley and Sons).

Metropolis, N., Rosenbluth, A. W., Rosenbluth, M. N., Teller, A. H., and Teller, E. (1953). Equation of state calculations by fast computing machines, *The Journal of Chemical Physics* **21**, 6, pp. 1087–1092, doi:10.1063/1.1699114.

Metropolis, N. and Ulam, S. (1949). The monte carlo method, *Journal of the American Statistical Association* **44**, 247, pp. 335–341, doi:10.1080/01621459.1949.10483310.

Michael, J. R., Schucany, W. R., and Haas, R. W. (1976). Generating random variables using transformations with multiple roots, *The American Statistician* **30**, 2, pp. 88–90, doi:10.2307/2683801.

Mitchell, T. M. (1997). *Machine Learning* (McGraw-Hill).

Monahan, G. E. (1982). State of the art—a survey of partially observable Markov decision processes: Theory, models, and algorithms, *Management Science* **28**, 1, pp. 1–16, doi:10.1287/mnsc.28.1.1.

Ong, S. C. W., Png, S. W., Hsu, D., and Lee, W. S. (2010). Planning under uncertainty for robotic tasks with mixed observability, *International Journal of Robotics Research* **29**, 8, pp. 1053–1068, doi:10.1177/0278364910369861.

Papadimitriou, C. H. and Tsitsiklis, J. N. (1987). The complexity of Markov decision processes, *Mathematics of Operations Research* **12**, 3, pp. 441–450, doi:10.1287/moor.12.3.441.

Pineau, J., Gordon, G., and Thrun, S. (2003). Point-based value iteration: An anytime algorithm for POMDPs, in *International Joint Conference on Artificial Intelligence (IJCAI)*, pp. 1025–1030.

Pineau, J., Gordon, G., and Thrun, S. (2006). Anytime point-based approximations for large POMDPs, *Journal of Artificial Intelligence Research* **27**, pp. 335–380, doi:10.1613/jair.2078.

Porta, J. M., Spaan, M. T. J., and Vlassis, N. (2005). Robot planning in partially observable continuous domains, in *Robotics: Science and Systems I* (Robotics: Science and Systems Foundation, Cambridge, USA), doi:10.15607/rss.2005.i.029.

Porta, J. M., Vlassis, N., Spaan, M. T. J., and Poupart, P. (2006). Point-based value iteration for continuous POMDPs, *Journal of Machine Learning Research* **7**, pp. 2329–2367.

Poupart, P., Vlassis, N., Hoey, J., and Regan, K. (2006). An analytic solution to discrete bayesian reinforcement learning, in *Proceedings of the 23rd International Conference on Machine Learning* (ACM Press), ISBN 1-59593-383-2, pp. 697–704, doi:10.1145/1143844.1143932.

Powell, W. B. (2011). *Approximate Dynamic Programming: Solving the Curse of Dimensionality*, 2nd edn., Wiley Series in Probability and Statistics (John Wiley & Sons), doi:10.1002/9781118029176.

Precup, D. (2000). *Temporal Abstraction in Reinforcement Learning*, Ph.D. thesis, University of Massachusetts Amherst.

Puterman, M. L. (1994). *Markov Decision Processes: Discrete Stochastic Dynamic Programming* (John Wiley & Sons), doi:10.1002/9780470316887.

Raiffa, H. and Schlaifer, R. (1961). *Applied Statistical Decision Theory*, Studies in Managerial Economics (Division of Research, Graduate School of Business Administration, Harvard University).

RAND Corporation (1955). *A Million Random Digits with 100,000 Normal Deviates* (The Free Press).

Ren, G. and Stachurski, J. (2021). Dynamic programming with value convexity, *Automatica* **130**, p. 109641, doi:10.1016/j.automatica.2021.109641.

Rivest, F. and Bengio, Y. (2011). Adaptive drift-diffusion process to learn time intervals, doi:10.48550/arXiv.1103.2382, arXiv:1103.2382.

Rivest, F., Kalaska, J. F., and Bengio, Y. (2010). Alternative time representation in dopamine models, *Journal of Computational Neuroscience* **28**, 1, pp. 107–130, doi:10.1007/s10827-009-0191-1.

Rivest, F. and Kohar, R. (2020). A new timing error cost function for binary time series prediction, *IEEE Transactions on Neural Networks and Learning Systems* **31**, 1, pp. 174–185, doi:10.1109/tnnls.2019.2900046.

Robbins, H. and Monro, S. (1951). A stochastic approximation method, *The Annals of Mathematical Statistics* **22**, 3, pp. 400–407, doi:10.1214/aoms/1177729586.

Robert, C. P. (1995). Simulation of truncated normal variables, *Statistics and Computing* **5**, 2, pp. 121–125, doi:10.1007/bf00143942.

Robert, C. P. (2007). *The Bayesian Choice: From Decision-Theoretic Foundations to Computational Implementation*, Springer Texts in Statistics (Springer), doi:10.1007/0-387-71599-1.

Ross, S., Chaib-draa, B., and Pineau, J. (2008a). Bayes-adaptive POMDPs, in J. C. Platt, D. Koller, Y. Singer, and S. T. Roweis (eds.), *Advances in Neural Information Processing Systems 20* (Curran Associates, Inc.), pp. 1225–1232.

Ross, S., draa, B. C., and Pineau, J. (2007). Bayes-adaptive POMDPs, Tech. Rep. SOCS-TR-2007.6, McGill University.

Ross, S., Pineau, J., Chaib-draa, B., and Kreitmann, P. (2011). A bayesian approach for learning and planning in partially observable Markov decision processes, *Journal of Machine Learning Research* **12**, pp. 1729–1770.

Ross, S., Pineau, J., Paquet, S., and Chaib-draa, B. (2008b). Online planning algorithms for POMDPs, *Journal of Artificial Intelligence Research* **32**, pp. 663–704, doi:10.1613/jair.2567.

Ross, S. M. (1970a). *Applied Probability Models with Optimization Applications* (Holden-Day).

Ross, S. M. (1970b). Average cost semi-Markov decision processes, *Journal of Applied Probability* **7**, 3, pp. 649–656, doi:10.2307/3211944.

Russell, S. and Norvig, P. (2010). *Artificial Intelligence: A Modern Approach*, 3rd edn. (Pearson, New Jersey).

Savage, L. J. (1954). *The Foundations of Statistics*, 1st edn. (John Wiley and Sons).

Sawaragi, Y. and Yoshikawa, T. (1970). Discrete-time Markovian decision processes with incomplete state observation, *The Annals of Mathematical Statistics* **41**, 1, pp. 78–86, doi:10.1214/aoms/1177697190.

Schmidhuber, J. (2007). 2006: Celebrating 75 years of AI—History and Outlook: The Next 25 Years, in M. Lungarella, F. Iida, J. Bongard, and R. Pfeifer (eds.), *50 Years of Artificial Intelligence: Essays Dedicated to the 50th Anniversary of Artificial Intelligence*, *Lecture Notes in Computer Science*, Vol. 4850 (Springer Berlin Heidelberg), ISBN 978-3-540-77295-8, pp. 29–41, doi:10.1007/978-3-540-77296-5_4.

Schrödinger, E. (1915). Zur theorie der fall- und steigversuche an teilchen mit brownscher bewegung, *Physikalische Zeitschrift* **16**, pp. 289–295.

Shapley, L. S. (1953). Stochastic games, *Proceedings of the National Academy of Sciences* **39**, 10, pp. 1095–1100, doi:10.1073/pnas.39.10.1095.

Shuster, J. (1968). On the inverse Gaussian distribution function, *Journal of the American Statistical Association* **63**, 324, pp. 1514–1516, doi:10.1080/01621459.1968.10480942.

Silver, D., Hubert, T., Schrittwieser, J., Antonoglou, I., Lai, M., Guez, A., Lanctot, M., Sifre, L., Kumaran, D., Graepel, T., Lillicrap, T., Simonyan, K., and Hassabis, D. (2018). A general reinforcement learning algorithm that masters chess, shogi, and go through self-play, *Science* **362**, 6419, pp. 1140–1144, doi:10.1126/science.aar6404.

Silver, D., Singh, S., Precup, D., and Sutton, R. S. (2021). Reward is enough, *Artificial Intelligence* **299**, 103535, pp. 1–13, doi:10.1016/j.artint.2021.103535.

Simen, P., Balci, F., de Souza, L., Cohen, J. D., and Holmes, P. (2011). A model of interval timing by neural integration, *Journal of Neuroscience* **31**, 25, pp. 9238–9253, doi:10.1523/jneurosci.3121-10.2011.

Smallwood, R. D. and Sondik, E. J. (1973). The optimal control of partially observable Markov processes over a finite horizon, *Operations Research* **21**, 5, pp. 1071–1088, doi:10.1287/opre.21.5.1071.

Smith, W. L. (1955). Regenerative stochastic processes, *Proceedings of the Royal Society A: Mathematical, Physical and Engineering Sciences* **232**, 1188, pp. 6–31, doi:10.1098/rspa.1955.0198.

Sondik, E. J. (1971). *The Optimal Control of Partially Observable Markov Processes*, Ph.D. thesis, Standford University.

Sondik, E. J. (1978). The optimal control of partially observable Markov processes over the infinite horizon: Discounted costs, *Operations Research* **26**, 2, pp. 282–304, doi:10.1287/opre.26.2.282.

Spaan, M. T. J. (2012). Partially observable Markov decision processes, in M. Wiering and M. van Otterlo (eds.), *Reinforcement Learning: State-of-the-Art*, *Adaptation, Learning, and Optimization*, Vol. 12, chap. 12 (Springer Berlin Heidelberg), pp. 387–414, doi:10.1007/978-3-642-27645-3_12.

Spaan, M. T. J. and Vlassis, N. (2005). Perseus: Randomized point-based value iteration for POMDPs, *Journal of Artificial Intelligence Research* **24**, pp. 195–220, doi:10.1613/jair.1659.

Stratonovich, R. L. (1960). Conditional Markov processes, *Theory of Probability and its Applications* **5**, 2, pp. 156–178, doi:10.1137/1105015.

Strens, M. (2000). A Bayesian framework for reinforcement learning, in *Proceeedings of the Seventeenth International Conference on Machine Learning*.

Sutton, R. S. (1984). *Temporal Credit Assignment in Reinforcement Learning*, Ph.D. thesis, University of Massachusetts.

Sutton, R. S. (1988). Learning to predict by the methods of temporal differences, *Machine Learning* **3**, 1, pp. 9–44, doi:10.1007/BF00115009.

Sutton, R. S. and Barto, A. G. (1998). *Reinforcement Learning: An Introduction*, Adaptive Computation and Machine Learning (MIT Press), ISBN 9780262193986.

Sutton, R. S. and Barto, A. G. (2018). *Reinforcement Learning: An Introduction*, 2nd edn., Adaptive Computation and Machine Learning (MIT Press).

Sutton, R. S., Precup, D., and Singh, S. (1999). Between MDPs and semi-MDPs: A framework for temporal abstraction in reinforcement learning, *Artificial Intelligence* **112**, 1-2, pp. 181–211, doi:10.1016/S0004-3702(99)00052-1.

Thrun, S. B. (1992). Efficient exploration in reinforcement learning, Tech. rep., Carnegie-Mellon University.

Tweedie, M. C. K. (1957). Statistical properties of inverse Gaussian distributions. I, *The Annals of Mathematical Statistics* **28**, pp. 362–377, doi:10.1214/aoms/1177706964.

von Neumann, J. (1928). Zur theorie der gesellschaftsspiele, *Mathematische Annalen* **100**, 1, pp. 295–320, doi:10.1007/bf01448847.

Wakuta, K. (1981). Semi-Markov decision processes with incomplete state observation: average cost criterion, *Journal of the Operations Research Society of Japan* **24**, 2, pp. 95–108, doi:10.15807/jorsj.24.95.

Wakuta, K. (1982). Semi-Markov decision processes with incomplete state observation: discounted cost criterion, *Journal of the Operations Research Society of Japan* **25**, 4, pp. 351–361, doi:10.15807/jorsj.25.351.

Wald, A. (1945). Sequential tests of statistical hypotheses, *The Annals of Mathematical Statistics* **16**, 2, pp. 117–186, doi:10.1214/aoms/1177731118.

Wald, A. (1947). Foundations of a general theory of sequential decision functions, *Econometrica* **15**, 4, p. 279, doi:10.2307/1905331.

Watkins, C. J. C. H. (1989). *Learning from Delayed Rewards*, Ph.D. thesis, King's College London.

Watkins, C. J. C. H. and Dayan, P. (1992). $Q$-Learning, *Machine Learning* **8**, 3, pp. 279–292, doi:10.1023/A:1022676722315.

Wetherill, G. B. (1961). Bayesian sequential analysis, *Biometrika* **48**, 3-4, pp. 281–292, doi:10.1093/biomet/48.3-4.281.

White, C. C. (1976). Procedures for the solution of a finite-horizon, partially observed, semi-Markov optimization problem, *Operations Research* **24**, 2, pp. 348–358, doi:10.1287/opre.24.2.348.

Young, S., Gasic, M., Thomson, B., and Williams, J. D. (2013). POMDP-based statistical spoken dialog systems: A review, *Proceedings of the IEEE* **101**, 5, pp. 1160–1179, doi:10.1109/jproc.2012.2225812.

Yu, H. (2006). *Approximate Solutions for Partially Observable Markov and Semi-Markov Decision Processes*, Ph.D. thesis, Massachusetts Institute of Technology.

Zhang, M. and Revie, M. (2017). Continuous-observation partially observable semi-Markov decision processes for machine maintenance, *IEEE Transactions on Reliability* **66**, 1, pp. 202–218, doi:10.1109/tr.2016.2626477.

Zhang, N. L. and Zhang, W. (2001). Speeding up the convergence of value iteration in partially observable Markov decision processes, *Journal of Artificial Intelligence Research* **14**, pp. 29–51, doi:10.1613/jair.761.

# Index