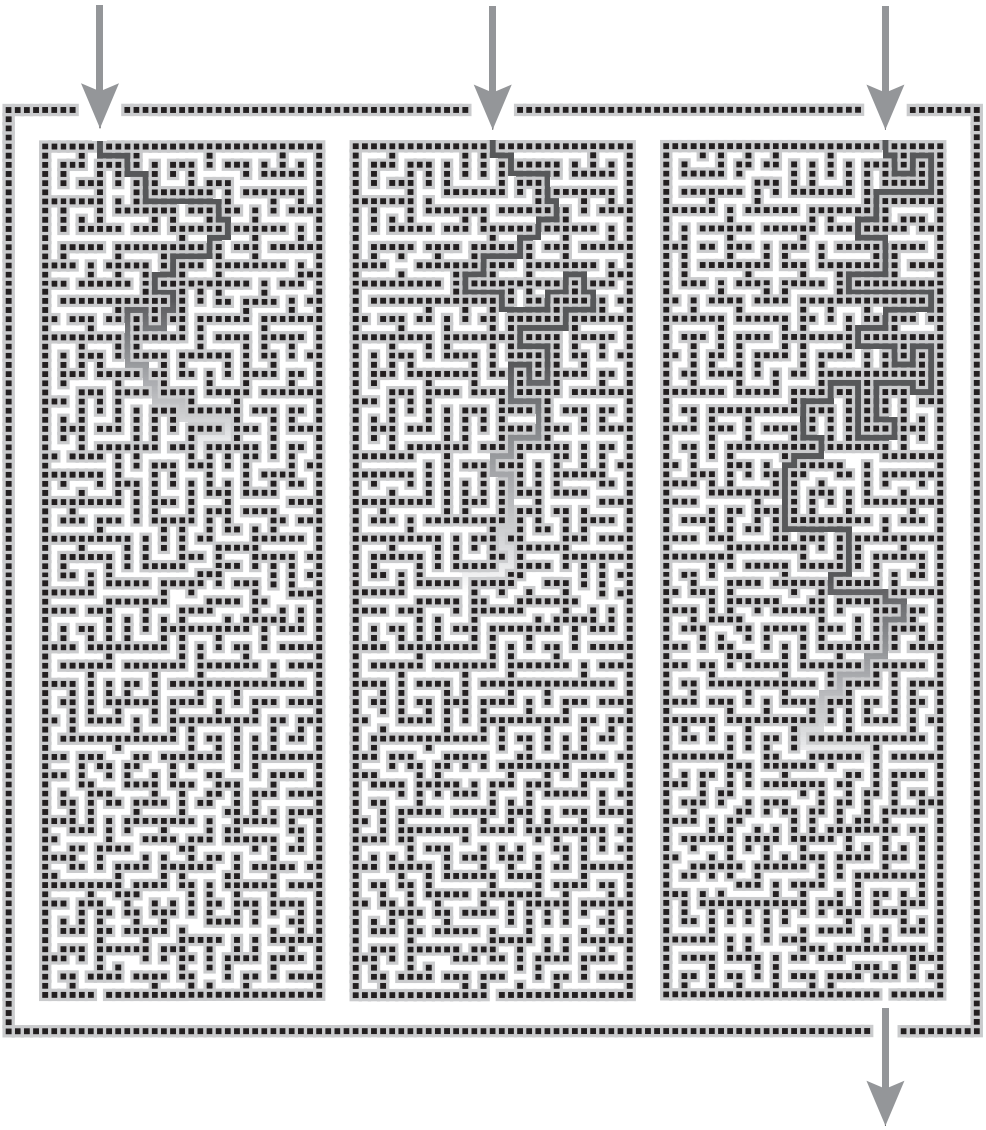
Master Thesis Computer Science

Marnix Suilen



Entropy Guided Decision Making in Multiple-Environment Markov Decision Processes

Master Thesis Computer Science

Radboud University

Entropy Guided Decision Making in Multiple-Environment Markov Decision Processes

Author: Marnix Suilen
Email: marnix@marnixsuilen.nl
Submitted on: August 11, 2020

First supervisor & assessor: Dr. Nils Jansen

Second supervisor: Dr. Sebastian Junges

Second assessor: Prof. dr. Frits Vaandrager

Cover illustration: Floris Suilen

Acknowledgements

This thesis was written in what we as of now understand to be *unprecedented times*. It was certainly not how I expected to finish my master, but now that this thesis is finished, I think it is fair to say that it all worked out.

This would not have been possible without the guidance and feedback from Nils and Sebastian. I hope there will be many other projects on which we can collaborate and look forward to doing so. I would also like to thank Frits for the many valuable comments and corrections, my brother Floris for designing the cover illustration, and my parents for their support in general.

Nijmegen, August 2020.

It's just a spark
But it's enough to keep me going
And when it's dark out, no one's around
It keeps glowing

—from *Last Hope* by Paramore

Abstract

We study *multiple-environment Markov decision processes* (MEMDPs). MEMDPs form a set of standard Markov decision processes (MDPs) that share the same states and actions, and only differ in their transition functions. A MEMDP thus forms a set of potential models, called environments, of a real-world system. By assumption, exactly one of these environments is the *true environment*, meaning it captures the behavior of the real-world system precisely.

A MEMDP can be seen as a partially observable MDP (POMDP) that exhibits a special structure. The belief update is now linear in the number of environments, versus quadratic in the number of states for arbitrary POMDPs. Furthermore, using the Shannon entropy as a measure of how much knowledge of the true environment we currently have, it is known that we cannot lose any information on average.

We show under which conditions our knowledge of the true environment will strictly increase on average, and under which conditions this knowledge stays the same. We use this to construct an iterative, greedy learning algorithm to infer the true environment. Based on the entropy we choose the action that is expected to lead to the largest increase in knowledge at every step. We discuss potential drawbacks to this algorithm and present an optimized version that has a lower time complexity per iteration.

Contents

Chapter 1

Introduction

Markov decision processes (MDPs) are the standard model to capture decision making in probabilistic environments [Puterman, 1994]. An MDP consists of a set of states, a set of actions, and a probabilistic transition function. The decision making is done a so-called *agent*. At every state in the model, the agent makes a decision by choosing one of the actions via a so-called *strategy*. Then, based on the action choice at the given state, the successor state is determined probabilistically.

For example, we can model a maze as an MDP [McCallum, 1993]. Every tile in the maze gets its own state. The actions are the four cardinal directions. The transition function simply moves to an adjacent tile of the chosen direction with probability 1 if this is possible. If no such tile exists, there is a transition of probability 1 to stay at the current tile. A more sophisticated model would be to make the maze *slippery*, meaning there is a small probability for the agent to end up on a different tile than the one they attempt to move towards to.

The goal is to compute a strategy, a sequence action choices, that takes the agent from the initial position I to the target position T. We illustrate such a maze in Figure 1.

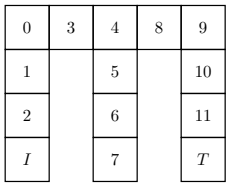


Figure 1: A fully observable map of a maze.

One of the key assumptions underlying MDPs is that the state-space is *fully observable*, meaning the agent always has perfect knowledge about the current state. In our maze example, this means that the agent always knows their exact position in the maze, hence assigning a unique number to each tile.

Partially observable Markov decision processes (POMDPs) remove this assumption by making the state-space *partially observable* [Kaelbling, Littman, and Cassandra, 1998]. As a consequence, the agent can no longer make decisions based on the current state, but only based on the observations.

This means that instead of knowing the exact state, the agent now has a *belief* of where they are. A belief is a probability distribution over the states that can be updated upon performing an action and receiving a new observation via Bayes' rule. This is called the *belief update*.

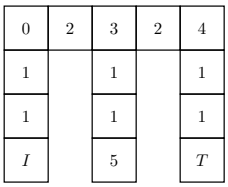


Figure 2: A partially observable map of a maze.

The maze example can be adapted to a POMDP. Instead of knowing their exact position, the agent now has to *infer* their position based on *observations*. The observations are the positions of the walls. For instance, the agent may observe a wall to their left and a wall to their right. This is illustrated in Figure 2. Note that all tiles with walls on the left and right have the same number, 1, that represents the observation.

When moving from I to T, this clearly is a problem. At the tiles with observation 1 directly above I the agent should move up, but at the tiles above T the agent should move down. But the agent only knows the current observation, making it impossible to distinguish between these cases.

If the agent were to be able to distinguish between said cases, which can be done by incorporating *finite memory* in the strategy, they can still reach T. Alternatively, when finite memory is not an option or the amount of memory needed is too large, *randomization* can help. Instead of moving either up or down at a state with observation 1, the agent can now probabilistically move up with some probability p and down with probability $1 - p$. Randomization can reduce the amount of memory needed [Chatterjee, De Alfaro, and Henzinger, 2004].

As a result there is a significant jump in complexity between MDPs and POMDPs. Problems that can be solved in polynomial time for MDPs, such as computing a strategy that satisfies some quantitative reachability or safety objective, are undecidable [Chatterjee, Chmelík, and Tracol, 2016], and at best (with additional assumptions) still NP-hard [Vlassis, Littman, and Barber, 2012] for POMDPs.

Multiple-environment Markov decision processes (MEMDPs) are a relatively new formalism defining a *finite set* of MDPs that share the same states and actions [Raskin and Sankur, 2014]. A MEMDP can be seen as a set of potential scenarios describing some real-world system. Each MDP in a MEMDP is called an *environment*. It is assumed that there is one *true environment* that describes the real-world system exactly. Which of the environments is the true environment is not known. But whichever environment is the true environment, it does not change over time.

We can adapt our maze example to a MEMDP. Instead of having an exact map of the maze, like in an MDP, we now have a number of maps, one of which is the real map and the others are not. This scenario is illustrated in Figure 3.

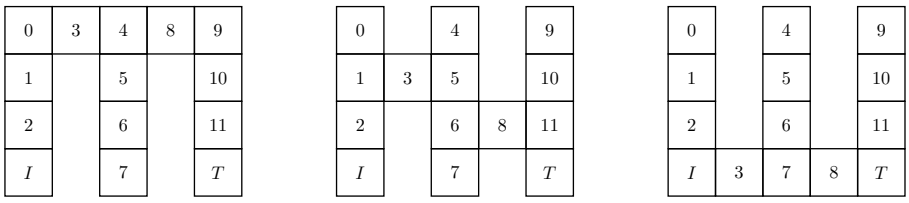


Figure 3: Three different possible maps of a maze.

If the agent wants to move from I to T, there are two approaches they can take. The first approach is via a *robust strategy*. A robust strategy is a strategy that works in *all* environments. Note that such a strategy may not exist, and if it does, it may be harder to compute, see for example quantitative reachability, which is NP-hard for MEMDPs [Raskin and Sankur, 2014], but solvable in polynomial time for MDPs [Baier and Katoen, 2008].

The second approach is to infer the true environment via *learning*. Suppose the map on the left of Figure 3 is the true environment. When the agent is currently at tile I, and tries to move to the right, it will stay at I. This can only happen in environments 1 (the true environment) and 2. In environment 3, a move to the right from I is possible and the agent would end up in tile 3. Thus, from observing that the agent stays at tile I we have learned that environment 3 *cannot* be the true environment.

Moving up to tile 1 and repeating the same procedure leads us to also exclude environment 2. As only environment 1 now remains, we know with certainty that this is the true environment.

Such transitions, that allow us to remove one of the environments from the MEMDP, are formalized as *reducing transitions*. A transition that instantly reveals the true environment, as we observe a successor state that is only possible in one of the environments, is called a *revealing transition*.

In the case of the slippery maze, we cannot draw such strong conclusions. Not observing a certain successor state may in the slippery maze be due to bad luck with the probabilities, instead of it being impossible to observe that successor state.

Thus, for the general case where there are not necessarily revealing or reducing transitions available, we need a more involved method to infer the true environment based on a sequence of observations. This is possible since any sequence of observations will have a different probability of occurring in each environment, *allowing us to distinct between the environments if we have enough observations*.

MEMDPs provide a computational advantage over POMDPs. Problems that are undecidable for POMDPs turn out to be decidable for MEMDPs. Compared to MDPs, these problems may be harder to solve though. Take for example the quantitative reachability problem mentioned before. This problem is undecidable in POMDPs, solvable in polynomial time for MDPs, and NP-hard in MEMDPs [Raskin and Sankur, 2014]. Besides the computational advantage of MEMDPs over POMDPs, many problems that are traditionally modeled as a POMDP can also be modeled as a MEMDP [Chatterjee et al., 2020]. Essentially, this means that POMDPs are stronger than necessary for many applications.

Problem statement

In this thesis, we develop a method for deriving a strategy that will *infer the true environment of a MEMDP by learning*.

Our motivation for this attempting to solve this problem is simple: once the true environment has been found, optimal solutions for many kinds of objectives can be computed more efficiently, as the true environment is just a standard MDP.

Contributions

Our contributions can be summarized as follows.

We relate MEMDPs to POMDPs to define the notion of a *belief over environments*. A belief is a probability distribution over the environments that tells us how likely each environment is the true environment. The belief update from POMDPs can be adapted to update the belief over environments, which enables us to transform our belief when performing an action and observing a successor state.

Using the Shannon entropy [Shannon, 2001], a standard function that measures the amount of information a probability distribution provides, it is known that *on average* we will *not lose* any information [Chatterjee et al., 2020] on the belief of the true environment in a MEMDP. We make this more precise by showing in which cases the average entropy *stays the same*. We conjecture that in any other case there is a *strict gain* in information on the true environment on average.

With the entropy, we can make more informed decisions than when just picking actions at random. This leads us to a naive greedy algorithm that learns the true environment by picking the action that, on average, will give us the greatest increase in knowledge on the true environment.

Using the Bhattacharyya distance [Bhattacharyya, 1943] to measure the distance between probability distributions on successor states we conjecture that the greater this distance is, the greater the gain in information will be on average. We use this to optimize our algorithm. The Bhattacharyya distance remains the same for every iteration of the algorithm, whereas the expected entropy has to be computed again at every iteration. Hence, replacing the expected entropy by the Bhattacharyya distance reduces the time complexity per iteration.

Besides our main results on learning the true environment in a MEMDP, we provide a number of additional results on MEMDPs. We define *bisimulations* between MEMDPs. We provide concrete *algorithms* to detect revealing and reducing transitions. We show how robust strategies for any objective can be computed by *approximating* the MEMDP by an uncertain MDP. We prove that the quantitative expected reward problem is *NP-hard in MEMDPs*. Finally, we give a mixed integer linear program that computes *deterministic* strategies that satisfy some objective.

Throughout the thesis the most important concepts are illustrated with examples. All lemmas and theorems are proven. Propositions are the results of others. Statements that have no formal proof associated with them are labeled as conjectures.

Structure of the thesis

The thesis is structured as follows.

- In Chapter 2 related work is discussed.
- In Chapter 3 we briefly cover some preliminaries.
- In Chapter 4 we give formal definitions of the relevant models and objectives, and compare various problems on their complexity results.
- In Chapter 5 we present a number of new results on MEMDPs, unrelated to the problem of learning the true environment.
- In Chapter 6 we present our main results on learning in MEMDPs. This is structured in the following sections.
  - In section 6.1 we relate MEMDPs to POMDPs and discuss how the belief update works in MEMDPs.
  - In Section 6.2 we introduce entropy as a measure on how much knowledge on the current environment we currently have, and how this knowledge changes upon taking an action.
  - In Section 6.3 we formalize the behavior in a number of theorems.
  - In Section 6.4 we construct a greedy learning algorithm that chooses the action that is expected to give the greatest increase of knowledge on the true environment.
  - In Section 6.5 we optimize this algorithm by preprocessing.
- In Chapter 7 we summarize our conclusions and provide a number of pointers for future work.

Chapter 2

Related work

MEMDPs were introduced in [Raskin and Sankur, 2014]. They primarily focus on computing strategies that solve standard problems for probabilistic models, such as reachability, safety, and parity objectives. The notion of i-revealing transitions is introduced. An i-revealing transition is only possible in one of the environments, thus once we observe such a transition, we know the true environment.

For limit-sure objectives, [Raskin and Sankur, 2014] introduce *double end-components*, named as such because they only consider MEMDPs of two environments. Double end-components are end-components [Baier and Katoen, 2008] that occur in each environment. They note that if there is a transition that is different in each environment in such a double end-component, one can distinguish between environments. This is however not used to reveal the true environment. Furthermore, we show that it is also possible to learn outside end-components.

In [Chatterjee et al., 2020], MEMDPs are studied further. They make the connection to POMDPs and the idea that the belief in a MEMDP is a probability distribution over environments. They show that, using the entropy as a measure for how much information we currently have on the true environment, on average *no information will be lost*. They do not detail this any further, and do not use it for e.g. learning or as a heuristic in their experiments. In contrast, we show in which cases there is expected to be no loss or gain on information, and in which cases there is a strict gain on average, and base our learning algorithm on this.

Before [Raskin and Sankur, 2014], it was [Chadès et al., 2012] who introduced *hidden model MDPs* (hmMDPs). Instead of extending MDPs, they start with *mixed observability MDPs* (MOMDPs) [Ong et al., 2010] and introduce restriction, effectively arriving at hmMDPs. A hmMDP

is effectively a set of fully observable MDPs, hence a MEMDP. They show that the finite horizon reward maximization problem is PSPACE-complete (just as in POMDPs) and apply their findings to a case study on bird preservation.

Besides the three main works on MEMDPs (or hmMDPs) mentioned above, some other works can easily be related to MEMDPs.

Uncertainty in MDPs

Interval MDPs [Nilim and El Ghaoui, 2005], robust MDPs [Wiesemann, Kuhn, and Rustem, 2013], or parametric MDPs [Junges et al., 2019] all consider MDPs with uncertainty in the transition probabilities. These can be seen as the continuous counterpart to the discrete uncertainty MEMDPs provide. Uncertain POMDPs [Suilen et al., 2020] extend the idea of uncertain transition probabilities to POMDPs.

Reinforcement learning in (PO)MDPs

Reinforcement learning [Kaelbling, Littman, and Moore, 1996] is a general topic in which one tries to find a strategy that performs well in an unknown environment, typically by trying to maximize some reward. In reinforcement learning, when applied to an MDP, the transition probabilities of the MDP are typically unknown. In contrast, in our approach to learn the true environment of the MEMDP, all transition probabilities are exactly known, but the true environment is unknown.

Standard reinforcement learning does not take any safety considerations into account. It is assumed the learning is unrestricted. This assumption is not valid often, giving rise to the need for *safe* reinforcement learning [Garcıa and Fernández, 2015]. In general terms, safe reinforcement learning is not only concerned with maximizing a reward, but also avoiding 'bad' states.

In [Fulton and Platzer, 2019], the idea of safe reinforcement learning is taken a step further. They introduce a method that learns a number of possible models, and then selects the most likely model based on observations, while satisfying some safety constraint during the entire process. They do, however, not consider probabilistic models such as MDPs.

Besides reinforcement learning there is also work on distinguishing a set of MDPs. These MDPs do not necessarily have the same state-space, thus this is a more general problem than learning the true environment in a MEMDP. It is shown that deciding whether a fixed strategy that distinguishes between MDPs exists is PSPACE-complete. If the strategy may adapted during runtime based on observations, this decision problem becomes EXPTIME-complete [Alur, Courcoubetis, and Yannakakis, 1995].

Learning hidden Markov models

A hidden Markov model (HMM) is a POMDP without any actions. The *learning problem for HMMs* is to find the transition function between states and the output distributions that generate the observations that best fit a given sequence of observations [Rabiner, 1989]. A weaker version of this problem, probably approximate correct learning, is already NP-hard [Terwijn, 2002].

Besides learning HMMs, there is also interest in *distinguishing* (also called *classifying*) HMMs. In this setting, a number of HMMs and a single sequence of observations are given. The problem is to figure out which of the HMMs generated the observations. This is a discrete analogue of the learning problem for HMMs, and can be solved in polynomial time [Akshay et al., 2019]. It should be noted that being unable to *distinguish* two HMMs does not mean they are the same [Kiefer and Sistla, 2016]. This matches with the result that comparing two HMMs via some norm is NP-hard [Lyngsø and Pedersen, 2002].

State identification

For non-probabilistic models, such as finite state machines (FSM), the *state identification* problem is studied. The state identification problem asks, given a FSM, to determine the initial state of the FSM by providing inputs and observing outputs [Lee and Yannakakis, 1994]. This problem is also considered for labeled transition systems (LTS) [van den Bos and Vaandrager, 2019].

State identification can be used to solve the non-probabilistic version of our problem of finding the true environment by taking the disjoint union of all models (like FSMs or LTSs). To the best of our knowledge, state identification for probabilistic systems, such as MDPs, has not been considered yet.

Entropy in (PO)MDPs

Entropy has been used before as a measure in POMDPs. Algorithms for solving POMDPs use the expected change in entropy as a heuristic [Cassandra, 1998]. In [Savas et al., 2018] strategies for MDPs that maximize the exploration of an MDP while satisfying a temporal logic constraint are considered. The amount of exploration is measured via the entropy.

Chapter 3

Preliminaries

For a finite set X we denote the number of elements of X by $|X|$. The set of all finite sequences over X, including the empty sequence, is denoted by X^*. The set of all infinite sequences over X is denoted by X^ω.

We assume the natural numbers $\mathbb{N}$ to start at 0, and denote the set of the natural numbers 1 to n as $[n] = \{1, \ldots, n\}$. The sets of integers, rationals, and reals are denoted as $\mathbb{Z}, \mathbb{Q}$ and $\mathbb{R}$ respectively. A *permutation* on $[n]$ is a bijective function $\phi\colon [n] \to [n]$. We use the symbol $\bowtie$ for any of the standard comparisons on numbers, i.e. $\bowtie \in \{\leq, <, =, >, \geq\}$.

We define the *asymptotic upper bound* of a function $g\colon \mathbb{N} \to \mathbb{N}$, denoted by $\mathcal{O}(g(n))$, as the set of functions $f\colon \mathbb{N} \to \mathbb{N}$ that are asymptotically bounded by g:

$$\mathcal{O}(g(n)) = \{f(n) \mid \exists c \in \mathbb{R}_{>0}, n_0 \in \mathbb{N}.\, \forall n \geq n_0.\, f(n) \leq c \cdot g(n)\}.$$

3.1 Directed graphs

A *directed graph* (digraph) is a tuple (V, E) where V is the set of vertices, and $E \subseteq V \times V$ is the set of edges.

A path in a digraph is a sequence of vertices $v_0 \ldots v_n$ such that each pair of subsequent vertices in the sequence have an edge between them: $\forall i \in [n].(v_{i-1}, v_i) \in E$. A cycle is a path of the form $v_0 \ldots v_n$ where $v_0 = v_n$. A cycle is *simple* if all vertices between the end points are distinct: $v_1 \neq \cdots \neq v_n$. A graph without any simple cycles is *acyclic*. Note that this definition of acyclicity still allows self-loops, e.g. edges of the form (v, v).

A *strongly connected component* is a maximal subset of vertices $C \subseteq V$ where each two vertices $v_1, v_2 \in C$ are mutually reachable, meaning there exists a path from v_1 to v_2 and from v_2 to v_1.

The set of strongly connected components of a digraph can be computed in linear time using depth-first search. For more on graphs, strongly connected components and graph-algorithms, see [Cormen et al., 2009].

3.2 Probability theory

A *discrete probability distribution* over a finite set X is defined as a function $\mu\colon X \to [0,1]$ with $\sum_{x \in X} \mu(x) = 1$. The set of all probability distributions over X is denoted by $\mathcal{D}istr(X)$. The *support* of a probability distribution is the set of all elements with a probability strictly greater than 0: $\mathrm{Supp}(\mu) = \{x \in X \mid \mu(x) > 0\}$.

Besides the function notation we introduce three other notations for probability distributions. For a probability distribution $\mu \in \mathcal{D}istr(X)$ with $|X| = n$ and $\mu(x_i) = p_i$ we also write μ as

$$\mu = \{x_1 \mapsto p_1, \ldots, x_n \mapsto p_n\}$$

in which we typically only consider the elements x_i that are in the support of μ.

We may also interpret μ as a vector of the probability values p_i. Vectors and matrices are written in bold, and their elements indiced: $\mathbf{p} = (p_1, \ldots, p_n)$.

Finally, μ can be written as a formal convex sum [Jacobs, Westerbaan, and Westerbaan, 2015]. This notation is particularly useful when dealing with a probability distribution over probability distributions, as we will encounter in Chapter 6. The convex sum, or 'ket-notation', for μ is given by

$$\mu = p_1 \cdot |x_1\rangle + \cdots + p_n \cdot |x_n\rangle = \sum_{i=1}^{n} p_i \cdot |x_i\rangle.$$

A *random variable* $\mathcal{X}$ denotes the outcome of an experiment, where the outcome is one out of a finite number of options given by the *sample space* X. The random variable $\mathcal{X}$ has a probability distribution $\mu \in \mathcal{D}istr(X)$. For more on random variables and probability theory in general, we refer to [Grinstead and Snell, 2012].

A *Dirac distribution* over a finite set X is a probability distribution where for exactly one $x \in X$ we have $\Pr(x) = 1$ and for all other

$x' \in X, x' \neq x$ we have $\Pr(x') = 0$. We denote the Dirac distribution that has probability 1 at $x \in X$ by $Dirac(x)$.

The expected value of a discrete random variable $\mathcal{X}$ denoted as $\mathbb{E}[X]$ and defined as

$$\mathbb{E}[\mathcal{X}] = \sum_{x \in X} x \cdot \mu(x).$$

The expected value satisfies the following properties:

$$\begin{aligned}
\mathbb{E}[\mathcal{X} + \mathcal{Y}] &= \mathbb{E}[\mathcal{X}] + \mathbb{E}[\mathcal{Y}], \\
\mathbb{E}[c \cdot \mathcal{X}] &= c \cdot \mathbb{E}[\mathcal{X}], \\
\mathbb{E}[g(\mathcal{X})] &= \sum_{x \in X} g(x) \cdot \mu(x),
\end{aligned}$$

where $\mathcal{X}$ and $\mathcal{Y}$ are discrete random variables, $c \in \mathbb{R}$ is a constant, and g is any function $g\colon X \to \mathbb{R}$.

Given two event x and y, the probability of x occurring given that y occurs is denoted by $\Pr(x \mid y)$. Bayes' rule is given by

$$\Pr(x \mid y) = \frac{\Pr(y \mid x) \Pr(x)}{\Pr(y)}.$$

3.3 Computability and complexity theory

A *decision problem* is a set of related questions that each can be answered by *yes* or *no*. A decision problem is *decidable* if there exists an algorithm that solves every instance of that decision problem. If such an algorithm does not exist, the problem is *undecidable*. The algorithm should be *complete* (being able to correctly answer every instance of the problem), *mechanistic* (consist of a finite number of elementary instructions), and *deterministic* (always give the same result on identical input). See e.g. [Sudkamp, 1997] for details.

A function $t\colon \mathbb{N} \to \mathbb{N}$ is called *time constructible* if there is a Turing machine which halts in exactly $t(n)$ steps for every input of length n. A function $s\colon \mathbb{N} \to \mathbb{N}$ is called *space constructible* if there is a Turing machine which halts in a configuration with exactly $s(n)$ non-blank tape cells and that used no other tape cells during computation, for every input of length n.

Using the functions defined above, we define the following base classes: TIME(t) is the class of all sets computed by a Turing machine within running time $t(n)$ for all inputs of length n. We define NTIME(t) analogously, but with a *non-deterministic* Turing machine. SPACE(s) is the

class of all sets computed by a Turing machine using at most $s(n)$ tape cells for any input of length n. Similarly, NSPACE(s) is defined the same as SPACE(s), but again with a non-deterministic Turing machine. Note that by Savitch's theorem, we have that SPACE = NSPACE [Arora and Barak, 2009].

In this thesis, the following complexity classes are used:

- $\mathrm{P} = \bigcup_i \mathrm{TIME}(n^i)$, the class of all *polynomial time computable sets*,
- $\mathrm{NP} = \bigcup_i \mathrm{NTIME}(n^i)$, the class of all *non-deterministic polynomial time computable sets*,
- $\mathrm{PSPACE} = \bigcup_i \mathrm{PSACE}(n^i)$, the class of all *polynomial space computable sets*,
- $\mathrm{EXPTIME} = \bigcup_i \mathrm{TIME}\left(2^{n^i}\right)$, the class of all *exponential time computable sets*,
- ETR, the class of problems with a polynomial many-one reduction to deciding membership in the *existential theory of the reals*, as defined in [Schaefer and Štefankovič, 2017].

We have the following standard inclusions between these classes:

$$\mathrm{P} \subseteq \mathrm{NP} \subseteq \mathrm{PSPACE} \subseteq \mathrm{EXPTIME},$$

as well as $\mathrm{NP} \subseteq \mathrm{ETR} \subseteq \mathrm{PSPACE}$ and $\mathrm{P} \subset \mathrm{EXPTIME}$ (by the time hierarchy theorem). It is conjectured that $\mathrm{P} \neq \mathrm{NP}$ and $\mathrm{P} \neq \mathrm{PSPACE}$.

Given two problems X and Y, we say that X is *polynomial many-to-one reducible (p-m-reducible)*, denoted as $X \leq_m^p Y$, if there exists a polynomial time computable function f such that $x \in X \iff f(x) \in Y$.

Given a problem X and a complexity class $\mathcal{C}$, we say that X is *hard* for $\mathcal{C}$ (or 'X is $\mathcal{C}$-hard') if for all $C \in \mathcal{C}$ we have $C \leq_m^p X$. If, additionally, we also have that $X \in \mathcal{C}$, we say that X is $\mathcal{C}$-complete. Note that if we have some problem Y which is known to be $\mathcal{C}$-hard or $\mathcal{C}$-complete, it suffices to show $Y \leq_m^p X$ to conclude that X is $\mathcal{C}$-hard.

For an extensive overview of complexity theory we refer to the following standard works on the subject: [Papadimitriou, 2003] and [Arora and Barak, 2009].

Chapter 4

Formal definitions & literature review

In this chapter we will formally introduce the models we work with, and provide an overview of the relevant literature that is available. We start with *Markov decision processes*, the standard model for decision making in a probabilistic environment, and will then introduce a number of extensions of this model, each introducing some form of *uncertainty* to the model. Most notably we will look at *multiple-environment Markov decision processes* and *partially observable Markov decision processes*.

4.1 Markov decision processes

The standard model for decision making in a probabilistic environment, i.e. one where the outcome of a decision is affected probabilistically, are Markov decision processes [Puterman, 1994]. The general idea is that the outside world is observed and translated into a state. This is done with certainty, i.e., the world is *fully observable*. At a state the *agent* makes a decision, usually referred to as an *action*, which leads to a (probabilistic) change in the world, and thus a new observation. This idea is illustrated in Figure 4, which is taken from [Kaelbling, Littman, and Cassandra, 1998], and formalized in Definition 1.

> **Definition 1: Markov decision process**
>
> A *Markov decision process* (MDP) is a tuple (S, A, δ), where S is a (finite or countably infinite) set of states, A is a (finite) set of

actions, and δ is the (partial) transition function defined as $\delta\colon S \times A \to \mathit{Distr}(S)$.

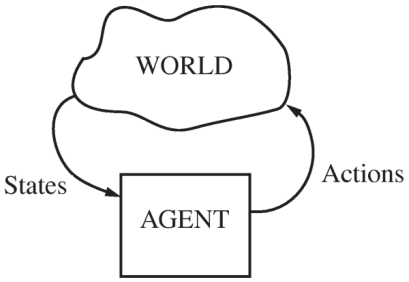


Figure 4: Schematic workings of an MDP.

In an MDP, each state-action pair $(s, a) \in S \times A$ is assigned a probability distribution over the successor states. For a state $s' \in S$, we denote the probability of reaching this state from (s, a) by $\delta(s, a, s')$. If an MDP (S, A, δ) has $|A| = 1$, it is a *discrete time Markov chain* (DTMC).

For an MDP M (and subsequent models) we introduce the following shorthand notations. For a state $s \in S$ and an action $a \in A$ we denote the set of *successor states* of this state-action pair by $\mathrm{Succ}(s, a) = \{s' \in S \mid \delta(s, a, s') > 0\}$. The set of *predecessor states* of a state $s' \in S$ is defined as $\mathrm{Pred}(s') = \{s \in S \mid \exists a \in A.\delta(s, a, s') > 0\}$ For a state s we define the set of *enabled actions* as $A(s) = \{a \in A \mid \exists s' \in S.\, \delta(s, a, s') > 0\}$.

The size of an MDP M, denoted $\mathrm{Size}(M)$, is determined by counting the number of transitions with positive probability, i.e., the total number of pairs $(s, a, s') \in S \times A \times S$ with $\delta(s, a, s') > 0$. Clearly, the size is bounded by $|S|^2 \cdot |A|$.

Definition 2: End-components in MDPs

An *end-component* in an MDP (S, A, δ) is a pair (S', A') with $S' \subseteq S$ and $A' \subseteq A$ such that for every state and action in these subsets all successor states are again in S':

$$s \in S', a \in A' \cap A(s), \forall s' \in S.\delta(s, a, s') > 0 \implies s' \in S',$$

and any state in the end-component can be reached from any other state in the end-component: $\forall s_0, s_{m+1} \in S'$ there exists a finite path $s_0 a_0 \ldots a_m s_{m+1} \in (S' \times A')^* \times S'$.

> An end-component is *maximal* if there is not other end-component (S'', A'') such that $S' \subseteq S''$ and $A' \subset A''$. The set of all maximal end-components in M is denoted by $\mathrm{MEC}(M)$.

The set of all maximal end-components can be computed in polynomial time [Courcoubetis and Yannakakis, 1995]. Any strategy of an MDP will end up in an end-component with probability 1 [De Alfaro, 1997, Theorem 3.2]. Since all end-components are included in one of the maximal end-components, we also have that any strategy will end up in a maximal end-component with probability 1.

It should be noted that it may be possible to leave an end-component (S', A'). If there is a state $s \in S'$ with an action $a \in A(s)$ that is not part of the end-component, i.e. $a \notin A'$. This means that an end-component acts as a strongly connected component that cannot be left under *some* strategy. However, there may exist end-components that act as strongly connected components that cannot be left under *all* strategies. These are, for example, sink states with a self-loop of probability 1.

An MDP is can be visually represented as a *state-based system* as in Figure 5. We may omit the actions if a transition is the same for all actions, see for example the transitions from s_1 to s_3, s_2 to s_3 or the self-loop at s_3. In this MDP, $(\{s_3\}, \{a_1\})$ and $(\{s_3\}, \{a_2\})$ form end-components, and $(\{s_3\}, \{a_1, a_2\})$ forms a maximal end-component.

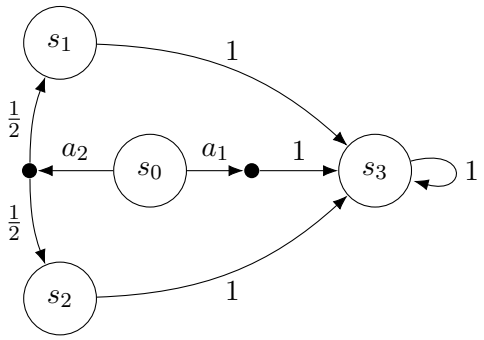


Figure 5: Graphical representation of an MDP.

4.2 Problems and solutions

For the following definitions, let $M = (S, A, \delta)$ be an MDP. A *run* of M is a sequence of states and actions $(s_1 a_1 s_2 a_2 \dots a_{n-1} s_n \dots)$ where for all $i \geq 1$ we have $\delta(s_i, a_i, s_{i+1}) > 0$.

Definition 3: Strategy

A (randomized) *strategy* (also called a scheduler or policy) for M is a function $\sigma\colon (SA)^*S \to Distr(A)$ that maps a finite run $h = (s_1a_1, \ldots, s_{n-1}a_{n-1}s_n)$ ending in some state s_n to a distribution over actions.

If a strategy σ only considers the current state, i.e. the run h consists of a single state (s), it is called *memoryless*. If the distribution over actions $Distr(A)$ is a *Dirac distribution* for all finite runs, σ is called *deterministic*. We write $\sigma(h, a)$ for the probability of performing action a in strategy σ after run h.

Applying a strategy σ to an MDP M resolves the non-determinism. It gives us an *induced Markov chain*, denoted by M^σ.

Definition 4: Induced Markov chain

Given an MDP $M = (S, A, \delta)$ and a *memoryless* strategy σ, the *induced Markov chain*, denoted by M^σ, is a tuple (S, δ') where S is the same set of states as in M, and $\delta'\colon S \to Distr(S)$ is the probabilistic transition function constructed by

$$\forall s, s' \in S. \quad \delta'(s, s') = \sum_{a \in A} \sigma(s, a) \cdot \delta(s, a, s').$$

In the case of finite memory strategies, the memory is encoded in the state-space of the DTMC, see [Baier and Katoen, 2008].

In an induced Markov chain, the non-determinism from the MDP is resolved by the strategy, and all transitions are purely probabilistic. This means we can compute the probability of reaching a given state. In an induced Markov chain M^σ we can compute the probability of reaching some state s from some initial state s_I by solving a linear equation system, see e.g. [Baier and Katoen, 2008] for details.

Given an MDP, we can define several objectives for which we want to compute a strategy that solves them.

Definition 5: Objectives

An *objective* in an MDP is a set of infinite sequences of states and actions, i.e. $\Phi \subseteq (S \times A)^\omega$.

A *reachability objective* for some target set T, denoted $\Diamond T$, is the set of all infinite runs that contain a state from T. Formally: $\Diamond T = \{s_0 a_0 s_1 a_1 \cdots \mid \exists i \geq 0. s_i \in T\}$. Intuitively, the objective is to visit at least one state in T at some point.

A *safety objective* for some target set T, denoted $\Box T$, is the set of all infinite runs that only contain states from T. Formally: $\Box T = \{s_0 a_0 s_1 a_1 \cdots \mid \forall i \geq 0. s_i \in T\}$. In other words, the objective is to always stay within T, where T is typically a set of states that are deemed safe.

A *parity objective* for a *priority function* $p\colon S \to \{0, 1, \ldots, d\} \subseteq \mathbb{N}$, denoted $\text{Parity}(p)$, requires the smallest priority to appear infinitely often to be even. Formally: $\text{Parity}(p) = \{s_0 a_0 s_1 a_1 \cdots \mid \min\{p(s_i) \mid s_i \in \mathit{inf}(s_0 a_0 s_1 a_1 \ldots)\} \text{ is even } \}$, where $\mathit{inf}(s_0 a_0 s_1 a_1 \ldots)$ is the set of states s_i that appear infinitely often in the given infinite run.

Formally, an objective Φ has to be Borel measurable, which means Φ has to be a set in the Cantor topology on $(S \times A)^\omega$, see [Kechris, 2012]. Besides the objectives defined in Definition 5 there are two other important classes of objectives: Büchi and coBüchi objectives. Büchi objectives require that at least one state in the target set T is visited infinitely often. CoBüchi objectives require that only states in T are visited infinitely often. Reachability and safety objectives are examples of Büchi objectives, and both Büchi and coBüchi objectives are in turn special cases of parity objectives [Chatterjee, Doyen, and Henzinger, 2010].

Given an objective Φ the problem is to compute a strategy σ such that the objective is satisfied against some probability threshold $\bowtie \lambda$ when starting in some initial state $s_I \in S$:

$$M^\sigma_{s_I} \models \mathbb{P}_{\bowtie\lambda}[\Phi] \iff \Pr(M^\sigma, \Phi, s_I) \bowtie \lambda,$$

where $\Pr(M^\sigma, \Phi, s_I)$ is the probability with which Φ is satisfied in the induced Markov chain M^σ with initial state s_I.

In the special case of $\mathbb{P}_{=1}[\Phi]$ we say that the objective Φ is to be satisfied *almost-surely*. In case of $\mathbb{P}_{\geq 1-\epsilon}$, for some $\epsilon > 0$, the problem is to be satisfied *limit-surely*. Any other case is referred to as the *quantitative* version of the problem.

In MDPs, computing a strategy that satisfies (any) parity objective can be done in polynomial time [Chatterjee, Jurdziński, and Henzinger, 2004]. As a consequence, satisfying reachability or safety objectives can also be done in polynomial time.

Besides the objectives mentioned here, we can also define other objectives using the logic PCTL [Hansson and Jonsson, 1994]. PCTL model checking for MDPs is in P, and thus for any objective defined in this logic, we can compute a strategy that satisfies it in polynomial time [Baier and Katoen, 2008].

4.2.1 Reward problems

We can extend an MDP by defining a *reward function*. Let $R\colon S \times A \to \mathbb{R}$ be the reward function giving the reward obtained, as a value in $\mathbb{R}$, to every action choice at each state. The intuitive idea behind rewards is that it gives us a goal (typically to maximize). Alternative definitions of reward function only consider the current state, i.e. $R\colon S \to \mathbb{R}$, or also include the successor state of some state-action pair, i.e. $R\colon S \times A \times S \to \mathbb{R}$.

We consider a number of different reward problems.

Definition 6: Expected reward problems

Let $\gamma \in (0, 1]$ be a *discount factor*. Intuitively, the discount factor is going to ensure that earlier steps have a bigger impact on the reward that later steps. Given an MDP $M = (S, A, \delta)$ with some reward function $R\colon S \times A \to \mathbb{R}$, and a *discount* $\gamma \in (0, 1]$, we define the following expected reward problems:

- The *finite horizon reward maximization problem* is to maximize
$$\mathbb{E}\left[\sum_{t=0}^{N} \gamma^t \cdot r_t\right]$$
for some given constant $N \in \mathbb{N}$.
- The *infinite horizon reward maximization problem* is to maximize
$$\mathbb{E}\left[\sum_{t=0}^{\infty} \gamma^t \cdot r_t\right]$$
- The *indefinite horizon reward maximization problem* is to maximize
$$\mathbb{E}\left[\sum_{t=0}^{N} \gamma^t \cdot r_t\right]$$

where N is *not given* (and as a consequence, finding N is part of the problem).

- The *quantitative reward problem* is to ensure that, given some threshold $\kappa \in \mathbb{R}$

$$\mathbb{E}\left[\sum_{t=0}^{\infty} \gamma^t \cdot r_t\right] \bowtie \kappa$$

In all of the above, r_t is the reward received on step t from the initial state. If the discount factor $\gamma = 1$ we speak of the *undiscounted* variants of these problems, where if $\gamma < 1$, we are dealing with the *discounted* variants.

The actual value of r_t depends on the strategy we use, and hence, the expected reward problems are to compute a strategy such that the expected reward is maximized accordingly. The value function $V_{\sigma,t} \colon S \to \mathbb{R}$ gives us the expected sum of the reward under some strategy σ after t steps. This value function can be computed recursively via the Bellman equation [Bellman, 2003] for every state $s \in S$ as

$$V_{\sigma,t}(s) = \sum_{a \in A} \sigma(s,a) \cdot R(s,a) + \sigma(s,a) \cdot \gamma \cdot \sum_{s' \in S} \delta(s,a,s') \cdot V_{\sigma,t-1}(s'),$$

with base case

$$V_{\sigma,1}(s) = \sum_{a \in A} \sigma(s,a) \cdot R(s,a).$$

In the infinite horizon case the time step counter t is simply removed from the equation, making the equation recurse infinitely. The goal of the maximization problems is to find a strategy (randomized) σ such that $V_{\sigma,t}$ is maximal. In the quantitative case, any strategy σ such that $V_{\sigma,t}(s_I) \bowtie \kappa$ will do.

A reward is typically interpreted as something for which a higher value is better. If instead we are interested in assigning some form of penalty to taking certain actions, we could define a *cost* function and an associated *expected cost* problem. Note that this is just about interpretation: the definitions we use for the reward function and the associated problem allow for both interpretations, even though we use the word reward in their definitions.

4.3 Partial observability

In an MDP we assume the world is fully observable, meaning every observation leads to a unique state. This assumption is not always valid. A better model would be to *derive* the state from the observation using a *state estimator* (SE). This idea is illustrated in Figure 6, again taken from [Kaelbling, Littman, and Cassandra, 1998], and formalized as a *partially observable* MDP in Definition 7.

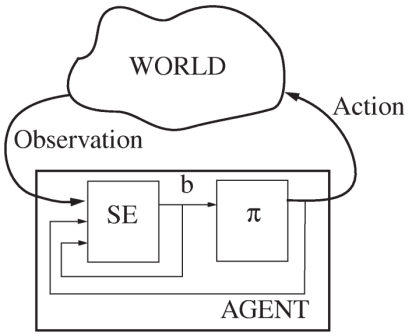


Figure 6: Schematic workings of a POMDP.

In Figure 6 the agent receives an observation that is translated into a *belief state* (b) by the *state estimator* (SE). A belief state is a probability distribution over the actual states. Hence, it summarizes where the agent 'believes' they are. Using the belief state, a policy π is developed to determine the next action to be taken.

> **Definition 7: Partially observable MDP**
>
> A *partially observable Markov decision process* (POMDP) is a tuple (S, A, δ, Z, O), where (S, A, δ) form a standard MDP, Z is a set of observations, and $O\colon S \times A \to Distr(Z)$ is a function mapping state-action pairs to a probability distribution over observations.

If the distribution the observation function maps to is a Dirac distribution for all state-action pairs, the observation function is *deterministic*. Furthermore, the observation function may be independent from the action, effectively defining it as $O\colon S \to Z$. All POMDPs occurring in this thesis are of this form, i.e., with deterministic and state-based observations. This forms no restriction: any POMDP with an observation function of the form $O\colon S \times A \to Distr(Z)$ can be transformed into a polynomially larger POMDP with state-based deterministic observations [Chatterjee et al., 2016].

As a POMDP extends an MDP, the definition of a reward function, and the short-hand notations $\mathrm{Succ}(s, a)$ and $A(s)$ remain the same for a POMDP.

The idea behind a POMDP is that, contrary to an MDP, we do not exactly know in which state we are, but instead are only aware of the *observation* the state emits. As such, computing a strategy for a POMDP and some problem Φ is now *observation based*.

A run $h = (s_1 a_1 \dots s_n a_n \dots)$ can be mapped to a sequence of observations by applying the observation function on each subsequent state-action pair in the run: $O(h) = (O(s_1, a_1), \dots, O(s_n, a_n), \dots)$.

To make a finite memory strategy σ observation based, we additionally require that for all histories of size k, if two histories h_1, h_2 have the same k observations in their observation sequence, they map to the same distribution over actions: $O(h_1) = O(h_2) \implies \sigma(h_1) = \sigma(h_2)$.

The problems we defined before remain the same in the context of POMDPs, though reward problems (see Definition 6) take a central place in the literature, especially in AI applications.

Just like MDPs, POMDPs can also be drawn as a state based system. The only difference being the observations, which are typically represented by giving states with different observations different colors. In Figure 7 we see the POMDP we get when taking the MDP from Figure 5 and adding observations $O(s_0) = \mathit{white}$, $O(s_1) = O(s_2) = \mathit{red}$, and $O(s_3) = \mathit{blue}$.

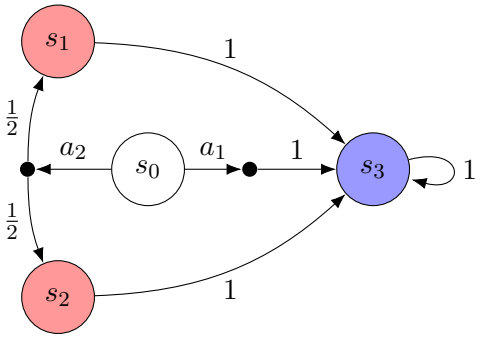


Figure 7: Graphical representation of a POMDP.

4.3.1 Belief states

Since a POMDP is *partially observable*, we have incomplete knowledge of the state we are currently in. Instead, in POMDPs we have the concept of *belief*.

> **Definition 8: Belief state**
>
> Given a POMDP $P = (S, A, \delta, Z, O)$, a *belief state* b is a probability distribution over the set of states S. We denote the probability of state s assigned by belief b by $b(s)$.

By the standard definitions of probabilities we have $0 \leq b(s) \leq 1$ and $\sum_{s \in S} b(s) = 1$. Typically, the belief state is something that evolves over time, as such we will denote the belief state at some time t by b_t. Given b_t and some action a that is to be performed, we may ask what the belief state looks like after performing this action. Hence, we want to update b_t to a new belief state b_{t+1}.

> **Definition 9: Belief update in POMDPs**
>
> Given the current belief state b_t, an action $a \in A$ and an observation $z \in Z$, we compute the *successor belief state* b_{t+1} by performing the following computation for every possible successor state s':
>
> $$\begin{aligned} b_{t+1}(s') &= \Pr(s' \mid a, z, b_t) \\ &= \frac{\Pr(z \mid s', a, b_t) \cdot \Pr(s' \mid a, b_t)}{\Pr(z \mid a, b_t)} \\ &= \frac{\Pr(z \mid s', a) \cdot \sum_{s \in S} \Pr(s' \mid a, b_t, s) \cdot \Pr(s \mid a, b_t)}{\Pr(z \mid a, b_t)} \\ &= \frac{O(s', a, z) \cdot \sum_{s \in S} \delta(s, a, s') \cdot b_t(s)}{\Pr(z \mid a, b_t)} \end{aligned}$$
>
> Where $\Pr(z \mid a, b_t)$ is a *normalization constant* [Cassandra, Kaelbling, and Littman, 1994], ensuring b_{t+1} is a valid probability distribution, defined by
>
> $$\Pr(z \mid a, b_t) = \sum_{s' \in S} O(s', a, z) \cdot \sum_{s \in S} \delta(s, a, s') \cdot b_t(s).$$

Using the belief update to compute a new belief state is quadratic in the number of states. The nominator has a complexity of $\mathcal{O}(|S|)$ due to the sum over all states, but for the total belief update this has to be

computed for every potential successor state $s' \in S$. The denominator is quadratic in the number of states, but is the same for every successor state, so only needs to be computed once, yielding a total complexity of $\mathcal{O}(|S|^2)$.

If we have defined some initial state s_I for a POMDP our initial belief b_0 is given by a Dirac distribution at s_I. In our definitions of the various kinds of expected reward maximization problems we assumed some initial state to count the number of steps from. This can also be generalized to some initial belief, hence a set of states, to start counting.

Using belief states we can map a POMDP to a fully observable continuous space *belief MDP*.

Definition 10: Belief MDP

For some POMDP $P = (S, A, \delta, Z, O)$, the associated *belief MDP* $(\mathcal{B}, A, \tau)$ is constructed as follows: the set of states $\mathcal{B}$ if formed by all belief states, the set of actions is the same as in the POMDP: A, and the transition function $\tau\colon \mathcal{B} \times A \to \mathit{Distr}(\mathcal{B})$ for belief states $b, b' \in \mathcal{B}$ and action $a \in A$ is defined by

$$\tau(b, a, b') = \Pr(b' \mid a, b) = \sum_{z \in Z} \Pr(b' \mid a, b, z) \cdot \Pr(z \mid a, b)$$

where $\Pr(z \mid a, b)$ is the normalization constant from the belief update (Definition 9), and

$$\Pr(b' \mid a, b, z) = \begin{cases} 1 & \text{if } \mathrm{SE}(b, a, z) = b' \\ 0 & \text{otherwise.} \end{cases}$$

with $SE(b, a, z)$ being the state estimator, determining the successor belief state of the current belief b, action a, and successor observation z.

If the POMDP P has some reward function R associated with it, the belief MDP has reward function $\rho\colon \mathcal{B} \times A \to \mathbb{R}$ defined by $\rho(b, a) = \sum_{s \in S} b(s) \cdot R(s, a)$.

All these definitions are standard and discussed extensively in e.g. [Kaelbling, Littman, and Cassandra, 1998].

4.3.2 Belief MDP example

Consider the POMDP in Figure 7. We construct the belief MDP, starting with an initial belief b_0 observing *white*, hence we have $b_0 = \{s_0 \mapsto 1\}$.

At s_0 we can either use action a_1 or a_2. Let b_1 be the belief state for taking action a_1 and observing *blue*. As the only blue state is s_3, we immediately know that $b_1 = \{s_3 \mapsto 1\}$.

When taking action a_2 form s_0 we will only observe *red*, but we do not know whether this is s_1 or s_2. From the transition probabilities it is clear the belief state will be $b_2 = \{s_1 \mapsto \frac{1}{2}, s_2 \mapsto \frac{1}{2}\}$, but let us use the formal belief update to check this.

We compute b_2, given the current belief b_0 and observing a red state. First, we note that the probability for any *non-red state* will be 0 in this belief state, since $O(s', a_2, z) = 0$ for any observation z that is not red. For any state with a red observation we have $O(s', a_2, red) = 1$, so based on the observation alone we know that the probabilities for s_0 and s_3 are 0 in b_2. The values for s_1 and s_2 are *potentially* not 0, as the observation matches, but we also need to account for the transition probabilities.

We compute the probability of being in s_1 for belief state b_2 via the belief update, given current belief b_0, action a_2, and observation *red*:

$$\begin{aligned} b_2(s_1) &= \Pr(s_1 \mid a_2, red, b_0) \\ &= \frac{O(s_1, a_2, red) \cdot \sum_{s \in S} \delta(s, a, s_1) \cdot b_0(s_0)}{\Pr(red \mid a_2, b_0)} \\ &= \frac{\delta(s_0, a_2, s_1) \cdot b_0(s_0)}{\Pr(red \mid a_2, b_0)} \\ &= \frac{\frac{1}{2}}{\Pr(red \mid a_2, b_0)}. \end{aligned}$$

And do the same for the probability of being in state s_2 in belief state b_2:

$$\begin{aligned} b_2(s_2) &= \Pr(s_2 \mid a_2, red, b_0) \\ &= \frac{O(s_2, a_2, red) \cdot \sum_{s \in S} \delta(s, a, s_2) \cdot b_0(s_0)}{\Pr(red \mid a_2, b_0)} \\ &= \frac{\delta(s_0, a_2, s_2) \cdot b_0(s_0)}{\Pr(red \mid a_2, b_0)} \\ &= \frac{\frac{1}{2}}{\Pr(red \mid a_2, b_0)}. \end{aligned}$$

The normalization constant is equal to 1, as we already have a valid probability distribution ($\frac{1}{2} + \frac{1}{2} = 1$), and red is the only possible observation from using action a_2 at b_0. But again, let us check this by computing the value of $\Pr(red \mid a_2, b_0)$

$$\begin{aligned}\Pr(red \mid a_2, b_0) &= \sum_{s' \in S} O(s', a_2, red) \cdot \sum_{s \in S} \delta(s, a, s') \cdot b_0(s) \\ &= O(s_1, a_2, red) \cdot \delta(s_0, a_2, s_1) + O(s_2, a_2, red) \cdot \delta(s_0, a_2, s_2) \\ &= \frac{1}{2} + \frac{1}{2} = 1.\end{aligned}$$

So we have $b_2 = \{s_1 \mapsto \frac{1}{2}, s_2 \mapsto \frac{1}{2}\}$.

The next belief state will observe *blue* from the current belief b_2. Since there is only one state with observation *blue* and the probability of reaching this state is 1 for both actions and both *red* states in b_2, we know that this belief state will be a Dirac distribution for s_3. Note that we already defined this belief state as b_1, so we can use that belief state again.

The only transition left is the self-loop out of s_3. But again, as we note that the transition probability is 1 for both actions, and there is only one successor state, namely s_3 itself, the successor belief state will again be a Dirac distribution in s_3, hence the belief b_1 does not change.

So in summary, we have the following belief states:

$$b_0 = \{s_0 \mapsto 1\}, \quad b_1 = \{s_3 \mapsto 1\}, \quad b_2 = \left\{s_1 \mapsto \frac{1}{2}, s_2 \mapsto \frac{1}{2}\right\}.$$

Next we construct the belief MDP. We already have the belief states: $\mathcal{B} = \{b_0, b_1, b_2\}$. The set of actions is the same as in the POMDP: $A = \{a_1, a_2\}$. The transition function is given by

$$\tau(b, a, b') = \sum_{z \in Z} \Pr(b' \mid a, b, z) \cdot \Pr(z \mid a, b)$$

where $\Pr(z \mid a, b)$ is the normalization constant we have seen before, and for which we already know to be 1 for all given actions a and belief states b in this example. The probability $\Pr(b' \mid a, b, z)$ is given by the state estimator as defined in Definition 10.

We have $\mathrm{SE}(b_0, a_1, blue) = b_1$ as we observe *blue* when taking action a_1 from s_0. When taking action a_2 at b_0 and observing *red* we get $\mathrm{SE}(b_0, a_2, red) = b_2$.

Similarly, we have $\mathrm{SE}(b_1, a_1, blue) = \mathrm{SE}(b_1, a_2, blue) = b_1$ when observing *blue* after any action at b_1, and $\mathrm{SE}(b_2, a_1, blue) = \mathrm{SE}(b_2, a_2, blue) = b_1$ when observing *blue* after any action at b_2. So we get the transition function τ defined by

$$\begin{aligned} \tau(b_0, a_1, b_1) &= 1, & \tau(b_0, a_2, b_2) &= 1, \\ \tau(b_1, a_1, b_1) &= 1, & \tau(b_1, a_2, b_1) &= 1, \\ \tau(b_2, a_1, b_1) &= 1, & \tau(b_2, a_2, b_1) &= 1. \end{aligned}$$

We draw the belief MDP $(\mathcal{B}, A, \tau)$ in Figure 8. Note that the belief MDP has fewer states than the original POMDP, and that all transition probabilities are 1. This is only for this specific example and certainly not true in general. In Section 6.1.1 we construct a belief MDP that has infinitely many reachable belief states. Finally, it should be noted that a belief MDP is *fully observable* and has no observations associated with it. Recall that every belief state is a distribution over states that share an observation. As such it makes sense to give the belief states the color of the observation they stem from, even though this has no further formal meaning.

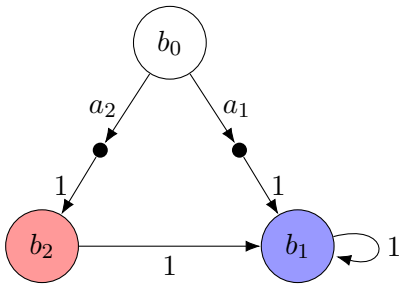


Figure 8: The belief MDP for the POMDP from Figure 7.

4.3.3 Complexity results for POMDPs

Where all problems we consider for MDPs can be solved in polynomial time, POMDPs turn out to be much more difficult. Furthermore, where for MDPs memoryless strategies suffice, the amount of memory that is available makes a difference for the computational complexity of various problems for POMDPs. We will give a brief overview of the important computability and complexity results that are available for POMDPs.

Results for parity objectives in POMDPs can be found in [Chatterjee, Chmelík, and Tracol, 2016]. Almost-sure parity objectives with infinite memory are undecidable, and EXPTIME-complete with finite memory.

Quantitative parity objectives are undecidable both for infinite and finite memory.

Furthermore, [Chatterjee, Chmelík, and Tracol, 2016] gives results for Büchi and coBüchi objectives in POMDPs. Reachability and safety objectives can be seen as special cases of these. Büchi and coBüchi objectives are in turn special cases of parity objectives.

Computing optimal strategies for the infinite-horizon POMDP problem is undecidable [Madani, Hanks, and Condon, 2000]. For the finite-horizon problem it is PSPACE-complete [Papadimitriou and Tsitsiklis, 1987], and still NP-hard when restricting to memoryless strategies [Vlassis, Littman, and Barber, 2012].

4.4 Multiple environments

Multiple-environment Markov decision processes (MEMDPs) are an extension of MDPs that add *discrete uncertainty*. Instead of a single transition function, we now have multiple transition functions. A MEMDP can thus be seen as a set of MDPs that share the same states and actions.

> **Definition 11: Multiple-environment MDP**
>
> A *multiple-environment Markov decision process* (MEMDP) is a tuple $(S, A, \{\delta_i\}_{i\in[n]})$, where S and A are a set of states and a set of action respectively, and $\{\delta_i\}_{i\in[n]}$ are n (different) transition functions $\delta_i \colon S \times A \to Distr(S)$.

When we fix a single transition function δ_i, we get a standard MDP. We call such an MDP an *environment* of the MEMDP. Whenever $M = (S, A, \{\delta_i\}_{i\in[n]})$ is a MEMDP, we write $M_i = (S, A, \delta_i)$ for the i-th environment in M, and may also write $M_i \in M$. A *sub-MEMDP* of M is a MEMDP $M' = (S, A, \{\delta_i\}_{i\in I})$ where $I \subseteq [n]$ is a subset of all the indexes. We may also write $M' \subseteq M$.

We assume that the set of enabled actions $A(s)$ for every state $s \in S$ is the same in every environment. Hence, environments cannot be revealed by being unable to perform a certain action.

Applying a strategy σ to M is done for each environment independently and yields n induced Markov chains: $M^\sigma = \{M_i^\sigma\}_{i\in[n]}$. Finally, we do not care about the order in which the transition functions are given. $M = (S, A, \delta_1, \delta_2)$ defines the same MEMDP as $M' = (S, A, \delta_2, \delta_1)$.

MEMDPs are by their definition *fully observable*. Furthermore, we do not require that the transition functions define the same graph in each environment. There may be transitions available in some of the environments and not in others. This gives rise the following definition.

> **Definition 12: Revealing transitions**
>
> A transition (s, a, s') is *i-revealing* for MEMDP $M = (S, A, \{\delta_i\}_{i\in[n]})$ if we have for some environment i that $\delta_i(s, a, s') > 0$ and for all other environments $j \neq i$ that $\delta_j(s, a, s') = 0$.

If we are in some state s, apply action a and observe that we end up in state s' we immediately know that environment $M_i \in M$ is the *true* environment. The idea behind the existence of a single true environment is that MEMDPs can be used to model systems with *discrete uncertainty*. That is, we have some system and we do not know exactly how it operates, but whatever the exact behavior is, it is one out of several predefined scenarios. Thus, the true environment is the scenario that exactly models our real-world system.

When dealing with MEMDPs of more than two environments, it may be the case that a transition does not reveal the true environment because it exists in more than one of the environments. However, it may exclude some of the environments.

> **Definition 13: Reducing transitions**
>
> For a MEMDP $M = (S, A, \{\delta_i\}_{i\in[n]})$ and a subset of indices $I \subseteq [n]$, a transition (s, a, s') *reduces* M to the sub-MEMDP $M' = (S, A, \{\delta_i\}_{i\in I})$ if for all $i \in I$ we have $\delta_i(s, a, s') > 0$ and for all $j \notin I$ we have $\delta_j(s, a, s') = 0$.

A reducing transition may not reveal the true environment, but it can shrink the MEMDP to a smaller set of potential true environments. Obviously, if the set I is a single index i, an I-reducing transition is also i-revealing.

End-components for MEMDPs were introduced in [Raskin and Sankur, 2014]. They only consider MEMDPs of two environments and call them *double end-components*. We extend the definition to the n-environment case, and change the name to reflect this.

Definition 14: End-components in MEMDPs

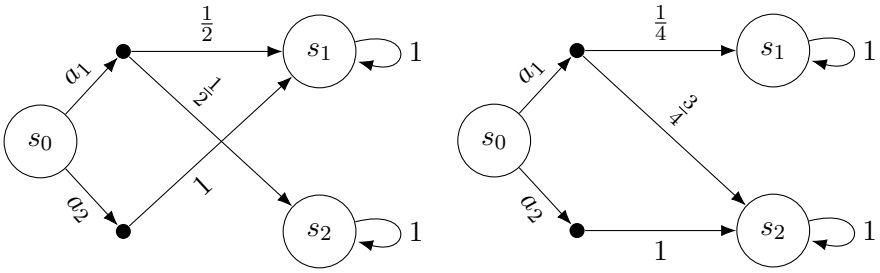
Given a MEMDP $M = (S, A, \{\delta_i\}_{i\in[n]})$, an *end-component* in M is a pair (S', A') that is an end-component in each environment $M_i \in M$.

An end-component (S', A') in M is *distinguishing* if there exists a state $s \in S'$ and an action $a \in A'$ such that for two environments $i \neq j$ we have $\delta_i(s, a) \neq \delta_j(s, a)$.

End-components for MEMDPs were introduced in [Raskin and Sankur, 2014] as *double end-components*, as they only consider MEMDPs of two environments. Our definition generalizes double end-components to n environments. We choose to just call them end-components, and make sure that the context makes clear whether we consider an end-component in an MDP or in a MEMDP.

Maximal end-components in MEMDPs can be computed by removing all actions at a state with different supports for the successor state distributions from M. That is, all (s, a) for which there exists an s' such that (s, a, s') is I-reducing for some $I \subset [n]$. Then, compute the multiple environment end-components in the remaining MEMDP. This procedure is a generalized version of the method presented in [Raskin and Sankur, 2014].

When drawing a MEMDP, we draw each environment separately, as in Figure 9. In this example, transition (s_0, a_2, s_1) is revealing for environment 1, and (s_0, a_2, s_2) is revealing for environment 2.



(a) Environment 1. (b) Environment 2.

Figure 9: An example MEMDP.

MEMDPs were introduced and extensively studied in [Raskin and Sankur, 2014]. They provide both complexity results and concrete algorithms for reachability, safety and parity objectives. Reward structures are not discussed. In summary, [Raskin and Sankur, 2014] provides the following

key insights into MEMDPs:

- Computing a strategy that satisfies an almost-sure reachability, safety, or parity objective is decidable in polynomial time.
- Computing a strategy that satisfies a limit-sure reachability, safety, or parity objective can be solved in polynomial time too.
- Computing a strategy that satisfies a quantitative reachability or safety objective is NP-hard on MEMDPs for both memoryless and finite-memory strategies, finite memory strategies can be computed in PSPACE.
- Computing a strategy that satisfies a quantitative parity objective can be reduced to satisfying a quantitative reachability objective in polynomial time, and thus is also NP-hard and solvable in PSPACE.

Besides the complexity results, concrete methods for solving each problem are provided.

4.5 Mixed observability

Besides MEMDPs and POMDPs there are many other ways of extending the modeling capabilities of MDPs. An other interesting type of model are *mixed observability* MDPs. The idea is to split the model into two parts, a *fully observable* part and a *partially observable* part. This is a special case of the more general idea of having *factored state-spaces*, as defined in *factored POMDPs* [Boutilier and Poole, 1996].

Definition 15: Mixed observation MDP

A *mixed observation Markov decision process* (MOMDP) is a tuple $(S_{fo} \times S_{po}, A, Z_{fo} \times Z_{po}, \delta_{fo}, \delta_{po}, O)$, where

- $S = S_{fo} \times S_{po}$ is the set of factored states, with S_{fo} being the fully observable states and S_{po} being the partially observable states, such that a pair (x, y) represents a state in the entire system.
- A is a set of actions.
- $Z = Z_{fo} \times Z_{po}$ is the set of observations with $Z_{fo} = S_{fo}$, such that the fully observable states all have a unique observation of their own, and Z_{po} is a set of observations for the partially

observable states.

- $\delta_{fo}\colon S_{fo} \times S_{po} \times A \to Distr(S_{fo})$ is the transition function on the fully observable states, and $\delta_{po}\colon S_{fo} \times S_{po} \times A \to Distr(S_{po})$ is the transition function the partially observable states. Together they map a state $(x, y) \in S$ to a distribution over successor states given by $(\delta_{fo}(x, a), \delta_{po}(y, a))$.
- $O\colon S_{fo} \times S_{po} \to Z_{fo} \times Z_{po}$ is the observation function that assigns an observation to each state. Note that each of the fully observable states gets a unique observation, so the interesting part is how it maps observations from Z_{po} to the partially observable states, and so we can simplify the observation function to $O\colon S_{po} \to Z_{po}$.

Where a POMDP can be seen as an MDP with continuous belief states [Kaelbling, Littman, and Cassandra, 1998], a MOMDP can be seen as an MDP with both discrete and continuous states, where the discrete states are given by the fully observable part of the MOMDP, and the continuous states by the belief over the partially observable part of the MOMDP [Ong et al., 2010].

The computational advantage of MOMDPs over POMDPs lies in that the fully observable part in a MOMDP tells us which partially observable states we should account for and which ones are certainly not possible in combination with the current fully observable state. As a consequence, instead of dealing with a single belief over the entire state space as in a POMDP, we now have to deal with a belief over the partially observable states associated with the current fully observable state. This usually leads to a smaller belief, which is easier to compute. Note that any POMDP can be modeled by a MOMDP by making the fully observable part is trivial. Thus complexity of problems for MOMDPs are at least as hard as for POMDPs.

4.5.1 Hidden model MDPs

By making additional assumptions about the behavior of a MOMDP, we get a computationally even easier model called a *hidden model MDP*.

Definition 16: Hidden model MDP

A *hidden model MDP* (hmMDP) is a MOMDP $(S_{fo} \times S_{po}, A, Z_{fo} \times Z_{po}, \delta_{fo}, \delta_{po}, O)$ with the following additional assumptions:

1. The set of fully observable states S_{fo} is a set of states. The set of partially observable states are n indices that number a set of predefined models. There is one real model $i_T \in S_{po}$.
2. The actions on the completely observable states have no effect on the partially observable states, and their transitions are also assumed to be independent of each other.
3. The real model i_T does not change over time. As a consequence, the transition function δ_{po} can be interpreted as an identity matrix.
4. The partially observable states S_{po} cannot be observed and the observation function O is only defined on the fully observable states.

As a consequence, a hidden model MDP is a simplified MOMDP that defines a finite set of MDPs that only differ in their transition functions [Chadès et al., 2012]. Hence, a hmMDP can be interpreted as a tuple $(S, A, \{\delta_i\}_{i\in[n]})$ of which the definition coincides with the definition of a MEMDP.

Note that MEMDPs were introduced in 2014 in [Raskin and Sankur, 2014], whereas hmMDPs originate from [Chadès et al., 2012]. Though both models are in the end exactly the same, their construction is different. MEMDPs are clearly build as an extension of MDPs, while hmMDPs are constructed by making additional assumptions on POMDPs.

Where [Raskin and Sankur, 2014] is a very in-depth paper with theoretical results and novel algorithms to deal with MEMDPs, [Chadès et al., 2012] only defines hmMDPs to simplify computation of MOMDP algorithms, in particular the MO-SARSOP [Ong et al., 2010] and MO-IP [Araya-López et al., 2010] algorithms. Both are in turn special versions of the POMDP methods SARSOP [Kurniawati, Hsu, and Lee, 2008] and IP [Cassandra, Littman, and Zhang, 1997] that account for the mixed observability of a MOMDP.

Chadès et al. does provide one interesting complexity result. We already discussed how MOMDPs are from a computational complexity perspective at least as difficult as POMDPs. For hmMDPs, and thus MEMDPs, we now also have that computing a strategy that maximizes the finite-horizon reward problem in a hmMDP is PSPACE-complete [Chadès et al., 2012]. This is the same as for POMDPs.

4.6 Parametric systems

The last types of models we briefly introduce are *uncertain Markov decision processes* and *parametric Markov chains*. An overview of various kinds of extensions of MDPs simply would not be complete without these. Furthermore, parametric Markov chains are closely related to POMDPs, and uncertain MDPs can play a nice role in over-approximating non-revealing MEMDPs, as we will see in Chapter 5.

We define V to be a finite set of *parameters*, and $\mathbb{Q}_V$ is the set of *rational polynomial functions over* V.

> **Definition 17: Parametric Markov chain**
>
> A *parametric Markov chain* (pMC) is a tuple $M = (S, V, \mathcal{P})$. As usual, S is a set of states and A is a set of actions. V is a set of parameters, and $\mathcal{P}$ is the *parametric transition function* defined by $\mathcal{P}\colon S \times S \to \mathbb{Q}_V$.

Essentially, a pMC is a standard Markov chain except that the transitions now have rational functions of parameters instead of concrete probabilities assigned to them. This can be seen as a form of *uncertainty*. There can be *multiple valid probability distributions* that can be formed by substituting the parameters with rational values. Hence, pMC defines a *set* of Markov chains. An especially interesting aspect of the uncertainty induced by the parameters is that it can also define *dependencies between transitions*. If the transition $\mathcal{P}(s_i, s_j)$ contains some parameter p, it is perfectly fine for this p to also show up in some other transition $\mathcal{P}(s_k, s_l)$, but when assigning a value to p, the same value has to be filled in in both transitions.

A *valuation* is a function $v\colon V \to [0, 1]$ that assigns a value to each parameter such that for every state s we have $\sum_{s' \in S} \mathcal{P}(s, s') = 1$. Applying a valuation to a pMC M, denoted $M[v]$ yields a standard Markov chain. The *parameter synthesis problem* is to find a valuation for a pMC such that the induced Markov chain $M[v]$ satisfies some reachability or specification. This problem is ETR-complete [Winkler et al., 2019].

Parametric Markov chains are particularly interesting for their connection to POMDPs. Any POMDP with a randomized (finite memory) strategy can be rewritten into a pMC, where the parameters represent the strategy choices. If due to the partial observability two states have the same strategy choice, this is reflected by assigning the same parameter to the outgoing transition from those states. As explained before, a

solution to the parameter synthesis problem ensures that each parameter is assigned a valid value, thus enforcing that the strategy is *observation based*.

The transformation from POMDP to pMC allows us to use the efficient techniques for parameter synthesis to solve POMDP problems. Further more, we can encode finite memory strategies by expanding the state space using *finite state controllers*. For full details of these techniques we refer to [Cubuktepe et al., 2018] and [Junges et al., 2018].

In [Junges et al., 2019] an overview of various methods to solve the parameter synthesis problem for pMCs and parametric MDPs is given.

4.7 Uncertain MDPs and POMDPs

Besides parameters there are also other methods for incorporating uncertainty in the transition probabilities, with intervals being the most straightforward way.

> **Definition 18: Uncertain MDP**
>
> An *uncertain Markov decision process* is a tuple $\mathcal{M} = (S, A, \Delta, \mathbb{I})$, where S and A are a set of states and a set of actions as usual. $\mathbb{I}$ is a set of probability intervals, and Δ is the *uncertain* transition function defined by $\Delta \colon S \times A \times S \to \mathbb{I}$.

An *uncertain POMDP* is defined similarly as a standard POMDP, except that we now have an *uncertain* MDP instead of a standard MDP underlying the model.

An *instantiation* of an uncertain MDP $\mathcal{M}$ is a transition function $\delta \colon S \times A \to Distr(S)$ such that for all successor states s' we have that probability of transitioning into s' lies inside the interval: $\delta(s, a, s') \in \Delta(s, a, s')$. We think of the uncertain transition function Δ as a set of all possible standard transition functions bounded by the intervals. Every instantiation of an uncertain MDP, denoted $\mathcal{M}[\delta]$ is a standard MDP.

A strategy σ *robustly satisfies* some objective Φ in an initial state s_I, denoted $\mathcal{M}^{\sigma}_{s_I} \models \Phi$, if it satisfies Φ in every instantiation:

$$\mathcal{M}^{\sigma}_{s_I} \models \Phi \iff \forall \delta \in \Delta . \mathcal{M}[\delta]^{\sigma}_{s_I} \models \Phi.$$

An uncertain (PO)MDP is called *graph preserving* if all intervals have a lower bound strictly greater than 0. This means that no instantiation

can put the value 0 on a transition, ensuring that all instantiations have the same underlying graph. Graph preservation is an assumption commonly made, see for example [Wiesemann, Kuhn, and Rustem, 2013].

Robust strategies can be computed in a number of ways. Dynamic programming is used in [Wolff, Topcu, and Murray, 2012] to compute strategies that maximizes the probability of robustly satisfying some LTL specification. In [Suilen et al., 2020] robust optimization [Ben-Tal, El Ghaoui, and Nemirovski, 2009] is used. Finally, the value iteration algorithm can be modified into a robust value iteration [Bagnell, Ng, and Schneider, 2001] that computes strategies for uncertain MDPs where the uncertainty is given by convex sets.

Uncertain MDPs can play a nice role in approximating MEMDPs, as we will see in Chapter 5.

4.8 Comparison between MDPs, MEMDPs, and POMDPs

Given MDPs and POMDPs, the two types of models that are *the* standard when it comes to modeling systems with decision making in a stochastic environment, we may wonder where MEMDPs fit in. We already discussed some key complexity and computability results from the literature for all three model types, and we will now summarize those results.

Model	Almost-sure reachability	Quantitative reachability
MDP	in P	in P
MEMDP	in P	NP-hard, in PSPACE
POMDP (k)	EXPTIME-complete	Undecidable
POMDP (∞)	EXPTIME-complete	Undecidable

Table 1: Complexity of reachability objectives.

Table 1 contains the complexity results for solving almost-sure and quantitative reachability objectives as introduced in Definition 5. The results for MDPs follow from the existence of a linear programming formulation that solves these problems for MDPs [Baier and Katoen, 2008], which can be solved in polynomial time. The results for MEMDPs are from [Raskin and Sankur, 2014].

For POMDPs we make the distinct between the different amounts of

memory that is available for the strategy, denoted by (k) for finite memory strategies, and (∞) for infinite memory strategies. The reachability problem is a specific instance of a Büchi objective. Complexity results for Büchi objectives in POMDPs are found in [Chatterjee, Chmelík, and Tracol, 2016].

Model	Almost-sure safety	Quantitative safety
MDP	in P	in P
MEMDP	in P	NP-hard, in PSPACE
POMDP (k)	EXPTIME-complete	Undecidable
POMDP (∞)	EXPTIME-complete	Undecidable

Table 2: Complexity of safety objectives.

Table 2 contains the results for the three types of safety objectives we consider. The results for MDPs follow from the fact that any PCTL-objective can be solved in polynomial time in MDPs [Baier and Katoen, 2008]. The results for MEMDPs are again found in [Raskin and Sankur, 2014]. For POMDPs, we again note that safety objectives are a special instance of Büchi objectives.

Model	Almost-sure parity	Quantitative parity
MDP	in P	in P
MEMDP	in P	NP-hard
POMDP (k)	EXPTIME-complete	Undecidable
POMDP (∞)	Undecidable	Undecidable

Table 3: Complexity of parity objectives.

The complexity for parity objectives is given in Table 3. For the results on MDPs, see [Chatterjee, Jurdziński, and Henzinger, 2004]. The results for MEMDPs are again found in [Raskin and Sankur, 2014], and the results for POMDPs are from [Chatterjee, Chmelík, and Tracol, 2016].

The known complexity results for various reward problems as defined in Definition 6 are given in Table 4. The quantitative reward problem can be solved by a linear program [Baier and Katoen, 2008]. The results on the finite and infinite horizon reward problems are found in [Papadimitriou and Tsitsiklis, 1987].

The result for MEMDPs in the finite horizon case follows from the PSPACE-completeness of this problem for hidden model MDPs given

Model	Quantitative reward	Finite horizon reward	Infinite horizon reward
MDP	in P	in P	in P
MEMDP	*NP-hard*	PSPACE-complete	?
POMDP	Undecidable	PSPACE-complete	Undecidable

Table 4: Complexity of reward objectives.

in [Chadès et al., 2012], and, as already noted, hmMDPs are in fact MEMDPs. The complexity of the quantitative and infinite horizon problems are, to the best of our knowledge, unknown as of yet. In Chapter 5 we prove that the quantitative reward problem is NP-hard in MEMDPs.

The undecidability of the infinite horizon problem for POMDPs is from [Madani, Hanks, and Condon, 2000], and the PSPACE-completeness for the finite horizon case is given by [Papadimitriou and Tsitsiklis, 1987]. The undecidability of the quantitative reward problem follows from the undecidability of the quantitative reachability problem and the fact that reward structures can model reachability problems [Arming et al., 2018].

MDPs are from a computational perspective clearly preferred. However they may lack in their capabilities to model real-world scenarios with a sufficient level of detail [Russell and Norvig, 2010]. On the other hand, POMDPs provide these modeling capabilities, but are computationally hard to solve, especially when we are interested in optimal solutions. MEMDPs provide a nice middle ground between MDPs and POMDPs. As noted before, their modeling capabilities (a distinct number of predefined scenarios) are sufficient enough for many real-world applications, and their computational complexity (maybe NP-hard, but at least still decidable) leaves enough room for computing strategies to solve the problems that arise in said applications.

4.9 Tool support

Many of the models, their problems and solution methods have been implemented. `Storm` [Dehnert et al., 2017] and `PRISM` [Kwiatkowska, Norman, and Parker, 2011, Norman, Parker, and Zou, 2017] are complete model checkers. For parametric models and the parameters synthesis problem dedicated tools, such as `PROPhESY` [Dehnert et al., 2015] or `PARAM` [Hahn et al., 2010], were developed.

Chapter 5

Additional results on MEMDPs

In this chapter we study some basic properties of MEMDPs. We use the fact that MEMDPs are a set of MDPs on the same states and actions to define *bisimulations between MEMDPs*. We give *concrete algorithms* for finding I-reducing and i-revealing transitions. We show that a MEMDP can be *approximated by an uncertain MDP* to compute robust strategies. We prove that the quantitative expected reward problem is NP-hard in MEMDPs. And we define a mixed integer linear program to compute *deterministic strategies* satisfying a quantitative reachability property.

5.1 Bisimulations

A relation $\mathcal{R} \subseteq S \times S$ on a set of states S is an *equivalence relation* if it satisfies the following properties:

$$
\begin{aligned}
&\forall s, s', s'' \in S. \\
&\quad (s,s) \in \mathcal{R} && \text{(Reflexivity)} \\
&\quad (s,s') \in \mathcal{R} \iff (s',s) \in \mathcal{R} && \text{(Symmetry)} \\
&\quad (s,s'),(s',s'') \in \mathcal{R} \implies (s,s'') \in \mathcal{R} && \text{(Transitivity)}
\end{aligned}
$$

Definition 19: Bisimulation between MDPs

A *bisimulation* between two MDPs with the same sets of states and actions $M_1 = (S, A, \delta_1)$ and $M_2 = (S, A, \delta_2)$ with initial states $s_{I_1} \in S$ for M_1 and $s_{I_2} \in S$ for M_2, is an equivalence relation

$\mathcal{R} \subseteq S \times S$ such that the following holds:

$(s_{I_1}, s_{I_2}) \in \mathcal{R}$,
$(s, s') \in \mathcal{R} \implies \forall a \in A. \forall E \in S/\mathcal{R}. \quad \sum_{t \in E} \delta_1(s, a, t) = \sum_{t \in E} \delta_2(s', a, t)$,

where $S/\mathcal{R}$ is the partition of S into $\mathcal{R}$-equivalence classes. Then $E \in S/\mathcal{R}$ denotes one such equivalence class.

If it is possible to construct a relation between two MDPs as in Definition 19 we have a bisimulation and say the MDPs are *bisimilar*, denoted as $M_1 \sim M_2$. If the MDPs also have reward structures R_1 and R_2 we additionally require that

$$(s, s') \in \mathcal{R} \implies \forall a \in A. R_1(s, a) = R_2(s', a).$$

Bisimulations between states of the same MDP, or between different MDPs are described in [Castro, Panangaden, and Precup, 2009, Larsen and Skou, 1991].

Our interest in using bismulations between MDPs in the context of MEMDPs is twofold. First, we can use bisimulations to reduce the number of environments in a MEMDP. Second, we can easily construct a new definition for bismulations between MEMDPs by using the fact that a MEMDP is nothing more than a set of MDPs.

5.1.1 Reducing the number of environments

For theoretical purposes, it is nice to assume that each environment in a MEMDP actually adds something to the MEMDP. That is, we do not want any duplicate MDPs in our set. This is something that is implicitly assumed in [Raskin and Sankur, 2014]. However, this assumption may not always be valid from the start.

Consider the case study in [Chadès et al., 2012]. Here the hmMDP – which, as we have seen, is actually a MEMDP – is constructed by involving a number of *domain experts*. A number of possible scenarios are proposed, where each such scenario is modeled by an MDP, and the set of all scenarios is then the resulting MEMDP. Depending on who does the actual construction of the MDPs, it is possible that we get multiple MDPs that may not *look* similar, but do have the same behavior (i.e., are bisimilar). Thus the resulting MEMDP may have one or more superfluous environments, that we wish to remove.

Definition 20: Minimal MEMDP

A MEMDP $M = (S, A, \{\delta_i\}_{i\in[n]})$ is minimal (in the number of environments) if none of the environments are bisimilar to each other:

$$\forall i \in [n].\forall j \in [n] \setminus \{i\}. \quad M_i \not\sim M_j.$$

5.1.2 Bisimilarity of MEMDPs

Since we consider MEMDPs as just sets of MDPs, we can easily adapt the definition of a bisimulation between MDPs to bisimilarity between MEMDPs.

Definition 21: Bisimulation between MEMDPs

Let M and M' be two MEMDPs on the same set of states and actions that are minimal as in Definition 20. M and M' are bisimilar if each environment in one MEMDP is bisimilar to an environment in the other MEMDP:

$$\forall M_i \in M.\exists M'_j \in M'. \quad M_i \sim M'_j \text{ by some relation } \mathcal{R}_{i,j}$$

By symmetry of bisimulations we have that if $M_i \sim M_j$ by $R_{i,j}$ then also $M_j \sim M_i$ by $R_{i,j}$. From the transitivity it follows that if two environments from $M_k, M_l \in M$ are bisimilar to the same environment of $M'_j \in M'$, M_k and M_l are also bisimilar to each other:

$$M_k \sim M'_j \sim M_l \implies M_k \sim M_l.$$

Note that this can only happen if M and M' are not minimal. As a consequence, when two (minimal) MEMDPs are bisimilar, they have the same number of environments.

Definition 21 essentially creates a relation $\mathcal{R} \subseteq (S \times S)^n$ composed of n bisimulations: $\mathcal{R} = \mathcal{R}_{1,\phi(1)} \times \cdots \times \mathcal{R}_{n,\phi(n)}$ where ϕ is a permutation of the indices.

5.2 Algorithms for detection of i-revealing and I-reducing transitions

As already mentioned at their introduction, i-revealing and I-reducing transitions play an important role in analyzing MEMDPs. And they are also closely related: any I-reducing transition with $|I| = 1$ is in fact i-revealing, and a MEMDP with no I-reducing transitions at all is *graph preserving*. This latter fact can be used to approximate the MEMDP by an uncertain MDP, which we will discuss in Section 5.3. This calls for some concrete algorithms to figure out whether such transitions exist, and if so, what they reveal exactly.

5.2.1 Graph preservation

We begin with a simple algorithm to detect whether a MEMDP is graph preserving. Simply put, we check whether a transition exists in *all* environments, or in *none*. Exactly when either one of those is true for all transitions, the MEMDP is graph preserving.

Algorithm 1: Graph preservation check algorithm.

Input: MEMDP $M = (S, A, \{\delta_i\}_{i\in[n]})$
Output: Is M graph preserving?
for $(s, a, s') \in S \times A \times S$ **do**
  for $i \in [n] \setminus \{1\}$ **do**
    if $\delta_1(s, a, s') > 0$ **and** $\delta_i(s, a, s') = 0$ **then**
      return False
    end
    if $\delta_1(s, a, s') = 0$ **and** $\delta_i(s, a, s') > 0$ **then**
      return False
    end
  end
end
return True

Algorithm 1 loops over all possible transitions. If a transition exists in the first environment, it should also exist in all other environments. If this is not the case we can return `False` and immediately terminate the algorithm. The same goes for the case if the transition does not exist in the first environment, then it should also not exist in any of the other environments. If the algorithm checked all transitions without terminating earlier, we know the graph preservation property holds for

this MEMDP and return `True`. The time complexity of this algorithm is linear in the size of the MEMDP: $\mathcal{O}(\text{Size}(M))$.

5.2.2 I-reducing and i-revealing transitions

We first construct a general purpose algorithm that computes the set consisting of *all* I-reducing transitions. That is, transitions and the set I associated with them.

Algorithm 2: Search algorithm for all I-reducing transitions.

Input: MEMDP $M = (S, A, \{\delta_i\}_{i\in[n]})$
Output: Set $\mathcal{I}$ of all I-reducing transitions

```
I ← ∅
for (s, a, s') ∈ S × A × S do
    I ← ∅
    for i ∈ [n] do
        if δ_i(s, a, s') > 0 then
            I ← I ∪ {i}
        end
    end
    if 0 < |I| < n then
        I ← I ∪ {((s, a, s'), I)}
    end
end
return I
```

Algorithm 2 has a time complexity that is linear in the size of the input MEMDP M: $\mathcal{O}(\text{Size}(M))$.

In fact, Algorithm 2 can also be used to compute whether M is graph preserving, by checking if the output $\mathcal{I}$ is empty. In that case there are no I-reducing transitions, meaning that each transition either occurs in all or in none of the environments, which is the precise definition for graph preservation.

Similarly, this algorithm can also be used for finding all i-revealing transitions. By definition, an I-reducing transition is i-revealing precisely when $I = \{i\}$. However, we construct a separate algorithm for finding i-revealing transitions that is slightly more efficient.

The optimization in Algorithm 3 consists of stopping the inner for-loop once the transition occurs in more than one environment, as it cannot be i-revealing anymore. This optimization has no consequences for the

Algorithm 3: Search algorithm for all i-revealing transitions.

Input: MEMDP $M = (S, A, \{\delta_i\}_{i\in[n]})$
Output: Set $\mathcal{I}$ of all i-revealing transitions
$\mathcal{I} \leftarrow \emptyset$
for $(s, a, s') \in S \times A \times S$ **do**
  $I \leftarrow \emptyset$
  for $i \in [n]$ **do**
    if $\delta_i(s, a, s') > 0$ **then**
      $I \leftarrow I \cup \{i\}$
    end
    if $|I| > 1$ **then**
      break i
    end
  end
  if $|I| = 1$ **then**
    $\mathcal{I} \leftarrow \mathcal{I} \cup \{((s, a, s'), I)\}$
  end
end
return $\mathcal{I}$

theoretical complexity, though it may prove useful in practice. Worst-case a transition is revealing for the n-th environment, requiring the full execution of the inner for-loop to be detected.

5.3 Approximating MEMDPs via uncertain MDPs

Besides POMDPs, MEMDPs can also be transformed into uncertain MDPs. This transformation, however, will introduce more uncertainty than necessary, as MEMDPs are discrete and uMDPs continuous in the different transition functions they define.

> **Definition 22: Approximation uMDP**
>
> Given a MEMDP $M = (S, A, \{\delta_i\}_{i\in[n]})$, we construct the following uncertain MDP such that it can be used to soundly approximate M: $\mathcal{M} = (S, A, \Delta, \mathbb{I})$. Clearly, the set of states and actions are the same as in M. The set of intervals $\mathbb{I}$ is constructed by finding the minimum and the maximum values for each transition over all

environments:

$$I_{(s,a,s')} = \left[\min_{i\in[n]}\{\delta_i(s,a,s')\}, \max_{i\in[n]}(\{\delta_i(s,a,s')\}\right],$$
$$\mathbb{I} = \bigcup_{(s,a,s')\in S\times A\times S} I_{(s,a,s')}.$$

Then, the uncertain transition function $\Delta\colon S\times A\times S \to \mathbb{I}$ is defined by $\Delta(s,a,s') = I_{(s,a,s')}$.

The construction from Definition 22 is general and may include transition with an interval of the form $[0,p]$, for some $p \in (0,1]$. This means there will be *non-graph preserving instantiations* in the uncertain MDP. In other words, there is an instantiation where the transition associated with this interval gets a probability value of 0, thus removing the transition from the uMDP. As already discussed in Section 4.7, having non-graph preserving instantiation makes computing a strategy that is robust for all instantiation in the uMDP harder. In fact, many techniques rely on the assumption that no such transitions exist in the uMDP.

This can be guaranteed by ensuring the MEMDP we are approximating has no I-reducing transitions, and is graph preserving. This can be checked easily using Algorithm 1.

Lemma 1: Graph preservation lemma

If a MEMDP $M = (S, A, \{\delta_i\}_{i\in[n]})$ is graph preserving, then the approximating uMDP $\mathcal{M} = (S, A, \Delta, \mathbb{I})$ is also graph preserving.

Proof. We know M is graph preserving. Thus, for all tuples $(s,a,s') \in S\times A\times S$ either $\delta_i(s,a,s') > 0$ for all environments i, or for none of them.

If $\delta_i(s,a,s') = 0$ for all environments i, then

$$\min_{i\in[n]}\{\delta_i(s,a,s')\} = \max_{i\in[n]}\{\delta_i(s,a,s')\} = 0,$$

and as a consequence $I_{(s,a,s')} = [0,0]$.

If $\delta_i(s,a,s') > 0$ for all environments i, then $\min_{i\in[n]}\{\delta_i(s,a,s')\} > 0$. Then the interval at this transition is of the form $I_{(s,a,s')} = [x,y]$ where

$$x = \min_{i\in[n]}\{\delta_i(s,a,s')\} > 0, \quad y = \max_{i\in[n]}\{\delta_i(s,a,s')\} \geq x.$$

Which means that we have $0 < x \leq y \leq 1$, and thus this transition cannot be removed by some instantiation, as 0 is not part of this interval.

From this we conclude that if the MEMDP M is graph preserving, then the approximating uMDP is also graph preserving.

□

If we compute a strategy for a MEMDP via the approximating uMDP, we get a *sound overapproximation*, meaning the strategy is also correct for the MEMDP.

> **Theorem 1: Strategies for the approximation uMDP are sound for the MEMDP**
>
> let $M = (S, A, \{\delta_i\}_{i\in[n]})$ be a MEMDP, and $\mathcal{M} = (S, A, \Delta, \mathbb{I})$ the uMDP approximating M as defined in Definition 22. Let σ be a strategy *robustly satisfying* some objective Φ at s_I in $\mathcal{M}$: $\mathcal{M}^\sigma_{s_I} \models \Phi$. Then σ also satisfies Φ in M for all environments: $M^\sigma_{s_I} \models \Phi$.

Proof. Clearly the value of each transition in any environment lies between the minimum and maximum over all environments:

$$\forall(s, a, s') \in S \times A \times S.\ \forall i \in [n].\\ \min_{j\in[n]}\{\delta_j(s, a, s')\} \leq \delta_i(s, a, s') \leq \max_{j\in[n]}\{\delta_j(s, a, s')\}.$$

Then it follows from the construction of $\mathcal{M}$ that each transition in the MEMDP M lies in the associated interval in $\mathcal{M}$:

$$\forall(s, a, s') \in S \times A \times S.\ \forall i \in [n]. \qquad \delta_i(s, a, s') \in I_{(s,a,s')},$$

and so we have $\delta_i \in \Delta$ for all environments i.

So we can fix any transition function δ_i from the MEMDP as an instantiation, resulting in exactly the i-th environment of M. Formally: $\mathcal{M}[\delta_i] = M_i$.

Thus the set of environments is a subset of all the instantiations of $\mathcal{M}$.

Since the strategy σ robustly satisfies Φ in $\mathcal{M}$, that is, for all instantiations of $\mathcal{M}$, it naturally also satisfies Φ for any subset of all the instantiations, and in particular the subset of instantiations identified with the MEMDP M.

□

By Theorem 1 we know that a solution for the approximation uMDP $\mathcal{M}$ of a MEMDP M also holds for M itself. It is, however, an *overapproximation*, meaning that the $\mathcal{M}$ accounts for more uncertainty than necessary. In fact, the strategy computed in $\mathcal{M}$ is robust for uncountably many instantiations, whereas it only needs to be robust for the finite number of environments in M. As a result, it may be the case that there does not exist a strategy for $\mathcal{M}$, while these does exist one for M itself.

5.4 Complexity of the quantitative expected reward problem

In Section 4.8 we noted that does not discuss reward structures for MEMDPs. From [Chadès et al., 2012] we know that the finite horizon reward maximization problem is PSPACE-complete for hmMDPs (and thus MEMDPs). To the best of our knowledge, this is the only complexity result regarding reward structures in MEMDPs. We show that the quantitative expected reward problem is NP-hard via a straightforward reduction from the quantitative reachability problem.

Proposition 1: Reachability as a reward structure in MDPs [Arming et al., 2018]

Let $M = (S, A, \delta)$ be an MDP with some initial state s_I with some reachability objective for a state $t \in S$. The MDP with rewards $M_r = (S, A, \delta_r)$ with reward structure $R\colon S \times A \times S \to \mathbb{R}$ has the same sets of states and actions, and a transition function δ_r defined as

$$\delta_r(s,a) = \begin{cases} Dirac(t) & \text{if } s = t \\ \delta(s,a) \end{cases}$$

and reward function R defined as

$$R(s,a,s') = \begin{cases} 1 & \text{if } s \neq t \text{ and } s' = t \\ 0 & \text{otherwise.} \end{cases}$$

Essentially, the construction in Proposition 1 copies the MDP, except that the target state t is made into a sink state. The reward structure gives a reward of 1 for any transition leading to t, and a reward of 0 for any other transition.

A strategy σ satisfies the reachability of t in M with some threshold λ if and only if that same strategy satisfies the expected reward of that same threshold λ in M_r.

Theorem 2: Quantitative expected reward is NP-hard for MEMDPs

Let $M = (S, A, \{\delta_i\}_{i\in[n]})$ be a MEMDP. Let $R_i \colon S \times A \times S \to \mathbb{R}$ be the reward function in environment i of M. Let $\kappa \in \mathbb{R}$ be the *reward threshold*, and $t \in S$ be a target state.

Computing a strategy σ for M such that the quantitative expected reward problem is solved with threshold $\bowtie \kappa$ is NP-hard.

Proof. We prove Theorem 2 by constructing a polynomial many-to-one reduction from the quantitative reachability problem for MEMDPs. Recall that this problem is NP-hard for MEMDPs by [Raskin and Sankur, 2014].

Each environment $M_i \in M$ is an MDP for which, by Proposition 1, we can construct an MDP $M_{i,r}$ with a reward structure $R_i \colon S \times A \times S \to \mathbb{R}$ as defined in the proposition.

This construction is linear in the size of the MEMDP. For every MDP we have to copy and adapt the transition function and construct the reward function, which can be done in $\mathcal{O}(|S| \cdot |A| \cdot |S|)$ operations. This is repeated for every MDP, hence a total computation time of $\mathcal{O}(n \cdot |S| \cdot |A| \cdot |S|)$, which simplifies to $\mathcal{O}(Size(M))$.

The set $M_r = \{M_{1,r}, \ldots, M_{n,r}\}$ is a set of n MDPs that only differ in their transition functions and reward functions. Hence, a valid MEMDP.

Now suppose a strategy σ solves the expected reward problem for threshold κ in every $M_{i,r} \in M_r$. Then, by construction of each $M_{i,r}$, this strategy also solves the reachability problem of reaching t with threshold κ.

Since the quantitative reachability problem is NP-hard for MEMDPs, it follows that the quantitative expected reward problem is also NP-hard.

□

5.5 MILP for deterministic memoryless strategies in MEMDPs

The quantitative reachability problem for MEMDPs can be solved by an quadratic equation system that results in a randomized finite memory strategy. Sometimes, randomization is not an option. Small scale implementations of an agent may not have access to a random number generator to randomly pick actions based on some precomputed distribution. We provide a mixed integer linear program (MILP) [Cohen and Parmentier, 2018] for computing *deterministic strategies* in MEMDPs. Note that this encoding is very general and can be extended satisfy multiple objectives. We present our encoding for maximizing the reachability of a target set.

> **Definition 23: MILP**
>
> A *mixed integer linear program* (MILP) is an optimization problem of the following form:
>
> $$\begin{aligned} &\text{minimize } f_0(x_1, \ldots, x_n) \\ &\text{subject to } f_i(x_1, \ldots, x_n) \leq 1, && i \in [m] \\ &\qquad x_j \in \mathbb{Z}, && j \in J \subseteq [n] \end{aligned}$$
>
> where $x_1, \ldots, x_n$ are n real-valued variables, $f_0, \ldots, f_m$ are $m+1$ linear functions over the variables, and J is a set of indices for which the variables are restricted to integers.

Let $\{\sigma_{s,a} \mid s \in S, a \in A\}$ be the integer variables that will encode the strategy σ. We define real-valued variables p_s^j, r_s^j and $t_{s,s'}^j$. for each environment j.

The variables p_s^j encode the probability of reaching a target state in T from s in environment j. Since we only consider a deterministic scheduler we need additional preprocessing of the MEMDP. It may be the case that the target set T becomes unreachable under some strategy. We define the set of states that can only reach the target set under *some* strategies in environment j as S_{PR}^j (for *problematic states*). The set of *problematic actions* in environment j is then given by

$$A_{\text{PR}} = \left\{(s,a) \in S \times A \mid a \in A(s) \wedge \text{Succ}(s,a) \subseteq S_{PR}^j\right\}.$$

Using these sets of problematic states and actions for each environment, we define the variables r_s^j as a *ranking* over the problematic states. Each

$s \in S^j_{\text{PR}}$ is assigned a ranking $r^j_s \in [0,1]$. The variables $t^j_{s,s'}$ are additional real numbered variables used to ensure correctness. The MILP encoding of the problem is given by the equations 1 – 6.

$$\text{maximize} \quad \sum_{j=1}^{n} p^j_{s_I} \tag{1}$$

subject to

$$\forall j \in [n]. \quad \forall s \in T. \quad p^j_s = 1 \tag{2}$$

$$\forall j \in [n]. \quad \forall s \in S \setminus T. \quad \sum_{a \in A(s)} \sigma_{s,a} = 1 \tag{3}$$

$$\forall j \in [n]. \quad \forall s \in S \setminus T. \quad \forall a \in A(s). \quad p^j_s \leq (1 - \sigma_{s,a}) + \sum_{s' \in \text{Succ}(s,a)} \delta_j(s,a,s') \cdot p^j_{s'} \tag{4}$$

$$\forall j \in [n]. \quad \forall (s,a) \in A^j_{\text{PR}}. \quad \forall s' \in \text{Succ}(s,a). \quad r^j_s < r^j_{s'} + 1 - t^j_{s,s'} \tag{5}$$

$$\forall j \in [n]. \quad \forall (s,a) \in A^j_{\text{PR}}. \quad p^j_s < 1 - \sigma_{s,a} + \sum_{s' \in \text{Succ}(s,a)} t^j_{s,s'} \tag{6}$$

Equation (1) defines the objective function. This can be, for example, maximizing the reachability of the target set T from initial state s_I for all environments. Constraint (2) ensures that for all states in the target set, and all environments j, the variables p^j_s are set to 1, as the probability of reaching the target set from these states is naturally 1. Constraint (3) ensures that our memoryless strategy that will be encoded by the σ-variables is a valid probability distribution. Constraint (4) recursively defines the probability of reach the target set from state s for each environment j. Constraints (5) and (6) are needed to deal with the special case where the target set becomes unreachable under some strategies. If no such strategies exists, these constraints can be omitted.

Note that this encoding is very similar to the MILP for computing deterministic strategies in POMDPs [Winterer et al., 2020]. This similarity is not a coincidence, as we are about to see in Chapter 6.

Chapter 6

Entropy guided decision making

In this chapter, we present our main results. By making the connection between MEMDPs and POMDPs, we bring the notion of belief to MEMDPs, and show how this can be used to infer the true environment by making a good choice between the available actions. This 'good choice' is determined via the *expected entropy*, a measure on how much information we expect to gain by choosing a certain action. We formalize this in a number of theorems, and construct a learning algorithm.

6.1 Belief over MEMDP environments

In Chapter 4 we have seen that MEMDPs can be placed between MDPs and POMDPs. We also noted that every MDP can naturally be rewritten into a POMDP by assigning each state a unique observation. In a similar way, we can rewrite every MEMDP into a POMDP, making all techniques known for solving problems in POMDPs available to MEMDPs.

> **Definition 24: Associated POMDP**
>
> Given a MEMDP $M = (S, A, \{\delta_i\}_{i\in[n]})$ we define the *associated POMDP of* M as the POMDP $P_M = (S', A, \delta', Z, O)$ where:
>
> - S' is the set of states of the POMDP, defined by the disjoint union over the states of each environment of the MEMDP. We do this in the following way: $S' = \{\langle s, i\rangle \mid s \in S, i \in [n]\}$ i.e.

we copy each state n times, once for every environment, and mark the states with the environment number.

- A is the set of actions, and is the same as in the MEMDP.
- δ' defines the partial transition function. In this POMDP, we have $\delta' \colon S' \times A \to Distr(S')$, and we define it as $\delta'(\langle s, i \rangle, a, s') = \delta_i(s, a, s')$.
- $Z = \{z_s \mid s \in S\}$ is the set of observations, in which we have one unique observation z_s for every state $s \in S$ in the MEMDP M.
- The (deterministic) observation function $O \colon S' \to Z$ is defined by $O(\langle s, i \rangle) = z_s$, such that we map each state $\langle s, i \rangle$ (regardless of the environment number i) to the observation defined for s: z_s.

Our construction in Definition 24 is similar to that of [Chatterjee et al., 2020], up to some different choices in notation.

The mapping from MEMDPs to POMDPs allows is to use any technique available for POMDPs on MEMDPs. Most notably, it allows us to talk about belief states and the belief MDP for a MEMDP, which we will discuss in detail in Chapter 6.

It is important to keep in mind that POMDPs are very hard to solve in general (recall the comparisons in Section 4.8), so solving MEMDP problems via POMDPs is only preferred if this is the only (known) way to solve that problem.

Let $P = (S', A, \delta, Z, O)$ be the POMDP constructed by M (see Definition 24). Assume we have some initial state $s_I \in S$ in our MEMDP, and that this state is initial for every environment i in M. If every environment is equally likely, our initial belief b_0 for P is defined as the uniform distribution over the environments observing s_I:

$$\begin{aligned} \forall i \in [n] \quad & b_0(\langle s_I, i \rangle) = \frac{1}{n}, \\ \forall s \neq s_I . \forall i \in [n] \quad & b_0(\langle s, i \rangle) = 0. \end{aligned}$$

Recall that the state space of P is the product of the states from the MEMDP and the set of environments, hence every state is a pair $\langle s, i \rangle$ with $s \in S$ and $i \in [n]$. The observations are deterministic in the state s. This means that any pairs $\langle s, i \rangle$ and $\langle s', j \rangle$ can be distinguished if $s \neq s'$, regardless of what i and j are. Hence, a belief state at some time t, associated with some observed state s, which we will denote as

$b_t(\langle s, \cdot \rangle)$, is a distribution over environments: $b(\langle s, \cdot \rangle) \in Distr([n])$. The probability of being in environment i is then given by $b_t(\langle s, i \rangle)$.

The belief update works by default as defined in Definition 9 for POMDPs, though the specific structure of MEMDPs allows us to make some simplifications.

Definition 25: Belief update for MEMDPs

Given the current belief $b_t(\langle s, \cdot \rangle)$ and observing the successor state s', we compute the successor belief state $b_t(\langle s, \cdot \rangle)$ by performing the following computation for all environments i.

$$b_{t+1}(\langle s', i \rangle) = \frac{b_t(\langle s, i \rangle) \cdot \delta_i(s, a, s')}{\sum_{j \in [n]} b_t(\langle s, j \rangle) \cdot \delta_j(s, a, s')}$$

Essentially, we compute the probability of being able to observe s' after choosing action a at state s in environment i, and normalize this probability over all environments. This can be interpreted as taking the transition probability $\delta_i(s, a, s')$ weighted by the probability that we are in environment i.

Computing the a new belief state in a MEMDP is computationally easier than computing a new belief state in a POMDP. The nominator is a single multiplication, that is repeated for every environment i. Just as in POMDPs, the denominator is independent of the input and only needs to be computed once. This is a single sum over the environments, thus linear in the number of environments. This means that the total complexity of the belief update in MEMDPs is $\mathcal{O}(n)$, in contrast to the belief update in POMDPs which is quadratic in the number of states.

The belief update for MEMDPs behaves exactly as we would expect in the presence of i-revealing transitions. That is, observing an i-revealing transition leads the belief update to return a Dirac distribution for environment i, which is formalized in Lemma 2.

For ease of reading, let the successor belief state observing s' after action a be $b_a(\langle s', \cdot \rangle)$.

Lemma 2: Belief update observing an i-revealing transition

Let b be an arbitrary belief state in the MEMDP $M = (S, A, \{\delta_i\}_{i \in [n]})$, observing a state $s \in S$. Suppose we take action $a \in A(s)$ and observe successor state $s' \in S$, and that we know that (s, a, s') is

i-revealing for environment $M_i \in M$. Then the successor belief $b_a(\langle s', \cdot \rangle)$ is a Dirac distribution for environment i.

Proof. Note that (s, a, s') is i-revealing, meaning $\delta_i(s, a, s') > 0$ and $\delta_k(s, a, s') = 0$ for all $k \neq i$. We simply apply the belief update as defined in Definition 25.

For any environment $k \neq i$ we get

$$\begin{aligned} b_a(\langle s', k \rangle) &= \frac{b(\langle s, k \rangle) \cdot \delta_k(s, a, s')}{\sum_{j \in [n]} b(\langle s, j \rangle) \cdot \delta_j(s, a, s')} \\ &= \frac{b(\langle s, k \rangle) \cdot 0}{\sum_{j \in [n]} b(\langle s, j \rangle) \cdot \delta_j(s, a, s')} \\ &= \frac{0}{\sum_{j \in [n]} b(\langle s, j \rangle) \cdot \delta_j(s, a, s')} \\ &= 0. \end{aligned}$$

And for the environment i we have

$$\begin{aligned} b_a(\langle s', i \rangle) &= \frac{b(\langle s, i \rangle) \cdot \delta_i(s, a, s')}{\sum_{j \in [n]} b(\langle s, j \rangle) \cdot \delta_j(s, a, s')} \\ &= \frac{b(\langle s, i \rangle) \cdot \delta_i(s, a, s')}{b(\langle s, i \rangle) \cdot \delta_i(s, a, s') + \sum_{j \in [n] \setminus \{i\}} b(\langle s, j \rangle) \cdot \delta_j(s, a, s')} \\ &= \frac{b(\langle s, i \rangle) \cdot \delta_i(s, a, s')}{b(\langle s, i \rangle) \cdot \delta_i(s, a, s') + \sum_{j \in [n] \setminus \{i\}} b(\langle s, j \rangle) \cdot 0} \\ &= \frac{b(\langle s, i \rangle) \cdot \delta_i(s, a, s')}{b(\langle s, i \rangle) \cdot \delta_i(s, a, s') + 0} \\ &= 1. \end{aligned}$$

Meaning that the successor belief $b_a(\langle s', \cdot \rangle)$ is a Dirac distribution for environment i.

□

Similarly, if the current belief is already a Dirac distribution over the environment, the belief update does not change this, as formalized in Lemma 3.

Lemma 3: Belief update in Dirac distributions

Let the current belief b observing some state s be Dirac in some environment i, thus $b(\langle s, i \rangle) = 1$ and $b(\langle s, k \rangle) = 0$ for all other environments $k \neq i$. Let b_a be the successor belief of b after performing

an arbitrary action a and observing an arbitrary successor state s'. Then b_a is still Dirac in i.

Proof. Apply the belief update as defined in Definition 25. First, we look at the belief update for any other environment $k \neq i$.

$$\begin{aligned} b_a(\langle s', k\rangle) &= \frac{b(\langle s, k\rangle) \cdot \delta_k(s, a, s')}{\sum_{j\in[n]} b(\langle s, j\rangle) \cdot \delta_j(s, a, s')} \\ &= \frac{0 \cdot \delta_i(s, a, s')}{\sum_{j\in[n]} b(\langle s, j\rangle) \cdot \delta_j(s, a, s')} \\ &= 0. \end{aligned}$$

And now for environment i:

$$\begin{aligned} b_a(\langle s', i\rangle) &= \frac{b(\langle s, i\rangle) \cdot \delta_i(s, a, s')}{\sum_{j\in[n]} b(\langle s, j\rangle) \cdot \delta_j(s, a, s')} \\ &= \frac{1 \cdot \delta_i(s, a, s')}{b(\langle s, i\rangle) \cdot \delta_i(s, a, s') + \sum_{j\in[n]\setminus\{i\}} b(\langle s, j\rangle) \cdot \delta_j(s, a, s')} \\ &= \frac{1 \cdot \delta_i(s, a, s')}{b(\langle s, i\rangle) \cdot \delta_i(s, a, s') + 0} \\ &= 1. \end{aligned}$$

Thus, the successor belief state $b_a(\langle s', \cdot\rangle)$ is again a Dirac distribution for environment i.

□

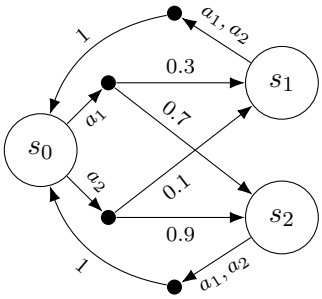
Lemma 3 tells us that if our belief is a Dirac distribution, which means we know which environment is the true environment with certainty, we cannot lose this information. This does, however, not mean that we cannot lose information on the true environment in other belief states. However, there is no monotone convergence from an initial belief to a Dirac belief, as the following example shows.

6.1.1 Belief over MEMDP environments: example

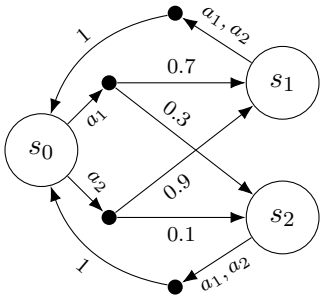
Consider the two-environment MEMDP and its associated POMDP in Figure 10.

First, we compute a small number of belief states for this MEMDP. Fix as strategy

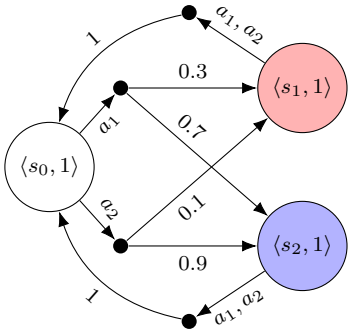
$$\sigma = a_1a_1a_2a_2a_1a_1a_2a_2\ldots$$

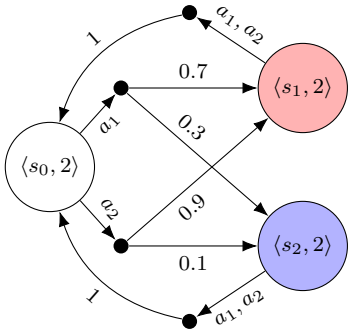


(a) MEMDP environment 1.



(b) MEMDP environment 2.





(c) Corresponding POMDP.

Figure 10: A MEMDP and its corresponding POMDP.

and suppose we observe the following sequence of observations:

$$s_0 s_1 s_0 s_1 s_0 s_2 s_0 s_1 s_0 s_2.$$

As there are no I-reducing or i-revealing transitions in this MEMDP, the given strategy and observation sequence are possible in both environments while not immediately revealing any information about the true environment.

Now assume that environment 1 is the true environment. As said before, the observation sequence is possible in this environment, but note that it is very unlikely to occur. Finally, we assume as out initial belief that both environments are just as likely: $b_0(\langle s_0, 1\rangle) = b_0(\langle s_0, 2\rangle) = 0.5$, and $b_0(\langle s_i, j\rangle) = 0$ for all other states s_i and environments j in this MEMDP.

Note that at every even step we will observe state s_0 as the MEMDP will move with probability 1 for both actions in both environments to s_0 when we are in state s_1 or s_2. As a consequence, the belief at every even time step will just shift the probabilities back to s_0 without changing their values. As such, we will only add the initial distribution b_0 and

the belief after one round (b_2) to the table, from then on all beliefs at even times are skipped.

We compute a couple of steps of the belief update, with the results in Table 5. For ease of reading we round the results up to three decimals, though it should be noted that since all transitions and the initial belief of this MEMDP can be expressed as rationals, the result of the belief update at each step is also a rational.

	$\langle s_0, 1\rangle$	$\langle s_1, 1\rangle$	$\langle s_2, 1\rangle$	$\langle s_0, 2\rangle$	$\langle s_1, 2\rangle$	$\langle s_2, 2\rangle$
b_0	0.5	0	0	0.5	0	0
b_1	0	0.3	0	0	0.7	0
b_2	0.3	0	0	0.7	0	0
b_3	0	0.045	0	0	0.955	0
b_5	0	0	0.1	0	0	0.9
b_7	0	0.012	0	0	0.988	0
b_9	0	0	0.028	0	0	0.972

Table 5: Belief updates for the MEMDP in Figure 10.

We summarize the difference between the two environments in Table 6.

	Env. 1	Env. 2
b_1	-0.2	$+0.2$
b_3	-0.255	$+0.255$
b_5	$+0.055$	-0.055
b_7	-0.088	$+0.088$
b_9	$+0.016$	-0.016

Table 6: Shifts between the beliefs from Table 5.

The numbers in Table 5 suggest that environment 2 is the true environment given the sequence of observations under the strategy we chose. However, we cannot actually draw that conclusion.

First, observe that while the belief clearly shifts towards environment 2, this is not monotone. For example, see how in steps 5 or 9 environment 1 becomes more likely again.

Second, we already mentioned that the given observation sequence is very unlikely for environment 1. And this is just for a finite (and very

short) sequence. Alternatively, consider the following sequence of observations that is very likely for environment 1 under the same strategy:

$$s_0 s_2 s_0 s_2 s_0 s_2 s_0 s_2 s_0 s_2 s_0 s_2.$$

Again we compute the belief after each step, as presented in Table 7.

	$\langle s_0, 1\rangle$	$\langle s_1, 1\rangle$	$\langle s_2, 1\rangle$	$\langle s_0, 2\rangle$	$\langle s_1, 2\rangle$	$\langle s_2, 2\rangle$
b_0	0.5	0	0	0.5	0	0
b_1	0	0	0.7	0	0	0.3
b_3	0	0	0.955	0	0	0.045
b_5	0	0	0.980	0	0	0.020
b_7	0	0	0.998	0	0	0.002
b_9	0	0	0.999	0	0	0.001

Table 7: Belief updates

This sequence suggest a monotone convergence to environment 1. Now we consider a (finite) part of the belief MDP associated with this example. We compute the following belief states:

$$b_0 = \left\{\langle s_0, 1\rangle \mapsto \frac{1}{2}, \langle s_0, 2\rangle \mapsto \frac{1}{2}\right\} \qquad b_1 = \left\{\langle s_1, 1\rangle \mapsto \frac{3}{10}, \langle s_1, 2\rangle \mapsto \frac{7}{10}\right\}$$
$$b_2 = \left\{\langle s_2, 1\rangle \mapsto \frac{7}{10}, \langle s_0, 2\rangle \mapsto \frac{3}{10}\right\} \qquad b_3 = \left\{\langle s_2, 1\rangle \mapsto \frac{3}{10}, \langle s_0, 2\rangle \mapsto \frac{7}{10}\right\}$$
$$b_4 = \left\{\langle s_0, 1\rangle \mapsto \frac{7}{10}, \langle s_0, 2\rangle \mapsto \frac{3}{10}\right\} \qquad b_5 = \left\{\langle s_1, 1\rangle \mapsto \frac{9}{58}, \langle s_1, 2\rangle \mapsto \frac{49}{58}\right\}$$
$$b_6 = \left\{\langle s_2, 1\rangle \mapsto \frac{1}{2}, \langle s_2, 2\rangle \mapsto \frac{1}{2}\right\} \qquad b_7 = \left\{\langle s_1, 1\rangle \mapsto \frac{1}{22}, \langle s_1, 2\rangle \mapsto \frac{21}{22}\right\}$$
$$b_8 = \left\{\langle s_2, 1\rangle \mapsto \frac{27}{34}, \langle s_2, 2\rangle \mapsto \frac{7}{34}\right\} \qquad b_9 = \left\{\langle s_1, 1\rangle \mapsto \frac{1}{10}, \langle s_1, 2\rangle \mapsto \frac{9}{10}\right\}$$
$$b_{10} = \left\{\langle s_2, 1\rangle \mapsto \frac{9}{10}, \langle s_2, 2\rangle \mapsto \frac{1}{10}\right\} \qquad b_{11} = \left\{\langle s_0, 1\rangle \mapsto \frac{1}{10}, \langle s_0, 2\rangle \mapsto \frac{9}{10}\right\}$$
$$b_{12} = \left\{\langle s_1, 1\rangle \mapsto \frac{1}{22}, \langle s_1, 2\rangle \mapsto \frac{21}{22}\right\} \qquad b_{13} = \left\{\langle s_2, 1\rangle \mapsto \frac{7}{34}, \langle s_2, 2\rangle \mapsto \frac{27}{34}\right\}$$
$$b_{14} = \left\{\langle s_1, 1\rangle \mapsto \frac{1}{82}, \langle s_1, 2\rangle \mapsto \frac{81}{82}\right\} \qquad b_{15} = \left\{\langle s_2, 1\rangle \mapsto \frac{1}{2}, \langle s_2, 2\rangle \mapsto \frac{1}{2}\right\}$$

We draw the corresponding part of the belief MDP in Figure 11.

White belief states are associated with observing s_0 in the MEMDP, red states with observing s_1, and blue states with observing s_2. Note that

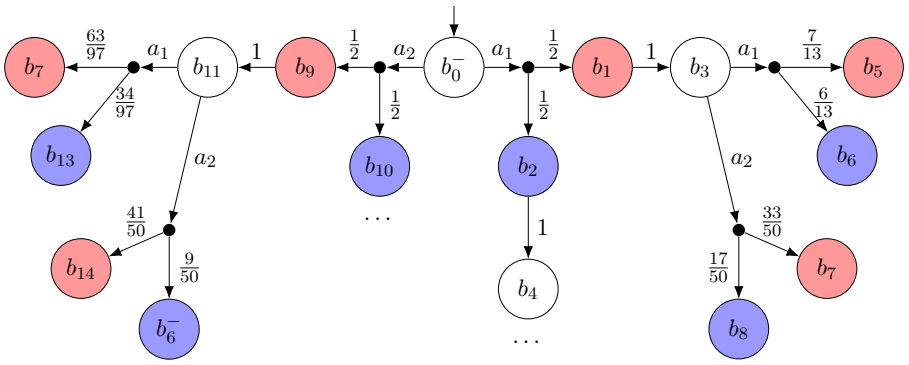


Figure 11: Initial part of the belief MDP.

some belief states are the same, even though we draw them twice to prevent arrows crossing the entire drawing, see for example b_7 and b_{12}. Using the belief MDP we can find out which belief states can be reached under some strategy, and with what probability they are reached.

Due to the cyclic nature of the MEMDP in this example, we can actually describe what an arbitrary part of the belief MDP looks like. Note that after any red or blue state we always end up in a white state, which is a belief state observing s_0. Similarly, after every white state there is a split over the two actions a_1 and a_2, each followed by a red state (a belief state observing s_1), and a blue state (a belief state observing s_2).

Suppose we have some (white) belief state $b_n = \{\langle s_0, 1\rangle \mapsto p, \langle s_0, 2\rangle \mapsto 1-p\}$, where p is an arbitrary probability value. This belief state is all we need to determine its successor states and the transitions between them. We compute all the successor belief states of the form $b_{n+1}^{a_k,s_l}$ where a_k is the action chosen, and s_l is the received observation.

$$b_{n+1}^{a_1 s_1} = \left\{ \langle s_1, 1\rangle \mapsto \frac{3p}{7-4p}, \langle s_1, 2\rangle \mapsto \frac{7-7p}{7-4p} \right\}$$
$$b_{n+1}^{a_1 s_2} = \left\{ \langle s_2, 1\rangle \mapsto \frac{7p}{4p+3}, \langle s_2, 2\rangle \mapsto \frac{3-3p}{4p+3} \right\}$$
$$b_{n+1}^{a_2 s_1} = \left\{ \langle s_1, 1\rangle \mapsto \frac{p}{9-8p}, \langle s_1, 2\rangle \mapsto \frac{9-9p}{9-8p} \right\}$$
$$b_{n+1}^{a_2 s_2} = \left\{ \langle s_2, 1\rangle \mapsto \frac{9p}{8p+1}, \langle s_2, 2\rangle \mapsto \frac{1-p}{8p+1} \right\}$$

Each of these belief states is then succeeded by a belief state $\hat{b}_{n+1}^{a_k,s_l}$ that has the same distribution over the environments, but shifted to observe s_0. We draw the composition of these belief states in Figure 12.

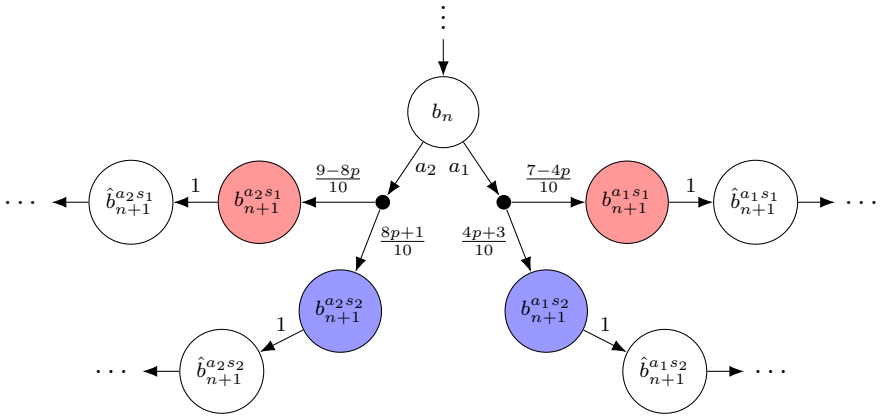


Figure 12: An arbitrary part of the belief MDP.

Each of the white states can now serve as a new "b_n" belief state. Hence the full belief MDP for the MEMDP in Figure 10 is the infinite composition of the structure given in Figure 12. We see this construction matches the initial part of the belief MDP in Figure 11.

6.2 Entropy as a measure of knowledge on beliefs

From the example in Section 6.1 we learn that there is no convergence of the belief in MEMDPs. The successor belief state depends on the observation we receive, and if this observation contradicts previous observations the belief is adjusted to this new information.

However, we also noted that certain observations are more likely than others in the true environment. So, intuitively, we want to weight the belief state with how likely it is to occur. Furthermore, we are not necessarily interested in the exact probability distribution over environments that a belief state provides. Our main interest is to measure how close (or far away) we are from reaching a Dirac distribution.

The *Shannon entropy* [Shannon, 2001] is a standard measure from information theory [Csiszár and Shields, 2004, Gray, 2011] that measures the amount of information in a random variable.

Definition 26: Shannon entropy

The *Shannon entropy* of a discrete probability distribution μ over a finite set X is defined as

$$H(\mu) = \sum_{x \in X} \mu(x) \cdot \log_b \left(\frac{1}{\mu(x)} \right),$$

where the base of the logarithm b is free to choose. By convention we have that, if $\mu(x) = 0$ then $\mu(x) \cdot \log_2(\frac{1}{\mu(x)}) = 0 \cdot \log_2(\frac{1}{0}) = 0$.

We will use the entropy on the belief over environments, hence $\mu \in Distr(n)$, and thus a natural choice for the base b is n. This ensures that the entropy of a uniform distribution on n environments is maximal and equal to 1, and for a Dirac distribution on n environments the entropy is minimal and equal to 0.

As an example, consider an arbitrary MEMDP of 2 environments. Let us consider some possible belief states for observing an arbitrary state s. First, let b be the uniform distribution, so

$$b = \left\{ \langle s, 1 \rangle \mapsto \frac{1}{2}, \langle s, 2 \rangle \mapsto \frac{1}{2} \right\}.$$

Then the entropy of b is given by

$$\begin{aligned} H(b) &= \sum_{i \in [2]} b(\langle s, i \rangle) \cdot \log_2 \left(\frac{1}{b(\langle s, i \rangle)} \right) \\ &= \frac{1}{2} \cdot \log_2(2) + \frac{1}{2} \cdot \log_2(2) \\ &= \frac{1}{2} + \frac{1}{2} = 1. \end{aligned}$$

Now consider a belief state that is Dirac in the first environment:

$$b = \{ \langle s, 1 \rangle \mapsto 1, \langle s, 2 \rangle \mapsto 0 \}.$$

Then the entropy is given by

$$\begin{aligned} H(b) &= \sum_{i \in [2]} b(\langle s, i \rangle) \cdot \log_2 \left(\frac{1}{b(\langle s, i \rangle)} \right) \\ &= \log_2(1) + 0 = 0. \end{aligned}$$

As a third and final example, consider the following two belief states:

$$b = \left\{ \langle s, 1 \rangle \mapsto \frac{1}{4}, \langle s, 2 \rangle \mapsto \frac{3}{4} \right\}, \quad \bar{b} = \left\{ \langle s, 1 \rangle \mapsto \frac{3}{4}, \langle s, 2 \rangle \mapsto \frac{1}{4} \right\}.$$

Again we compute the entropy for both, and note

$$\begin{aligned} H(b) &= \frac{1}{4} \cdot \log_2(4) + \frac{3}{4} \cdot \log_2(\frac{4}{3}) \\ &= \frac{3}{4} \cdot \log_2(\frac{4}{3}) + \frac{1}{4} \cdot \log_2(4) = H(\bar{b}). \end{aligned}$$

This symmetry means that we cannot derive *which* environment is the most likely from the entropy of a belief state, we just know how close (or far away) we are from a Dirac distribution for *some* environment.

6.2.1 The setting and notation

Since our focus is real-time decision making we do not consider the whole MEMDP. At a given state, all we need are the available actions and the distributions over successor states for every environment. Before we get to the decision making, i.e., determining which action to choose at a given state, we first investigate the behavior at a given state s and a fixed action a. This setting is illustrated in Figure 13.

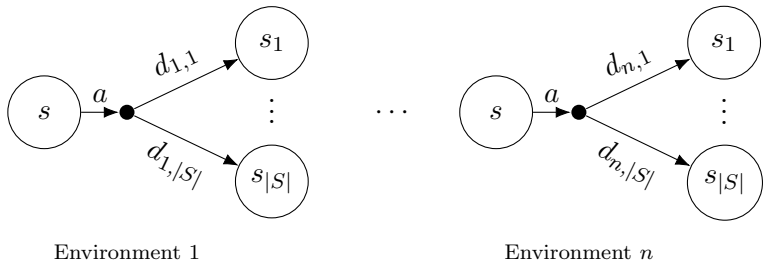


Figure 13: The setting in which we operate.

We introduce the following short-hand notation. We fix an enumeration $1, \ldots, |S|$ over all possible successor states and use this for all of the following definitions.

- We have $\mathbf{b} = (b_1, \ldots, b_n)$ as the current belief, observing state s.
- For every environment i we have the distribution over successor states $\mathbf{d}_i = \delta_i(s, a) = (d_{i,1}, \ldots, d_{i,|S|})$.
- For every possible successor state $s_k \in S$ we define the successor belief state observing s_k after action a as $\mathbf{B}_k = (B_{k,1} \ldots, B_{k,n})$.
- The random vector $\mathbf{B} = (\mathbf{B}_1, \ldots, \mathbf{B}_{|S|})$ consists of all successor belief states.

The vector $\mathbf{B}$ forms a probability distributions over the distributions $\mathbf{B}_1, \ldots, \mathbf{B}_{|S|}$ by weighting each distribution with the probability of observing that successor state. This is determined by the probability of transitioning into that successor state for a given environment, $d_{i,k}$, weighted by the probability of being in that environment. Explicitly, the probability of having distribution $\mathbf{B}_k$ is $\sum_{i \in [n]} b_i \cdot d_{i,k}$.

$$\mathbf{B} = \left(\sum_{i \in [n]} b_i \cdot d_{i,1} \right) \cdot |\mathbf{B}_1\rangle + \cdots + \left(\sum_{i \in [n]} b_i \cdot d_{i,|S|} \right) \cdot |\mathbf{B}_{|S|}\rangle.$$

Each successor belief observing state s_k, $\mathbf{B}_k$, is a distribution over the environments, determined by the belief update:

$$\mathbf{B}_k = \left(\frac{b_1 \cdot d_{1,k}}{\sum_{j \in [n]} b_j \cdot d_{j,k}} \right) \cdot |1\rangle + \cdots + \left(\frac{b_n \cdot d_{n,k}}{\sum_{j \in [n]} b_j \cdot d_{j,k}} \right) \cdot |n\rangle.$$

Proposition 2: Expected entropy over the successor belief states will not increase [Chatterjee et al., 2020]

Given a MEMDP $M = (S, A, \{\delta_i\}_{i \in [n]})$, any action $a \in A$ and an arbitrary belief state $\mathbf{b}$ observing state s, the *expected entropy* of $\mathbf{B}$ is bounded by the entropy of $\mathbf{b}$:

$$\mathbb{E}[H(\mathbf{B})] \leq H(\mathbf{b}).$$

Where the expected entropy of $\mathbf{B}$, $\mathbb{E}[H(\mathbf{B})]$ is computed as follows:

$$\mathbb{E}[H(\mathbf{B})] = \sum_{k \in [|S|]} \left(\sum_{i \in [n]} b_i \cdot d_{i,k} \right) \cdot H(\mathbf{B}_k).$$

The statement in Proposition 2 says that *on average* the entropy will decrease monotonically, and thus on an infinite run we will not lose any information about the true environment. However, this proposition cannot guarantee any convergence towards a Dirac distribution on the true environment, as it is possible to get stuck in a belief where the expected entropy of all successor beliefs is the same as the entropy of the current belief.

Formally speaking, $\mathbb{E}[H(\mathbf{B})]$ denotes the *expected entropy* over $\mathbf{B}$, where $\mathbf{B}$ is the vector of all successor belief states given the current belief state $\mathbf{b}$. For simplicity, we will refer to $\mathbb{E}[H(\mathbf{B})]$ just as *the expected entropy* when $\mathbf{B}$ is clear from the context.

It should be noted that Proposition 2 only holds for MEMDPs, and not arbitrary POMDPs.

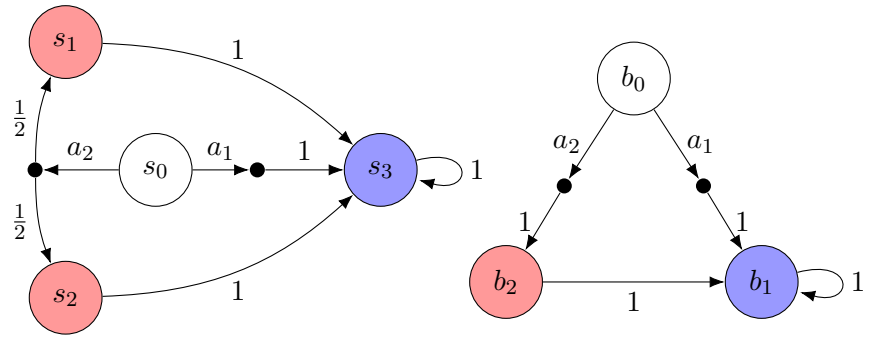
Lemma 4

The statement in Proposition 2 only holds in MEMDPs, and not in arbitrary POMDPs.

Proof by counter example. Consider the POMDP in Figure 14. This is the same POMDP and its belief MDP we used as an example in Section 4.3.2. The belief states are given by

$$b_0 = \{s_0 \mapsto 1\}, \quad b_1 = \{s_3 \mapsto 1\}, \quad b_2 = \left\{s_1 \mapsto \frac{1}{2}, s_2 \mapsto \frac{1}{2}\right\}.$$

These distributions have the following entropies: $H(b_0) = 0$, $H(b_1) = 0$, and $H(b_2) = 1$.



(a) The POMDP. (b) The associated belief MDP.

Figure 14: A POMDP where the expected entropy is not monotonic.

The initial belief is b_0. The expected entropy is the sum of all entropies of the successor beliefs, weighted by the probability of observing that successor belief. When taking action a_1 we can only end up in belief state b_1 with probability 1. So the expected entropy of all successor beliefs is in this case just the entropy of b_1, which is also 0. So for a_1 it indeed holds that

$$\mathbb{E}[H(\mathbf{B})] = \mathbb{E}[H(b_1)] = H(b_1) = 0 = H(b_0).$$

However, when choosing action a_2 the following happens. With the same reasoning, we conclude that the expected entropy is just the entropy of b_2. Thus, we get

$$\mathbb{E}[H(\mathbf{B})] = \mathbb{E}[H(b_2)] = H(b_2) = 1 \not\leq 0 = H(b_0).$$

Thus, the statement in Proposition 2 does not hold for general POMDPs.

□

We implement the setting described in Figure 13 in Python to get a sense of the behavior under various cases. In particular, we compute the expected entropy and compare it to the entropy of the current belief state. The code is available at `https://gitlab.science.ru.nl/msuilen/memdp-entropy-explorer`, and only serves to get a sense of what is going on.

We consider three cases and describe the observed behavior of our decision making setting in each case.

1. If the current belief is a Dirac distribution, and the distributions over successor states are randomly generated, the expected entropy is *equal to* 0.

2. If the distribution over successor states is the same for every environment, and the current belief is an arbitrary non-Dirac distribution, the expected entropy is *the same* as the current belief entropy.

3. If the current belief is any non-Dirac distribution, and the distributions over successor states are different for at least two environments, the expected entropy is *strictly less* than the current belief entropy.

These three observed cases can be interpreted as follows.

- In 1. we already have *maximum knowledge* of the true environment, as the current belief is already a Dirac distribution, so our *knowledge cannot increase* any further.

- In 2. the distributions over successor states are the same for all environments, which means we *cannot distinguish* between environments, and thus *cannot learn* anything.

- In 3. there is still room to learn, as we do not have a Dirac distribution yet, and there is also *opportunity to learn*, as the distributions over successor states are different, and thus we are also *expected to learn*.

We formalize cases 1. and 2. as theorems, and case 3. as a conjecture in Section 6.3.2. Together, they cover all possible cases that can occur in our setting, and combined arrive at the statement made in Proposition 2.

6.3 Theorems on the expected entropy

Before formalizing and proving the theorems, we introduce a number of lemmas that show how the entropy and the expected entropy can be rewritten in such a way that the sums on the outside of the logarithm become products inside the logarithm.

6.3.1 Two helpful lemmas

In our proofs we will use the following standard facts for logarithms:

$$\log_2(x \cdot y) = \log_2(x) + \log_2(y) \qquad \text{(Product rule)}$$

$$\log_2\left(\frac{x}{y}\right) = \log_2(x) - \log_2(y) \qquad \text{(Quotient rule)}$$

$$\log_2(x^y) = y \cdot \log_2(x) \qquad \text{(Power rule)}$$

$$x < y \implies \log_2(x) < \log_2(y) \qquad \text{(Strict monotonicity)}$$

as well as the standard rule $x^y \cdot x^z = x^{y+z}$ for any $x, y, z \in \mathbb{R}$. Now recall the definition of the belief update in MEMDPs (Definition 25), modified to our short-hand notation:

$$B_{k,i} = \frac{b_i \cdot d_{i,k}}{\sum_{j \in [n]} b_j \cdot d_{j,k}},$$

the definition of the Shannon entropy (Definition 26), also adapted to the new notation we introduced:

$$H(\mathbf{b}) = \sum_{i \in [n]} b_i \cdot \log_n\left(\frac{1}{b_i}\right), \qquad H(\mathbf{B}_k) = \sum_{i \in [n]} B_{k,i} \cdot \log_n\left(\frac{1}{B_{k,i}}\right),$$

and the definition of the expected entropy of $\mathbf{B}$ that follows:

$$\mathbb{E}[H(\mathbf{B})] = \sum_{k \in [|S|]} \left(\sum_{i \in [n]} b_i \cdot d_{i,k}\right) \cdot H(\mathbf{B}_k).$$

The first lemma is a simple rewriting of the definition of the entropy.

Lemma 5

Let $\mu \in Distr(X)$ be an arbitrary probability distribution over X with $|X| = n$. The entropy of μ can be rewritten as

$$H(\mu) = \log_n \left(\prod_{x \in X} \mu(x)^{-\mu(x)} \right).$$

Proof. Expand the definition of the entropy and rewrite using the power and product rules for logarithms.

$$\begin{aligned} H(\mu) &= \sum_{x \in X} \mu(x) \cdot \log_n \left(\frac{1}{\mu(x)} \right) \\ &= \sum_{x \in X} \log_n \left(\mu(x)^{-\mu(x)} \right) && \text{(by power rule)} \\ &= \log_n \left(\prod_{x \in X} \mu(x)^{-\mu(x)} \right). && \text{(by product rule)} \end{aligned}$$

□

The next lemma is a rewriting of the expected entropy of the successor beliefs.

Lemma 6

Let $\mathbf{B} = (\mathbf{B}_1, \ldots, \mathbf{B}_{|S|})$ be the vector of successor belief states. Let $N_k = \sum_{l \in [n]} b_l \cdot d_{l,k}$. Then

$$\mathbb{E}[H(\mathbf{B})] = \log_n \left(\prod_{k \in [|S|]} \prod_{j \in [n]} \left(\frac{b_j \cdot d_{j,k}}{N_k} \right)^{-b_j \cdot d_{j,k}} \right).$$

Proof. First, we show that

$$\mathbb{E}[H(\mathbf{B})] = \log_n \left(\prod_{k \in [|S|]} \prod_{i \in [n]} \prod_{j \in [n]} B_{k,j}^{-B_{k,j} \cdot d_{i,k} \cdot b_i} \right)$$

by rewriting using Lemma 5, the power rule, and the product rule:

$$\begin{aligned}
\mathbb{E}[H(\mathbf{B})] &= \sum_{k\in[|S|]} \left(\sum_{i\in[n]} d_{i,k} \cdot b_i \right) \cdot H(\mathbf{B}_k) \\
&= \sum_{k\in[|S|]} \sum_{i\in[n]} d_{i,k} \cdot b_i \cdot \log_n \left(\prod_{i\in[n]} B_{k,i}^{-B_{k,i}} \right) && \text{(by Lemma 5)} \\
&= \sum_{k\in[|S|]} \sum_{i\in[n]} \log_n \left(\prod_{j\in[n]} B_{k,j}^{-B_{k,j}\cdot d_{i,k}\cdot b_i} \right) && \text{(by power rule)} \\
&= \log_n \left(\prod_{k\in[|S|]} \prod_{i\in[n]} \prod_{j\in[n]} B_{k,j}^{-B_{k,j}\cdot d_{i,k}\cdot b_i} \right). && \text{(by product rule)}
\end{aligned}$$

Now, we show that

$$\begin{aligned}
&\log_n \left(\prod_{k\in[|S|]} \prod_{i\in[n]} \prod_{j\in[n]} B_{k,j}^{-B_{k,j}\cdot d_{i,k}\cdot b_i} \right) \\
&\qquad = \log_n \left(\prod_{k\in[|S|]} \prod_{j\in[n]} \left(\frac{b_j \cdot d_{j,k}}{N_k} \right)^{-b_j \cdot d_{j,k}} \right).
\end{aligned}$$

We use that index i only occurs in the exponents, that $N_k = \sum_{l\in[n]} b_l \cdot d_{l,k}$ and that

$$B_{k,j} = \frac{b_j \cdot d_{j,k}}{\sum_{l\in[n]} b_l \cdot d_{l,k}} = \frac{b_j \cdot d_{j,k}}{N_k},$$

and start rewriting:

$$\begin{aligned}
&\log_n \left(\prod_{k\in[|S|]} \prod_{i\in[n]} \prod_{j\in[n]} B_{k,j}^{-B_{k,j}\cdot d_{i,k}\cdot b_i} \right) \\
&= \log_n \left(\prod_{k\in[|S|]} \prod_{j\in[n]} B_{k,j}^{-B_{k,j}\cdot \sum_{i\in[n]} d_{i,k}\cdot b_i} \right) && \text{(use } x^y \cdot x^z = x^{y+z}\text{)} \\
&= \log_n \left(\prod_{k\in[|S|]} \prod_{j\in[n]} \left(\frac{b_j \cdot d_{j,k}}{N_k} \right)^{\frac{-b_j\cdot d_{j,k}}{N_k}\cdot N_k} \right) && \text{(rewrite } B_{k,j}\text{)} \\
&= \log_n \left(\prod_{k\in[|S|]} \prod_{j\in[n]} \left(\frac{b_j \cdot d_{j,k}}{N_k} \right)^{-b_j\cdot d_{j,k}} \right).
\end{aligned}$$

Note that by the commutativity of the product on real numbers, we may change the order of the products over j and k into whatever order we prefer.

□

6.3.2 The theorems on the expected entropy

We now formalize our findings from Section 6.2.1.

> **Theorem 3: Expected entropy on Dirac beliefs**
>
> Suppose the belief at state s is a Dirac distribution for the first environment, i.e. $\mathbf{b} = (1, 0, \ldots, 0)$, and let the distributions over successor states $\mathbf{d_1}, \ldots, \mathbf{d_n}$ be arbitrary. Then the expected entropy is equal to the entropy of the current belief:
>
> $$\mathbb{E}[H(\mathbf{B})] = H(\mathbf{b}) = 0.$$

Proof. First, note that since $\mathbf{b}$ is Dirac, we have $H(\mathbf{b}) = 0$.

We expand the definition of the expected entropy.

$$\mathbb{E}[H(\mathbf{B})] = \sum_{k \in [|S|]} \left(\sum_{i \in [n]} d_{i,k} \cdot b_i \right) \cdot H(\mathbf{B}_k).$$

By definition of $\mathbf{b}$ we have that $b_1 = 1$ and $b_i = 0$ for all $i \neq 1$. Thus we simplify

$$\begin{aligned} \mathbb{E}[H(\mathbf{B})] &= \sum_{k \in [|S|]} \left(\sum_{i \in [n]} d_{i,k} \cdot b_i \right) \cdot H(\mathbf{B}_k) \\ &= \sum_{k \in [|S|]} \left(d_{1,k} \cdot b_1 \cdot H(\mathbf{B}_k) \right). \qquad \text{(use } b_j = 0 \text{ for } j \geq 2) \end{aligned}$$

We know that applying the belief update on a Dirac belief returns that same Dirac belief (Lemma 3), so we have that $\mathbf{B}_k$ is a Dirac distribution, for all k. We know that the entropy of a Dirac distribution is 0. Thus we get:

$$\sum_{k\in[|S|]} (d_{1,k} \cdot b_1 \cdot H(\mathbf{B}_k))$$
$$= \sum_{k\in[|S|]} (d_{1,k} \cdot b_1 \cdot 0)$$
$$= \sum_{k\in[|S|]} 0 = 0.$$

We have $\mathbb{E}[H(\mathbf{B})] = 0 = H(\mathbf{b})$ and we conclude that once we have reached a Dirac distribution over the environments we are also expected to remain in a Dirac distribution.

□

The statement in Theorem 3 assumes the belief $\mathbf{b}$ is Dirac in the first environment, but this is an assumption we can make without loss of generality. From the proof it clearly follows the statement also holds when $\mathbf{b}$ is Dirac in any other environment. Furthermore, because a MEMDP is just a set in which ordering of the environments does not matter, we may reorder the environments in such a way that the environment in which $\mathbf{b}$ is Dirac becomes the first environment.

Theorem 4: Constant expected entropy

Let the belief state $\mathbf{b} = (b_1, \ldots, b_n)$ at state s be arbitrary. Suppose for all environments the distributions over successor states are the same. Thus, $\mathbf{d}_1 = \cdots = \mathbf{d}_n$. Then the expected belief entropy does not change from the current entropy:

$$\mathbb{E}[H(\mathbf{B})] = H(\mathbf{b}).$$

Proof. First note that the distributions over successor states are all equal, thus $\forall i, j \in [n]$. $d_{i,k} = d_{j,k}$, which means we can simplify to a single distribution over successor states $\mathbf{d} = (d_1 \ldots, d_{|S|})$. The belief update is then also simplified:

$$
\begin{aligned}
B_{k,i} &= \frac{b_i \cdot d_k}{\sum_{j \in [n]} b_j \cdot d_k} \\
&= \frac{b_i \cdot d_k}{d_k \cdot \sum_{j \in [n]} b_j} \\
&= \frac{b_i}{\sum_{j \in [n]} b_j} \\
&= b_i.
\end{aligned}
$$

The entropy of $\mathbf{b}$ can be rewritten by Lemma 5 into

$$
H(\mathbf{b}) = \sum_{i \in [n]} b_i \cdot \log_n \left(\frac{1}{b_i} \right) = \log_n \left(\prod_{i \in [n]} b_i^{-b_i} \right)
$$

The expected entropy of $\mathbf{B}$ is given by

$$
\begin{aligned}
\mathbb{E}[H((B))] &= \sum_{k \in [|S|]} \left(\sum_{i \in [n]} d_{i,k} \cdot b_i \right) \cdot H(\mathbf{B}_k) \\
&= \sum_{k \in [|S|]} \left(\sum_{i \in [n]} d_k \cdot b_i \right) \cdot H(\mathbf{B}_k) \\
&= \sum_{k \in [|S|]} d_k \cdot \left(\sum_{i \in [n]} b_i \right) \cdot H(\mathbf{B}_k) \\
&= \sum_{k \in [|S|]} d_k \cdot H(\mathbf{B}_k).
\end{aligned}
$$

Now we apply Lemma 5 on $\mathbf{B}_k$ and use the simplified belief update to get to $H(\mathbf{b})$.

$$
\begin{aligned}
&\sum_{k\in[|S|]} d_k \cdot H(\mathbf{B}_k) \\
&= \sum_{k\in[|S|]} d_k \cdot \log_n \left(\prod_{i\in[n]} B_{k,i}^{-B_{k,i}} \right) && \text{(by Lemma 5)} \\
&= \sum_{k\in[|S|]} d_k \cdot \log_n \left(\prod_{i\in[n]} b_i^{-b_i} \right) && \text{(by simplified belief update)} \\
&= \log_n \left(\prod_{i\in[n]} b_i^{-b_i} \right) \cdot \sum_{k\in[|S|]} d_k \\
&= \log_n \left(\prod_{i\in[n]} b_i^{-b_i} \right) && \text{(as } \mathbf{d} \text{ is a probability distribution)} \\
&= H(\mathbf{b}).
\end{aligned}
$$

□

The third case we observed, that says that if there is opportunity to learn, we are also expected to learn, is left as a conjecture.

> **Conjecture 1: Strict decrease of expected entropy**
>
> Let $\mathbf{b}$ be an arbitrary non-Dirac belief state. For the successor distributions, let $\mathbf{d}_1 \neq \mathbf{d}_2$, and $\mathbf{d}_k$ arbitrary for $k = 3, \ldots, |S|$. Then
>
> $$\mathbb{E}[H(\mathbf{B})] < H(\mathbf{b})$$

6.4 Decision making

In Section 6.2 the setting in which we investigated the behavior of the expected entropy of the successor beliefs for an observed state and a given action. Now, we wish to use our findings to choose between the available actions at a given state.

This new setting is described in Figure 15. Now we have $l = |A(s)|$ available actions to choose from, and each action gives us a distribution over successor states for every environment, denoted $\mathbf{d}_i^{a_j} = (d_{i,1}^{a_j}, \ldots, d_{i,|S|}^{a_j})$.

The problem is now to figure out which action allows us to learn about the true environment, given we currently observe state s.

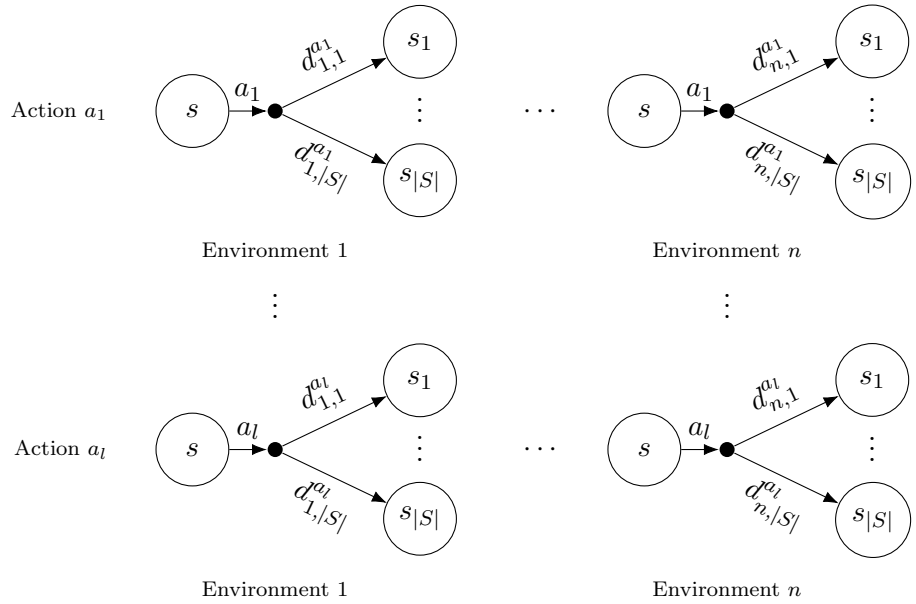


Figure 15: The setting in which decision making takes place.

We start with a (naive) greedy algorithm, given in Algorithm 4, and discuss a number of problems with this naive approach.

The basic idea is to compute the expected entropy for every available action and pick the action that gives the biggest decrease. If all actions have the same expected entropy, an action is chosen at random uniformly. It may be the case that this action leads us to a successor state in which we have the same situation. Looking further ahead, however, is computationally expensive, as it requires unfolding the belief MDP up to an arbitrary number of steps. There is no way to predict how many steps are needed, hence the choice for the possibly less optimal but computationally preferred option of just picking an action at random.

One problem in this algorithm is the repeated computation of the expected entropy. In order to compute $\mathbb{E}[H(\mathbf{B}^a)]$, the following computations are performed:

1. The vector $\mathbf{B}^a$, which requires $\mathcal{O}(|S|)$ belief updates. Each belief update has a complexity of $\mathcal{O}(n)$, yielding a total complexity of $\mathcal{O}(n \cdot |S|)$ for the vector $\mathbf{B}^a$.

2. The expected entropy $\mathbb{E}[H(\mathbf{B}^a)]$, which has a complexity of $\mathcal{O}(n \cdot |S|)$.

Algorithm 4: Naive greedy learning

Input: MEMDP $M = (S, A, \{\delta_i\}_{i\in[n]})$, initial belief $\mathbf{b}_0$ observing initial state s_I

```
s ← s_I
b ← b_0
CurrentEntropy ← H(b)
while CurrentEntropy > 0 do
    BestAction ← Random(A(s))
    BestExpectedEntropy ← CurrentEntropy
    for a ∈ A(s) do
        Compute B^a
        Compute 𝔼[H(B^a)]
        if 𝔼[H(B^a)] < BestExpectedEntropy then
            BestAction ← a
            BestExpectedEntropy ← 𝔼[H(B^a)]
        end
    end
    Perform action BestAction
    s ← observed successor state
    b ← belief update
    CurrentEntropy ← H(b)
end
```

The computations above are repeated for every action a, hence, every iteration of the while loop has a worst-case complexity of $\mathcal{O}(n \cdot |S| \cdot |A|)$.

Since the belief update relies on the current belief $\mathbf{b}$, which in the worst case (when the belief MDP is infinite) is unique every iteration, we have to perform the entire computation at every iteration for every available action.

Another problem is that we may get stuck in a maximal end-component (recall Definition 14) in which the distributions over successor states are all the same for every environment, rendering us unable to learn anything by Theorem 4. It should be noted that by definition of an end-component, it may still possible to leave the end-component.

We can detect that we are stuck in an end-component (S', A') that cannot be left, simply by checking the following conditions:

1. The current state s is in S',
2. There is no action that can take us out of the end-component: for all $s' \in S'$ we have $A(s') = A'$.

3. We cannot learn inside the end-component. By Theorem 4, the distributions over successor states are the same for every state and action of the end-component: for all $(s, a) \in S' \times A'$ we have $\delta_i(s, a) = \delta_j(s, a)$ for all environments $i, j \in [n]$.

When stuck, our only option is to reset and start the learning again from the initial state. From a theoretical perspective this is fine, and we can even bring our last known belief $\mathbf{b}$ as new initial belief. In practice, however, it may be impossible to reset the system. See for example the bird preservation case study from [Chadès et al., 2012].

We can preprocess a MEMDP M to get all states for which we are stuck in an end-component in which we cannot learn. We denote these states by the boolean vector **Stuck** with **Stuck**$[s] =$ `True` if and only if there is an end-component (S', A') with $s \in S'$ for which the learning algorithm is stuck.

6.5 Optimizing by preprocessing

Besides the statement in Conjecture 1, we observe something else in our implementation of the setting of Figure 13. Whenever the distributions over the successor states look to be further apart, the expected entropy decreases at a greater rate than when the distributions over the successor states are closer to each other. We will make this more precise using the Bhattacharyya distance [Bhattacharyya, 1943].

The Bhattacharyya distance is a distance measure between two probability distributions. It is straightforward to compute and easily extends to the more general case where we wish to compute the difference between any arbitrary number of probability distributions.

Definition 27: Bhattacharyya coefficient

Given two discrete probability distributions μ_1, μ_2 over the same finite set X, the *Bhattacharyya coefficient* [Bhattacharyya, 1943] BC of μ_1 and μ_2 is given by

$$BC(\mu_1, \mu_2) = \sum_{x \in X} \sqrt{\mu_1(x) \cdot \mu_2(x)}.$$

The Bhattacharyya coefficient can be adapted to an n-dimensional version [Kang and Wildes, 2015]. The n-dimensional Bhattacharyya coefficient of discrete probability distributions $\mu_1, \ldots, \mu_n$ over the

same finite set X is defined as

$$BC(\mu_1, \ldots, \mu_n) = \sum_{x \in X} \sqrt[n]{\prod_{i=1}^{n} \mu_i(x)}.$$

The Bhattacharyya coefficient measures the amount of overlap between the probability distributions. The coefficient is minimal and equal to 0 if there is no overlap between the probability distributions, and maximal and equal to 1 if all probability distributions are the same. These properties are also met by the n-dimensional variant.

The Bhattacharyya coefficient can be converted into the Bhattacharyya distance by taking the negative logarithm of the coefficient.

Definition 28: Bhattacharyya distance

The Bhattacharyya distance if defined as

$$D_B(\mu_1, \ldots, \mu_n) = -\ln\left(BC(\mu_1, \ldots, \mu_n)\right).$$

It follows that the Bhattacharyya distance then lies between 0 and ∞, being 0 in the case all distributions are the same, and ∞ if there is no overlap at all, thus, being at maximal distance from each other. The latter is for example the case when all distributions are Dirac in a different element of the set X.

We extends our Python implementation to account for the decision making setting in Figure 15. Again, this implementation only serves to get a sense of the behavior of our decision making setting under various inputs. We use the Bhattacharyya distance to measure the distance between the distributions over the successor states for every action. We formalize the observed findings in Conjecture 2

Conjecture 2

Given two different actions a_1, a_2 at a state s, if

$$D_B(\mathbf{d}_1^{a_1}, \ldots, \mathbf{d}_n^{a_1}) > D_B(\mathbf{d}_1^{a_2}, \ldots, \mathbf{d}_n^{a_2})$$

then

$$\mathbb{E}[H(\mathbf{B}^{a_1})] < \mathbb{E}[H(\mathbf{B}^{a_2})]$$

Conjecture 2 naturally extends to the more general case where we compare more than two actions.

The advantage of the Bhattacharyya distance, in combination with Conjecture 2, is that the distributions over successor states remain fixed for every iteration of Algorithm 4. Hence, instead of computing the expected entropy for all actions over and over again, we can compute all the Bhattacharyya distances once, as a preprocessing step. We store the distance between environments at a given state-action pair (s, a) in the $|S| \times |A|$ matrix $\mathbf{D}$ with $\mathbf{D}[s, a] = D_B(\delta_1(s, a), \ldots, \delta_n(s, a))$.

The complexity per iteration of the while loop is now a single belief update and a single entropy computation, both of $\mathcal{O}(n)$, and a single loop over all actions. Assuming we use an efficient data structure with $\mathcal{O}(1)$ access to store and read the distances, the total complexity per iteration is $\mathcal{O}(\max\{n, |A|\})$. The optimized algorithm is given in Algorithm 5.

Algorithm 5: Optimized greedy learning

Input: MEMDP $M = (S, A, \{\delta_i\}_{i\in[n]})$, initial belief $\mathbf{b}_0$ observing initial state s_I, distance matrix $\mathbf{D}$, vector **Stuck**

```
s ← s_I
b ← b_0
CurrentEntropy ← H(b)
while CurrentEntropy > 0 do
    BestAction ← Random(A(s))
    for a ∈ A(s) do
        if D[s, a] > D[s, BestAction] then
            BestAction ← a
        end
    end
    Perform action BestAction
    s ← observed successor state
    b ← belief update
    CurrentEntropy ← H(b)
    if Stuck[s] then
        Restart with initial belief b_0 = b observing s_I = s
    end
end
```

Implementing Algorithm 5 to get experimental results is left for future work.

Chapter 7

Conclusions & future work

We presented a method for inferring the true environment in a MEMDP via learning.

By mapping the MEMDP to a POMDP, we obtained the notion of a belief over environments. We investigated how this belief behaves over time, and applied the Shannon entropy to measure how much knowledge on the true environment the belief at any given moment provides. We showed how the expected entropy can be used to measure the change in knowledge on average, and note under which conditions the expected entropy stays the same, and under which conditions there is a strict increase in knowledge on average.

We used this to construct an iterative learning algorithm. At the core of the learning algorithm is the idea that we can determine which action we expect to give the biggest increase in knowledge on the true environment. We optimized the algorithm by preprocessing, and discussed under which conditions the algorithm can get stuck and requires a restart.

Additionally, this thesis provides a literature study into MDPs, POMDPs, MEMDPs and a number of related models. We defined bisimulations between MEMDPs, proved that the quantitative expected reward problem is NP-hard, and defined a sound overapproximation method for computing robust strategies for MEMDPs via uncertain MDPs. We also defined a mixed integer linear program that computed deterministic strategies for one or more objectives in a MEMDP.

There is, however, plenty of work left to be done, both on learning the true environment and other topics within MEMDPs. We end this thesis with some pointers for directions future work might take.

Future work

This thesis provides a theoretical treatment of the problem of learning the true environment in MEMDPs. As such, the algorithm we introduced in Chapter 6 should be implemented to get experimental results on the performance in practice.

Furthermore, we have two conjectures that have no formal proof as of yet. Perhaps a proof assistant, such as e.g. `Lean` [de Moura et al., 2015], can help out. From the other theorems and lemmas it seems likely that the proofs for Conjecture 1 and Conjecture 2 rely entirely on rewriting using standard rules on real numbers and logarithms, with the main problem being the high number of variables to account for. Alternatively, one could look at standard inequalities on the entropy, such as Gibbs' inequality [Brémaud, 2012], or Jensen's inequality [Abramovich, Jameson, and Sinnamon, 2004].

There are several directions future work on learning the true environment of a MEMDP can take. Our approach implicitly assumes that it is possible to safely explore the model. That is, there are no bad states that should be avoided. This may not be the case. As such combining the learning while also satisfying another objective, such as a safety objective should be explored.

The learning algorithm we introduce is real-time, it makes a deterministic choice between the actions given the current state. An alternative would be to compute a fixed strategy (possibly randomized and with finite memory) that satisfies some kind of learning objective. We may also wonder how much an arbitrary given strategy learns on average.

Finally, in [Araya et al., 2010] POMDPs where the reward function depends on the current belief state instead of the actual state are introduced. This is especially interesting when combined with the POMDPs we derive from MEMDPs, as the belief states there are probability distributions over the environments. Hence, one could define a belief dependent reward function that changes the reward based on how much knowledge we have about the true environment.

Bibliography

[Abramovich, Jameson, and Sinnamon, 2004] Abramovich, S.; Jameson, G.; and Sinnamon, G. 2004. Refining jensen's inequality. *Bulletin mathématique de la Société des Sciences Mathématiques de Roumanie* 3–14.

[Akshay et al., 2019] Akshay, S.; Bazille, H.; Fabre, E.; and Genest, B. 2019. Classification among hidden Markov models. In *FSTTCS 2019 - 39th IARCS Annual Conference on. Foundations of Software Technology and Theoretical Computer Science*, volume 150, 1–14. LIPIcs.

[Alur, Courcoubetis, and Yannakakis, 1995] Alur, R.; Courcoubetis, C.; and Yannakakis, M. 1995. Distinguishing tests for nondeterministic and probabilistic machines. In *Proceedings of the twenty-seventh annual ACM symposium on Theory of computing*, 363–372.

[Araya et al., 2010] Araya, M.; Buffet, O.; Thomas, V.; and Charpillet, F. 2010. A POMDP extension with belief-dependent rewards. In *Advances in neural information processing systems*, 64–72.

[Araya-López et al., 2010] Araya-López, M.; Thomas, V.; Buffet, O.; and Charpillet, F. 2010. A closer look at MOMDPs. In *22nd IEEE International Conference on Tools with Artificial Intelligence, ICTAI 2010, Arras, France, 27-29 October 2010 - Volume 2*, 197–204. IEEE Computer Society.

[Arming et al., 2018] Arming, S.; Bartocci, E.; Chatterjee, K.; Katoen, J.-P.; and Sokolova, A. 2018. Parameter-independent strategies for pmdps via pomdps. In *International Conference on Quantitative Evaluation of Systems*, 53–70. Springer.

[Arora and Barak, 2009] Arora, S., and Barak, B. 2009. *Computational complexity: a modern approach*. Cambridge University Press.

[Bagnell, Ng, and Schneider, 2001] Bagnell, J. A.; Ng, A. Y.; and Schneider, J. G. 2001. *Solving uncertain Markov decision processes*. Carnegie Mellon University.

[Baier and Katoen, 2008] Baier, C., and Katoen, J.-P. 2008. *Principles of model checking*. The MIT Press.

[Bellman, 2003] Bellman, R. E. 2003. *Dynamic Programming*. USA: Dover Publications, Inc.

[Ben-Tal, El Ghaoui, and Nemirovski, 2009] Ben-Tal, A.; El Ghaoui, L.; and Nemirovski, A. 2009. *Robust optimization*, volume 28. Princeton University Press.

[Bhattacharyya, 1943] Bhattacharyya, A. 1943. On a measure of divergence between two statistical populations defined by their probability distributions. *Bull. Calcutta Math. Soc.* 35:99–109.

[Boutilier and Poole, 1996] Boutilier, C., and Poole, D. 1996. Computing optimal policies for partially observable decision processes using compact representations. In Clancey, W. J., and Weld, D. S., eds., *Proceedings of the Thirteenth National Conference on Artificial Intelligence and Eighth Innovative Applications of Artificial Intelligence Conference, AAAI 96, IAAI 96, Portland, Oregon, USA, August 4-8, 1996, Volume 2*, 1168–1175. AAAI Press / The MIT Press.

[Brémaud, 2012] Brémaud, P. 2012. *An introduction to probabilistic modeling*. Springer Science & Business Media.

[Cassandra, Kaelbling, and Littman, 1994] Cassandra, A. R.; Kaelbling, L. P.; and Littman, M. L. 1994. Acting optimally in partially observable stochastic domains. In *Aaai*, volume 94, 1023–1028.

[Cassandra, Littman, and Zhang, 1997] Cassandra, A. R.; Littman, M. L.; and Zhang, N. L. 1997. Incremental pruning: A simple, fast, exact method for partially observable Markov decision processes. In Geiger, D., and Shenoy, P. P., eds., *UAI '97: Proceedings of the Thirteenth Conference on Uncertainty in Artificial Intelligence, Brown University, Providence, Rhode Island, USA, August 1-3, 1997*, 54–61. Morgan Kaufmann.

[Cassandra, 1998] Cassandra, A. R. 1998. *Exact and approximate algorithms for partially observable Markov decision processes*. Brown University.

[Castro, Panangaden, and Precup, 2009] Castro, P. S.; Panangaden, P.; and Precup, D. 2009. Equivalence relations in fully and partially observable Markov decision processes. In *Twenty-First International Joint Conference on Artificial Intelligence*.

[Chadès et al., 2012] Chadès, I.; Carwardine, J.; Martin, T. G.; Nicol, S.; Sabbadin, R.; and Buffet, O. 2012. MOMDPs: a solution for

modelling adaptive management problems. In *Twenty-Sixth AAAI Conference on Artificial Intelligence*.

[Chatterjee et al., 2016] Chatterjee, K.; Chmelík, M.; Gupta, R.; and Kanodia, A. 2016. Optimal cost almost-sure reachability in POMDPs. *Artificial Intelligence* 234:26–48.

[Chatterjee et al., 2020] Chatterjee, K.; Chmelík, M.; Karkhanis, D.; Novotnỳ, P.; and Royer, A. 2020. Multiple-environment Markov decision processes: Efficient analysis and applications. In *Proceedings of the International Conference on Automated Planning and Scheduling*, volume 30, 48–56.

[Chatterjee, Chmelík, and Tracol, 2016] Chatterjee, K.; Chmelík, M.; and Tracol, M. 2016. What is decidable about partially observable Markov decision processes with ω-regular objectives. *Journal of Computer and System Sciences* 82(5):878 – 911.

[Chatterjee, De Alfaro, and Henzinger, 2004] Chatterjee, K.; De Alfaro, L.; and Henzinger, T. A. 2004. Trading memory for randomness. In *First International Conference on the Quantitative Evaluation of Systems, 2004. QEST 2004. Proceedings.*, 206–217. IEEE.

[Chatterjee, Doyen, and Henzinger, 2010] Chatterjee, K.; Doyen, L.; and Henzinger, T. A. 2010. Qualitative analysis of partially-observable Markov decision processes. In *International Symposium on Mathematical Foundations of Computer Science*, 258–269. Springer.

[Chatterjee, Jurdziński, and Henzinger, 2004] Chatterjee, K.; Jurdziński, M.; and Henzinger, T. A. 2004. Quantitative stochastic parity games. In *Proceedings of the fifteenth annual ACM-SIAM symposium on Discrete algorithms*, 121–130. Citeseer.

[Cohen and Parmentier, 2018] Cohen, V., and Parmentier, A. 2018. Linear programming for decision processes with partial information. *arXiv preprint arXiv:1811.08880*.

[Cormen et al., 2009] Cormen, T. H.; Leiserson, C. E.; Rivest, R. L.; and Stein, C. 2009. *Introduction to algorithms*. MIT press.

[Courcoubetis and Yannakakis, 1995] Courcoubetis, C., and Yannakakis, M. 1995. The complexity of probabilistic verification. *Journal of the ACM (JACM)* 42(4):857–907.

[Csiszár and Shields, 2004] Csiszár, I., and Shields, P. C. 2004. *Information theory and statistics: A tutorial*. Now Publishers Inc.

[Cubuktepe et al., 2018] Cubuktepe, M.; Jansen, N.; Junges, S.; Katoen, J.; and Topcu, U. 2018. Synthesis in pMDPs: A tale of 1001

parameters. In Lahiri, S. K., and Wang, C., eds., *Automated Technology for Verification and Analysis - 16th International Symposium, ATVA 2018, Los Angeles, CA, USA, October 7-10, 2018, Proceedings*, volume 11138 of *Lecture Notes in Computer Science*, 160–176. Springer.

[De Alfaro, 1997] De Alfaro, L. 1997. *Formal verification of probabilistic systems*. Stanford university.

[de Moura et al., 2015] de Moura, L.; Kong, S.; Avigad, J.; Van Doorn, F.; and von Raumer, J. 2015. The lean theorem prover (system description). In *International Conference on Automated Deduction*, 378–388. Springer.

[Dehnert et al., 2015] Dehnert, C.; Junges, S.; Jansen, N.; Corzilius, F.; Volk, M.; Bruintjes, H.; Katoen, J.; and Ábrahám, E. 2015. Prophesy: A probabilistic parameter synthesis tool. In Kroening, D., and Pasareanu, C. S., eds., *Computer Aided Verification - 27th International Conference, CAV 2015, San Francisco, CA, USA, July 18-24, 2015, Proceedings, Part I*, volume 9206 of *Lecture Notes in Computer Science*, 214–231. Springer.

[Dehnert et al., 2017] Dehnert, C.; Junges, S.; Katoen, J.; and Volk, M. 2017. A storm is coming: A modern probabilistic model checker. In Majumdar, R., and Kuncak, V., eds., *Computer Aided Verification - 29th International Conference, CAV 2017, Heidelberg, Germany, July 24-28, 2017, Proceedings, Part II*, volume 10427 of *Lecture Notes in Computer Science*, 592–600. Springer.

[Fulton and Platzer, 2019] Fulton, N., and Platzer, A. 2019. Verifiably safe off-model reinforcement learning. In *International Conference on Tools and Algorithms for the Construction and Analysis of Systems*, 413–430. Springer.

[Garcıa and Fernández, 2015] Garcıa, J., and Fernández, F. 2015. A comprehensive survey on safe reinforcement learning. *Journal of Machine Learning Research* 16(1):1437–1480.

[Gray, 2011] Gray, R. M. 2011. *Entropy and information theory*. Springer Science & Business Media.

[Grinstead and Snell, 2012] Grinstead, C. M., and Snell, J. L. 2012. *Introduction to probability*. American Mathematical Soc.

[Hahn et al., 2010] Hahn, E. M.; Hermanns, H.; Wachter, B.; and Zhang, L. 2010. Param: A model checker for parametric Markov models. In *International Conference on Computer Aided Verification*, 660–664. Springer.

[Hansson and Jonsson, 1994] Hansson, H., and Jonsson, B. 1994. A logic for reasoning about time and reliability. *Formal aspects of computing* 6(5):512–535.

[Jacobs, Westerbaan, and Westerbaan, 2015] Jacobs, B.; Westerbaan, B.; and Westerbaan, B. 2015. States of convex sets. In *International Conference on Foundations of Software Science and Computation Structures*, 87–101. Springer.

[Junges et al., 2018] Junges, S.; Jansen, N.; Wimmer, R.; Quatmann, T.; Winterer, L.; Katoen, J.; and Becker, B. 2018. Finite-state controllers of POMDPs using parameter synthesis. In Globerson, A., and Silva, R., eds., *Proceedings of the Thirty-Fourth Conference on Uncertainty in Artificial Intelligence, UAI 2018, Monterey, California, USA, August 6-10, 2018*, 519–529. AUAI Press.

[Junges et al., 2019] Junges, S.; Ábrahám, E.; Hensel, C.; Jansen, N.; Katoen, J.; Quatmann, T.; and Volk, M. 2019. Parameter synthesis for Markov models. *CoRR* abs/1903.07993.

[Kaelbling, Littman, and Cassandra, 1998] Kaelbling, L. P.; Littman, M. L.; and Cassandra, A. R. 1998. Planning and acting in partially observable stochastic domains. *Artificial intelligence* 101(1-2):99–134.

[Kaelbling, Littman, and Moore, 1996] Kaelbling, L. P.; Littman, M. L.; and Moore, A. W. 1996. Reinforcement learning: A survey. *Journal of artificial intelligence research* 4:237–285.

[Kang and Wildes, 2015] Kang, S. M., and Wildes, R. P. 2015. The n-distribution bhattacharyya coefficient. *EECS-2015-02*.

[Kechris, 2012] Kechris, A. 2012. *Classical descriptive set theory*, volume 156. Springer Science & Business Media.

[Kiefer and Sistla, 2016] Kiefer, S., and Sistla, A. P. 2016. Distinguishing hidden Markov chains. In *Proceedings of the 31st Annual ACM/IEEE Symposium on Logic in Computer Science*, 66–75.

[Kurniawati, Hsu, and Lee, 2008] Kurniawati, H.; Hsu, D.; and Lee, W. S. 2008. Sarsop: Efficient point-based pomdp planning by approximating optimally reachable belief spaces. In *Robotics: Science and systems*, volume 2008. Zurich, Switzerland.

[Kwiatkowska, Norman, and Parker, 2011] Kwiatkowska, M. Z.; Norman, G.; and Parker, D. 2011. PRISM 4.0: Verification of probabilistic real-time systems. In Gopalakrishnan, G., and Qadeer, S., eds., *Computer Aided Verification - 23rd International Conference,*

CAV 2011, Snowbird, UT, USA, July 14-20, 2011. Proceedings, volume 6806 of *Lecture Notes in Computer Science*, 585–591. Springer.

[Larsen and Skou, 1991] Larsen, K. G., and Skou, A. 1991. Bisimulation through probabilistic testing. *Information and Computation* 94(1):1 – 28.

[Lee and Yannakakis, 1994] Lee, D., and Yannakakis, M. 1994. Testing finite-state machines: State identification and verification. *IEEE Transactions on computers* 43(3):306–320.

[Lyngsø and Pedersen, 2002] Lyngsø, R. B., and Pedersen, C. N. 2002. The consensus string problem and the complexity of comparing hidden Markov models. *Journal of Computer and System Sciences* 65(3):545–569.

[Madani, Hanks, and Condon, 2000] Madani, O.; Hanks, S.; and Condon, A. 2000. On the undecidability of probabilistic planning and infinite-horizon partially observable Markov decision problems. *Proceedings of the National Conference on Artificial Intelligence*.

[McCallum, 1993] McCallum, R. A. 1993. Overcoming incomplete perception with utile distinction memory. In *Proceedings of the Tenth International Conference on Machine Learning*, 190–196.

[Nilim and El Ghaoui, 2005] Nilim, A., and El Ghaoui, L. 2005. Robust control of Markov decision processes with uncertain transition matrices. *Operations Research* 53(5):780–798.

[Norman, Parker, and Zou, 2017] Norman, G.; Parker, D.; and Zou, X. 2017. Verification and control of partially observable probabilistic systems. *Real-Time Systems* 53(3):354–402.

[Ong et al., 2010] Ong, S. C. W.; Png, S. W.; Hsu, D.; and Lee, W. S. 2010. Planning under uncertainty for robotic tasks with mixed observability. *The International Journal of Robotics Research* 29(8):1053–1068.

[Papadimitriou and Tsitsiklis, 1987] Papadimitriou, C. H., and Tsitsiklis, J. N. 1987. The complexity of Markov decision processes. *Mathematics of operations research* 12(3):441–450.

[Papadimitriou, 2003] Papadimitriou, C. H. 2003. *Computational complexity*. John Wiley and Sons Ltd.

[Puterman, 1994] Puterman, M. L. 1994. *Markov Decision Processes: Discrete Stochastic Dynamic Programming*. USA: John Wiley & Sons, Inc., 1st edition.

[Rabiner, 1989] Rabiner, L. R. 1989. A tutorial on hidden Markov models and selected applications in speech recognition. *Proceedings of the IEEE* 77(2):257–286.

[Raskin and Sankur, 2014] Raskin, J.-F., and Sankur, O. 2014. Multiple-environment Markov decision processes. In *FSTTCS*.

[Russell and Norvig, 2010] Russell, S. J., and Norvig, P. 2010. *Artificial Intelligence - A Modern Approach, Third International Edition*. Pearson Education.

[Savas et al., 2018] Savas, Y.; Ornik, M.; Cubuktepe, M.; and Topcu, U. 2018. Entropy maximization for constrained Markov decision processes. In *2018 56th Annual Allerton Conference on Communication, Control, and Computing (Allerton)*, 911–918. IEEE.

[Schaefer and Štefankovič, 2017] Schaefer, M., and Štefankovič, D. 2017. Fixed points, nash equilibria, and the existential theory of the reals. *Theory of Computing Systems* 60(2):172–193.

[Shannon, 2001] Shannon, C. E. 2001. A mathematical theory of communication. *ACM SIGMOBILE mobile computing and communications review* 5(1):3–55.

[Sudkamp, 1997] Sudkamp, T. A. 1997. *Languages and Machines: An Introduction to the Theory of Computer Science*. USA: Addison-Wesley Longman Publishing Co., Inc.

[Suilen et al., 2020] Suilen, M.; Jansen, N.; Cubuktepe, M.; and Topcu, U. 2020. Robust policy synthesis for uncertain POMDPs via convex optimization. In Bessiere, C., ed., *Proceedings of the Twenty-Ninth International Joint Conference on Artificial Intelligence, IJCAI 2020 [scheduled for July 2020, Yokohama, Japan, postponed due to the Corona pandemic]*, 4113–4120. ijcai.org.

[Terwijn, 2002] Terwijn, S. A. 2002. On the learnability of hidden Markov models. In *International Colloquium on Grammatical Inference*, 261–268. Springer.

[van den Bos and Vaandrager, 2019] van den Bos, P., and Vaandrager, F. 2019. State identification for labeled transition systems with inputs and outputs. In *International Conference on Formal Aspects of Component Software*, 191–212. Springer.

[Vlassis, Littman, and Barber, 2012] Vlassis, N.; Littman, M. L.; and Barber, D. 2012. On the computational complexity of stochastic controller optimization in POMDPs. *ACM Trans. Comput. Theory* 4(4).

[Wiesemann, Kuhn, and Rustem, 2013] Wiesemann, W.; Kuhn, D.; and Rustem, B. 2013. Robust Markov decision processes. *Math. Oper. Res.* 38(1):153–183.

[Winkler et al., 2019] Winkler, T.; Junges, S.; Pérez, G. A.; and Katoen, J. 2019. On the complexity of reachability in parametric Markov decision processes. In Fokkink, W., and van Glabbeek, R., eds., *30th International Conference on Concurrency Theory, CONCUR 2019, August 27-30, 2019, Amsterdam, the Netherlands*, volume 140 of *LIPIcs*, 14:1–14:17. Schloss Dagstuhl - Leibniz-Zentrum für Informatik.

[Winterer et al., 2020] Winterer, L.; Wimmer, R.; Jansen, N.; and Becker, B. 2020. Strengthening deterministic policies for POMDPs. *arXiv preprint arXiv:2007.08351*.

[Wolff, Topcu, and Murray, 2012] Wolff, E. M.; Topcu, U.; and Murray, R. M. 2012. Robust control of uncertain Markov decision processes with temporal logic specifications. In *2012 IEEE 51st IEEE Conference on Decision and Control (CDC)*, 3372–3379. IEEE.