Master thesis
Computing Science

Radboud University

The Underlying Belief Model of Uncertain Partially Observable Markov Decision Processes

Author:
Eline Bovy
s1007567

Daily supervisor:
Marnix Suilen MSc
marnix.suilen@ru.nl

First supervisor/assessor:
dr. Nils Jansen
n.jansen@cs.ru.nl

Second assessor:
dr. Sebastian Junges
sebastian.junges@ru.nl

April 24, 2023

Abstract

Markov Decision Processes (MDPs) are the standard model for reasoning about sequential decision-making in a probabilistic environment. Extending MDPs with partial observability and model uncertainty allows real-world sequential decision-making problems to be modeled more realistically. The MDP and its extensions are used to compute optimal policies, i.e., best courses of action with respect to some predefined goals.

As both partial observability and model uncertainty add uncertainty to the model, computing optimal policies becomes increasingly hard. For Partially Observable Markov Decision Processes (POMDPs), it is common practice to define the underlying belief model: the belief MDP. Many POMDP optimal policy computation algorithms use the belief MDP or the corresponding belief update function. The underlying belief model for uncertain POMDPs (uPOMDPs) has not yet been defined.

In this thesis, we identify the underlying belief model of uPOMDPs. As this model is computationally untractable, we define an over-approximation of this model, which we refer to as the belief uMDP. We define the belief uMDP similarly to how the belief MDP is defined for POMDPs, adjusting it to work with uncertainty intervals. The added model uncertainty causes the uncertain versions of the belief update, belief state transition, and reward functions to give rise to optimization problems. The uncertain belief update, in particular, gives rise to non-convex quadratic fractional programs.

To make the computation of new uncertain belief states tractable, we approximate the non-convex quadratic fractional programs. We consider under-approximation via random sampling and over-approximation by decoupling constraints between the numerator and denominator functions. We discuss both full and partial decoupling of these functions.

We combine the full- and partial-decoupling over-approximation methods with the property that the transition function is independent between states. This combination allows us to transform part of the computation to precomputable linear programs. The resulting programs for the full- and partial-decoupling methods are linear and linear fractional, respectively. As a final step, we can use the Charnes-Cooper transformation to turn the fractional linear program of the partial-decoupling method into a linear program.

We implement the random sampling under-approximation and the partial-decoupling over-approximation method. We implement the partial-decoupling method both with and without the Charnes-Cooper transformation step. We use the mathematical optimization solver Gurobi to solve resulting linear (fractional) programs. We evaluate the implementations on uPOMDP versions of the cheese maze, the tiger problem, and the grid-world robot POMDPs. Additionally, we evaluate the aircraft benchmark problem.

The partial-decoupling method, in combination with the Charnes-Cooper transformation, is the fastest in all experiments and returns the same values as the partial-decoupling method without the Charnes-Cooper transformation. On the aircraft benchmark problem, the partial-decoupling method with the Charnes-Cooper transformation is able to explore up to horizon 17 within one hour.

Contents

Chapter 1

Introduction

In *sequential decision making*, an agent, such as a human, animal, or computer, must repeatedly choose between two or more possible actions. These actions can influence the state of the world in the short or long term. The agent must try to make decisions in an optimal way with respect to some goal it is trying to reach. Sequential decision-making problems arise in many situations. Think, for example, of a spoken dialogue system, where a computer gets auditory input and must choose how to react with the goal of quickly satisfying the user's requests, therefore maximizing the user's happiness [Young et al., 2010]; or a mouse in a maze that has to choose what direction to move in with the goal of reaching a piece of cheese placed somewhere in the maze [McCallum, 1993].

A *Markov decision process* (MDP) provides a framework for sequential decision-making where the outcome of an action is probabilistic [Puterman, 1994]. With this, you can model, for example, a slippery version of the cheese maze, meaning that whenever the mouse tries to move to another square, there is a chance it slips and stays in the same square. An MDP consists of a set of states that mimic the state of the world, a set of actions, and a state transition function. This state transition function defines for each state-action pair the probability of reaching successor states. Additionally, the MDP can have a reward function, which assigns a reward to each state-action pair. This corresponds to the benefits of reaching your goal or the cost of taking risks. Once you have an MDP, you can compute the optimal *policy*, i.e., the best course of action with respect to the predefined goal, often maximizing the expected reward. Optimal policies for MDPs can be computed via, for instance, value iteration or linear programming [Baier and Katoen, 2008].

One underlying assumption of MDPs is full state observability for the agent. Furthermore, defining an MDP precisely requires full knowledge of the problem. When trying to model real-world sequential decision-making problems, state observability for the agent and full knowledge are not always possible. For example, when the available data is imprecise or incomplete. We give some more concrete examples below. Various extensions of MDPs have been developed to deal with different types of lack of information.

One such extension is the *partially observable MDP* (POMDP) [Kaelbling et al., 1998]. POMDPs deal with a lack of information in observability for the agent: States are only partially observable and cannot always be distinguished from one another. For example, when the mouse is placed inside the maze, it might be able to tell that it is in a vertical hallway but not in which one. When subject to partial observability, an agent can no longer choose actions based on the true state of the world. However, by keeping track of the action-observation history, the agent can summarize its knowledge about the true state of the world

in a probability distribution over the states, called the *belief*. The probability for each state describes the chance of that state being the true state of the world. After every action and resulting observation, the new belief can be computed with the *belief update function*.

Another extension is the *uncertain MDP* (uMDP). uMDPs deal with a lack of information in the model: Transitions can not be described by a single probability but instead lie in an uncertainty set [Nilim and Ghaoui, 2005, Wiesemann et al., 2013, Suilen et al., 2022]. For example, in the cheese maze, it is difficult to give the exact probability with which the mouse will slip. Instead, we can give an interval in which this probability lies. By fixing a probability distribution in the uncertainty set for each state-action pair, we can instantiate a nominal version of the model. A uMDP is essentially a, possibly infinite, set of MDPs.

The lack of information in observability and transitions can also be combined into an *uncertain partially observable MDP* (uPOMDP) [Itoh and Nakamura, 2007, Osogami, 2015, Suilen et al., 2020, Cubuktepe et al., 2021]. This is the type of model we concern ourselves with in this thesis.

The underlying belief model

Computing an optimal policy for POMDPs is a hard problem [Madani et al., 2003]. Since uPOMDPs are, possibly infinite, sets of POMDPs, optimal policy computation for uPOMDPs is at least as hard as for POMDPs.

When computing policies for POMDPs, it is common practice to look at the underlying belief model: the *belief Markov decision process* (belief MDP). In the belief MDP, the states represent the belief in the state of the world [Kaelbling et al., 1998]. As there are infinitely many probability distributions, we, in general, cannot compute an entire belief MDP. However, we can compute the belief MDP up to a finite horizon and then use MDP policy computation algorithms to get a finite horizon policy for the corresponding POMDP. Furthermore, various POMDP policy computation algorithms use the belief update function, such as point-based value iteration [Shani et al., 2013, Walraven and Spaan, 2019].

A similar underlying belief model can be defined for uPOMDPs. This model, which we call the *belief uncertain Markov decision process* (belief uMDP), is the set of belief MDPs of the POMDPs contained in the uPOMDP. Figure 1.1 depicts these relations between the models. The blue arrows represent the underlying belief model and the red arrows represent membership. So, if a POMDP is contained in a uPOMDP, then the underlying belief model of that POMDP, i.e., the belief MDP, is contained in the underlying belief model of that uPOMDP, i.e., the belief uMDP.

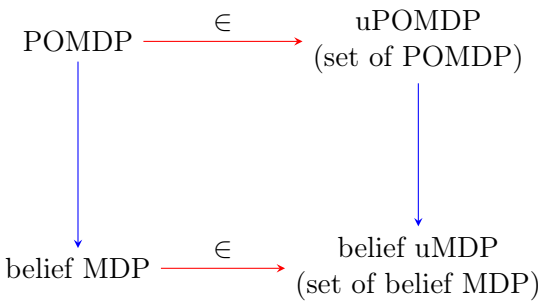


Figure 1.1: Overview of relations between the underlying belief models.

Although there are some belief-based approaches for computing policies on uPOMDPs [Osogami, 2015, Chamie and Mostafa, 2018, Itoh and Nakamura, 2007, Ahmadi et al., 2018], none of them defines the underlying belief model for uPOMDPs. [Osogami, 2015] mentions the idea but quickly disregards it because of the finite horizon limitations we also have for the belief MDP. However, defining this underlying belief model has many benefits.

Most importantly, defining the belief uMDP paves the way for lifting belief-based POMDP approaches to the uncertain setting. It allows us to compute finite horizon policies on uPOMDPs using uMDP algorithms like we use MDP algorithms to compute them on POMDPs. Furthermore, having an uncertain version of the belief update function is the first step in adjusting various belief-based POMDP policy computation algorithms to work with uPOMDPs.

Moreover, the belief uMDP acts as a saving point, which reduces recomputation when multiple finite horizon policies with different objectives need to be computed on the same uPOMDP. In existing uPOMDP policy computation algorithms, the objective is integrated into the algorithm, so even slight adjustments to the objective require complete recomputation.

Finally, the belief uMDP provides a visual representation of the uncertainty in a model up to a finite horizon. The states of the belief uMDP summarize the uncertain belief of the agent, making it possible to identify which uncertain transition in the uPOMDP causes the most uncertainty in the agent's belief. This information can, for example, be used to make more effective decisions regarding system improvement.

Contributions

Despite all the benefits, the belief uMDP has one crucial problem: it is computationally untractable. Especially when the number of POMDPs in a uPOMDP is infinite, for example, when we use interval uncertainties. As a more tractable alternative, we define an approximate belief uMDP, which over-approximates the true belief uMDP. By over-approximating the true belief uMDP, we ensure that the belief MDPs of all POMDPs contained in the uPOMDP are still contained in the approximate belief uMDP. Figure 1.2 extends Figure 1.1 with the approximate belief uMDP. Again, the blue arrows represent the underlying belief model, and the red arrows represent membership. The cyan arrow represents subset inclusion.

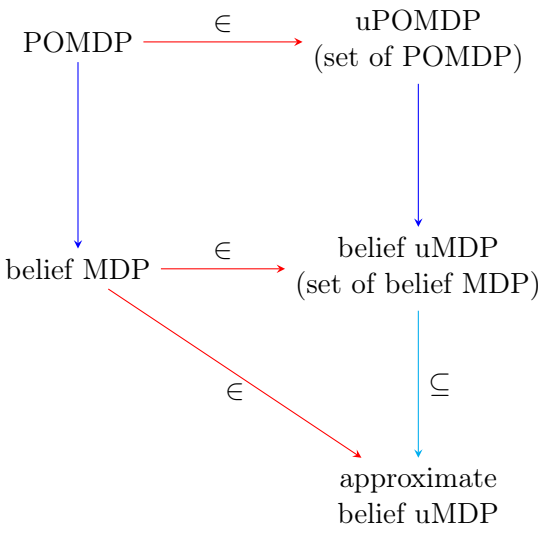


Figure 1.2: Overview of the relations between the underlying belief models and the approximate belief uMDP.

Note that, by over-approximating, an error is introduced into the model. Policies computed using the approximate belief uMDP or the approximate uncertain belief update might become overly pessimistic or optimistic. The tighter the approximation, the smaller the error.

For readability purposes, we will refer to the approximate belief uMDP as the belief uMDP throughout the rest of the report. Furthermore, to prevent confusion, we will emphasize when we talk about the true belief uMDP by explicitly naming it as such.

The uncertain belief state

We begin by defining the uncertain belief state. This uncertain belief state must contain all possible true beliefs given the action-observation history. We use probability intervals to represent the uncertain belief in each state and add additional constraints to minimize the over-approximation of the true uncertain belief state.

The belief uMDP

Based on the set of uncertain belief states, we define the belief uMDP. We extend the belief MDP definitions of the belief update, belief state transition, and reward functions to uncertain versions by turning them into optimization problems over the old uncertain belief state and the uncertain state transition function. We explain how these optimization problems ensure that all true uncertain beliefs, belief state transitions, and rewards are contained.

Optimization problems

Our definition of the belief uMDP results in three different types of optimization problems: linear, quadratic, and quadratic fractional programs. Where linear programs can be solved exactly, up to a given accuracy, in polynomial time [Boyd and Vandenberghe, 2014], quadratic and quadratic fractional programs are, in general, harder to solve [Sahni, 1974].

We discuss a couple of ways to approximate the quadratic programs. Additionally, we discuss how, in case of independence between transitions of different states, we can transform the quadratic programs into multiple linear ones. Most of these resulting linear programs can be precomputed, as they only rely on the uncertain state transition function.

Furthermore, we discuss several ways to approximate the quadratic fractional programs. We discuss under-approximation by sampling on the uncertainty set, applying the quadratic fractional function, and taking the maximum or minimum. Furthermore, as over-approximation is more desirable than under-approximation, we discuss the decoupling of the uncertainty sets of the numerator and denominator. We consider both full decoupling and partial decoupling. Full decoupling results in two quadratic programs, one for the numerator and one for the denominator. Partial decoupling, on the other hand, does not change the type of problem we need to deal with. However, if we again consider the independence between transitions of different states, we end up with a fractional linear program and a couple of precomputable linear programs. As a final step, the Charnes-Cooper transformation can be applied to transform the linear fractional program into a linear one.

Implementation

We implement finite horizon computation of the belief uMDP with the three most promising approximation methods for the non-convex quadratic fractional programs: the random sampling approach and the partial decoupling approach with and without the Charnes-Cooper transformation. We use the mathematical optimization solver Gurobi to solve all linear (fractional) programs.

We compare the approximation methods based on the number of explored uncertain belief states within a given time limit and the sizes of the probability intervals of the uncertain belief states. Furthermore, we investigate the effect of increased uncertainty and increased model size. Most notably, with the partial decoupling approach with the Charnes-Cooper transformation, we can explore the belief uMDP of a relatively large uPOMDP up to horizon 17 within one hour.

Chapter overview

We begin by discussing existing research on uPOMDPs in Chapter 2. Next, in Chapter 3, we introduce the theory necessary for understanding this thesis, and in Chapter 4, we give the formal definitions of the MDP and its relevant extensions. In Chapter 5, we define the uncertain belief state, the belief uMDP and all related functions. In Chapter 6, we elaborate on solving and approximating the optimization problems that follow from the definitions in Chapter 5. In Chapter 7, we discuss our implementation of the finite horizon computation of the belief uMDP, the numerical experiments we conducted to compare the most promising approximation methods of Chapter 6, and the results thereof. We shortly reflect on the thesis in Chapter 8, discussing some improvement points. Finally, we conclude the thesis in Chapter 9, summarizing our work and discussing future research directions.

Chapter 2

Related Work

The notion of uncertain probabilities in POMDPs is first introduced in [Itoh and Nakamura, 2007]. They define POMDPs with imprecise parameters (POMDPIPs) with two versions of uncertainty in the transition and observation functions: the interval case and the point-set case. They use second-order beliefs, the belief in the different possible models, to induce hypothetical POMDPs, which in turn can be used to compute policies. As the true second-order belief of a POMDPIP is unknown, they use multiple arbitrary possibly-correct second-order beliefs to compute so-called quasi-optimal policies. They regard any quasi-optimal policy as a solution to the corresponding POMDPIP.

[Osogami, 2015] focuses on finding a robust policy with respect to the expected reward, i.e., a policy that maximizes the expected reward considering the worst-case of the uncertainty. They define a robust version of the value function using a maximin approach and prove this function is convex when the uncertainty set is as well. Using this convexity, they define robust point-based value iteration on the belief space to compute robust policies.

In addition to the expected reward, [Suilen et al., 2020] focuses on robustly satisfying temporal logic constraints. They encode the underlying optimization problem as a semi-infinite non-convex quadratically constrained quadratic program (QCQP) [Boyd and Vandenberghe, 2014] and convexify this around an initial policy using the convex-concave procedure [Lipp and Boyd, 2016]. Additionally, they restrict the uncertainty set to convex polytopes, which allows them to reduce the quadratic constraints to a finite number. The finite convex QCQP can be directly solved by convex optimization solvers, resulting in a robust memoryless policy for the uPOMDP, i.e., a robust policy that chooses actions based only on the current state, not taking into account execution history.

Similar to [Suilen et al., 2020], [Cubuktepe et al., 2021] uses a form of convex optimization to compute robust policies. [Cubuktepe et al., 2021] extends policy computation to finite-memory policies by combining the uPOMDP with the memory update of a finite state controller, i.e., an automaton consisting of a finite set of states, an action mapping function, and a (memory) state update function [Meuleau et al., 1999]. The underlying problem is, again, a semi-infinite non-convex QCQP. They transform the uPOMDP into a simple uPOMDP, where a state can have either only action choices or only uncertain outcomes. They use dualization to reduce the constraints to a finite number. This dualization relies on the uPOMDP being simple. Finally, they use a sequential convex programming (SCP) method to solve the finite non-convex problem. SCP iteratively linearizes the quadratic constraints around previous solutions, starting with an arbitrary one, until a solution satisfies the expected reward specification.

Instead of robust policies, [Ahmadi et al., 2018] concentrates on checking robust satisfaction of safety and optimality requirements. They treat uPOMDPs as switched systems [Liberzon, 2003], where switching between different possible models takes place depending on belief. Based on this representation, the safety and optimality requirements of a POMDP can be verified using barrier certificates [Prajna et al., 2007].

[Chamie and Mostafa, 2018] looks at post-processing a policy to become more robust against changes in the observation function. They keep track of a belief region, which encapsulates the possible error of the observation function. They formulate a linear program to compute a robust value function given a belief region and use this to post-process the original policy to select actions more robustly.

Like most approaches above, we look at uPOMDPs restricted to a convex uncertainty set. A critical difference between our research and previous research is the purpose: instead of finding an optimal robust policy or verifying specifications, we aim to define the underlying belief model. This model can later be used to compute policies or verify specifications on a finite horizon by applying existing methods for uMDPs. Finite horizon policy algorithms for on uMDPs use robust versions of techniques such as dynamic programming and value iteration [Nilim and Ghaoui, 2005, Wiesemann et al., 2013, Wolff et al., 2012]

The research done in [Chamie and Mostafa, 2018] has the most overlap with our research, considering we also keep track of belief regions instead of induced belief points. However, [Chamie and Mostafa, 2018] only considers uncertainty in the observation function and relies on a previously computed policy to linearize around. We, on the other hand, consider uncertainty in the transition function. Note that we can do this without loss of generality, as the observation function can be made deterministic by expanding the state space, which moves the observation uncertainty to the transition function [Chatterjee et al., 2016]. Furthermore, we do not linearize around another policy.

Chapter 3

Preliminaries

In this chapter, we introduce our notation conventions (Section 3.1) and briefly explain the mathematical theory needed for understanding the thesis. We discuss the first and second derivatives of multivariate functions in Section 3.2 and optimization problems in Section 3.3.

3.1 Notation

Throughout this report, we use the following notations:

$\Delta(S)$	:	The set of probability distributions over S.
$\mathbb{I}_{\mathbb{R}}$	:	The set of real valued intervals: $\{[a, b] \mid a, b \in \mathbb{R} \wedge a \leq b\}$.
$\mathbb{I}_{\mathbb{P}}$	:	The set of probability intervals: $\{[a, b] \mid a, b \in [0, 1] \wedge a \leq b\}$.
$[n]$	:	The set of the integers 1 to n: $\{z \mid z \in \mathbb{Z}^+ \wedge z \leq n\}$.
$\vec{x}$	:	Vector notation.
A	:	Matrix notation.
∇f	:	The gradient of function f.
$\nabla^2 f$	:	The Hessian of function f.

3.2 Gradient and Hessian

Given a function $f\colon \mathbb{R}^n \to \mathbb{R}$, the derivative $f'(\vec{x})$ is a row vector of length n. The gradient of f is the transpose of this vector [Boyd and Vandenberghe, 2014].

$$\begin{aligned}\nabla f(\vec{x}) &= f'(\vec{x})^T \\ &= \begin{bmatrix} \dfrac{\partial f}{\partial x_1}(\vec{x}) \\ \vdots \\ \dfrac{\partial f}{\partial x_n}(\vec{x}) \end{bmatrix}\end{aligned}$$

The Hessian of f is the second derivative [Boyd and Vandenberghe, 2014].

$$\nabla^2 f(\vec{x}) = \begin{bmatrix} \frac{\partial^2 f}{\partial x_1^2}(\vec{x}) & \frac{\partial^2 f}{\partial x_1 \partial x_2}(\vec{x}) & \dots & \frac{\partial^2 f}{\partial x_1 \partial x_n}(\vec{x}) \\ \frac{\partial^2 f}{\partial x_2 \partial x_1}(\vec{x}) & \frac{\partial^2 f}{\partial x_2^2}(\vec{x}) & \dots & \frac{\partial^2 f}{\partial x_2 \partial x_n}(\vec{x}) \\ \vdots & \vdots & \ddots & \vdots \\ \frac{\partial^2 f}{\partial x_n \partial x_1}(\vec{x}) & \frac{\partial^2 f}{\partial x_n \partial x_2}(\vec{x}) & \dots & \frac{\partial^2 f}{\partial x_n^2}(\vec{x}) \end{bmatrix}$$

3.3 Optimization problems

Optimization problems form the core problem for computing optimal policies. An optimization problem has the following standard form [Boyd and Vandenberghe, 2014]:

$$\begin{aligned} \text{Minimize} \quad & f_0(\vec{x}) \\ \text{subject to} \quad & f_i(\vec{x}) \leq c_i, \quad i \in [m], \end{aligned}$$

where

$$\begin{aligned} \vec{x} = \begin{bmatrix} x_1 & \dots & x_n \end{bmatrix} \quad & \text{is the } \textit{optimization variable}, \\ f_0 \colon \mathbb{R}^n \to \mathbb{R} \quad & \text{is the } \textit{objective} \text{ function}, \\ f_i \colon \mathbb{R}^n \to \mathbb{R}, \quad i \in [m] \quad & \text{are the } \textit{constraint} \text{ functions}, \\ c_i, \quad i \in [m] \quad & \text{are the } \textit{constraints}. \end{aligned}$$

The standard form minimizes the objective function. The maximum can be obtained by minimizing the negation.

There are many subclasses of optimization problems defined by the types of objective and constraint functions. We summarize the most relevant terms below.

Linearly constrained An optimization problem with linear constraint functions is called linearly constrained. We write a linear function in canonical form:

$$f(\vec{x}) = \vec{b}^T \vec{x}$$

Quadratically constrained An optimization problem with quadratic constraint functions is called quadratically constrained. We write a quadratic function in canonical form:

$$f(\vec{x}) = \tfrac{1}{2} \vec{x}^T Q \vec{x} + \vec{b}^T \vec{x}$$

We assume the Q to be symmetric. A non-symmetric square matrix can be rewritten to its symmetric part $\frac{1}{2}(Q + Q^T)$, where $\forall \vec{x} \in \mathbb{R}^n : \vec{x}^T Q \vec{x} = \vec{x}^T \frac{1}{2}(Q + Q^T) \vec{x}$ [Chong and Żak, 2013].

Linear program An optimization problem with a linear objective function is called a linear program. Unless otherwise noted, a linear program is assumed to be linearly constrained.

Linear programs can be solved exactly, up to a given accuracy, in polynomial time with methods like Dantzig's simplex method and interior-point methods [Boyd and Vandenberghe, 2014].

Quadratic program An optimization problem with a quadratic objective function is called a quadratic program. Unless otherwise noted, a quadratic program is assumed to be linearly constrained.

The general global optimization of quadratic programs is NP-hard [Sahni, 1974].

Fractional program An optimization problem with a fraction of two functions as the objective function is called a fractional program. Unless otherwise noted, a fractional program is assumed to be linearly constrained.

3.3.1 Convexity

A set $C \subset \mathbb{R}^d$ with $d \in \mathbb{Z}^+$ is convex if $\forall \vec{x}, \vec{y} \in C$, the line segment connecting $\vec{x}$ and $\vec{y}$ is contained in C.

Given a convex set C and a function $f\colon C \to \mathbb{R}$:

f is convex	if	$\forall \vec{x}, \vec{y} \in C, \forall \alpha, \beta \in \mathbb{R}_{\geq 0}, \alpha + \beta = 1$:	$f(\alpha\vec{x} + \beta\vec{y}) \leq \alpha f(\vec{x}) + \beta f(\vec{y})$
f is concave	if	$\forall \vec{x}, \vec{y} \in C, \forall \alpha, \beta \in \mathbb{R}_{\geq 0}, \alpha + \beta = 1$:	$f(\alpha\vec{x} + \beta\vec{y}) \geq \alpha f(\vec{x}) + \beta f(\vec{y})$
f is strictly convex	if	$\forall \vec{x}, \vec{y} \in C, \forall \alpha, \beta \in \mathbb{R}_{\geq 0}, \alpha + \beta = 1$:	$f(\alpha\vec{x} + \beta\vec{y}) < \alpha f(\vec{x}) + \beta f(\vec{y})$
f is strictly concave	if	$\forall \vec{x}, \vec{y} \in C, \forall \alpha, \beta \in \mathbb{R}_{\geq 0}, \alpha + \beta = 1$:	$f(\alpha\vec{x} + \beta\vec{y}) > \alpha f(\vec{x}) + \beta f(\vec{y})$

If f is none of the above, f is non-convex.

Convex functions have the property that every local minimum is a global minimum, and concave functions have the property that every local maximum is a global maximum [Kochenderfer and Wheeler, 2019]. Additionally, the strictly convex and concave functions have at most one global minimum and maximum, respectively.

Checking convexity

Let A be an $n \times n$ symmetric real matrix and let $\vec{0}$ be the n-dimensional zero vector:

A is positive semi-definite	iff $\forall \vec{x} \in \mathbb{R}^n$:	$\vec{x}^T A \vec{x} \geq 0$	which we denote as	$A \succeq 0$
A is negative semi-definite	iff $\forall \vec{x} \in \mathbb{R}^n$:	$\vec{x}^T A \vec{x} \leq 0$	which we denote as	$A \preceq 0$
A is positive definite	iff $\forall \vec{x} \in \mathbb{R}^n \setminus \{\vec{0}\}$:	$\vec{x}^T A \vec{x} > 0$	which we denote as	$A \succ 0$
A is negative definite	iff $\forall \vec{x} \in \mathbb{R}^n \setminus \{\vec{0}\}$:	$\vec{x}^T A \vec{x} < 0$	which we denote as	$A \prec 0$

If A is none of the above, A is indefinite.

The convexity of a function $f\colon C \to \mathbb{R}$ can be checked by looking at the definiteness of the Hessian of f [Boyd and Vandenberghe, 2014].

$$\begin{array}{lll} f \text{ is convex} & \text{iff} \quad \forall \vec{x} \in C: & \nabla^2 f(\vec{x}) \text{ is positive semi-definite} \\ f \text{ is concave} & \text{iff} \quad \forall \vec{x} \in C: & \nabla^2 f(\vec{x}) \text{ is negative semi-definite} \\ f \text{ is strictly convex} & \text{iff} \quad \forall \vec{x} \in C: & \nabla^2 f(\vec{x}) \text{ is positive definite} \\ f \text{ is strictly concave} & \text{iff} \quad \forall \vec{x} \in C: & \nabla^2 f(\vec{x}) \text{ is negative definite} \end{array}$$

The Hessian of a quadratic function $f(\vec{x}) = \frac{1}{2}\vec{x}^T Q\vec{x} + \vec{b}^T\vec{x}$ is $\nabla^2 f(\vec{x}) = Q$. Hence, the convexity of a quadratic function can be determined by looking at the definiteness of the quadratic part of the function.

3.3.2 Convex polytopes

A polyhedron is the solution set of a finite number of linear equalities and inequalities [Boyd and Vandenberghe, 2014]. A convex polytope is a bounded polyhedron. In this thesis, we focus on optimization problems where the optimization variable is completely bounded by probability intervals, i.e., linear inequalities, hence forming a convex polytope.

Given optimization variable $\vec{x} = \begin{bmatrix} x_1 & \dots & x_n \end{bmatrix}$ and a linear (in)equality:

$$\sum_{i \in [n]} e_n x_n \bowtie e^*$$

We call

$$\begin{array}{rl} \vec{e} = \begin{bmatrix} e_1 & \dots & e_n \end{bmatrix} & \text{the coefficient vector,} \\ e^* & \text{the constant,} \\ \bowtie \in \{=, <, >, \leq, \geq\} & \text{the operator.} \end{array}$$

In this thesis, we only deal with equalities and non-strict inequalities. We can rewrite the linear (in)equalities with the $=$ or $\geq$ operators to linear inequalities with the $\leq$ operator as follows:

$$\begin{array}{lcl} \displaystyle\sum_{i \in [n]} e_n x_n \geq e^* & \Longrightarrow & \displaystyle\sum_{i \in [n]} -e_n x_n \leq -e^* \\ \displaystyle\sum_{i \in [n]} e_n x_n = e^* & \Longrightarrow & \displaystyle\sum_{i \in [n]} -e_n x_n \leq -e^* \wedge \sum_{i \in [n]} e_n x_n \leq e^* \end{array}$$

Having transformed all linear (in)equalities to $\leq$-inequalities, we can write a convex polytope in the canonical matrix form:

$$A\vec{x} \leq \vec{c},$$

where each column in A forms the coefficients of a linear inequality and the corresponding element in $\vec{c}$ forms the constant.

Chapter 4

Formal models

This chapter formally discusses the Markov decision process and the three extensions that follow from partial observability and model uncertainty. We begin by introducing the running example used throughout this thesis in Section 4.1. Next, in Section 4.2, we formally define the Markov decision process and discuss the related theory of policies, horizons, and value functions. Finally, we discuss the partial observability extension in Section 4.3 and the model uncertainty extensions in Section 4.4.

4.1 Running example

Throughout this report, we use the *cheese maze* example to demonstrate the theory [McCallum, 1993]. This example was briefly touched upon in Chapter 1. A mouse is placed in a maze and can move in any of the cardinal directions that are not blocked by a wall. Somewhere in the maze lies a piece of cheese, and the mouse's goal is to reach it.

The specific maze we look at consists of fourteen squares labeled 0 through 13. The cheese is located at square 13. Figure 4.1 depicts the maze.

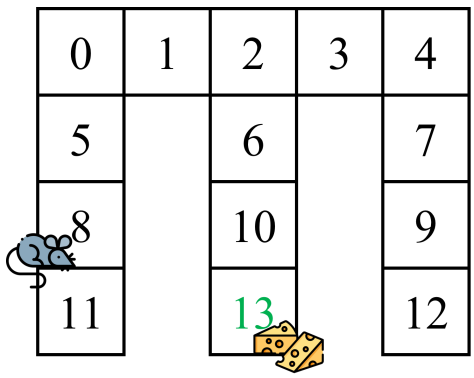


Figure 4.1: Cheese maze problem.

4.2 Markov Decision Processes

Markov decision processes (MDPs) are the standard model for sequential decision-making in a probabilistic environment [Puterman, 1994]. As shown in Figure 4.2, based on [Kaelbling

et al., 1998], an MDP models the decision-making loop, where the agent observes the state and takes an action, and the world receives an action and updates the state accordingly.

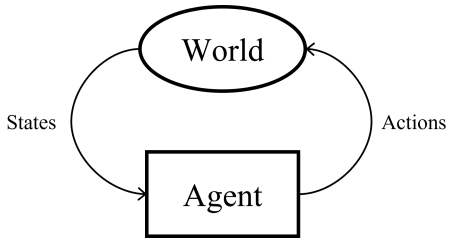


Figure 4.2: Systematic workings of an MDP agent.

Modeling a sequential decision-making problem with an MDP requires full knowledge of the problem. Specifically, it requires full state observability for the agent, meaning the agent always knows the true state, and full knowledge about the model, meaning the transition probabilities are known.

Definition 1: Markov decision process

An MDP is a tuple $M = (S, A, P, R)$, where:

- S the finite set of states.
- A the finite set of actions.
- $P \colon S \times A \to \Delta(S)$ the (partial) state transition function.
- $R \colon S \times A \to \mathbb{R}$ the optional reward function.

We use $P(s, a, s')$ as shorthand notation for $P(s, a)(s')$. Moreover, we use $P(\cdot, a) \colon S \to \Delta(S)$ to denote the state transition function limited to the a-transitions, so $P(\cdot, a)(s) = P(s, a)$.

4.2.1 Policies

A *policy*, also called a *scheduler* or *strategy*, is a function mapping a sequence of states and actions, i.e., an execution history, to a distribution over the set of actions:

$$\sigma \colon (SA)^* S \to \Delta(A)$$

Given execution history $h \in (SA)^* S$, we denote the probability of taking action $a \in A$ in policy σ as $\sigma(h, a)$.

A policy resolves the non-deterministic choices in an MDP based on the execution history. If a policy only depends on the last state of the sequence, i.e., of type $\sigma \colon S \to \Delta(A)$, the policy is called *memoryless*. If a policy maps the execution history, or state, to a single action instead of a distribution over the actions, i.e., of type $\sigma \colon (SA)^* S \to A$, the policy is called *deterministic*. A policy induces a *Markov chain*.

Definition 2: Induced Markov chain

Given an MDP $M = (S, A, P, R)$ and a policy σ, the induced Markov chain is a tuple $M^\sigma = (S^\sigma, P^\sigma)$, where:

- $S^\sigma \subseteq (SA)^*S$ the set of states.
- $P^\sigma \colon S^\sigma \to \Delta(S^\sigma)$ the state transition function.

Given state $h = (s_0\, a_0\, \ldots\, s_{n-1}\, a_{n-1}\, s_n) \in S^\sigma$ and action $a_n \in A$, the transition probability is computed as:

$$P^\sigma(h, h\, a_n\, s_{n+1}) = \sigma(h, a_n) \cdot P(s_n, a_n, s_{n+1})$$

In an induced Markov chain, all transitions are probabilistic. You can compute the probability of different executions of the Markov chain, also called paths, by multiplying the transition probabilities [Baier and Katoen, 2008]. These path probabilities can, in turn, be used to compute more meaningful probabilities, such as the probability of reaching a desirable state or not reaching an undesirable state.

Policies can be computed for various objectives, such as satisfying *probabilistic computation tree logic* (PCTL) formulae [Baier and Katoen, 2008] or, in case a reward function is given, maximizing the expected reward. In this thesis, we focus on policies for maximizing the expected reward.

Horizons

When computing optimal policies with respect to the expected reward, there are three types of frameworks: finite, indefinite, and infinite horizons [Kaelbling et al., 1998]. Let r_t be the reward the agent receives at t steps from the initial state. In a finite horizon framework, the agent must maximize the expected reward of the next k states:

$$\mathbb{E}\left[\sum_{t=0}^{k-1} r_t\right]$$

The indefinite horizon framework is similar, except that the number k of next states to maximize over is unknown:

$$\mathbb{E}\left[\sum_{t=0}^{k-1} r_t\right]$$

Alternatively, in the infinite horizon framework, the agent must maximize the expected reward over an infinite lifetime, taking into account a discounting factor $0 < \gamma < 1$ to give more weight to early rewards:

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

The value of the discounting factor determines the degree to which future expected reward is taken into account. The closer to 1 the discounting factor is, the more effect future expected rewards have on the agent's decisions. Furthermore, the discounting factor is required to ensure convergence of the expected reward. We can also add a discounting factor to the finite and indefinite horizon frameworks, where setting it to 1 will result in the original definition.

Value function

Given an MDP $M = (S, A, P, R)$, a memoryless deterministic policy σ, and a discounting factor $0 < \gamma < 1$, the value function computes the expected reward of executing σ. The value function can be used to evaluate policies [Kaelbling et al., 1998]. Let $V_{\sigma,t}(s)$ be the expected reward from starting in state s and executing policy σ for t steps:

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

For the infinite horizon case, we can drop the steps parameter from the above definition:

$$V_{\sigma}(s) = R(s, \sigma(s)) + \gamma \cdot \sum_{s' \in S} P(s, \sigma(s), s') \cdot V_{\sigma}(s')$$

4.2.2 Example

We can model the cheese maze problem as an MDP. The set of states consists of the fourteen squares of the maze. The set of actions consists of the four cardinal directions plus an extra action for when the mouse has found the cheese by reaching square 13. This can be, for example, a do-nothing action or a reset action.

The state transition function defines for each state-action pair the probability distribution of reaching the other states. We assume the maze is slippery: for each action, the mouse has an 85% chance to reach the intended square and a 15% chance to stay where it is. Figure 4.3 illustrates the cheese maze MDP as described above.

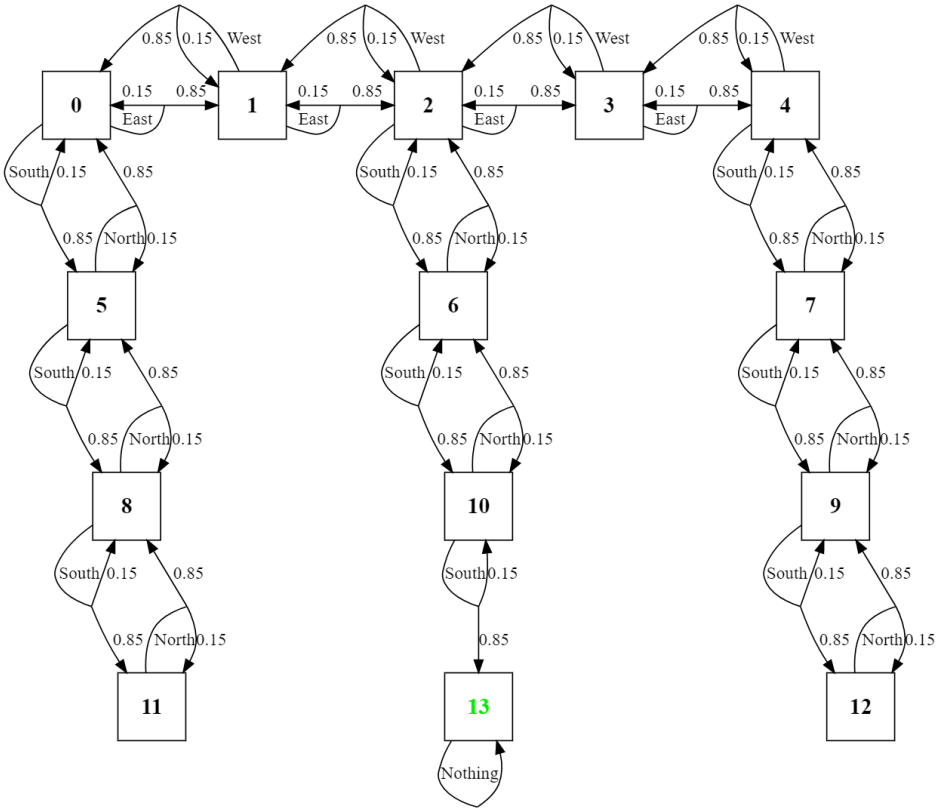


Figure 4.3: Cheese maze MDP.

We did not define any probabilities for taking an action towards a wall. If we want to define a total state transition function, we can add self-loop transitions with probability 1.

4.3 Partially Observability

The assumption that an agent has full state observability is often unrealistic when looking at real-world sequential decision-making problems. For example, in the maze problem, the mouse might be able to tell that it is in a vertical hall but not in which one. The mouse must, therefore, choose actions based on its action-observation history instead of its true position. Figure 4.4 illustrates the maze problem where undistinguishable squares have the same color.

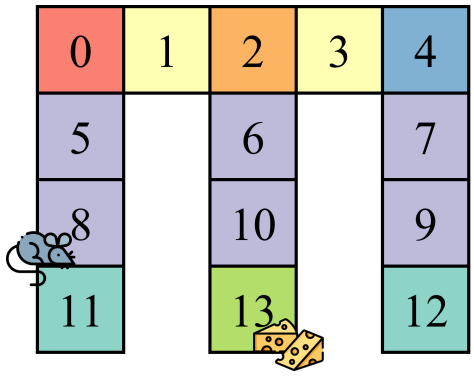


Figure 4.4: Cheese maze problem with partial observability.

The *Partially observable Markov decision processes* (POMDP) adds a set of observations and an observation function to the MDP definition, allowing sequential decision-making problems without full state observability to be modeled.

Definition 3: Partially observable Markov decision process

A POMDP is a tuple $M = (S, A, P, R, Z, O)$, where:

- S the finite set of states.
- A the finite set of actions.
- $P\colon S \times A \to \Delta(S)$ the (partial) state transition function.
- $R\colon S \times A \to \mathbb{R}$ the optional reward function.
- Z the finite set of observations.
- $O\colon S \to \Delta(Z)$ the observation function.

The observation function can be simplified to a state-based deterministic function $O\colon S \to Z$ by expanding the state space [Chatterjee et al., 2016]. Therefore, this thesis will only focus on this simpler observation function. We denote the boolean value of observing observation $o \in Z$ in state $s \in S$ or not as $O(s, o)$.

4.3.1 Belief

As an alternative for knowing the true state, the agent can keep track of a *belief state*, which summarizes the agent's previous actions and observations in a probability distribution over the set of states [Kaelbling et al., 1998]. The probabilities in a belief state indicate the degree to which the agent believes it is in the different states. As belief states are based on the action-observation history of the agent, each belief state is linked to a single observation. The set of belief states $\mathcal{B}$ is a subset of all probability distributions over the set of states $\Delta(S)$. Given a belief state $b \in \mathcal{B}$, we use the function $b\colon S \to [0, 1]$ to indicate the belief in the different states.

Figure 4.5, based on [Kaelbling et al., 1998], illustrates the idea of belief: the agent internally consists of a *state estimator* (SE), which produces a belief state, and a policy, which chooses the action based on this belief.

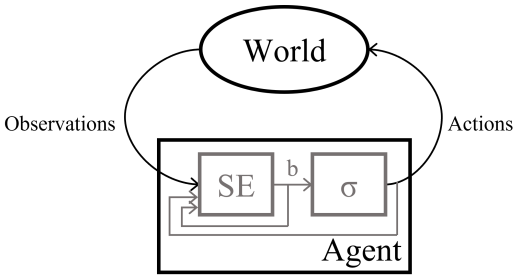


Figure 4.5: Systematic workings of a POMDP agent.

Starting with an initial belief, the state estimator needs to update the belief state after every action and resulting observation. The resulting belief state is given by the function $\mathrm{SE}\colon \mathcal{B} \times A \times Z \to \mathcal{B}$ and is computed using the *belief update function*.

Definition 4: Belief update function

Given a POMDP $M = (S, A, P, R, Z, O)$, a belief state b, an action $a \in A$, and an observation $o \in Z$, the new belief state b' is computed by $\forall s' \in S$:

$$\begin{aligned} b'(s') &= \Pr(\, s' \mid a, o, b\,) \\ &= \frac{\Pr(\, o \mid s', a, b\,) \cdot \Pr(\, s' \mid a, b\,)}{\Pr(\, o \mid a, b\,)} \\ &= \frac{\Pr(\, o \mid s', a\,) \cdot \sum_{s \in S} \Pr(\, s' \mid a, b, s\,) \cdot \Pr(\, s' \mid a, b\,)}{\Pr(\, o \mid a, b\,)} \\ &= \frac{O(s', o) \cdot \sum_{s \in S} P(s, a, s') \cdot b(s)}{\Pr(\, o \mid a, b\,)} \end{aligned}$$

Where $\Pr(\, o \mid a, b\,)$ is the normalization constant: $\sum_{s' \in S} O(s', o) \cdot \sum_{s \in S} P(s, a, s') \cdot b(s)$. Hence, we can write the belief update function explicitly as:

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

As belief states summarize the agent's action-observation history, they can be used to compute policies on POMDPs. For this purpose, we define the underlying belief model of the POMDP: the *belief MDP*.

Definition 5: Belief Markov decision process

Given a POMDP $M = (S, A, P, R, Z, O)$, the associated belief MDP is a tuple $\mathcal{M} = (\mathcal{B}, A, \mathcal{P}, \mathcal{R})$, where:

- $\mathcal{B} \subseteq \Delta(S)$ the set of belief states.
- A the finite set of actions.
- $\mathcal{P}\colon \mathcal{B} \times A \to \Delta(\mathcal{B})$ the (partial) belief state transition function.
- $\mathcal{R}\colon \mathcal{B} \times A \to \mathbb{R}$ the optional reward function.

Given belief states $b, b' \in \mathcal{B}$ and action $a \in A$, the transition probability is computed as:

$$\begin{aligned}\mathcal{P}(b, a, b') &= \Pr(\, b' \mid a, b\,) \\ &= \sum_{o \in Z} \Pr(\, b' \mid a, b, o\,) \cdot \Pr(\, o \mid a, b\,)\end{aligned}$$

Where $\Pr(\, o \mid a, b\,)$ is again the normalization constant and $\Pr(\, b' \mid a, b, o\,)$ indicates whether belief state b' is the successor of belief state b given action a and observation o:

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

Given belief state $b \in \mathcal{B}$ and action $a \in A$, the reward is computed as:

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

Horizons

Each belief state has at most $|A|$ outgoing transitions, and each outgoing transition can, because each belief state represents a single observation, go to at most $|Z|$ new belief states. Therefore, each belief state can produce at most $|A| \times |Z|$ new belief states. Since we defined the set of actions A and the set of observations Z as finite sets, looking at a finite horizon will also result in a finite belief MDP.

4.3.2 Policies

Policies for POMDPs cannot rely on the true state of the model. One type of policy available is the observation-based policy, where an observation sequence or action-observation sequence is mapped to a distribution over the set of actions:

$$\sigma \colon (ZA)^* Z \to \Delta(A)$$

An important requirement for observation-based policies is that executions with the same observation sequence are mapped to the same action.

Another type of policy for POMDPs is the belief-based policy, where a belief state is mapped to a distribution over the set of actions. Finite horizon belief-based policies can be computed using the belief MDP.

$$\sigma \colon \mathcal{B} \to \Delta(A)$$

Similar to the state-based policies for MDPs, observation-based and belief-based policies induce a Markov chain.

4.3.3 Example

As already mentioned in the introduction of this section, it is not obvious that when placing the mouse in the cheese maze, it immediately knows in which square it is. It is reasonable

to assume that the mouse has some sense of direction and can therefore deduct that it is, for example, in a vertical hallway, but there are no visual cues in the maze to distinguish the squares beyond that.

We extend the cheese maze MDP of Section 4.2.2 to a POMDP by adding an observation set and observation function. The observation set consists of the different wall configurations plus an observation for the goal square containing the cheese. Indicating a wall configuration by the letters of the positions of the walls it has and the cheese square by a 'C' gives us the following observation set:

$$\{\text{NW, NS, N, NE, EW, ESW, C}\}$$

We assume the observation function O is deterministic and get the following mapping:

$$O(s) = \begin{cases} \text{NW} & \text{if } s = s0 \\ \text{NS} & \text{if } s \in \{s1, s3\} \\ \text{N} & \text{if } s = s2 \\ \text{NE} & \text{if } s = s4 \\ \text{EW} & \text{if } s \in \{s5, s6, s7, s8, s9, s10\} \\ \text{ESW} & \text{if } s \in \{s11, s12\} \\ \text{C} & \text{if } s = s13 \end{cases}$$

Figure 4.6 illustrates the cheese maze POMDP described above, where states with the same observation have the same color.

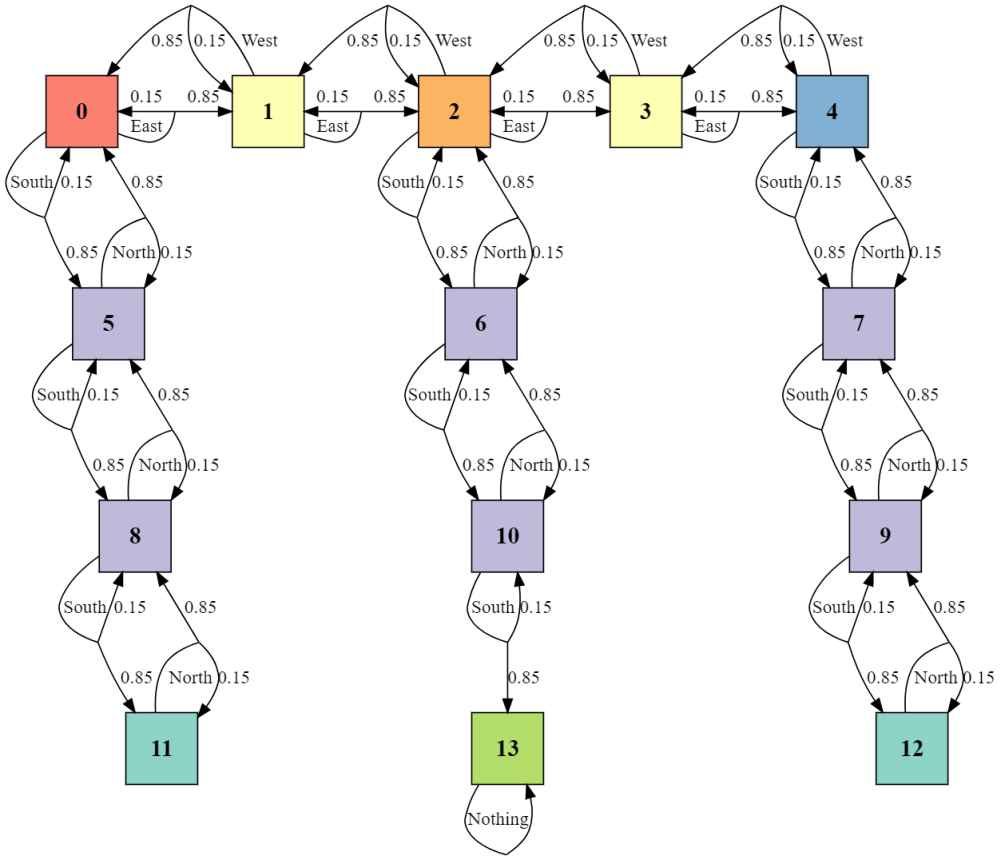


Figure 4.6: Cheese maze POMDP.

We can now compute the belief MDP of the cheese maze POMDP. We take the initial belief that there is an 80% chance that the mouse is in square 8 and a 10% chance that it is in square 9 or 10. The belief MDP results in an infinite amount of belief states. For example purposes, we shall only expand two steps from the initial belief. Figure 4.7 illustrates the partial belief MDP.

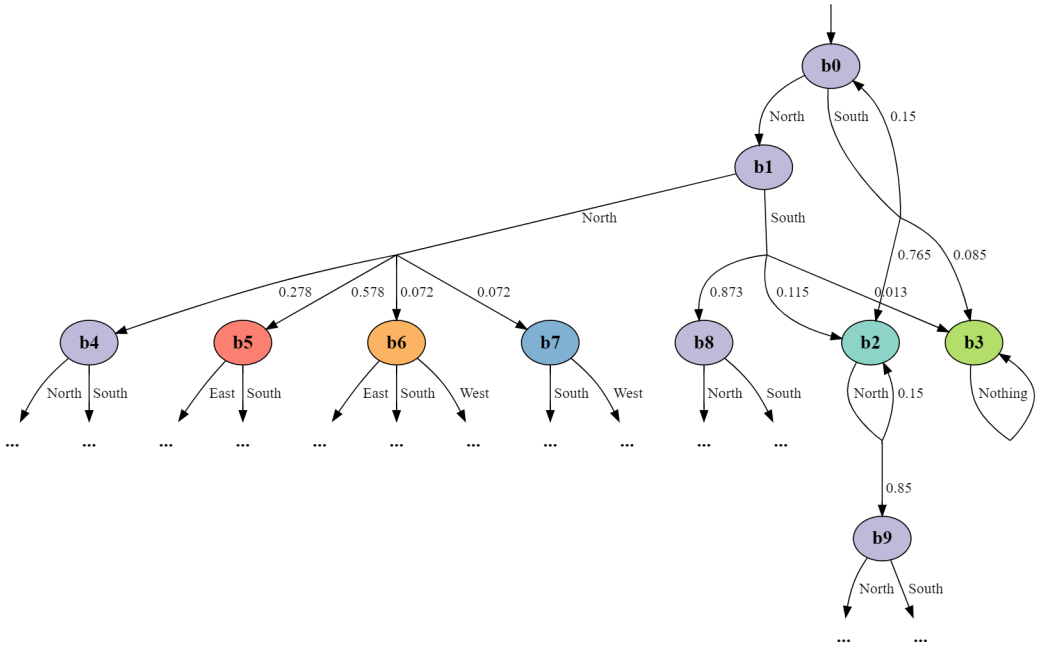


Figure 4.7: Cheese maze belief MDP.

The colors only indicate the observations the belief states correspond to: the belief states are fully observable and hence distinguishable for the agent. The belief states correspond to the following probability distributions, rounded to three decimals. We omit the states that are mapped to 0.

$b0\colon \{s8 \mapsto 0.8,\ s9 \mapsto 0.1,\ s10 \mapsto 0.1\}$

$b1\colon \{s5 \mapsto 0.68,\ s6 \mapsto 0.085,\ s7 \mapsto 0.085,\ s8 \mapsto 0.12,\ s9 \mapsto 0.015,\ s10 \mapsto 0.015\}$

$b2\colon \{s11 \mapsto 0.889,\ s12 \mapsto 0.111\}$

$b3\colon \{s13 \mapsto 1\}$

$b4\colon \{s5 \mapsto 0.735,\ s6 \mapsto 0.092,\ s7 \mapsto 0.092,\ s8 \mapsto 0.065,\ s9 \mapsto 0.008,\ s10 \mapsto 0.008\}$

$b5\colon \{s0 \mapsto 1\}$

$b6\colon \{s2 \mapsto 1\}$

$b7\colon \{s4 \mapsto 1\}$

$b8\colon \{s5 \mapsto 0.117,\ s6 \mapsto 0.015,\ s7 \mapsto 0.015,\ s8 \mapsto 0.683,\ s9 \mapsto 0.085,\ s10 \mapsto 0.085\}$

$b9\colon \{s8 \mapsto 0.889,\ s9 \mapsto 0.111\}$

We demonstrate how to compute belief state $b2$ and the transition $\mathcal{P}(b0, \text{South}, b2)$.

Belief state $b2$ describes the belief after the agent takes action South and observes observation ESW starting in belief state $b0$. Plugging this information into the belief update function gives us the following formula:

$\forall s' \in S$:

$$b2(s') = \frac{O(s', \text{ESW}) \cdot \sum_{s \in S} P(s, \text{South}, s') \cdot b0(s)}{\sum_{s' \in S} O(s', \text{ESW}) \cdot \sum_{s \in S} P(s, \text{South}, s') \cdot b0(s)}$$

Since the observation function O is deterministic and $O(s, \text{ESW}) = 1$ iff state $s \in \{s11, s12\}$, we can make the following split:

$\forall s' \in S \setminus \{s11, s12\}$:

$$b2(s') = 0$$

$\forall s' \in \{s11, s12\}$:

$$b2(s') = \frac{\sum_{s \in S} P(s, \text{South}, s') \cdot b0(s)}{\sum_{s' \in \{s11, s12\}} \sum_{s \in S} P(s, \text{South}, s') \cdot b0(s)}$$

Furthermore, we only need to look at the transitions starting in a state with a non-zero belief in belief state b0, which are states $s8, s9$, and $s10$.

$\forall s' \in \{s11, s12\}$:

$$\begin{aligned} b2(s') &= \frac{\sum_{s \in \{s8, s9, s10\}} P(s, \text{South}, s') \cdot b0(s)}{\sum_{s' \in \{s11, s12\}} \sum_{s \in \{s8, s9, s10\}} P(s, \text{South}, s') \cdot b0(s)} \\ &= \frac{\sum_{s \in \{s8, s9, s10\}} P(s, \text{South}, s') \cdot b0(s)}{0.85 \cdot 0.8 + 0 \cdot 0.1 + 0 \cdot 0.1 + 0 \cdot 0.8 + 0.85 \cdot 0.1 + 0 \cdot 0.1} \\ &= \frac{\sum_{s \in \{s8, s9, s10\}} P(s, \text{South}, s') \cdot b0(s)}{0.85 \cdot 0.9} \\ &= \frac{\sum_{s \in \{s8, s9, s10\}} P(s, \text{South}, s') \cdot b0(s)}{0.765} \end{aligned}$$

$0.765 = \Pr(\text{ ESW } \mid \text{ South}, b0\,)$ is the normalization constant, which we also need for the computation of the transition between belief states $b0$ and $b2$.

$$\begin{aligned} b2(s11) &= \frac{\sum_{s\in\{s8,s9,s10\}} P(s, \text{South}, s11) \cdot b0(s)}{0.765} \\ &= \frac{0.85 \cdot 0.8 + 0 \cdot 0.1 + 0 \cdot 0.1}{0.765} \\ &= \frac{0.85 \cdot 0.8}{0.765} \\ &= \frac{0.68}{0.765} \\ &\approx 0.889 \end{aligned}$$

$$\begin{aligned} b2(s12) &= \frac{\sum_{s\in\{s8,s9,s10\}} P(s, \text{South}, s12) \cdot b0(s)}{0.765} \\ &= \frac{0 \cdot 0.8 + 0.85 \cdot 0.1 + 0 \cdot 0.1}{0.765} \\ &= \frac{0.85 \cdot 0.1}{0.765} \\ &= \frac{0.085}{0.765} \\ &\approx 0.111 \end{aligned}$$

We hence end up with the belief state $b2$:

$$b2\colon \{s11 \mapsto 0.889, s12 \mapsto 0.111\}$$

Next, we compute the transition $\mathcal{P}(b0, \text{South}, b2)$. We defined $b2$ as the belief state that describes the belief after the agent takes action South and observes observation ESW starting in belief state $b0$, so $\text{SE}(b0, \text{South}, o) = b2$ iff observation $o = \text{ESW}$. Therefore we get that:

$$\Pr(\, b2 \mid \text{South}, b0, o\,) \begin{cases} 1 & \text{if } o = \text{ESW} \\ 0 & \text{otherwise} \end{cases}$$

We can use this to simplify the transition computation:

$$\begin{aligned} \mathcal{P}(b0, \text{South}, b2) &= \sum_{o\in Z} \Pr(\, b2 \mid \text{South}, b0, o\,) \cdot \Pr(\, o \mid \text{South}, b0\,) \\ &= \Pr(\, b2 \mid \text{South}, b0, \text{ESW}\,) \cdot \Pr(\, \text{ESW} \mid \text{South}, b0\,) \\ &= \Pr(\, \text{ESW} \mid \text{South}, b0\,) \end{aligned}$$

We end up with the normalization constant, which we already computed for the belief state.

$$\mathcal{P}(b0, \text{South}, b2) = 0.765$$

4.4 Model uncertainty

Another way of making MDPs more applicable to real-world sequential decision-making problems is to relax the full model knowledge requirement. It is often difficult or impossible to precisely estimate the probabilities of the transitions in the MDP models. Consider, for example, incomplete, insufficient, or noisy data or probabilities prone to changes in external factors such as temperature. Model uncertainty extends the MDP to the *uncertain MDP* (uMDP) and the POMDP to the *uncertain POMDP* (uPOMDP).

Uncertain models use uncertainty sets instead of exact probabilities. It is common to assume the uncertainty set is convex. Throughout this thesis, we will focus on interval uncertainty sets, as this is a straightforward approach which is also the focus in most other uPOMDP research [Cubuktepe et al., 2021, Suilen et al., 2020, Itoh and Nakamura, 2007, Ahmadi et al., 2018] and is a subfield in uMDP research [Wu and Koutsoukos, 2008, Givan et al., 2000]. An element from the uncertainty set, an instantiation, induces a nominal version of the model.

4.4.1 Uncertain MDPs

Definition 6: Uncertain Markov decision process

A uMDP is a tuple $\boldsymbol{M} = (S, A, \boldsymbol{P}, R)$, where:

- S the finite set of states.
- A the finite set of actions.
- $\boldsymbol{P}\colon S \times A \to (S \to \mathbb{I}_{\mathbb{P}})$ the (partial) uncertain state transition function.
- $R\colon S \times A \to \mathbb{R}$ the optional reward function.

We use $\boldsymbol{P}(s, a, s')$ as shorthand notation for $\boldsymbol{P}(s, a)(s')$. Moreover, we use $\boldsymbol{P}(\cdot, a)\colon S \to (S \to \mathbb{I}_{\mathbb{P}})$ to denote the uncertain state transition function limited to the a-transitions, so $\boldsymbol{P}(\cdot, a)(s) = \boldsymbol{P}(s, a)$.

Example

We can extend the cheese maze MDP of Section 4.2.2 to a uMDP by adding an uncertainty set. We now assume the mouse's chance of reaching its intended square lies between 85% and 95%. Furthermore, we assume the mouse's chance of slipping and staying in the same square lies between 5% and 15%. Note that these intervals do not have to match up; they only need to allow valid instantiations. Figure 4.8 illustrates the cheese maze uMDP.

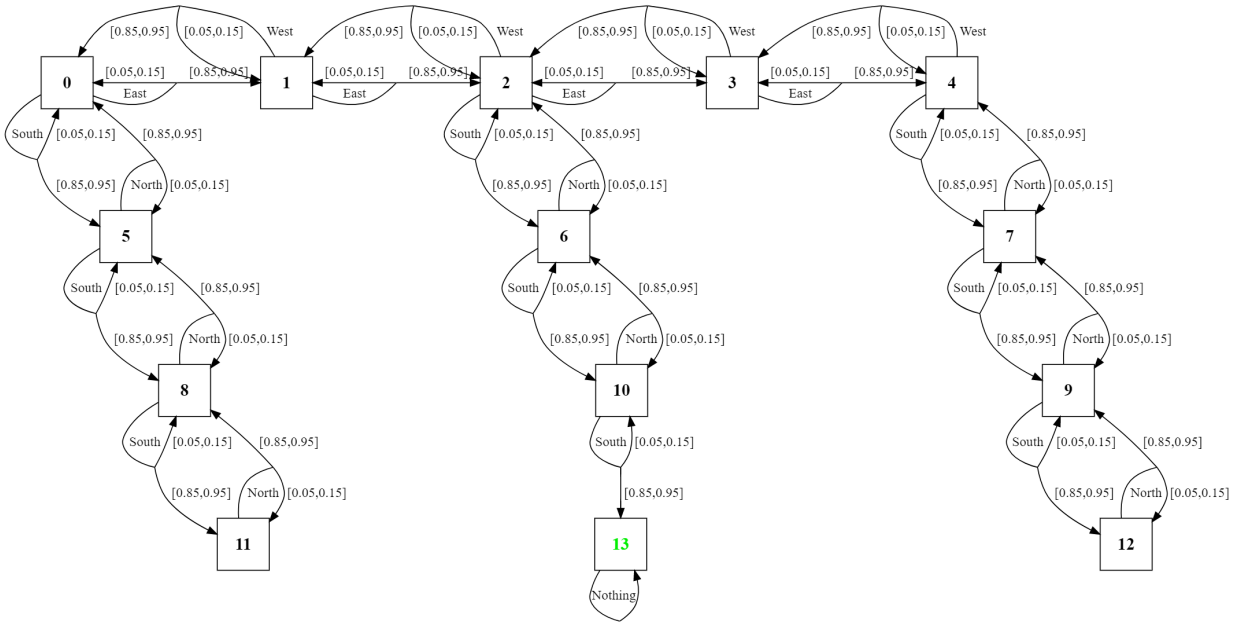


Figure 4.8: Cheese maze uMDP.

4.4.2 Uncertain POMDPs

Definition 7: Uncertain partially observable Markov decision process

A uPOMDP is a tuple $\boldsymbol{M} = (S, A, \boldsymbol{P}, R, Z, \boldsymbol{O})$, where:

- S the finite set of states.
- A the finite set of actions.
- $\boldsymbol{P}\colon S \times A \to (S \to \mathbb{I}_{\mathbb{P}})$ the (partial) uncertain state transition function.
- $R\colon S \times A \to \mathbb{R}$ the optional reward function.
- Z the finite set of observations.
- $\boldsymbol{O}\colon S \times Z \to \mathbb{I}_{\mathbb{P}}$ the uncertain observation function.

Like with standard POMDPs, the observation function can be simplified to a state-based deterministic function $\boldsymbol{O}\colon S \to Z$ by expanding the state space [Chatterjee et al., 2016]. The upper and lower bounds of the transitions to the new states are computed by multiplying the upper bounds, respectively lower bounds, of the original transition and the observation intervals.

Example

We can extend the cheese maze MDP of Section 4.2.2 to a uPOMDP by adding an uncertainty set, observation set, and observation function. We use the same uncertainty set and uncertain state transition function as in Section 4.4.1 and the same observation set and observation function as in Section 4.3.3. Figure 4.9 illustrates the cheese maze uPOMDP.

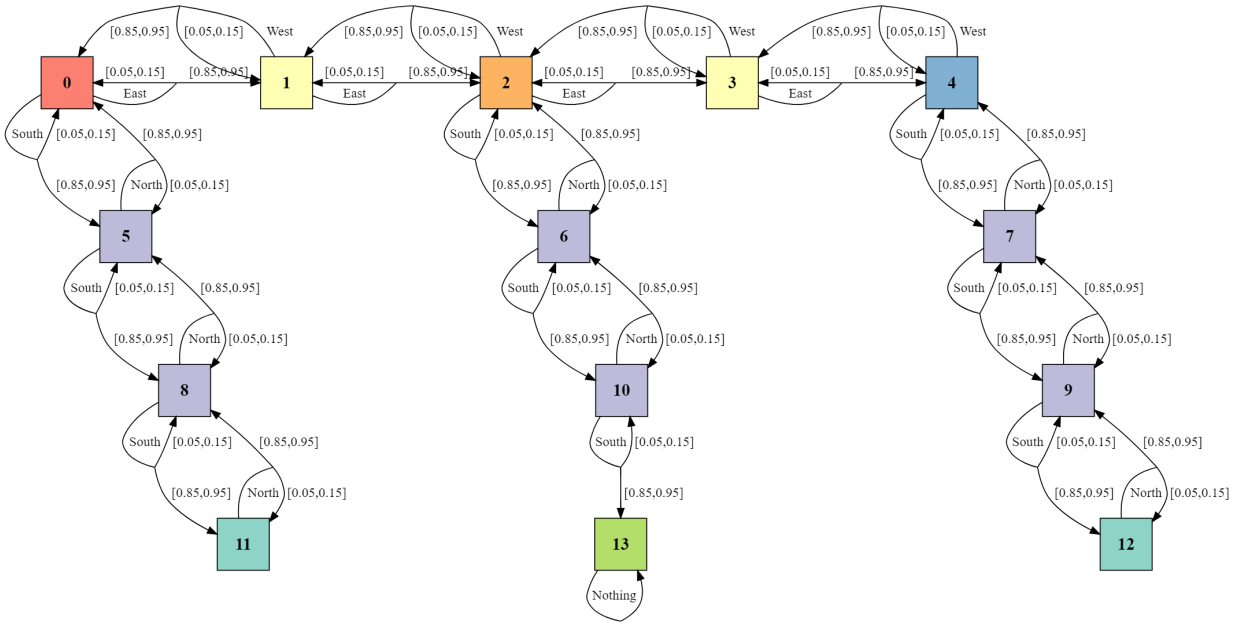


Figure 4.9: Cheese maze uPOMDP.

4.4.3 Model assumptions

We make three assumptions about the uncertain models. First, we assume graph preservation, which states that all lower bounds are strictly greater than 0. In other words, instantiations cannot eliminate transitions [Suilen et al., 2020, Winkler et al., 2019].

Furthermore, we assume (s,a)-rectangularity, which states that the transition function is independent for each state-action pair [Iyengar, 2005, Wiesemann et al., 2013].

$$\boldsymbol{P} = \bigtimes_{s \in S, a \in A} \boldsymbol{P}(s,a)$$

Our final assumption is that the models are dynamic, which means that whenever a state action pair is revisited, a different probability can be chosen from the uncertainty set [Iyengar, 2005].

Chapter 5

Underlying belief model for uPOMDPs

In this chapter, we discuss our new framework for approximating the underlying belief model of the uPOMDP: the *belief uncertain Markov decision process* (belief uMDP). We introduce the uncertain belief state in Section 5.1, followed by the formal definition of the belief uMDP in Section 5.2. In Sections 5.3 to 5.5, we discuss the uncertain versions of the belief update, belief state transition, and reward functions that follow from this definition. In Section 5.6, we show the correctness of our definition. We illustrate the belief uMDP framework on the running example in Section 5.7. Finally, we discuss the benefits of the belief uMDP framework in Section 5.8.

Note that, as explained in the Introduction, we refer to our approximative definition of the underlying belief model as the belief uMDP and the true underlying belief model as the true belief uMDP.

5.1 Uncertain belief state

To define the underlying belief model of the uPOMDP, we need to define the notion of an uncertain belief state. Since the uncertain belief state must represent the belief of each belief MDP in the true belief uMDP, a single probability distribution no longer suffices to capture the action-observation history of the agent: Even if we start with a single probability distribution as belief when taking a transition and receiving an observation, we do not know with how much chance we ended up in state s or in state s'. We hence cannot form a new probability distribution over the states to represent the agent's new belief.

We do, however, know the intervals in which the chances to transition to each state lie. Each probability distribution contained in the uncertainty intervals of the uncertain state transition function, i.e., each instantiation $P \in \boldsymbol{P}$, gives rise to a new belief probability distribution. To represent the action-observation history of the agent, the new uncertain belief state must contain all true possible belief probability distributions the agent can have given the uncertainty in the state transition function and the initial belief.

We define the uncertain belief state $\boldsymbol{B}$ as a convex polytope over $|S|$ variables, representing the belief in each state of the uPOMDP. The convex polytope consists of at least $2|S| + 2$

linear inequalities, which we can divide into three sets $\boldsymbol{B}_{ui}$, $\boldsymbol{B}_{pd}$, and $\boldsymbol{B}_{dc}$. The subscripts ui, pd, and cd stand for uncertainty interval, probability distribution, and distribution constraint.

The first set, $\boldsymbol{B}_{ui}$, always contains $2|S|$ inequalities that each restrict a single variable. Let $\vec{y}$ be the $|S|$-sized variable vector over which the uncertain belief states are defined, where y_i represents the belief in state i. For each $i \in S$, $\boldsymbol{B}_{ui}$ contains the following two inequalities:

$$-y_i \leq -a_i,$$
$$y_i \leq b_i,$$

where parameters a_i and b_i are the upper and lower bounds of the belief uncertainty interval of i. This belief uncertainty interval contains all probabilities the belief in state i could be given the action-observation history and initial belief. For simplicity, we can represent $\boldsymbol{B}_{ui}$ as a set of $|S|$ probability intervals: $\boldsymbol{B}_{ui} \in \mathbb{I}_{\mathbb{P}}^{|S|}$. We use the function $\boldsymbol{B}_{ui} \colon S \to \mathbb{I}_{\mathbb{P}}$ to indicate the belief uncertainty intervals of the different states of the uPOMDP.

The second set, $\boldsymbol{B}_{pd}$, always consists of two inequalities that ensure that the belief vectors contained in the convex polytope are valid probability distributions:

$$\sum_{i \in S} -y_i \leq -1,$$
$$\sum_{i \in S} y_i \leq 1.$$

The final set, $\boldsymbol{B}_{dc}$, is a placeholder for any additional constraints on the distribution of the belief over the states of the uPOMDP. The necessity and content of this set will be explained in Sections 5.3.2 and 5.3.3, respectively.

We denote the set of all uncertain belief states as $\boldsymbol{\mathcal{B}}$. We associate to each uncertain belief state $\boldsymbol{B} \in \boldsymbol{\mathcal{B}}$ a tuple $(\boldsymbol{B}_{ui}, \boldsymbol{B}_{pd}, \boldsymbol{B}_{dc})$ containing the three sets of linear inequalities as defined in the paragraphs above. Furthermore, we write $b \in \boldsymbol{B}$ to denote that belief $b \in \Delta(S)$ is an instantiation of uncertain belief state $\boldsymbol{B}$, meaning b is a solution of the system of linear inequalities formed by $\boldsymbol{B}_{ui}$, $\boldsymbol{B}_{pd}$, and $\boldsymbol{B}_{dc}$.

5.2 Definition

Definition 8: Belief uncertain Markov decision process

Given uPOMDP $\boldsymbol{M} = (S, A, \boldsymbol{P}, R, Z, \boldsymbol{O})$, we define the belief uMDP as a tuple $\boldsymbol{\mathcal{M}} = (\boldsymbol{\mathcal{B}}, A, \boldsymbol{\mathcal{P}}, \boldsymbol{\mathcal{R}})$, where:

- $\boldsymbol{\mathcal{B}}$ the set of uncertain belief states.
- A the set of actions.
- $\boldsymbol{\mathcal{P}} \colon \boldsymbol{\mathcal{B}} \times A \to (\boldsymbol{\mathcal{B}} \to \mathbb{I}_{\mathbb{P}})$ the (partial) uncertain belief state transition function.
- $\boldsymbol{\mathcal{R}} \colon \boldsymbol{\mathcal{B}} \times A \to \mathbb{I}_{\mathbb{R}}$ the optional uncertain reward function.

We use $\boldsymbol{\mathcal{P}}(\boldsymbol{B}, a, \boldsymbol{B'})$ as shorthand notation for $\boldsymbol{\mathcal{P}}(\boldsymbol{B}, a)(\boldsymbol{B'})$. A new uncertain belief state is computed given an old uncertain belief state, an action, and an observation, using the so-called *uncertain belief update function*. We discuss the uncertain belief update in Section 5.3.

5.2.1 Interchangeable parts

The computation methods of the uncertain belief update, belief state transition, and reward functions are interchangeable: it does not influence the structure of the belief uMDP, only the values. Where exact computation of the uncertainty intervals is impossible, the computed intervals should over-approximate the true probability intervals to ensure the belief uMDP over-approximates the true belief uMDP.

Corollary 5.2.1:

$$\{f(x) \mid x \in [\text{lb}, \text{ub}]\} \subseteq \{f(x) \mid x \in [\text{lb} - d, \text{ub} + e]\}$$

The true uncertainty interval should hence always be a subinterval of the computed interval.

5.2.2 Horizon

Just like the standard belief states of the belief MDP, the uncertain belief states of the belief uMDP can produce at most $|A| \times |Z|$ new uncertain belief states. Since we defined the set of actions A and the set of observations Z as finite sets, looking at a finite horizon will also result in a finite belief uMDP.

5.3 Uncertain belief update

Given a uPOMDP $\boldsymbol{M} = (S, A, \boldsymbol{P}, R, Z, \boldsymbol{O})$, an old uncertain belief state $\boldsymbol{B} \in \boldsymbol{\mathcal{B}}$, and an action $a \in A$, we compute a successor uncertain belief state $\boldsymbol{B'} \in \boldsymbol{\mathcal{B}}$ for each possible observation $o \in Z$. We do this via the uncertain belief update, which consists of two steps: computing $\boldsymbol{B'_{ui}}$ and computing $\boldsymbol{B'_{dc}}$. Note that we do not need to compute $\boldsymbol{B'_{pd}}$, as the probability distribution constraints remain the same for all uncertain belief states: $\boldsymbol{B'_{pd}} = \boldsymbol{B_{pd}}$.

As explained in Section 5.1, an uncertain belief state needs to contain all true possible beliefs given the action-observation history and the initial uncertain belief. To ensure that the successor uncertain belief state $\boldsymbol{B'}$ still satisfies this, we need to take into account all the possible instantiated beliefs b of the old uncertain belief state $\boldsymbol{B}$, and all instantiated state transition functions P of the uncertain state transition function $\boldsymbol{P}$.

5.3.1 Belief uncertainty intervals

Given an instantiated belief $b \in \boldsymbol{B}$ and state transition function $P \in \boldsymbol{P}$, we can use the belief update of standard POMDPs to compute a new possible belief b'. $\forall s' \in S$:

$$\begin{aligned} b'(s') &= \frac{\boldsymbol{O}(s', o) \sum_{s \in S} P(s, a, s') b(s)}{\Pr(\, o \mid a, b\,)} \\ &= \frac{\boldsymbol{O}(s', o) \sum_{s \in S} P(s, a, s') b(s)}{\sum_{s' \in S} \boldsymbol{O}(s', o) \sum_{s \in S} P(s, a, s') b(s)} \end{aligned}$$

To capture all new possible beliefs b', we can maximize and minimize this belief update function over the instantiations of the old uncertain belief state $\boldsymbol{B}$ and uncertain state transition function $\boldsymbol{P}$. Let $\boldsymbol{B'}$ be the successor uncertain belief state after taking action $a \in A$ from

old uncertain belief state $\boldsymbol{B}$ and observing observation $o \in Z$. Given state $s' \in S$, we get the following optimization problem to compute $\boldsymbol{B'}_{ui}(s')$:

Let $n = |S|$. We use the optimization variable:

$$\vec{x} = \begin{bmatrix} x_{1,1} & \dots & x_{n,n} & y_1 & \dots & y_n \end{bmatrix}$$

This optimization variable consists of n^2 elements representing the transitions between the states and n elements representing the belief in each state. We distinguish the subvectors that group the transition variables per source state:

$$\vec{x_i} = \begin{bmatrix} x_{i,1} & \dots & x_{i,n} \end{bmatrix}, i \in [n]$$

And the subvector that groups the belief variables:

$$\vec{y} = \begin{bmatrix} y_1 & \dots & y_n \end{bmatrix}$$

Then we need to minimize/maximize:

$$\frac{\boldsymbol{O}(s', o) \sum_{s \in S} x_{s,s'} y_s}{\sum_{s' \in S} \boldsymbol{O}(s', o) \sum_{s \in S} x_{s,s'} y_s}$$

Subject to:

$$\begin{aligned} &\vec{x_i} \in \boldsymbol{P}(i, a), \quad i \in [n] \\ &\vec{y} \in \boldsymbol{B} \end{aligned}$$

The objective function of this optimization problem is non-convex and requires some form of approximation. We elaborate on this in Chapter 6. Approximating this problem gives us upper and lower bounds for the belief uncertainty interval of one state s' in the new uncertain belief state $\boldsymbol{B'}$. To compute the whole new linear inequality set $\boldsymbol{B'}_{ui}$, this problem must be solved for each state $s \in S$ with $\boldsymbol{O}(s) = o$, since states with a different observation will always get a zero interval.

5.3.2 Information loss

Having computed the belief uncertainty intervals $\boldsymbol{B'}_{ui}$, we now know the intervals containing at least all possible probabilities the true belief could be in uncertain belief state $\boldsymbol{B'}$. This sounds like we have reached our goal: the uncertain belief state contains all true possible beliefs given the action-observation history and the initial uncertain belief. However, by only considering the new belief uncertainty intervals $\boldsymbol{B'}_{ui}$, we overapproximate the set of true possible beliefs more than necessary: The old belief uncertainty intervals $\boldsymbol{B}_{ui}$ contain information about the distribution of the belief which is not represented in the new uncertainty intervals[1].

[1]Thanks to Merlijn Krale for pointing this out to me.

We illustrate the information loss using the partial uPOMDP in Figure 5.1. We only consider the first six states; the rest of the uPOMDP is not required. The six states all have the same observation, indicated by the shape of the state, and there is only a single possible action from this observation, namely α.

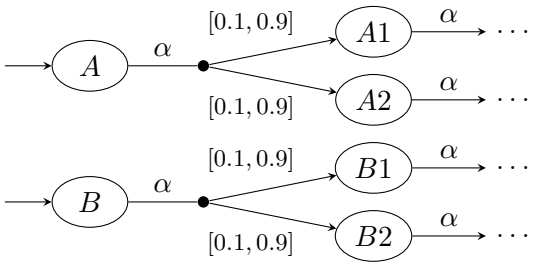


Figure 5.1: uPOMDP information loss example.

Let $\boldsymbol{B}^{\text{init}} \in \boldsymbol{\mathcal{B}}$ be the initial uncertain belief state. For simplicity, we assume the initial uncertain belief is known and evenly divided over states A and B. There are no additional constraints yet: $\boldsymbol{B}^{\text{init}}_{dc} = \emptyset$. We compute the successor uncertain belief state $\boldsymbol{B}^{\text{suc}} \in \boldsymbol{\mathcal{B}}$ using the belief update function as defined up to this point. The belief uncertainty intervals will then change as follows:

$$
\begin{array}{rcll}
\boldsymbol{B}^{\text{init}}_{ui} = \{ & A & \mapsto & [0.5, 0.5], \\
& B & \mapsto & [0.5, 0.5], \\
& A1 & \mapsto & [0, 0], \\
& A2 & \mapsto & [0, 0], \\
& B1 & \mapsto & [0, 0], \\
& B2 & \mapsto & [0, 0]\}
\end{array}
\quad \Longrightarrow \quad
\begin{array}{rcll}
\boldsymbol{B}^{\text{suc}}_{ui} = \{ & A & \mapsto & [0, 0], \\
& B & \mapsto & [0, 0], \\
& A1 & \mapsto & [0.05, 0.45], \\
& A2 & \mapsto & [0.05, 0.45], \\
& B1 & \mapsto & [0.05, 0.45], \\
& B2 & \mapsto & [0.05, 0.45]\}
\end{array}
$$

As mentioned before, the probability distribution set does not change: $\boldsymbol{B}^{\text{suc}}_{pd} = \boldsymbol{B}^{\text{init}}_{pd}$. Furthermore, as we have not defined the computation yet, the additional distribution constraint set remains empty: $\boldsymbol{B}^{\text{suc}}_{dc} = \emptyset$.

If we now look at the probability distributions contained in the belief uncertainty intervals $\boldsymbol{B}^{\text{suc}}_{ui}$, we find, among others, the following belief $b \in \boldsymbol{B}^{\text{suc}}_{ui}$:

$$
\begin{array}{rcll}
b = \{ & A & \mapsto & 0, \\
& B & \mapsto & 0, \\
& A1 & \mapsto & 0.45, \\
& A2 & \mapsto & 0.45, \\
& B1 & \mapsto & 0.05, \\
& B2 & \mapsto & 0.05\}
\end{array}
$$

The problem with belief b is that 90% of the belief ended up in the successor states of states A, while state A originally only contained 50% of the belief. This is hence a belief that is contained in the successor belief state $\boldsymbol{B}^{\text{suc}}$, while we know it is not a true possible belief.

Every time we ignore the belief distribution of the old uncertain belief state during the computation of a successor belief state, we lose information and end up with unnecessary uncertainty. To limit this information loss and, hence, restrict the distribution of the belief over the states, we add linear inequalities to the $\boldsymbol{B}_{dc}$ set of an uncertain belief state $\boldsymbol{B}$.

5.3.3 Belief distribution constraints

There are two difficulties with defining the additional distribution constraints $\boldsymbol{B}_{dc}$: states with multiple possible belief sources and normalization of the belief. We first look at the simplest case as illustrated in Figure 5.1, where there are no states with multiple incoming transitions, and all states have the same observation. This means we do not yet need to take into account multiple sources of belief or normalization.

Given an uncertain belief state $\boldsymbol{B} \in \boldsymbol{\mathcal{B}}$, action $a \in A$ and observation $o \in Z$, let $\boldsymbol{B'}$ be the successor uncertain belief state after taking action a and receiving observation o, computed with the uncertain belief update as defined up to this point. Let $\vec{y}$ be the variable over which the uncertain belief states are defined.

We define the set of observable successors of a state $s \in S$ given action a and observation o:

$$\text{OS}(s, a, o) = \{s' \in S \mid \boldsymbol{P}(s, a, s') \neq [0, 0] \wedge \boldsymbol{O}(s') = o\}.$$

These observable successors are exactly those states to which the belief in state s could go and are, therefore, also the states that we need to constrain by the amount of uncertain belief in s. Given $\boldsymbol{B}_{ui}(s) = [v_s, w_s]$, we compute two linear inequalities for each state $s \in S$:

$$\sum_{s' \in \text{OS}(s,a,o)} -y_{s'} \leq -v_s,$$

$$\sum_{s' \in \text{OS}(s,a,o)} y_{s'} \leq w_s.$$

Note that we can limit the computation of linear inequalities to those states with a non-zero belief in old uncertain belief state $\boldsymbol{B}$. The number of linear inequalities in set $\boldsymbol{B'}_{dc}$ is at most $2|S|$ but will, in practice, often be lower. We write all linear inequalities in $\boldsymbol{B'}_{dc}$ as $\leq$-inequalities.

Multiple belief sources

Now that we have the base version of the constraints, we can extend them to work for uP-OMDPs with states with multiple belief sources. We define a belief source as follows:

> **Definition 9: Belief source**
>
> Given a state $s' \in S$, its belief sources are those states $s \in S$ of which s' could receive belief, i.e., the states with an uncertain transition to state s':
>
> $$\{s \in S \mid \boldsymbol{P}(s, a, s') \neq [0, 0]\}$$

Figure 5.2 illustrates a uPOMDP with states with multiple belief sources: State AB is a successor state of both state A and state B and can therefore receive belief from both. Note that all states still have the same observation, so we do not yet have to take normalization into account.

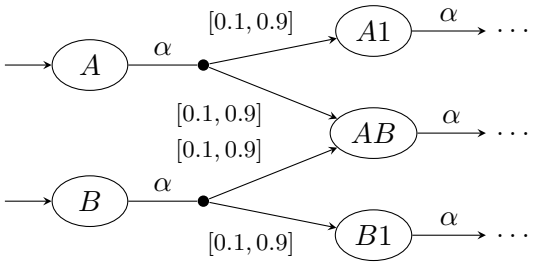


Figure 5.2: uPOMDP information loss example multiple belief source.

If we now try to use the simple distribution constraints defined in the previous subsection, we constrain the total uncertain belief of state AB by the bounds of a state that only causes part of that uncertain belief. One way to circumvent this is to combine the belief distribution constraints of all states that share an observable successor. The problem with that approach is that the constraints become a lot less strict. Thus, we instead compute the intervals of each state's belief contribution to the new uncertain belief of its observable successors.

Given an instantiated belief $b \in \boldsymbol{B}$ and state transition function $P \in \boldsymbol{P}$, we can compute the belief contribution of state s to the belief in state s' in successor belief $b' \in \boldsymbol{B'}$ as follows:

$$\frac{\boldsymbol{O}(s', o)P(s, a, s')b(s)}{\boldsymbol{O}(s', o)\sum_{s \in S} P(s, a, s')b(s)}$$

To capture the bounds of the belief contribution of s' to the uncertain belief of state s', we can maximize and minimize this function over the instantiations of the old uncertain belief state $\boldsymbol{B}$ and uncertain state transition function $\boldsymbol{P}$. Using the same optimization variable and subvectors as in Section 5.3.1, and given states $s, s' \in S$, we get the following optimization problem:

Minimize/maximize:

$$\frac{\boldsymbol{O}(s', o)x_{s,s'}y_s}{\boldsymbol{O}(s', o)\sum_{s \in S} x_{s,s'}y_s}$$

Subject to:

$$\vec{x_i} \in \boldsymbol{P}(i, a), \quad i \in [n]$$
$$\vec{y} \in \boldsymbol{B}$$

The objective function of this optimization problem is non-convex and requires some form of approximation. We elaborate on this in Chapter 6. Note that if state s is the only belief source for state s', the numerator and denominator become the same, resulting in the required total contribution.

Let $\widehat{r_{s,s'}}$ and $r_{s,s'}$ be the results of minimizing and maximizing this optimization problem for states $s, s' \in \widetilde{S}$, respectively. So $\widehat{r_{s,s'}}$ and $r_{s,s'}$ are the upper and lower bounds of the belief contribution of state s to the new uncertain belief of state s'. Given $\boldsymbol{B}_{ui}(s) = [v_s, w_s]$, we now get the following linear inequalities for each state $s \in S$:

$$\sum_{s' \in S} -\widehat{r_{s,s'}} y_{s'} \leq -v_s,$$
$$\sum_{s' \in S} \widecheck{r_{s,s'}} y_{s'} \leq w_s.$$

Note that we changed the summation over the observable successors to a summation over the set of states S. The contribution coefficients $\widehat{r_{s,s'}}$ and $\widecheck{r_{s,s'}}$ are zero for states outside of the observable successor set, which makes the restricted summation superfluous.

Normalization of distribution constraints

The final adjustment we need to make to the belief distribution constraints is to take normalization into account. Figure 5.3 shows a uPOMDP where we need to normalize the linear inequalities. In this figure, state $B2$ has a different observation from all the other states, indicated by the square state. This means that the belief in state B in an uncertain belief state will be divided over two successor uncertain belief states, one with observation oval and one with observation square; Neither uncertain belief state gets the full belief.

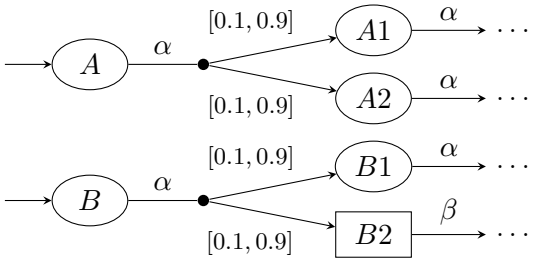


Figure 5.3: uPOMDP information loss example normalization.

To account for the lost belief in a successor uncertain belief state $\boldsymbol{B'}$, we have to normalize the linear inequalities in $\boldsymbol{B'}_{dc}$. Note that the belief uncertainty intervals $\boldsymbol{B'}_{ui}$ are already normalized by the denominator of the objective function.

The coefficients we added to the linear inequalities in $\boldsymbol{B'}_{dc}$ to deal with multiple belief sources are computed via a fraction of unnormalized terms. Normalizing these terms would cancel each other out. Therefore, the only aspect of the linear inequalities we still need to normalize is the constants, which, until now, were formed by the bounds of the uncertain belief intervals of old uncertain belief state $\boldsymbol{B}$. As the old uncertain belief in a state s can now end up in multiple successor uncertain belief states, we need to use the portion of the uncertain belief in state s that goes to the successor uncertain belief state under consideration to form the constants of the linear inequalities.

Given an instantiated belief $b \in \boldsymbol{B}$ and state transition function $P \in \boldsymbol{P}$, we can normalize the portion of the belief in a state s that goes to the successor belief $b' \in \boldsymbol{B'}$ with the total belief in b' as follows:

$$\frac{\sum_{s' \in S} \boldsymbol{O}(s', o) P(s, a, s') b(s)}{\sum_{s' \in S} \boldsymbol{O}(s', o) \sum_{s \in S} P(s, a, s') b(s)}$$

To capture the bounds of the normalized uncertain belief in state s that goes to successor uncertain belief state $\boldsymbol{B'}$, we can maximize and minimize this function over the instantiations of the old uncertain belief state $\boldsymbol{B}$ and uncertain state transition function $\boldsymbol{P}$. Using the same optimization variable and subvectors as in Section 5.3.1, and given state $s \in S$, we get the following optimization problem:

Minimize/maximize:

$$\frac{\sum_{s' \in S} \boldsymbol{O}(s', o) x_{s,s'} y_s}{\sum_{s' \in S} \boldsymbol{O}(s', o) \sum_{s \in S} x_{s,s'} y_s}$$

Subject to:

$$\vec{x_i} \in \boldsymbol{P}(i, a), \quad i \in [n]$$
$$\vec{y} \in \boldsymbol{B}$$

We elaborate on how to approximate this optimization problem in Chapter 6. Let $\widehat{t_s}$ and $\underset{\smile}{t_s}$ be the results of minimizing and maximizing this optimization problem for state $s \in S$, respectively. So $\widehat{t_s}$ and $\underset{\smile}{t_s}$ are the upper and lower bounds of the old uncertain belief in state s that goes to successor uncertain belief state $\boldsymbol{B'}$ normalized to the total belief of $\boldsymbol{B'}$. We now get the following linear inequalities for each state $s \in S$:

$$\sum_{s' \in S} -\widehat{r_{s,s'}} y_{s'} \leq -\underset{\smile}{t_s},$$
$$\sum_{s' \in S} \underset{\smile}{r_{s,s'}} y_{s'} \leq \widehat{t_s}.$$

These at most $2|S|$ linear inequalities form the $\boldsymbol{B'_{dc}}$ set of successor uncertain belief state $\boldsymbol{B'}$. Having computed both $\boldsymbol{B'_{ui}}$ and $\boldsymbol{B'_{dc}}$, the uncertain belief update is done, and the successor uncertain belief state $\boldsymbol{B'}$ is complete.

5.3.4 Correctness of the uncertain belief update function

Below, we show the correctness of the uncertain belief update, meaning that the successor beliefs b' of all valid beliefs $b \in \boldsymbol{B}$ in an uncertain belief state $\boldsymbol{B}$ are contained in the successor uncertain belief state $\boldsymbol{B'}$ computed with the uncertain belief update.

Theorem 5.3.1:

Let $\boldsymbol{B} \in \boldsymbol{\mathcal{B}}$ be an uncertain belief state and let $b \in \boldsymbol{B}$ be one of the valid beliefs contained in $\boldsymbol{B}$. Let $P \in \boldsymbol{P}$ be a valid state transition function. Furthermore, given an action $a \in A$ and observation $o \in Z$, let:

- $b' \in \Delta(S)$ be the successor belief of b after taking action a and observing observation o computed via the belief update function (Section 4.3.1, definition 4).
- $\boldsymbol{B'} \in \boldsymbol{\mathcal{B}}$ be the successor uncertain belief state of $\boldsymbol{B}$ after taking action a and observing observation o computed via the uncertain belief update function (Section 5.3).

Then we have:

$$b' \in \boldsymbol{B'}$$

Proof.
We proof $b' \in \boldsymbol{B'}$ in three steps: $b' \in \boldsymbol{B'_{ui}}$, $b' \in \boldsymbol{B'_{pd}}$, and $b' \in \boldsymbol{B'_{dc}}$.

Part A: $b' \in \boldsymbol{B'_{ui}}$

A1. Take an arbitrary state $s' \in S$.

A2. Let $\widehat{b_{s'}} = \max_{b \in \boldsymbol{B}, P \in \boldsymbol{P}} \frac{\boldsymbol{O}(s', o) \sum_{s \in S} P(s, a, s') b(s)}{\sum_{s' \in S} \boldsymbol{O}(s', o) \sum_{s \in S} P(s, a, s') b(s)}$
and $\underbrace{b_{s'}} = \min_{b \in \boldsymbol{B}, P \in \boldsymbol{P}} \frac{\boldsymbol{O}(s', o) \sum_{s \in S} P(s, a, s') b(s)}{\sum_{s' \in S} \boldsymbol{O}(s', o) \sum_{s \in S} P(s, a, s') b(s)}$ (Section 5.3.1).

A3. Then $\underbrace{b_{s'}} \leq b'(s') \leq \widehat{b_{s'}}$.

A4. By definition, $\boldsymbol{B'_{ui}}(s') = [\underbrace{b_{s'}}, \widehat{b_{s'}}]$.

A5. Then from (A3) and (A4), it follows that $b'(s') \in \boldsymbol{B'_{ui}}(s')$.

A6. Then from (A1) and (A5), it follows that $b' \in \boldsymbol{B'_{ui}}$.

Part B: $b' \in \boldsymbol{B'_{pd}}$

B1. $b' \in \Delta(S)$, so by definition $\sum_{s \in S} b'(s) = 1$.

b2. Hence, $b' \in \boldsymbol{B'_{pd}}$.

Part C: $b' \in \boldsymbol{B'_{dc}}$

C1. Assume $b' \notin \boldsymbol{B'_{dc}}$.

C2. Then $\exists s \in S$ such that $\sum_{s' \in S} \widehat{r_{s,s'}} b'(s') < \underbrace{t_s}$ or $\sum_{s' \in S} \underbrace{r_{s,s'}} b'(s') > \widehat{t_s}$ (Section 5.3.3).

C3. Let $s \in S$ be a state with $\sum_{s' \in S} \widehat{r_{s,s'}} b'(s') < \underbrace{t_s}$ or $\sum_{s' \in S} \underbrace{r_{s,s'}} b'(s') > \widehat{t_s}$.

C4. First assume $\sum_{s' \in S} \widehat{r_{s,s'}} b'(s') < \underbrace{t_s}$.

C5. By definition, we know $\forall s' \in S : r_{s,s'} \leq \widehat{r_{s,s'}}$ and $t_s \geq \underset{\smile}{t_s}$.

C6. From (C4) and (C5) it follows that $\sum_{s' \in S} r_{s,s'} b'(s') < t_s$.

C7. Filling in the definitions of $r_{s,s'}$, $b'(s')$ and t_s (Section 5.3.3), we get:

$$\sum_{s' \in S} \frac{\boldsymbol{O}(s',o)P(s,a,s')b(s)}{\boldsymbol{O}(s',o)\sum_{s \in S} P(s,a,s')b(s)} \cdot \frac{\boldsymbol{O}(s',o)\sum_{s \in S} P(s,a,s')b(s)}{\sum_{s' \in S} \boldsymbol{O}(s',o)\sum_{s \in S} P(s,a,s')b(s)} < \frac{\sum_{s' \in S} \boldsymbol{O}(s',o)P(s,a,s')b(s)}{\sum_{s' \in S} \boldsymbol{O}(s',o)\sum_{s \in S} P(s,a,s')b(s)}$$

$$\sum_{s' \in S} \frac{\boldsymbol{O}(s',o)P(s,a,s')b(s)}{\sum_{s' \in S} \boldsymbol{O}(s',o)\sum_{s \in S} P(s,a,s')b(s)} < \frac{\sum_{s' \in S} \boldsymbol{O}(s',o)P(s,a,s')b(s)}{\sum_{s' \in S} \boldsymbol{O}(s',o)\sum_{s \in S} P(s,a,s')b(s)}$$

$$\frac{\sum_{s' \in S} \boldsymbol{O}(s',o)P(s,a,s')b(s)}{\sum_{s' \in S} \boldsymbol{O}(s',o)\sum_{s \in S} P(s,a,s')b(s)} < \frac{\sum_{s' \in S} \boldsymbol{O}(s',o)P(s,a,s')b(s)}{\sum_{s' \in S} \boldsymbol{O}(s',o)\sum_{s \in S} P(s,a,s')b(s)}$$

C8. (C7) gives a contradiction, hence our assumption in (C4) is incorrect.

C9. Then, because of (C3), we must assume $\sum_{s' \in S} \underset{\smile}{r_{s,s'}} b'(s') > \widehat{t_s}$.

C10. By definition, we know $\forall s' \in S : r_{s,s'} \geq \underset{\smile}{r_{s,s'}}$ and $t_s \leq \widehat{t_s}$.

C11. From (C9) and (C10) it follows that $\sum_{s' \in S} r_{s,s'} b'(s') > t_s$.

C12. Filling in the definitions of $r_{s,s'}$, $b'(s')$ and t_s (Section 5.3.3), we get:

$$\sum_{s' \in S} \frac{\boldsymbol{O}(s',o)P(s,a,s')b(s)}{\boldsymbol{O}(s',o)\sum_{s \in S} P(s,a,s')b(s)} \cdot \frac{\boldsymbol{O}(s',o)\sum_{s \in S} P(s,a,s')b(s)}{\sum_{s' \in S} \boldsymbol{O}(s',o)\sum_{s \in S} P(s,a,s')b(s)} > \frac{\sum_{s' \in S} \boldsymbol{O}(s',o)P(s,a,s')b(s)}{\sum_{s' \in S} \boldsymbol{O}(s',o)\sum_{s \in S} P(s,a,s')b(s)}$$

$$\sum_{s' \in S} \frac{\boldsymbol{O}(s',o)P(s,a,s')b(s)}{\sum_{s' \in S} \boldsymbol{O}(s',o)\sum_{s \in S} P(s,a,s')b(s)} > \frac{\sum_{s' \in S} \boldsymbol{O}(s',o)P(s,a,s')b(s)}{\sum_{s' \in S} \boldsymbol{O}(s',o)\sum_{s \in S} P(s,a,s')b(s)}$$

$$\frac{\sum_{s' \in S} \boldsymbol{O}(s',o)P(s,a,s')b(s)}{\sum_{s' \in S} \boldsymbol{O}(s',o)\sum_{s \in S} P(s,a,s')b(s)} > \frac{\sum_{s' \in S} \boldsymbol{O}(s',o)P(s,a,s')b(s)}{\sum_{s' \in S} \boldsymbol{O}(s',o)\sum_{s \in S} P(s,a,s')b(s)}$$

C13. (C12) gives a contradiction, hence our assumption in (C10) is incorrect.

C14. From (C2), (C3), (C8), and (C13), it follows that (C1) is incorrect.

C15. Hence $b' \in \boldsymbol{B'_{dc}}$.

Since $b' \in \boldsymbol{B'_{ui}}$, $b' \in \boldsymbol{B'_{pd}}$, and $b' \in \boldsymbol{B'_{dc}}$, we know that $b' \in \boldsymbol{B'}$.

□

5.4 Uncertain belief state transitions

Having computed a new uncertain belief state $\boldsymbol{B'} \in \boldsymbol{\mathcal{B}}$ from an old uncertain belief state $\boldsymbol{B} \in \boldsymbol{\mathcal{B}}$ using action $a \in A$ and resulting observation $o \in Z$, we need to compute the uncertain a-transition between $\boldsymbol{B}$ and $\boldsymbol{B'}$. We combine the a-transitions in the uPOMDP of all states with a non-zero uncertain belief in uncertain belief state $\boldsymbol{B}$ to all states with observation o. The a-transitions in the uPOMDP are intervals, and the resulting a-transition in the belief uMDP needs to be an interval as well.

Given an instantiated belief $b \in \boldsymbol{B}$ and instantiated state transition function $P \in \boldsymbol{P}$, we can compute the combined belief state transition probability $\mathcal{P}$, which is an instantiation of the uncertain belief state transition function $\boldsymbol{\mathcal{P}}$, for action a to the successor belief $b' \in \boldsymbol{B'}$ with $\text{SE}(b, a, o) = b'$ as follows:

$$\mathcal{P}(b, a, b') = \sum_{s' \in S} \boldsymbol{O}(s', o) \cdot \sum_{s \in S} P(s, a, s') b(s)$$

To capture all true possible belief state transition probabilities $\mathcal{P}$, we can maximize and minimize the above function over the instantiations of the uncertain belief state $\boldsymbol{B}$ and uncertain state transition function $\boldsymbol{P}$. Using the same optimization variable and subvectors as in Section 5.3.1, we get the following optimization problem to compute $\boldsymbol{\mathcal{P}}(\boldsymbol{B}, a, \boldsymbol{B'})$:

Minimize/maximize:

$$\sum_{s' \in S} \boldsymbol{O}(s', o) \sum_{s \in S} x_{s,s'} y_s$$

Subject to:

$$\begin{aligned} &\vec{x_i} \in \boldsymbol{P}(i, a), \quad i \in [n] \\ &\vec{y} \in \boldsymbol{B} \end{aligned}$$

We elaborate on how to solve this optimization problem in Chapter 6. Note that this function is the same as the denominator of the belief uncertainty interval function. This is the uncertain normalization constant, which represents how much of the belief of the old uncertain belief state goes to this particular successor uncertain belief state.

5.4.1 Correctness of the uncertain belief state transition function

Below, we show the correctness of the uncertain belief state transition function, meaning that the a-transitions between all valid beliefs $b \in \boldsymbol{B}$ in an uncertain belief state $\boldsymbol{B}$ and their successor beliefs b' (which are contained in $\boldsymbol{B'}$ as shown by Theorem 5.3.1) are contained in $\boldsymbol{\mathcal{P}}(\boldsymbol{B}, a, \boldsymbol{B'})$.

Theorem 5.4.1:

Let $\boldsymbol{B} \in \boldsymbol{\mathcal{B}}$ be an uncertain belief state and let $b \in \boldsymbol{B}$ be one of the valid beliefs contained in $\boldsymbol{B}$. Let $P \in \boldsymbol{P}$ be a valid state transition function. Furthermore, given an action $a \in A$ and observation $o \in Z$, let:

- $b' \in \Delta(S)$ be the successor belief of b after taking action a and observing observation o computed via the belief update function (Section 4.3.1, definition 4).
- $\boldsymbol{B'} \in \boldsymbol{\mathcal{B}}$ be the successor uncertain belief state of $\boldsymbol{B}$ after taking action a and observing observation o computed via the uncertain belief update function (Section 5.3).
- $p = \mathcal{P}(b, a, b') \in [0, 1]$ be the a-transition between b and b' computed via the belief state transition function (Section 4.3.1, definition 5).
- $\boldsymbol{p} = \boldsymbol{\mathcal{P}}(\boldsymbol{B}, a, \boldsymbol{B'}) \in \mathbb{I}_{\mathbb{P}}$ be the uncertain a-transition between $\boldsymbol{B}$ and $\boldsymbol{b'}$ computed via the uncertain belief state transition function (Section 5.4).

Then we have:

$$p \in \boldsymbol{p}$$

Proof.

1. Let $\widehat{p} = \max\limits_{b \in \boldsymbol{B}, P \in \boldsymbol{P}} \sum\limits_{s' \in S} \boldsymbol{O}(s', o) \cdot \sum\limits_{s \in S} P(s, a, s') b(s)$
   and $\underset{\smile}{p} = \min\limits_{b \in \boldsymbol{B}, P \in \boldsymbol{P}} \sum\limits_{s' \in S} \boldsymbol{O}(s', o) \cdot \sum\limits_{s \in S} P(s, a, s') b(s)$ (Section 5.4).
2. Then $\underset{\smile}{p} \leq p \leq \widehat{p}$.
3. By definition, $\boldsymbol{p} = [\underset{\smile}{p}, \widehat{p}]$.
4. Then from (2) and (3), it follows that $p \in \boldsymbol{p}$.

□

5.5 Uncertain rewards

Given an uncertain belief state $\boldsymbol{B} \in \boldsymbol{\mathcal{B}}$ and action $a \in A$, we combine the reward for taking that action for all states with a non-zero uncertain belief in $\boldsymbol{B}$. Since the uncertain belief of a state is an interval, the reward will also be an interval.

Given an instantiation $b \in \boldsymbol{B}$, we can compute the combined reward for taking action a as follows:

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

To capture all true possible rewards $\mathcal{R} \in \boldsymbol{\mathcal{R}}$, we can maximize and minimize the above function over the instantiations of the uncertain belief state $\boldsymbol{B}$. We get the following optimization problem to compute $\boldsymbol{\mathcal{R}}(\boldsymbol{B}, a)$:

We use the optimization variable:

$$\vec{y} = \begin{bmatrix} y_1 & \dots & y_n \end{bmatrix}$$

Then we need to minimize/maximize:

$$\sum_{s \in S} y_s R(s, a)$$

Subject to:

$$\vec{y} \in \boldsymbol{B}$$

We again elaborate on how to solve this optimization problem in Chapter 6.

5.5.1 Correctness of the uncertain reward function

Below, we show the correctness of the uncertain reward function, meaning that the a-rewards of all valid beliefs $b \in \boldsymbol{B}$ in an uncertain belief state $\boldsymbol{B}$ are contained in $\boldsymbol{\mathcal{R}}(\boldsymbol{B}, a)$.

Theorem 5.5.1:

Let $\boldsymbol{B} \in \boldsymbol{\mathcal{B}}$ be an uncertain belief state and let $b \in \boldsymbol{B}$ be one of the valid beliefs contained in $\boldsymbol{B}$. Furthermore, given an action $a \in A$, let:

- $r = \mathcal{R}(b, a) \in \mathbb{R}$ be the reward after taking action a from belief b computed via the reward function (Section 4.3.1, definition 5).
- $\boldsymbol{r} = \boldsymbol{\mathcal{R}}(\boldsymbol{B}, a) \in \mathbb{I}_{\mathbb{R}}$ be the uncertain reward after taking action a from uncertain belief state $\boldsymbol{B}$ computed via the uncertain reward function (Section 5.5).

Then we have:

$$r \in \boldsymbol{r}$$

Proof.

1. Let $\widehat{r} = \max\limits_{b \in \boldsymbol{B}} \sum\limits_{s \in S} b(s) R(s, a)$
   and $\underset{\smile}{r} = \min\limits_{b \in \boldsymbol{B}} \sum\limits_{s \in S} b(s) R(s, a)$ (Section 5.5).
2. Then $\underset{\smile}{r} \leq r \leq \widehat{r}$.
3. By definition, $\boldsymbol{r} = [\underset{\smile}{r}, \widehat{r}]$.
4. Then from (2) and (3), it follows that $r \in \boldsymbol{r}$.

□

5.6 Correctness of the belief uMDP

Now that we have our definition of the belief uMDP, we show that it indeed over-approximates the true belief uMDP, meaning the belief MDP of all POMDPs of the uPOMDP are contained in the belief uMDP.

Note that we need an initial (uncertain) belief to compute the belief MDPs and belief uMDP. We can manually set this. The initial uncertain belief state needs to contain all true initial beliefs. We include this assumption in our correctness theorem.

Theorem 5.6.1:

Let $\boldsymbol{M}$ be a uPOMDP and let $M \in \mathcal{M}$ be one of the POMDPs in $\boldsymbol{M}$. Furthermore, let:

- $\mathcal{M} = (\mathcal{B}, A, \mathcal{P}, \mathcal{R})$ be the belief MDP of POMDP M.
- $\boldsymbol{\mathcal{M}} = (\boldsymbol{\mathcal{B}}, A, \boldsymbol{\mathcal{P}}, \boldsymbol{\mathcal{R}})$ be the belief uMDP of uPOMDP $\boldsymbol{M}$ as defined in this chapter.
- $b_0 \in \mathcal{B}$ be the initial belief state of $\mathcal{M}$.
- $\boldsymbol{B_0} \in \boldsymbol{\mathcal{B}}$ be the initial uncertain belief state of $\boldsymbol{\mathcal{M}}$.

Then we have:

$$b_0 \in \boldsymbol{B_0} \longrightarrow \mathcal{M} \in \boldsymbol{\mathcal{M}}$$

Proof.

1. Assume $b_0 \in \boldsymbol{B_0}$.
2. As shown by Theorem 5.3.1, a successor uncertain belief state $\boldsymbol{B'}$ contains all successor beliefs b' of all valid beliefs $b \in \boldsymbol{B}$ contained in the previous uncertain belief state.
3. From (1) and (2), it follows that all successor beliefs of b_0, forming the belief states of belief MDP $\mathcal{M}$, are contained in the uncertain belief states of belief uMDP $\boldsymbol{\mathcal{M}}$.
4. As shown by Theorem 5.4.1, all transitions between beliefs $b \in \boldsymbol{B}$ and successor beliefs $b' \in \boldsymbol{B'}$ in uncertain belief state $\boldsymbol{B}$ and successor uncertain belief state $\boldsymbol{B'}$ are contained in the uncertain belief state transition function $\boldsymbol{\mathcal{P}}$.
5. Hence, if the uncertain belief states contain all beliefs of belief MDP $\mathcal{M}$, the uncertain belief state transition function $\boldsymbol{\mathcal{P}}$ contains all belief state transitions of belief MDP $\mathcal{M}$.
6. So from (3) and (5), it follows that the uncertain belief state transition function $\boldsymbol{\mathcal{P}}$ contains the belief state transition function $\mathcal{P}$.
7. As shown by Theorem 5.5.1, all rewards of valid beliefs $b \in \boldsymbol{B}$ in an uncertain belief state $\boldsymbol{B}$ are contained in the uncertain reward function $\boldsymbol{\mathcal{R}}$.
8. Hence, if the uncertain belief states contain all beliefs of belief MDP $\mathcal{M}$, the uncertain reward function $\boldsymbol{\mathcal{R}}$ contains all true rewards of belief MDP $\mathcal{M}$.
9. So from (3) and (8), it follows that the uncertain reward function $\boldsymbol{\mathcal{R}}$ contain the reward function $\mathcal{R}$.

10. From (1), (3), (6), and (9), we conclude that

$$b_0 \in \boldsymbol{B_0} \longrightarrow \mathcal{M} \in \boldsymbol{\mathcal{M}}$$

.

□

5.7 Example

We now have the complete definition of the belief uMDP. As mentioned multiple times in the previous sections, we discuss how to deal with the various optimization problems that follow from this definition in Chapter 6. Using that information, we can give a small example belief uMDP of the cheese maze uPOMDP, just like we did with the belief MDP of the cheese maze POMDP in Section 4.3.3. Figure 5.4 illustrates the cheese maze uPOMDP, where states with the same observation have the same colour.

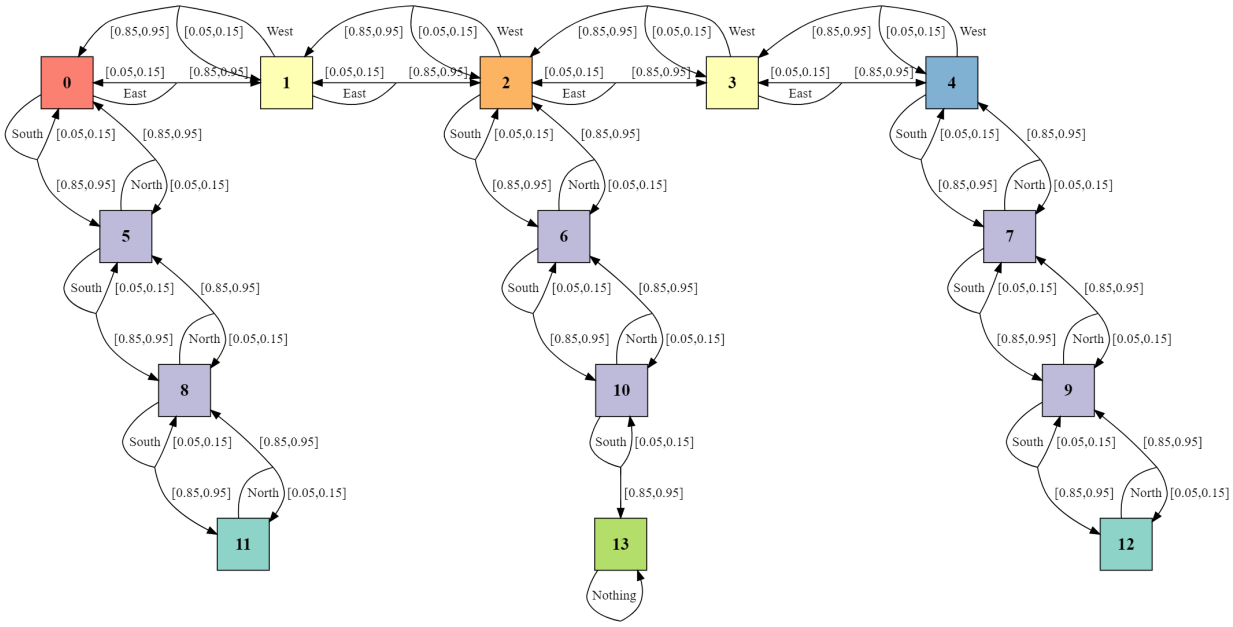


Figure 5.4: Cheese maze uPOMDP.

We again take the initial belief that there is an 80% chance that the mouse is in square 8 and a 10% chance that it is in square 9 or 10. The belief uMDP results in an infinite amount of belief states. For example purposes, we shall only expand the uncertain belief states that correspond with the belief states in the belief MDP example. We compute the partial belief uMDP using one of the implementations discussed in Chapter 7. Figure 5.5 illustrates the partial belief uMDP.

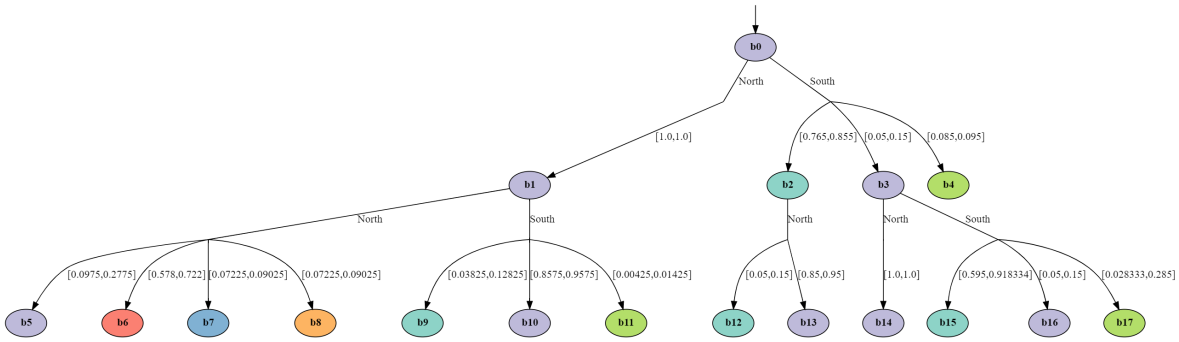


Figure 5.5: Cheese maze belief uMDP.

The colours only indicate the observations the belief states correspond to: the belief states are fully observable and hence distinguishable for the agent. Below we provide the belief uncertainty intervals $\boldsymbol{B}_{ui}$ and additional distribution constraints $\boldsymbol{B}_{dc}$ for each uncertain belief state $\boldsymbol{B}$ in the example. We omit states with a zero belief and distribution constraints concerning a single state. Note that we did not draw loops in this example belief uMDP, hence some states have the same $\boldsymbol{B}_{ui}$ and $\boldsymbol{B}_{dc}$

$\boldsymbol{b0}_{ui}$: $\{s8 \mapsto [0.8, 0.8], \quad s9 \mapsto [0.1, 0.1], \quad s10 \mapsto [0.1, 0.1]\}$

$\boldsymbol{b1}_{ui}$: $\{s5 \mapsto [0.68, 0.76], \quad s6 \mapsto [0.085, 0.095], \quad s7 \mapsto [0.085, 0.095], s8 \mapsto [0.04, 0.12],$
$s9 \mapsto [0.005, 0.015], s10 \mapsto [0.005, 0.015]\}$

$\boldsymbol{b1}_{dc}$: $\{y_{s5} + y_{s8} = 0.8, \;\; y_{s7} + y_{s9} = 0.1, \;\; y_{s6} + y_{s10} = 0.8\}$

$\boldsymbol{b2}_{ui}$: $\{s11 \mapsto [0.795, 0.993], s12 \mapsto [0.099, 0.124]\}$

$\boldsymbol{b3}_{ui}$: $\{s8 \mapsto [0.267, 1], \quad s9 \mapsto [0.033, 0.3]\} \quad s10 \mapsto [0.033, 0.3]\}$

$\boldsymbol{b4}_{ui}$: $\{s13 \mapsto [1, 1]\}$

$\boldsymbol{b5}_{ui}$: $\{s5 \mapsto [0.344, 1], \quad s6 \mapsto [0.033, 0.252], \quad s7 \mapsto [0.033, 0.252], s8 \mapsto [0.009, 0.104],$
$s9 \mapsto [0.001, 0.021], s10 \mapsto [0.001, 0.021]\}$

$\boldsymbol{b5}_{dc}$: $\{0.838y_{s5} + y_{s8} \geq 0.191, \;\; 0.838y_{s7} + y_{s9} \geq 0.019, \;\; 0.838y_{s6} + y_{s10} \geq 0.019,$
$0.224y_{s5} + y_{s8} \leq 0.691, \;\; 0.224y_{s7} + y_{s9} \leq 0.140, \;\; 0.224y_{s6} + y_{s10} \leq 0.140\}$

$\boldsymbol{b6}_{ui}$: $\{s0 \mapsto [1, 1]\}$

$\boldsymbol{b7}_{ui}$: $\{s4 \mapsto [1, 1]\}$

$\boldsymbol{b8}_{ui}$: $\{s2 \mapsto [1, 1]\}$

$\boldsymbol{b9}_{ui}$: $\{s11 \mapsto [0.651, 1.0], \quad s12 \mapsto [0.036, 0.305]\}$

$\boldsymbol{b10}_{ui}$: $\{s5 \mapsto [0.038, 0.122], s6 \mapsto [0.004, 0.016], \quad s7 \mapsto [0.004, 0.016], s8 \mapsto [0.657, 0.780],$

$$
\begin{aligned}
& s9 \mapsto [0.077, 0.105], s10 \mapsto [0.077, 0.105]\} \\
\boldsymbol{b10}_{dc}\colon\ & \{y_{s5} + y_{s8} \geq 0.764,\ \ y_{s7} + y_{s9} \geq 0.090,\ \ y_{s6} + y_{s10} \geq 0.090, \\
& y_{s5} + 0.870 y_{s8} \leq 0.814,\ \ y_{s7} + 0.870 y_{s9} \leq 0.110,\ \ y_{s6} + 0.870 y_{s10} \leq 0.110\}
\end{aligned}
$$

$$
\boldsymbol{b11}_{ui}\colon\ \{s13 \mapsto [1, 1]\}
$$

$$
\boldsymbol{b12}_{ui}\colon\ \{s11 \mapsto [0.292, 1.0],\quad s12 \mapsto [0.033, 0.373]\}
$$

$$
\boldsymbol{b13}_{ui}\colon\ \{s8 \mapsto [0.784, 1.0],\quad s9 \mapsto [0.089, 0.139]\}
$$

$$
\begin{aligned}
\boldsymbol{b14}_{ui}\colon\ & \{s5 \mapsto [0.34, 0.887],\ \ s6 \mapsto [0.028, 0.285],\quad s7 \mapsto [0.028, 0.285], s8 \mapsto [0.02, 0.14], \\
& s9 \mapsto [0.002, 0.045], s10 \mapsto [0.002, 0.045]\} \\
\boldsymbol{b14}_{dc}\colon\ & \{y_{s5} + y_{s8} \geq 0.4,\ \ y_{s7} + y_{s9} \geq 0.033,\ \ y_{s6} + y_{s10} \geq 0.033, \\
& y_{s5} + y_{s8} \leq 0.933,\ \ y_{s7} + y_{s9} \leq 0.3,\ \ y_{s6} + y_{s10} \leq 0.3\}
\end{aligned}
$$

$$
\boldsymbol{b15}_{ui}\colon\ \{s11 \mapsto [0.511, 1.0],\quad s12 \mapsto [0.031, 0.479]\}
$$

$$
\boldsymbol{b16}_{ui}\colon\ \{s8 \mapsto [0.133, 1],\qquad s9 \mapsto [0.011, 0.9]\}\qquad s10 \mapsto [0.011, 0.9]\}
$$

$$
\boldsymbol{b17}_{ui}\colon\ \{s13 \mapsto [1, 1]\}
$$

We can now see that the probability distributions of the belief states of the belief MDP of Section 4.3.3 are contained in the probability intervals of the corresponding uncertain belief states in the belief uMDP:

Belief MDP		Belief uMDP
$\boldsymbol{b0}$	$\in$	$\boldsymbol{b0}$
$\boldsymbol{b1}$	$\in$	$\boldsymbol{b1}$
$\boldsymbol{b2}$	$\in$	$\boldsymbol{b2}$
$\boldsymbol{b3}$	$\in$	$\boldsymbol{b4}$
$\boldsymbol{b4}$	$\in$	$\boldsymbol{b5}$
$\boldsymbol{b5}$	$\in$	$\boldsymbol{b6}$
$\boldsymbol{b6}$	$\in$	$\boldsymbol{b8}$
$\boldsymbol{b7}$	$\in$	$\boldsymbol{b7}$
$\boldsymbol{b8}$	$\in$	$\boldsymbol{b10}$
$\boldsymbol{b9}$	$\in$	$\boldsymbol{b13}$

Looking at the transitions, we see that the belief MDP probability is not always contained in the belief uMDP probability interval. However, this is because we rounded the transitions in the belief MDP, which we did not do in the belief uMDP.

5.8 Benefits of the belief uMDP

We briefly discussed the benefits of the belief uMDP in the Introduction. Here, we repeat those benefits and give a bit more detail.

The main benefit of defining the belief uMDP is that it paves the way for transforming belief-based approaches for POMDPs to uncertain belief-based approaches for uPOMDPs. Most obviously, as the belief uMDP is a uMDP, we can use uMDP algorithms to compute finite horizon policies for uPOMDPs [Nilim and Ghaoui, 2005, Wiesemann et al., 2013, Givan et al., 2000]. Moreover, there are various POMDP policy computation algorithms that use the belief update function, which we can try and lift to the uncertain setting using the uncertain belief update [Shani et al., 2013, Walraven and Spaan, 2019].

Another possible application of the uncertain belief update function is robust reinforcement learning for POMDPs [Kaelbling et al., 1996]. Note that both suggested applications of the uncertain belief update function require future research.

Furthermore, once a finite horizon belief uMDP is computed, it can be reused to compute policies with objectives such as worst-case expected reward or satisfaction of temporal logic formula [Baier and Katoen, 2008]. For direct policy computation on uPOMDPs, policies for different objectives require full recomputation. The belief uMDP works as a saving point and reduces the recomputation requirement to the finite horizon uMDP policy computation.

Moreover, the belief uMDP provides a visual representation of the uncertainty up to a finite horizon. This visual representation can provide insight into which uncertain transitions have a lot of influence on the amount of uncertainty in the agent's belief, i.e., the size of the belief uncertainty intervals and which do not. Identification of problematic uncertainties could help make effective choices regarding system improvement.

Chapter 6

Solving and approximating the optimization problems

In this chapter, we discuss how we deal with the optimization problems that follow from our definition of the belief uMDP and the associated functions as defined in Chapter 5. We begin by elaborating on the feasible sets that the various optimization problems maximize and minimize over. Next, in Section 6.2, we give an overview of the optimization problems and categorize them per type. In Sections 6.3 to 6.5, we flesh out different ways of solving or approximating the non-linear optimization problems. We discuss two concrete approximation methods in Sections 6.6 to 6.7. Finally, in Section 6.8, we evaluate the suggested approximation methods.

Note that when we talk about approximation, we mean approximation of an interval. An under-approximation is a sub-interval of the true interval, and an over-approximation is a super-interval of the true interval.

6.1 Overview of the feasible sets

The optimization problems from Chapter 5 are all defined over an uncertain belief state, the uncertain state transition function, or both. Below we demonstrate how to write these feasible sets as convex polytopes in the canonical matrix form.

6.1.1 Optimizing over an uncertain belief state

Let $n = |S|$. We use the optimization variable:

$$\vec{y} = \begin{bmatrix} y_1 & \dots & y_n \end{bmatrix}$$

This optimization variable consists of n elements to hold the belief in each state. Let $\boldsymbol{B}$ be the uncertain belief state we optimize over. We list the linear equalities we get per set of $\boldsymbol{B}$.

Let $[v_k, w_k] = \boldsymbol{B}_{ui}(k)$. We get the following linear inequalities:

$\forall k \in [n]$:

$$-y_k \leq -v_k$$
$$y_k \leq w_k$$

Furthermore, we get the two linear inequalities from $\boldsymbol{B}_{pd}$ that restrict instantiations of uncertain belief state $\boldsymbol{B}$ to probability distributions.

$$\sum_{k \in [n]} -y_k \leq -1$$
$$\sum_{k \in [n]} y_k \leq 1$$

And finally, we get the linear inequalities of $\boldsymbol{B}_{dc}$. Let $m = |\boldsymbol{B}_{dc}|$. Given linear inequality $e_h \in \boldsymbol{B}_{dc}$ with $h \in [m]$, let $\vec{e_h}$ be the n-sized coefficient vector and let e_h^* be the constant. We get the following linear inequalities:

$\forall h \in [m]$:

$$\sum_{k \in [n]} e_{h,k} y_k \leq e_h^*$$

We can write the convex polytope formed by the uncertain belief state $\boldsymbol{B}$ in the canonical matrix form, based on [Suilen et al., 2020]:

$$A_{\boldsymbol{B}} \vec{y} \leq \vec{c}_{\boldsymbol{B}}$$

Where:

$$E = \begin{bmatrix} \vec{e_h}^T & \text{for } h \in [m] \end{bmatrix}$$
$$A_{\boldsymbol{B}}^T = \begin{bmatrix} -I_n & I_n & H_n^T & -H_n^T & E \end{bmatrix}$$

$$\vec{c}_{\boldsymbol{B}}^T = \begin{bmatrix} -v_1 & \dots & -v_n & w_1 & \dots & w_n & 1 & -1 & e_1^* & \dots & e_m^* \end{bmatrix}$$

Where I_n is the $n \times n$ identity matrix.
Where H_n is the $1 \times n$ single row matrix consisting of only ones.

6.1.2 Optimizing over the uncertain state transition function

Given an action $a \in A$, we first describe how to define the convex polytope for the uncertain a-transitions from a single state $i \in S$ and then combine them into a big convex polytope for the entire uncertain state transition function $\boldsymbol{P}$ for action a.

Uncertain a-transitions from a single state

We use the optimization variable:

$$\vec{x} = \begin{bmatrix} x_1 & \dots & x_n \end{bmatrix}$$

This optimization variable consists of n elements representing the a-transitions from state i to the other states.

Let $[r_j, t_j] = \boldsymbol{P}(i, a, j)$. We get the following linear inequalities:

$\forall j \in [n]$:

$$-x_j \leq -r_j \qquad \sum_{j \in [n]} -x_j \leq -1$$

$$x_j \leq t_j \qquad \sum_{j \in [n]} x_j \leq 1$$

We can write the convex polytope formed by the uncertain transition function for state i and action a in the canonical matrix form, based on [Suilen et al., 2020]:

$$A_{\boldsymbol{P}(i,a)}\vec{x} \leq \vec{c}_{\boldsymbol{P}(i,a)}$$

Where:

$$A^T_{\boldsymbol{P}(i,a)} = \begin{bmatrix} -I_n & I_n & H_n^T & -H_n^T \end{bmatrix}$$
$$\vec{c}^T_{\boldsymbol{P}(i,a)} = \begin{bmatrix} -r_1 & \dots & -r_n & t_1 & \dots & t_n & 1 & -1 \end{bmatrix}$$

Where I_n is the $n \times n$ identity matrix.
Where H_n is the $1 \times n$ single row matrix consisting of only ones.

Uncertain a-transitions from all states

We use the optimization variable:

$$\vec{x} = \begin{bmatrix} x_{1,1} & \dots & x_{n,n} \end{bmatrix}$$

This optimization variable consists of n^2 elements representing the states' transitions. We can write the combined convex polytope formed by the uncertain transition function for action a for each state $i \in S$ in the canonical matrix form, based on [Suilen et al., 2020]:

$$A_{\boldsymbol{P}(\cdot,a)}\vec{x} \leq \vec{c}_{\boldsymbol{P}(\cdot,a)}$$

Where:

$$O = \begin{bmatrix} O_n & O_n & K_n^T & K_n^T \end{bmatrix}$$

$$A^T_{\boldsymbol{P}(\cdot,a)} = \begin{bmatrix} A^T_{\boldsymbol{P}(1,a)} & O & \dots & O \\ O & A^T_{\boldsymbol{P}(2,a)} & \dots & O \\ \vdots & \vdots & \ddots & \vdots \\ O & O & \dots & A^T_{\boldsymbol{P}(n,a)} \end{bmatrix}$$

$$\vec{c}^T_{\boldsymbol{P}(\cdot,a)} = \begin{bmatrix} \vec{c}^T_{\boldsymbol{P}(1,a)} & \dots & \vec{c}^T_{\boldsymbol{P}(n,a)} & 1 & -1 \end{bmatrix}$$

Where O_n is the $n \times n$ zero matrix.
Where K_n is the $1 \times n$ single row matrix consisting of only zeros.

6.1.3 Combining the feasible sets

We combine the feasible sets of an uncertain belief state $\boldsymbol{B}$ and the uncertain state transition function $\boldsymbol{P}$ for action a into a single convex polytope

We use the optimization variable:

$$\vec{x} = \begin{bmatrix} x_{1,1} & \dots & x_{n,n} & y_1 & \dots & y_n \end{bmatrix}$$

This optimization variable consists of n^2 elements representing the transitions between the states and n elements representing the belief in each state.

We can write the combined convex polytope in the canonical matrix form, based on [Suilen et al., 2020]:

$$A_{\boldsymbol{B},\boldsymbol{P}(\cdot,a)}\vec{x} \leq \vec{c}_{\boldsymbol{B},\boldsymbol{P}(\cdot,a)}$$

Where:

$$O = \begin{bmatrix} O_n & O_n & K_n^T & K_n^T \end{bmatrix}$$

$$L = O + \begin{bmatrix} K_n^T & \text{for } h \in [m] \end{bmatrix}$$

$$A^T_{\boldsymbol{B},\boldsymbol{P}(\cdot,a)} = \begin{bmatrix} A^T_{\boldsymbol{P}(1,a)} & O & \dots & O & L \\ O & A^T_{\boldsymbol{P}(2,a)} & \dots & O & L \\ \vdots & \vdots & \ddots & \vdots & \vdots \\ O & O & \dots & A^T_{\boldsymbol{P}(n,a)} & L \\ O & O & \dots & O & A^T_{\boldsymbol{B}} \end{bmatrix}$$

$$\vec{c}^T_{\boldsymbol{B},\boldsymbol{P}(\cdot,a)} = \begin{bmatrix} \vec{c}^T_{\boldsymbol{P}(1,a)} & \dots & \vec{c}^T_{\boldsymbol{P}(n,a)} & \vec{c}^T_{\boldsymbol{B}} \end{bmatrix}$$

Where O_n is the $n \times n$ zero matrix.
Where K_n is the $1 \times n$ single row matrix consisting of only zeros.

6.2 Overview of the optimization problems

Below, we list all the optimization problems we need to deal with to compute a belief uMDP. We categorize them based on the type of optimization problem. All optimization problems are linearly constrained.

Linear programs

Optimization variables	Objective function (min/max)	Feasible set
Uncertain reward function (Section 5.5)		
$\vec{y} = \begin{bmatrix} y_1 & \dots & y_n \end{bmatrix}$	$\sum_{s \in S} y_s R(s,a)$	$A_{\boldsymbol{B}} \vec{y} \leq \vec{c}_{\boldsymbol{B}}$

Quadratic programs

Optimization variables	Objective function (min/max)	Feasible set
Uncertain transition function (Section 5.4)		
$\vec{x} = \begin{bmatrix} x_{1,1} & \dots & x_{n,n} & y_1 & \dots & y_n \end{bmatrix}$	$\sum_{s' \in S} \boldsymbol{O}(s',o) \sum_{s \in S} x_{s,s'} y_s$	$A_{\boldsymbol{B},\boldsymbol{P}(\cdot,a)} \vec{x} \leq \vec{c}_{\boldsymbol{B},\boldsymbol{P}(\cdot,a)}$

Quadratic fractional programs

Optimization variables	Objective function (min/max)	Feasible set
Belief uncertainty intervals (Section 5.3.1)		
$\vec{x} = \begin{bmatrix} x_{1,1} & \dots & x_{n,n} & y_1 & \dots & y_n \end{bmatrix}$	$\frac{\boldsymbol{O}(s',o) \sum_{s \in S} x_{s,s'} y_s}{\sum_{s' \in S} \boldsymbol{O}(s',o) \sum_{s \in S} x_{s,s'} y_s}$	$A_{\boldsymbol{B},\boldsymbol{P}(\cdot,a)} \vec{x} \leq \vec{c}_{\boldsymbol{B},\boldsymbol{P}(\cdot,a)}$
Belief contribution for states with multiple belief sources (Section 5.3.3)		
$\vec{x} = \begin{bmatrix} x_{1,1} & \dots & x_{n,n} & y_1 & \dots & y_n \end{bmatrix}$	$\frac{\boldsymbol{O}(s',o) x_{s,s'} y_s}{\boldsymbol{O}(s',o) \sum_{s \in S} x_{s,s'} y_s}$	$A_{\boldsymbol{B},\boldsymbol{P}(\cdot,a)} \vec{x} \leq \vec{c}_{\boldsymbol{B},\boldsymbol{P}(\cdot,a)}$
Normalization of belief distribution constraints (Section 5.3.3)		
$\vec{x} = \begin{bmatrix} x_{1,1} & \dots & x_{n,n} & y_1 & \dots & y_n \end{bmatrix}$	$\frac{\sum_{s' \in S} \boldsymbol{O}(s',o) x_{s,s'} y_s}{\sum_{s' \in S} \boldsymbol{O}(s',o) \sum_{s \in S} x_{s,s'} y_s}$	$A_{\boldsymbol{B},\boldsymbol{P}(\cdot,a)} \vec{x} \leq \vec{c}_{\boldsymbol{B},\boldsymbol{P}(\cdot,a)}$

6.3 Solving linear programs

As previously mentioned in Section 3.3, linear programs can be solved exactly, upto a given accuracy, in polynomial time with methods like Dantzig's simplex method and interior-point

methods [Boyd and Vandenberghe, 2014]. State-of-the-art mathematical optimization solvers use these methods to solve linear programs in seconds. In this thesis, we use the optimization solver Gurobi to solve the linear programs [Gurobi Optimization, LLC, 2023].

6.4 Solving and approximating quadratic programs

As shown in [Sahni, 1974], the general global optimization of quadratic programs is NP-hard. Polynomial-time algorithms exist for convex quadratic programs, such as interior-point methods [Boyd and Vandenberghe, 2014], but the function we need to optimize is non-convex. We prove the non-convexity of this function in Appendix A.1.
Given a non-convex quadratic program, some mathematical optimization solvers, like Gurobi [Gurobi Optimization, LLC, 2023], are able to find globally optimal solutions. For those cases where such an optimization solver is unavailable or unable to find an optimal solution within a reasonable time, we also consider two approximation methods.

As mentioned in Section 5.2.1, any approximations used to compute a finite horizon belief uMDP should over-approximate. The two approximation methods we consider for computing bounds on uncertainty intervals do not meet this requirement, as both return a value within the true possible interval as bound. The first method we consider is random sampling, which can be used for any optimization problem. The second function we consider is *Two-phase Successive Linear Programming Algorithm* (2pSLPA), which only applies to quadratic programs. We discuss the two approximation methods in Sections 6.6 and 6.7, respectively.

6.4.1 Independent transitions

As explained above, without any additional information, finding an exact solution for a non-convex quadratic program is difficult. However, because of the (s,a)-rectangularity assumption (see Section 4.4.3), we have additional information about our optimization problems: The uncertain state transition function $\boldsymbol{P}$ is independent between the states. This means the outgoing transitions of one state are completely unrelated to those of another state. This allows us to transform the quadratic program into multiple linear ones.

We begin by transforming the quadratic function of the uncertain belief state transition function a bit. Remember that we maximize and minimize over the instantiations $b \in \boldsymbol{B}$ of the uncertain belief state $\boldsymbol{B}$ and $P \in \boldsymbol{P}$ of the uncertain state transition function $\boldsymbol{P}$ and $\mathrm{SE}(b, a, o) = b'$. Using the commutative, associative, and distributive properties of summations and multiplications, we can rewrite it as follows:

$$\begin{aligned}
\mathcal{P}(b, a, b') &= \sum_{s' \in S} \boldsymbol{O}(s', o) \cdot \sum_{s \in S} P(s, a, s') b(s) \\
&= \sum_{s' \in S} \sum_{s \in S} \boldsymbol{O}(s', o) b(s) P(s, a, s') \\
&= \sum_{s' \in S} \sum_{s \in S} b(s) \boldsymbol{O}(s', o) P(s, a, s') \\
&= \sum_{s \in S} b(s) \cdot \sum_{s' \in S} \boldsymbol{O}(s', o) P(s, a, s')
\end{aligned}$$

The subfunction $\sum_{s' \in S} \boldsymbol{O}(s', o) P(s, a, s')$ does not depend on the belief. We always need the maximum of the subfunction to maximize or the minimum of the subfunction to minimize the total optimization problem. We hence get two lemmas. We provide the proof for the minimization Lemma 6.4.1. The maximization proof directly follows by symmetry.

Lemma 6.4.1:

$$\min_{b \in \boldsymbol{B}, P(\cdot,a) \in \boldsymbol{P}(\cdot,a)} \sum_{s \in S} b(s) \cdot \sum_{s' \in S} \boldsymbol{O}(s', o) P(s, a, s') = \min_{b \in \boldsymbol{B}} \sum_{s \in S} b(s) \cdot \min_{P(s,a) \in \boldsymbol{P}(s,a)} \sum_{s' \in S} \boldsymbol{O}(s', o) P(s, a, s')$$

Proof.

1. Let $v = \min_{b \in \boldsymbol{B}, P(\cdot,a) \in \boldsymbol{P}(\cdot,a)} \sum_{s \in S} b(s) \cdot \sum_{s' \in S} \boldsymbol{O}(s', o) P(s, a, s')$.
2. Let $b \in \boldsymbol{B}$ and $P(\cdot, a) \in \boldsymbol{P}(\cdot, a)$ be the instatiations that resulted in v.
3. Assume that $\exists s \in S : b(s) \neq 0 \wedge \sum_{s' \in S} \boldsymbol{O}(s', o) P(s, a, s') \neq \min_{P'(s,a) \in \boldsymbol{P}(s,a)} \sum_{s' \in S} \boldsymbol{O}(s', o) P'(s, a, s')$.
4. Let $P'(s, a)$ be the partial instatiation that minimizes $\min_{P'(s,a) \in \boldsymbol{P}(s,a)} \sum_{s' \in S} \boldsymbol{O}(s', o) P'(s, a, s')$.
5. Then $\sum_{s' \in S} \boldsymbol{O}(s', o) P'(s, a, s') < \sum_{s' \in S} \boldsymbol{O}(s', o) P(s, a, s')$.
6. And $b(s) \cdot \sum_{s' \in S} \boldsymbol{O}(s', o) P'(s, a, s') < b(s) \cdot \sum_{s' \in S} \boldsymbol{O}(s', o) P(s, a, s')$.
7. Since the uncertain state transition function is independent between the states, we can set the rest of $P'(\cdot, a)$ equal to $P(\cdot, a)$.
8. Hence $\forall s'' \in S \setminus \{s\} : b(s'') \cdot \sum_{s' \in S} \boldsymbol{O}(s', o) P'(s'', a, s') = b(s'') \cdot \sum_{s' \in S} \boldsymbol{O}(s', o) P(s'', a, s')$.
9. Then, from (6) and (8), it follows that:

$$\sum_{s \in S} b(s) \cdot \sum_{s' \in S} \boldsymbol{O}(s', o) P'(s, a, s') < \sum_{s \in S} b(s) \cdot \sum_{s' \in S} \boldsymbol{O}(s', o) P(s, a, s') = v.$$

10. Hence, from (1), (2), and (9), we get a contradiction.
11. Therefore, the assumption made in (3) is false.
12. So we get $\forall s \in S$:

$$b(s) \neq 0 \rightarrow \sum_{s' \in S} \boldsymbol{O}(s', o) P(s, a, s') = \min_{P'(s,a) \in \boldsymbol{P}(s,a)} \sum_{s' \in S} \boldsymbol{O}(s', o) P'(s, a, s').$$

13. Furthermore, $\forall s \in S$:

$$b(s) = 0 \rightarrow b(s) \cdot \sum_{s' \in S} \boldsymbol{O}(s', o) P(s, a, s') = b(s) \cdot \min_{P'(s,a) \in \boldsymbol{P}(s,a)} \sum_{s' \in S} \boldsymbol{O}(s', o) P'(s, a, s').$$

14. Then from (1), (12), and (13), it follows that:

$$\min_{b\in\boldsymbol{B},P(\cdot,a)\in\boldsymbol{P}(\cdot,a)}\sum_{s\in S}b(s)\cdot\sum_{s'\in S}\boldsymbol{O}(s',o)P(s,a,s')=\min_{b\in\boldsymbol{B}}\sum_{s\in S}b(s)\cdot\min_{P(s,a)\in\boldsymbol{P}(s,a)}\sum_{s'\in S}\boldsymbol{O}(s',o)P(s,a,s').$$

□

Lemma 6.4.2:

$$\max_{b\in\boldsymbol{B},P(\cdot,a)\in\boldsymbol{P}(\cdot,a)}\sum_{s\in S}b(s)\cdot\sum_{s'\in S}\boldsymbol{O}(s',o)P(s,a,s')=\max_{b\in\boldsymbol{B}}\sum_{s\in S}b(s)\cdot\max_{P(s,a)\in\boldsymbol{P}(s,a)}\sum_{s'\in S}\boldsymbol{O}(s',o)P(s,a,s')$$

Now that we have proven the equivalences, we can split the optimization problem into two steps. First, we precompute the minimums and maximums of the subfunction $\sum_{s'\in S}\boldsymbol{O}(s',o)P(s,a,s')$ over the instantiations of the uncertain transition function $\boldsymbol{P}$ for each state-action-observation triple. Second, we use these minimums to minimize, and maximums to maximize the total function over the instantiations of the old uncertain belief state $\boldsymbol{B}$.

Given a state $i\in S$, action $a\in A$, and observation $o\in Z$, we get the following optimization problems for the first step.

Optimization variables	Objective function (min/max)	Feasible set
Precomputing observable transition sums		
$\vec{x}=\begin{bmatrix}x_1 & \dots & x_n\end{bmatrix}$	$\sum_{s'\in S}\boldsymbol{O}(s',o)x_{s'}$	$A_{\boldsymbol{P}(i,a)}\vec{x}\leq\vec{c}_{\boldsymbol{P}(i,a)}$

Let $\widehat{x_i}$ be the maximum observable transition sum for state i
Let $\underset{\smile}{x_i}$ be the minimum observable transition sum for state i

We get the following optimization problems for the second step.

Optimization variables	Objective function (max)	Feasible set
Upper bound uncertain belief state transition		
$\vec{y}=\begin{bmatrix}y_1 & \dots & y_n\end{bmatrix}$	$\sum_{s\in S}y_s\widehat{x_s}$	$A_{\boldsymbol{B}}\vec{y}\leq\vec{c}_{\boldsymbol{B}}$
Optimization variables	Objective function (min)	Feasible set
Lower bound uncertain belief state transition		
$\vec{y}=\begin{bmatrix}y_1 & \dots & y_n\end{bmatrix}$	$\sum_{s\in S}y_s\underset{\smile}{x_s}$	$A_{\boldsymbol{B}}\vec{y}\leq\vec{c}_{\boldsymbol{B}}$

The new optimization problems are all linear programs and can hence, as described in Section 6.3, be solved exactly in polynomial time.

6.5 Approximating quadratic fractional programs

The final three optimization problems are all quadratic fractional programs. All three functions are non-convex, as shown in Appendices A.2 to A.4. Moreover, the quadratic sub-functions that form the numerators and denominators are non-convex as well, as shown in Appendices A.5 to A.7.

We investigated whether Gurobi could solve the non-convex quadratic fractional programs, but Gurobi does not allow for a fraction of two quadratic functions as the objective function. Additionally, we tried to move the denominator to the constraints by using an extra variable, but Gurobi also does not allow this.

Existing research on approximating quadratic fractional programs all rely on some form of convexity or are restricted in the type or number of constraint functions [Beck and Teboulle, 2010, Nguyen et al., 2014, Xu and Zhou, 2021] and are therefore not applicable to our optimization problems.

As explained in Section 6.4, we can under-approximate any function using random sampling. We discuss this approximation method in Section 6.6. To get an over-approximation of the non-convex quadratic programs, we can instead decide to decouple some of the optimization variables between the numerator and denominator. We discuss full decoupling and partial decoupling in the next two Sections.

6.5.1 Full decoupling

To over-approximate the quadratic fractional programs, we can compute the bounds of the numerators and denominators separately. This means we can use different instantiations of the old uncertain belief state $\boldsymbol{B}$ and the uncertain state transition function $\boldsymbol{P}$ to optimize the numerator than we use to optimize the denominator. We completely decouple the optimization variables between the two functions.

Given a fractional function $f(\vec{x}) = \dfrac{g(\vec{x})}{h(\vec{x})}$ with U the feasible set of $\vec{x}$.

$$\text{Let } \widehat{n} = \max_{\vec{x} \in U} g(\vec{x}).$$
$$\text{Let } \underset{\smile}{n} = \min_{\vec{x} \in U} g(\vec{x}).$$
$$\text{Let } \widehat{d} = \max_{\vec{x} \in U} h(\vec{x}).$$
$$\text{Let } \underset{\smile}{d} = \min_{\vec{x} \in U} h(\vec{x}).$$

Then we can compute an upper bound $\widehat{f(\vec{x})}$ and lower bound $\underset{\smile}{f(\vec{x})}$ as follows:

$$\widehat{f(\vec{x})} = \frac{\widehat{n}}{\underset{\smile}{d}},$$
$$\underset{\smile}{f(\vec{x})} = \frac{\underset{\smile}{n}}{\widehat{d}}.$$

Which over-approximates the true bounds of $f(\vec{x})$:

$$\underbrace{f(\vec{x})} \leq \min_{\vec{x} \in U} f(\vec{x}) \leq \max_{\vec{x} \in U} f(\vec{x}) \leq \widehat{f(\vec{x})}.$$

Note that the computed bounds of the numerator and denominator functions must either be exact or over-approximations of the true bounds of those functions for this full decoupling approach to over-approximates the true bounds of the entire quadratic fractional programs. Otherwise, it becomes unclear whether the computed bounds are under or over-approximating the true bounds.

We get the following new optimization problems for the numerators of the three quadratic fractional programs. We can leave out the denominator functions, as other quadratic functions already cover these.

Optimization variables	Objective function (min/max)	Feasible set
Belief uncertainty intervals (numerator)		
$\vec{x} = \begin{bmatrix} x_{1,1} & \dots & x_{n,n} & y_1 & \dots & y_n \end{bmatrix}$	$\boldsymbol{O}(s', o) \sum_{s \in S} x_{s,s'} y_s$	$A_{\boldsymbol{B},\boldsymbol{P}(\cdot,a)} \vec{x} \leq \vec{c}_{\boldsymbol{B},\boldsymbol{P}(\cdot,a)}$
Belief contribution for states with multiple belief sources (numerator)		
$\vec{x} = \begin{bmatrix} x_{1,1} & \dots & x_{n,n} & y_1 & \dots & y_n \end{bmatrix}$	$\boldsymbol{O}(s', o) x_{s,s'} y_s$	$A_{\boldsymbol{B},\boldsymbol{P}(\cdot,a)} \vec{x} \leq \vec{c}_{\boldsymbol{B},\boldsymbol{P}(\cdot,a)}$
Normalization of belief distribution constraints (numerator)		
$\vec{x} = \begin{bmatrix} x_{1,1} & \dots & x_{n,n} & y_1 & \dots & y_n \end{bmatrix}$	$\sum_{s' \in S} \boldsymbol{O}(s', o) x_{s,s'} y_s$	$A_{\boldsymbol{B},\boldsymbol{P}(\cdot,a)} \vec{x} \leq \vec{c}_{\boldsymbol{B},\boldsymbol{P}(\cdot,a)}$

As mentioned above, the numerator and denominator functions are non-convex quadratic functions, so we end up with non-convex quadratic programs. We discuss how to solve or approximate this type of program in Section 6.4. Note that we can again use the independence of transitions between the states to solve these non-convex quadratic programs using linear programs, as discussed in Section 6.4.1.

To use the linear transformation for the new quadratic programs, we need to prove six lemmas. One for minimizing and one for maximizing each new quadratic function. Note that in the lemmas below, we have already rewritten the functions as we did in Section 6.4.1, moving the belief optimization variables to the front. We omit the proofs of these lemmas, as they follow from the proof for Lemma 6.4.1.

Lemma 6.5.1:

$$\min_{b \in \boldsymbol{B}, P(\cdot,a) \in \boldsymbol{P}(\cdot,a)} \sum_{s \in S} b(s) \cdot \boldsymbol{O}(s', o) P(s, a, s') = \min_{b \in \boldsymbol{B}} \sum_{s \in S} b(s) \cdot \min_{P(s,a) \in \boldsymbol{P}(s,a)} \boldsymbol{O}(s', o) P(s, a, s')$$

Lemma 6.5.2:

$$\max_{b \in \boldsymbol{B}, P(\cdot,a) \in \boldsymbol{P}(\cdot,a)} \sum_{s \in S} b(s) \cdot \boldsymbol{O}(s', o) P(s, a, s') = \max_{b \in \boldsymbol{B}} \sum_{s \in S} b(s) \cdot \max_{P(s,a) \in \boldsymbol{P}(s,a)} \boldsymbol{O}(s', o) P(s, a, s')$$

Lemma 6.5.3:

$$\min_{b \in \boldsymbol{B}, P(\cdot,a) \in \boldsymbol{P}(\cdot,a)} b(s) \cdot \boldsymbol{O}(s', o) P(s, a, s') = \min_{b \in \boldsymbol{B}} b(s) \cdot \min_{P(s,a) \in \boldsymbol{P}(s,a)} \boldsymbol{O}(s', o) P(s, a, s')$$

Lemma 6.5.4:

$$\max_{b \in \boldsymbol{B}, P(\cdot,a) \in \boldsymbol{P}(\cdot,a)} b(s) \cdot \boldsymbol{O}(s', o) P(s, a, s') = \max_{b \in \boldsymbol{B}} b(s) \cdot \max_{P(s,a) \in \boldsymbol{P}(s,a)} \boldsymbol{O}(s', o) P(s, a, s')$$

Lemma 6.5.5:

$$\min_{b \in \boldsymbol{B}, P(\cdot,a) \in \boldsymbol{P}(\cdot,a)} b(s) \cdot \sum_{s' \in S} \boldsymbol{O}(s', o) P(s, a, s') = \min_{b \in \boldsymbol{B}} b(s) \cdot \min_{P(s,a) \in \boldsymbol{P}(s,a)} \sum_{s' \in S} \boldsymbol{O}(s', o) P(s, a, s')$$

Lemma 6.5.6:

$$\max_{b \in \boldsymbol{B}, P(\cdot,a) \in \boldsymbol{P}(\cdot,a)} b(s) \cdot \sum_{s' \in S} \boldsymbol{O}(s', o) P(s, a, s') = \max_{b \in \boldsymbol{B}} b(s) \cdot \max_{P(s,a) \in \boldsymbol{P}(s,a)} \sum_{s' \in S} \boldsymbol{O}(s', o) P(s, a, s')$$

6.5.2 Partial decoupling

Instead of completely splitting the numerator and denominator functions, we can decouple only the uncertain state transition optimization variables. This approach leads to a smaller over-approximation than full decoupling but is only applicable because we have an uncertain state transition function with independence between the states. For similar problems without this independence, the optimization problem remains a quadratic fractional program after partial decoupling.

Similar to the approach for quadratic programs, as explained in Section 6.4.1, we begin by transforming the objective functions of the quadratic fractional programs a bit. Remember that we maximize and minimize over the instantiations $b \in \boldsymbol{B}$ of the uncertain belief state $\boldsymbol{B}$ and $P \in \boldsymbol{P}$ of the uncertain state transition function $\boldsymbol{P}$. Using the commutative, associative, and distributive properties of summations and multiplications, we can rewrite the three functions as follows:

$$\frac{\boldsymbol{O}(s', o) \sum_{s \in S} P(s, a, s') b(s)}{\sum_{s' \in S} \boldsymbol{O}(s', o) \sum_{s \in S} P(s, a, s') b(s)} \Longrightarrow \frac{\sum_{s \in S} b(s) \boldsymbol{O}(s', o) P(s, a, s')}{\sum_{s \in S} b(s) \sum_{s' \in S} \boldsymbol{O}(s', o) P(s, a, s')}$$

$$\frac{\boldsymbol{O}(s', o) P(s, a, s') b(s)}{\boldsymbol{O}(s', o) \sum_{s \in S} P(s, a, s') b(s)} \Longrightarrow \frac{b(s) \boldsymbol{O}(s', o) P(s, a, s')}{\sum_{s \in S} b(s) \boldsymbol{O}(s', o) P(s, a, s')}$$

$$\frac{\sum_{s' \in S} \boldsymbol{O}(s', o) P(s, a, s') b(s)}{\sum_{s' \in S} \boldsymbol{O}(s', o) \sum_{s \in S} P(s, a, s') b(s)} \Longrightarrow \frac{b(s) \sum_{s' \in S} \boldsymbol{O}(s', o) P(s, a, s')}{\sum_{s \in S} b(s) \sum_{s' \in S} \boldsymbol{O}(s', o) P(s, a, s')}$$

Next, we decouple the uncertain state transition function between the numerator and the denominator. We add an extra optimization variable $\vec{z} = \begin{bmatrix} z_{1,1} & \dots & z_{n,n} \end{bmatrix}$ with feasible set $A_{\boldsymbol{P}(\cdot,a)} \vec{z} \leq c_{\boldsymbol{P}(\cdot,a)}$ to the optimization problems to represent the second instantiation of the uncertain state transition function. For simplicity, we do not combine this optimization variable and feasible set with those of the original problem. We end up with the following three quadratic fractional programs:

Optimization variables	Objective function (min/max)	Feasible set
Belief uncertainty intervals		
$\vec{x} = \begin{bmatrix} x_{1,1} & \dots & x_{n,n} & y_1 & \dots & y_n \end{bmatrix}$ $\vec{z} = \begin{bmatrix} z_{1,1} & \dots & z_{n,n} \end{bmatrix}$	$\frac{\sum_{s\in S} y_s \boldsymbol{O}(s',o) x_{s,s'}}{\sum_{s\in S} y_s \sum_{s'\in S} \boldsymbol{O}(s',o) z_{s,s'}}$	$A_{\boldsymbol{B},\boldsymbol{P}(\cdot,a)}\vec{x} \leq \vec{c}_{\boldsymbol{B},\boldsymbol{P}(\cdot,a)}$ $A_{\boldsymbol{P}(\cdot,a)}\vec{z} \leq c_{\boldsymbol{P}(\cdot,a)}$
Belief contribution for states with multiple belief sources		
$\vec{x} = \begin{bmatrix} x_{1,1} & \dots & x_{n,n} & y_1 & \dots & y_n \end{bmatrix}$ $\vec{z} = \begin{bmatrix} z_{1,1} & \dots & z_{n,n} \end{bmatrix}$	$\frac{y_s \boldsymbol{O}(s',o) x_{s,s'}}{\sum_{s\in S} y_s \boldsymbol{O}(s',o) z_{s,s'}}$	$A_{\boldsymbol{B},\boldsymbol{P}(\cdot,a)}\vec{x} \leq \vec{c}_{\boldsymbol{B},\boldsymbol{P}(\cdot,a)}$ $A_{\boldsymbol{P}(\cdot,a)}\vec{z} \leq c_{\boldsymbol{P}(\cdot,a)}$
Normalization of belief distribution constraints		
$\vec{x} = \begin{bmatrix} x_{1,1} & \dots & x_{n,n} & y_1 & \dots & y_n \end{bmatrix}$ $\vec{z} = \begin{bmatrix} z_{1,1} & \dots & z_{n,n} \end{bmatrix}$	$\frac{y_s \sum_{s'\in S} \boldsymbol{O}(s',o) x_{s,s'}}{\sum_{s\in S} y_s \sum_{s'\in S} \boldsymbol{O}(s',o) z_{s,s'}}$	$A_{\boldsymbol{B},\boldsymbol{P}(\cdot,a)}\vec{x} \leq \vec{c}_{\boldsymbol{B},\boldsymbol{P}(\cdot,a)}$ $A_{\boldsymbol{P}(\cdot,a)}\vec{z} \leq c_{\boldsymbol{P}(\cdot,a)}$

For all three functions, both the numerator and the denominator have subfunctions that do not rely on the belief. Because of the partial decoupling, we now always need to minimize the numerator subfunction and maximize the denominator subfunction to minimize the total optimization problems, and vice versa to maximize it. We get six lemmas, one for minimizing and one for maximizing each partially decoupled quadratic fractional program. We only provide the proof for Lemma 6.5.7. The rest of the proofs follow the same steps.

Lemma 6.5.7:

$$\min_{\substack{b\in\boldsymbol{B},P(\cdot,a)\in\boldsymbol{P}(\cdot,a),\\ P'(\cdot,a)\in\boldsymbol{P}(\cdot,a)}} \frac{\sum_{s\in S} b(s)\boldsymbol{O}(s',o)P(s,a,s')}{\sum_{s\in S} b(s)\sum_{s'\in S}\boldsymbol{O}(s',o)P'(s,a,s')} = \min_{b\in\boldsymbol{B}} \frac{\sum_{s\in S} b(s) \min_{P(\cdot,a)\in\boldsymbol{P}(s,a)} \boldsymbol{O}(s',o)P(s,a,s')}{\sum_{s\in S} b(s) \max_{P'(\cdot,a)\in\boldsymbol{P}(s,a)} \sum_{s'\in S}\boldsymbol{O}(s',o)P'(s,a,s')}$$

Proof.

Part A: Minimizing the numerator.

A1. Let $v = \min_{\substack{b\in\boldsymbol{B},P(\cdot,a)\in\boldsymbol{P}(\cdot,a),\\ P'(\cdot,a)\in\boldsymbol{P}(\cdot,a)}} \frac{\sum_{s\in S} b(s)\boldsymbol{O}(s',o)P(s,a,s')}{\sum_{s\in S} b(s)\sum_{s'\in S}\boldsymbol{O}(s',o)P'(s,a,s')}$.

A2. Let $b \in \boldsymbol{B}$ and $P(\cdot,a), P'(\cdot,a) \in \boldsymbol{P}(\cdot,a)$ be the instatiations that resulted in v.

A3. Assume that $\exists s \in S : b(s) \neq 0 \ \wedge \ \boldsymbol{O}(s',o)P(s,a,s') \neq \min_{P^\circ(s,a)\in\boldsymbol{P}(s,a)} \boldsymbol{O}(s',o)P^\circ(s,a,s')$.

A4. Let $P^\circ(s,a)$ be the partial instatiation that minimizes $\min_{P^\circ(s,a)\in\boldsymbol{P}(s,a)} \boldsymbol{O}(s',o)P^\circ(s,a,s')$.

A5. Then $\boldsymbol{O}(s',o)P^\circ(s,a,s') < \boldsymbol{O}(s',o)P(s,a,s')$.

A6. And $\dfrac{b(s)\boldsymbol{O}(s',o)P^{\circ}(s,a,s')}{\sum_{s\in S} b(s)\sum_{s'\in S}\boldsymbol{O}(s',o)P'(s,a,s')} < \dfrac{b(s)\boldsymbol{O}(s',o)P(s,a,s')}{\sum_{s\in S} b(s)\sum_{s'\in S}\boldsymbol{O}(s',o)P'(s,a,s')}$.

A7. Since the uncertain state transition function is independent between the states, we can set the rest of $P^{\circ}(\cdot,a)$ equal to $P(\cdot,a)$.

A8. Hence $\forall s''\in S\setminus\{s\}: \dfrac{b(s'')\boldsymbol{O}(s',o)P^{\circ}(s'',a,s')}{\sum_{s\in S} b(s)\sum_{s'\in S}\boldsymbol{O}(s',o)P'(s,a,s')} = \dfrac{b(s'')\boldsymbol{O}(s',o)P(s'',a,s')}{\sum_{s\in S} b(s)\sum_{s'\in S}\boldsymbol{O}(s',o)P'(s,a,s')}$.

A9. Then, from (A6) and (A8), it follows that:

$$\frac{\sum_{s\in S} b(s)\boldsymbol{O}(s',o)P^{\circ}(s,a,s')}{\sum_{s\in S} b(s)\sum_{s'\in S}\boldsymbol{O}(s',o)P'(s,a,s')} < \frac{\sum_{s\in S} b(s)\boldsymbol{O}(s',o)P(s,a,s')}{\sum_{s\in S} b(s)\sum_{s'\in S}\boldsymbol{O}(s',o)P'(s,a,s')} = v.$$

A10. Hence, from (A1), (A2), and (A9), we get a contradiction.

A11. Therefore, the assumption made in (A3) is false.

A12. So we get $\forall s\in S$:

$$b(s)\neq 0 \rightarrow \boldsymbol{O}(s',o)P(s,a,s') = \min_{P'(s,a)\in\boldsymbol{P}(s,a)} \boldsymbol{O}(s',o)P'(s,a,s').$$

A13. Furthermore, $\forall s\in S$:

$$b(s)=0 \rightarrow b(s)\boldsymbol{O}(s',o)P(s,a,s') = b(s)\min_{P'(s,a)\in\boldsymbol{P}(s,a)} \boldsymbol{O}(s',o)P'(s,a,s').$$

A14. Then from (A1), (A12) and (A13) it follows that:

$$\min_{\substack{b\in\boldsymbol{B},P(\cdot,a)\in\boldsymbol{P}(\cdot,a),\\ P'(\cdot,a)\in\boldsymbol{P}(\cdot,a)}} \frac{\sum_{s\in S} b(s)\boldsymbol{O}(s',o)P(s,a,s')}{\sum_{s\in S} b(s)\sum_{s'\in S}\boldsymbol{O}(s',o)P'(s,a,s')} = \min_{b\in\boldsymbol{B},P'(\cdot,a)\in\boldsymbol{P}(\cdot,a)} \frac{\sum_{s\in S} b(s)\min_{P(\cdot,a)\in\boldsymbol{P}(s,a)}\boldsymbol{O}(s',o)P(s,a,s')}{\sum_{s\in S} b(s)\sum_{s'\in S}\boldsymbol{O}(s',o)P'(s,a,s')}$$

Part B: Maximizing the denominator.

B1. Now assume that $\exists s \in S$:

$$b(s) \neq 0 \ \wedge \ \sum_{s' \in S} \boldsymbol{O}(s', o) P'(s, a, s') \neq \max_{P^{\triangle}(s,a) \in \boldsymbol{P}(s,a)} \sum_{s' \in S} \boldsymbol{O}(s', o) P^{\triangle}(s, a, s').$$

B2. Let $P^{\triangle}(s, a)$ be the partial instatiation that maximizes $\max_{P^{\triangle}(s,a) \in \boldsymbol{P}(s,a)} \sum_{s' \in S} \boldsymbol{O}(s', o) P^{\triangle}(s, a, s')$.

B3. Then $\sum_{s' \in S} \boldsymbol{O}(s', o) P^{\triangle}(s, a, s') > \sum_{s' \in S} \boldsymbol{O}(s', o) P(s, a, s')$.

B4. Since the uncertain state transition function is independent between the states, we can set the rest of $P^{\triangle}(\cdot, a)$ equal to $P(\cdot, a)$.

B5. Hence $\forall s'' \in S \setminus \{s\} : \sum_{s' \in S} \boldsymbol{O}(s', o) P^{\triangle}(s, a, s') = \sum_{s' \in S} \boldsymbol{O}(s', o) P'(s, a, s')$.

B6. Then, from (B3) and (B5), it follows that:

$$\sum_{s \in S} b(s) \sum_{s' \in S} \boldsymbol{O}(s', o) P^{\triangle}(s, a, s') > \sum_{s \in S} b(s) \sum_{s' \in S} \boldsymbol{O}(s', o) P'(s, a, s').$$

B7. And hence:

$$\frac{\sum_{s \in S} b(s) \boldsymbol{O}(s', o) P(s, a, s')}{\sum_{s \in S} b(s) \sum_{s' \in S} \boldsymbol{O}(s', o) P^{\triangle}(s, a, s')} < \frac{\sum_{s \in S} b(s) \boldsymbol{O}(s', o) P(s, a, s')}{\sum_{s \in S} b(s) \sum_{s' \in S} \boldsymbol{O}(s', o) P'(s, a, s')} = v.$$

B8. Then, from (A1), (A2), and (B7), we get a contradiction.

B9. Therefore, the assumption made in (B1) is false.

B10. So we get $\forall s \in S$:

$$b(s) \neq 0 \rightarrow \sum_{s' \in S} \boldsymbol{O}(s', o) P'(s, a, s') = \max_{P^{\triangle}(s,a) \in \boldsymbol{P}(s,a)} \sum_{s' \in S} \boldsymbol{O}(s', o) P^{\triangle}(s, a, s').$$

B11. Furthermore, $\forall s \in S$:

$$b(s) = 0 \rightarrow b(s) \sum_{s' \in S} \boldsymbol{O}(s', o) P'(s, a, s') = b(s) \max_{P^{\triangle}(s,a) \in \boldsymbol{P}(s,a)} \sum_{s' \in S} \boldsymbol{O}(s', o) P^{\triangle}(s, a, s').$$

B12. Then from (A1), (A14), (B10) and (B11) it follows that:

$$\min_{\substack{b \in \boldsymbol{B}, P(\cdot,a) \in \boldsymbol{P}(\cdot,a), \\ P'(\cdot,a) \in \boldsymbol{P}(\cdot,a)}} \frac{\sum_{s \in S} b(s) \boldsymbol{O}(s', o) P(s, a, s')}{\sum_{s \in S} b(s) \sum_{s' \in S} \boldsymbol{O}(s', o) P'(s, a, s')} = \min_{b \in \boldsymbol{B}} \frac{\sum_{s \in S} b(s) \min_{P(\cdot,a) \in \boldsymbol{P}(s,a)} \boldsymbol{O}(s', o) P(s, a, s')}{\sum_{s \in S} b(s) \max_{P'(\cdot,a) \in \boldsymbol{P}(\cdot,a)} \sum_{s' \in S} \boldsymbol{O}(s', o) P'(s, a, s')}$$

□

Lemma 6.5.8:

$$\max_{\substack{b\in\boldsymbol{B},P(\cdot,a)\in\boldsymbol{P}(\cdot,a),\\ P'(\cdot,a)\in\boldsymbol{P}(\cdot,a)}} \frac{\sum_{s\in S} b(s)\boldsymbol{O}(s',o)P(s,a,s')}{\sum_{s\in S} b(s)\sum_{s'\in S}\boldsymbol{O}(s',o)P'(s,a,s')} = \max_{b\in\boldsymbol{B}} \frac{\sum_{s\in S} b(s)\max_{P(s,a)\in\boldsymbol{P}(s,a)}\boldsymbol{O}(s',o)P(s,a,s')}{\sum_{s\in S} b(s)\min_{P'(s,a)\in\boldsymbol{P}(s,a)}\sum_{s'\in S}\boldsymbol{O}(s',o)P'(s,a,s')}$$

Lemma 6.5.9:

$$\min_{\substack{b\in\boldsymbol{B},P(\cdot,a)\in\boldsymbol{P}(\cdot,a),\\ P'(\cdot,a)\in\boldsymbol{P}(\cdot,a)}} \frac{b(s)\boldsymbol{O}(s',o)P(s,a,s')}{\sum_{s\in S} b(s)\boldsymbol{O}(s',o)P'(s,a,s')} = \min_{b\in\boldsymbol{B}} \frac{b(s)\min_{P(s,a)\in\boldsymbol{P}(s,a)}\boldsymbol{O}(s',o)P(s,a,s')}{\sum_{s\in S} b(s)\max_{P'(s,a)\in\boldsymbol{P}(s,a)}\boldsymbol{O}(s',o)P'(s,a,s')}$$

Lemma 6.5.10:

$$\max_{\substack{b\in\boldsymbol{B},P(\cdot,a)\in\boldsymbol{P}(\cdot,a),\\ P'(\cdot,a)\in\boldsymbol{P}(\cdot,a)}} \frac{b(s)\boldsymbol{O}(s',o)P(s,a,s')}{\sum_{s\in S} b(s)\boldsymbol{O}(s',o)P'(s,a,s')} = \max_{b\in\boldsymbol{B}} \frac{b(s)\max_{P(s,a)\in\boldsymbol{P}(s,a)}\boldsymbol{O}(s',o)P(s,a,s')}{\sum_{s\in S} b(s)\min_{P'(s,a)\in\boldsymbol{P}(s,a)}\boldsymbol{O}(s',o)P'(s,a,s')}$$

Lemma 6.5.11:

$$\min_{\substack{b\in\boldsymbol{B},P(\cdot,a)\in\boldsymbol{P}(\cdot,a),\\ P'(\cdot,a)\in\boldsymbol{P}(\cdot,a)}} \frac{b(s)\sum_{s'\in S}\boldsymbol{O}(s',o)P(s,a,s')}{\sum_{s\in S} b(s)\sum_{s'\in S}\boldsymbol{O}(s',o)P'(s,a,s')} = \min_{b\in\boldsymbol{B}} \frac{b(s)\min_{P(s,a)\in\boldsymbol{P}(s,a)}\sum_{s'\in S}\boldsymbol{O}(s',o)P(s,a,s')}{\sum_{s\in S} b(s)\max_{P'(s,a)\in\boldsymbol{P}(s,a)}\sum_{s'\in S}\boldsymbol{O}(s',o)P'(s,a,s')}$$

Lemma 6.5.12:

$$\max_{\substack{b\in\boldsymbol{B},P(\cdot,a)\in\boldsymbol{P}(\cdot,a),\\ P'(\cdot,a)\in\boldsymbol{P}(\cdot,a)}} \frac{b(s)\sum_{s'\in S}\boldsymbol{O}(s',o)P(s,a,s')}{\sum_{s\in S} b(s)\sum_{s'\in S}\boldsymbol{O}(s',o)P'(s,a,s')} = \max_{b\in\boldsymbol{B}} \frac{b(s)\max_{P(s,a)\in\boldsymbol{P}(s,a)}\sum_{s'\in S}\boldsymbol{O}(s',o)P(s,a,s')}{\sum_{s\in S} b(s)\min_{P'(s,a)\in\boldsymbol{P}(s,a)}\sum_{s'\in S}\boldsymbol{O}(s',o)P'(s,a,s')}$$

Having these equivalences, we can split the three quadratic fractional optimization problems into two steps. First, we precompute the minimums and maximums of the subfunctions $\boldsymbol{O}(s',o)P(s,a,s')$ and $\sum_{s'\in S}\boldsymbol{O}(s',o)P(s,a,s')$ over the instantiations of the uncertain transition function $\boldsymbol{P}$. Second, we use these minimums to minimize, and maximums to maximize the total function over the instantiations of the old uncertain belief state $\boldsymbol{B}$.

We get one new optimization problem for the first steps of the optimization problems for the subfunction $\boldsymbol{O}(s',o)P(s,a,s')$. Note that we already discuss the second subfunction $\sum_{s'\in S}\boldsymbol{O}(s',o)P(s,a,s')$ in Section 6.4.1, as it also appears in the uncertain transition function. Given states $i,j\in S$, action $a\in A$, and observation $o\in Z$, we get:

Optimization variables	Objective function (min/max)	Feasible set
Precomputing observable transitions		
$\vec{x}=\begin{bmatrix}x_1 & \dots & x_n\end{bmatrix}$	$\boldsymbol{O}(j,o)x_j$	$A_{\boldsymbol{P}(i,a)}\vec{x}\leq\vec{c}_{\boldsymbol{P}(i,a)}$

Let $\widehat{w_{i,j}}$ be the maximum observable transition from state i to state j
Let $\underbrace{w_{i,j}}$ be the minimum observable transition from state i to state j

We use the notations introduced in Section 6.4.1 for the second subfunction:
Let $\widehat{x_i}$ be the maximum observable transition sum for state i
Let $\underset{\smile}{x_i}$ be the minimum observable transition sum for state i

We get the following optimization problems for the second step:

Optimization variables	Objective function (max)	Feasible set
Belief uncertainty intervals		
$\vec{y} = \begin{bmatrix} y_1 & \dots & y_n \end{bmatrix}$	$\frac{\sum_{s \in S} y_s \widehat{w_{s,s'}}}{\sum_{s \in S} y_s \underset{\smile}{x_s}}$	$A_{\boldsymbol{B}} \vec{y} \leq \vec{c}_{\boldsymbol{B}}$
Belief contribution for states with multiple belief sources		
$\vec{y} = \begin{bmatrix} y_1 & \dots & y_n \end{bmatrix}$	$\frac{y_s \widehat{w_{s,s'}}}{\sum_{s \in S} y_s \underset{\smile}{w_{s,s'}}}$	$A_{\boldsymbol{B}} \vec{y} \leq \vec{c}_{\boldsymbol{B}}$
Normalization of belief distribution constraints		
$\vec{y} = \begin{bmatrix} y_1 & \dots & y_n \end{bmatrix}$	$\frac{y_s \widehat{x_s}}{\sum_{s \in S} y_s \underset{\smile}{x_s}}$	$A_{\boldsymbol{B}} \vec{y} \leq \vec{c}_{\boldsymbol{B}}$
Optimization variables	Objective function (min)	Feasible set
Belief uncertainty intervals		
$\vec{y} = \begin{bmatrix} y_1 & \dots & y_n \end{bmatrix}$	$\frac{\sum_{s \in S} y_s \underset{\smile}{w_{s,s'}}}{\sum_{s \in S} y_s \widehat{x_s}}$	$A_{\boldsymbol{B}} \vec{y} \leq \vec{c}_{\boldsymbol{B}}$
Belief contribution for states with multiple belief sources		
$\vec{y} = \begin{bmatrix} y_1 & \dots & y_n \end{bmatrix}$	$\frac{y_s \underset{\smile}{w_{s,s'}}}{\sum_{s \in S} y_s \widehat{w_{s,s'}}}$	$A_{\boldsymbol{B}} \vec{y} \leq \vec{c}_{\boldsymbol{B}}$
Normalization of belief distribution constraints		
$\vec{y} = \begin{bmatrix} y_1 & \dots & y_n \end{bmatrix}$	$\frac{y_s \underset{\smile}{x_s}}{\sum_{s \in S} y_s \widehat{x_s}}$	$A_{\boldsymbol{B}} \vec{y} \leq \vec{c}_{\boldsymbol{B}}$

The new optimization problems are all linear fractional programs. As explained in the section below, we can transform these into linear programs with the Charnes-Cooper transformation.

Alternatively, we can try to solve the linear fractional programs by directly using the mathematical optimization solver Gurobi. Gurobi again does not allow for a fraction as the objective function, but this time moving the denominator to the constraints and introducing an extra variable does work.

$$\begin{aligned} \text{Minimize} \quad & \frac{f(\vec{x})}{g(\vec{x})} \\ \text{subject to} \quad & A\vec{x} \leq \vec{b} \end{aligned}$$

We can transform it to:

$$\begin{aligned} \text{Minimize} \quad & z \\ \text{subject to} \quad & A\vec{x} \leq \vec{b} \\ & z \cdot g(\vec{x}) = f(\vec{x}) \end{aligned}$$

The resulting programs that Gurobi solves are quadratically constrained linear programs.

Charnes-Cooper transformation

Given a non-empty and bounded feasible set, the Charnes-Cooper transformation [Charnes and Cooper, 1962] states that given an optimization problem over an optimization variable $\vec{x}$:

$$\begin{aligned} \text{Maximize} \quad & \frac{\vec{c}^T\vec{x} + \alpha}{\vec{d}^T\vec{x} + \beta} \\ \text{subject to} \quad & A\vec{x} \leq \vec{b} \end{aligned}$$

We can transform it to:

$$\begin{aligned} \text{Maximize} \quad & \vec{c}^T\vec{y} + \alpha t \\ \text{subject to} \quad & A\vec{y} \leq \vec{b}t \\ & \vec{d}^T\vec{y} + \beta t = 1 \\ & t \geq 0 \end{aligned}$$

Using that:

$$\begin{aligned} \vec{y} &= \frac{1}{\vec{d}^T\vec{x} + \beta} \cdot \vec{x} \\ t &= \frac{1}{\vec{d}^T\vec{x} + \beta} \end{aligned}$$

Then given the $\vec{y}^*$ and t^* that maximize the transformed problem, we obtain the solution $\vec{x}^*$ to the original optimization problem by setting:

$$\vec{x}^* = \frac{1}{t^*}\vec{y}^*$$

In the same way, we can transform a minimization problem.

Since the linear numerator and denominator functions of our linear fractional programs do not have a constant part, we can simplify the transformation. Given an optimization problem:

$$\begin{aligned} \text{Maximize} \quad & \frac{\vec{c}^T\vec{x}}{\vec{d}^T\vec{x}} \\ \text{subject to} \quad & A\vec{x} \leq \vec{b} \end{aligned}$$

We can transform it to:

$$\begin{aligned} \text{Maximize} \quad & \vec{c}^T \vec{y} \\ \text{subject to} \quad & A\vec{y} \leq \vec{b}t \\ & \vec{d}^T \vec{y} = 1 \\ & t \geq 0 \end{aligned}$$

Using that:

$$\vec{y} = \frac{1}{\vec{d}^T \vec{x}} \cdot \vec{x}$$
$$t = \frac{1}{\vec{d}^T \vec{x}}$$

Then, again, given the $\vec{y}^*$ and t^* that maximize the transformed problem, we obtain the solution $\vec{x}^*$ to the original optimization problem by setting:

$$\vec{x}^* = \frac{1}{t^*}\vec{y}^*$$

Using this simplified transformation, we end up with the following six linear optimization problems. To prevent confusion, we use a fresh optimization variable $\vec{v} = \begin{bmatrix} v_1 & \dots & v_n \end{bmatrix}$ and add optimization variable t:

Optimization variables	Objective function (max)	Feasible set	Subject to
Belief uncertainty intervals			
$\vec{v} = \begin{bmatrix} v_1 & \dots & v_n \end{bmatrix}, t$	$\sum_{s \in S} v_s \widehat{w_{s,s'}}$	$A_{\boldsymbol{B}}\vec{v} \leq \vec{c}_{\boldsymbol{B}}t$	$\sum_{s \in S} v_s \underset{\smile}{x_s} = 1$ $t \geq 0$
Belief contribution for states with multiple belief sources			
$\vec{v} = \begin{bmatrix} v_1 & \dots & v_n \end{bmatrix}, t$	$v_s \widehat{w_{s,s'}}$	$A_{\boldsymbol{B}}\vec{v} \leq \vec{c}_{\boldsymbol{B}}t$	$\sum_{s \in S} v_s \underbrace{w_{s,s'}} = 1$ $t \geq 0$
Normalization of belief distribution constraints			
$\vec{v} = \begin{bmatrix} v_1 & \dots & v_n \end{bmatrix}, t$	$v_s \widehat{x_s}$	$A_{\boldsymbol{B}}\vec{v} \leq \vec{c}_{\boldsymbol{B}}t$	$\sum_{s \in S} v_s \underset{\smile}{x_s} = 1$ $t \geq 0$

Optimization variables	Objective function (min)	Feasible set	Subject to
Belief uncertainty intervals			
$\vec{v} = \begin{bmatrix} v_1 & \dots & v_n \end{bmatrix}, t$	$\sum_{s \in S} v_s \underbrace{w_{s,s'}}$	$A_{\boldsymbol{B}} \vec{v} \leq \vec{c}_{\boldsymbol{B}} t$	$\sum_{s \in S} v_s \widehat{x_s} = 1$ $t \geq 0$
Belief contribution for states with multiple belief sources			
$\vec{v} = \begin{bmatrix} v_1 & \dots & v_n \end{bmatrix}, t$	$v_s \underbrace{w_{s,s'}}$	$A_{\boldsymbol{B}} \vec{v} \leq \vec{c}_{\boldsymbol{B}} t$	$\sum_{s \in S} v_s \widehat{w_{s,s'}} = 1$ $t \geq 0$
Normalization of belief distribution constraints			
$\vec{v} = \begin{bmatrix} v_1 & \dots & v_n \end{bmatrix}, t$	$v_s \underbrace{x_s}$	$A_{\boldsymbol{B}} \vec{v} \leq \vec{c}_{\boldsymbol{B}} t$	$\sum_{s \in S} v_s \widehat{x_s} = 1$ $t \geq 0$

Note that we can transform the linear equality constraints in the last column to $\leq$-inequalities, as explained in Section 3.3.2, and combine them into the feasible set convex polytope. However, for the readability of the new linear optimization problems, we leave them in this form.

Solving these linear programs completes the second and final step of solving the partially decoupled quadratic fractional programs.

6.6 Random sampling

Given an optimization variable $\vec{x}$, objective function f, and feasible set $A\vec{x} \leq \vec{c}$, we can generate a set of random samples in the feasible set. A sample is an instantiation of the optimization variable that satisfies all linear inequalities of the feasible set. Applying the objective function to each sample and taking the maximum and minimum gives us an under-approximation of the true upper and lower bounds of $f(\vec{x})$.

The more samples you generate, the higher the chance of sampling the instantiations that maximize and minimize the objective function, or at least instantiations that get close to it. As not all feasible sets have an equal amount of uncertainty, we determine the number of samples to generate using a `spud`-parameter (`s`amples `p`er `u`ncertainty `d`istance): we compute the amount of uncertainty based on the interval constraints and multiply this by the `spud`. Note that additional, non-interval constraints that might decrease the total amount of uncertainty are not taken into account here.

We use a straightforward algorithm for generating samples. For simplicity, we assume the feasibility set has a probability distribution constraint over the entire optimization variable. All of the feasible sets in this thesis either have this constraint or consist of smaller independent feasible sets that have that constraint. Those smaller independent feasible sets can then each be sampled in the way described below and straightforwardly combined to form a sample for the entire feasible set.

1. Start with an all-zero sample $\vec{z}$.
2. Let X_{nz} be the variables $x \in \vec{x}$ with non-zero intervals $[a_x, b_x]$ in the feasible set.
3. For each $x \in X_{nz}$ except the last one: Generate a random value $v_x \in [a_x, b_x]$.
4. For the last $x' \in X_{nz}$: Compute $v_{x'} = 1 - \sum_{x \in X_{nz}} v_x$.
5. Check whether $v_{x'} \in [a_{x'}, b_{x'}]$.
   - If so, $\forall x \in X_{nz}$ set $z_x = v_x$.
   - If not, return to step 3.
6. Check whether $\vec{z}$ meets the additional, non-interval constraints (if there are any).
   - If so, $\vec{z}$ is a valid sample.
   - If not, set $\vec{z}$ back to a zero vector and return to step 3.

As mentioned before, we can compute samples from independent smaller feasible sets separately and combine them later. We call the samples from the uncertain state transition function feasible sets $A_{\boldsymbol{P}(i,a)}\vec{x} \leq c_{\boldsymbol{P}(i,a)}$ the i-transition samples. We call the samples from an uncertain belief state feasible set $A_{\boldsymbol{B}}\vec{x} \leq c_{\boldsymbol{B}}$ the belief samples.

As the uncertain state transition function never changes, we can precompute a set of i-transition samples for each state $i \in S$. Whenever we need a complete sample to apply the objective function, we can randomly select samples from the precomputed transition samples and combine them with freshly computed belief samples.

This precomputation approach can greatly improve the computation time. However, it does result in a possible partial overlap between samples and hence reduced randomness. The more transitions are precomputed, the more randomness remains. The number of precomputed transition samples should therefore be a factor larger than the `spud` parameter used for the belief samples. We use a `factor` parameter to indicate the factor.

6.7 Two-phase Succesive Linear Programming Algorithm

Given an optimization variable $\vec{x}$, non-convex quadratic objective function f and feasible set $A\vec{x} \leq \vec{c}$, we can compute upper and lower bounds of $f(\vec{x})$ using the approximate algorithm *Two-phase Successive Linear Programming Algorithm* (2pSLPA) given in [Bentobache et al., 2022]. We chose this algorithm as it nicely fits the constraints of our problem. Let n be the length of $\vec{x}$ and let m be the length of $\vec{c}$, the number of constraints in $A\vec{x} \leq \vec{c}$. We summarize the steps below:

We begin by describing the function in its general form:

$$f(\vec{x}) \;=\; \tfrac{1}{2}\vec{x}^T D\vec{x} + \vec{c}^T\vec{x}$$

Where $\vec{x}, \vec{c} \in \mathbb{R}^n$ and $D \in \mathbb{R}^{n\times n}$ and D is indefinite and symmetric. Since none of the quadratic functions in this thesis have a linear part, we can leave that out and simplify some steps in the algorithm.

$$f(\vec{x}) \;=\; \tfrac{1}{2}\vec{x}^T D\vec{x}$$

Let U be the feasible set described by the feasible set $A\vec{x} \leq \vec{c}$. We first provide the SLPA algorithm. Note that we adjusted the notation to match with the rest of this thesis. Furthermore, we omitted some irrelevant steps because of the indefiniteness of D.

1. Choose $r \in \mathbb{N}^*$ and set k = 0
2. Choose an $(n \times r)$-matrix $H = (\vec{h_j} \mid j \in \{1 \dots r\}, \vec{h_j} \in \mathbb{R}^n)$ and set $J = \emptyset$;
3. For $j \in \{1 \dots r\}$: if $\vec{h_j}^T D\vec{h_j} \neq 0$, then calculate:
   - $\gamma_j = \frac{-2\vec{h_j}^T(D\vec{x_k}+\vec{c})}{\vec{h_j}^T D\vec{h_j}} (= \frac{-2\vec{h_j}^T D\vec{x_k}}{\vec{h_j}^T D\vec{h_j}}$ because our functions have no linear parts)
   - $\vec{y_j} = \vec{x_k} + \gamma_j \vec{h_j}$
   - $J = J \cup \{j\}$

   $\{\vec{y_j} \mid j \in J\}$ forms an approximation set[1] of $f(\vec{x_0})$. Not all $\vec{y_j}$ are necessarily feasible.
4. For $j \in J$, calculate the vectors $\vec{u_j} \in \arg\min_{\vec{x} \in U} \vec{x}^T \nabla f(\vec{y_j})$
5. Calculate the index $p \in J$, such that $f(\vec{u_p}) = \min_{j \in J} f(\vec{u_j})$
6. If $f(\vec{u_p}) < f(\vec{x_k})$:
   - $\vec{u_p}$ improves the current local solution
   - Set $k = k + 1$
   - Got to step 2

⋮

13. Calculate $\overline{\vec{x_k}} \in \arg\min_{\vec{x} \in U} \vec{x}^T \nabla f(\vec{x_k})$.
    If $(\overline{\vec{x_k}} - \vec{x_k})^T \nabla f(\vec{x_k}) \geq 0$, then the algorithm stops with the improved stationary point $\vec{x}^* = \vec{x_k}$. Else, go to step 14
14. By starting at $\vec{x_k}$, apply the Conditional Gradient Algorithm (CGA) to get an improved stationary point $\vec{x}^*$.

For efficiently solving the quadratic program, [Bentobache et al., 2022] proposes the following Two-phase version
Phase 1:

1. Calculate $\vec{x_0} \in \arg\min_{\vec{x} \in U} \vec{c}^T \vec{x}$
   Compute a minimal solution for the linear part of the non-convex quadratic function. Since our numerator and denominator do not have a linear part, we can take a random feasible point here.
2. Compute a stationary point $\vec{x}^*$ by following steps 1 through 12 of the *SLPA* algorithm with $r = n$ and $H = I_n$. When H is the identity matrix, step 3 of the SLPA algorithm will check for any squares in D. Since neither the numerator nor the denominator has this, J will remain the empty set, and the other steps will be skipped.

Phase 2:

3. Choose
$$r = \begin{cases} 10n & \text{if } n < 100 \text{ and } m > 0 \\ n & \text{otherwise} \end{cases}$$
   and generate random matrix $H = (h_{i,j} \mid i \in \{1 \dots n\}, j \in \{1 \dots r\})$ using the uniform distribution on $[-1, 1]$. Apply the *SLPA* algorithm starting from $\vec{x}^*$, which still is $\vec{x_0}$ in our case.

In our case, because of the lack of a linear part, this two-phase version primarily provides a starting point for applying the standard SLPA algorithm.

[1] An approximation set of a point $\vec{z}$ and function f is a finite subset of the level curve: the set of all points $\vec{y} \in \mathbb{R}^n$ with $f(\vec{y}) = f(\vec{z})$.

6.8 Evaluation of the approximations

Throughout this chapter, we have discussed various ways of solving and approximating the optimization problems that follow from our definition of the belief uMDP. We summarize them below.

Linear programs:

- Solve using a mathematical optimization solver.

Quadratic programs:

- Random sampling.
- Two-phase Successive Linear Programming Algorithm.
- Linear transformation.

Quadratic fractional programs:

- Random sampling.
- Full decoupling.
  - Random sampling.
  - Two-phase Successive Linear Programming Algorithm.
  - Linear transformation.
- Partial decoupling.
  - Linear fractional transformation.
    - Solve using a mathematical optimization solver.
    - Charnes-Cooper transformation.

Because of the state independence of the uncertain state transition function, we can solve many of the optimization (sub-)problems using linear programs. The quadratic fractional programs still require approximation. In Chapter 7, we implement the belief uMDP computation, which allows us to compare the different approximation methods. As full decoupling leads to a larger over-approximation than partial decoupling, and both can be transformed into linear programs, we do not consider full decoupling worth comparing. We do consider the random sampling approach a promising method, as a tight under-approximation could be more useful than a big over-approximation. The three most promising approximation methods are hence random sampling and partial decoupling with and without the Charnes-Cooper transformation.

Chapter 7

Numerical experiments

In this chapter, we compare the three most promising approximation methods for the non-convex quadratic fractional programs, introduced in Chapter 6, via numerical experiments. In Section 7.1, we discuss, on a high level, what we implemented to perform the numerical experiments. Section 7.2 describes the four uPOMDP problems we use for the experiments. Next, we describe our experiments in Section 7.3. And finally, we present and analyze the results of the experiments in Section 7.4. The code used for these numerical experiments is available at: GitHub

7.1 Implementation

We implement the entire computation process of a belief uMDP. This roughly consists of reading the uPODMP problem from an input file, precomputing various linear programs, and unfolding uncertain belief states, so computing all successor uncertain belief states and corresponding uncertain transitions and rewards until a time limit is reached. We chose to work with a time limit for the sake of the numerical experiments, but this can easily be changed to work with a horizon-based limit instead. We do not check for duplicate uncertain belief states during the unfolding process. We use a precision parameter to round computed values. Consequently, we need a lower bound to prevent values from becoming 0. We base this lower bound on the precision parameter.

As discussed in Section 6.8, the three most promising approximation methods are random sampling and partial decoupling with and without the Charnes-Cooper transformation. We only split the implementation between these three methods in the uncertain belief update, using the same code for the rest of the computation. We use the mathematical optimization solver Gurobi to solve all linear programs for the computation of the uncertain transition and reward functions and all linear (fractional) programs that follow from the chosen approximation methods.

We store all computed uncertain belief states, transitions, and rewards data. Additionally, we store the time after each completely explored uncertain belief state, meaning all successor uncertain belief states have been computed.

7.2 Problem descriptions

We compare the different approximation methods on three small uPOMDPs based on well-known POMDP problems: the cheese maze [McCallum, 1993], the tiger problem [Kaelbling et al., 1998], and the grid-world robot [Littman et al., 1995]. We extend them to uPOMDPs, as in [Suilen et al., 2020, Itoh and Nakamura, 2007]. The base versions of these problems use uncertainty intervals of size 0.1. We define four uncertainty variants and two size variants of each base problem to compare the effects of increased uncertainty and size on the approximation methods. Hence, we end up with seven versions in total. The four uncertainty variants each increase the uncertainty interval by 0.1. We discuss the two size variants separately for each smaller uPOMDP.

Additionally, we consider a larger, more realistic uPOMDP: the robust aircraft collision avoidance problem [Kochenderfer, 2015]. We refer to this problem as the aircraft problem. This problem is also used to benchmark the uPOMDP policy computation algorithms in [Cubuktepe et al., 2021, Suilen et al., 2020]. Our implementation of this problem is based on [Cubuktepe et al., 2021].

Below, we introduce the problems in more detail[1]. The fully specified uPOMDP tuples of the base versions of the smaller problems can be found in Appendix B. We do not provide the fully specified uPOMDP tuple for the aircraft problem, but instead, we give the PRISM model [Kwiatkowska et al., 2011] and the parser used to generate the input data on GitHub. The headers in the overview tables in the subsections have the following meanings:

Problem	The name of the problem.
Size	The size version of the problem
Unc.	The size of the uncertainty intervals.
$\lvert S \rvert$	The number of states.
$\lvert A \rvert$	The number of actions.
$\lvert Z \rvert$	The number of observations.

7.2.1 Cheese maze

As explained in Section 4.1, the cheese maze problem is about a mouse looking for a piece of cheese in a maze. The maze consists of 14 squares, and the mouse can only observe the position of the walls, not its true position. The mouse can move in any cardinal direction unless blocked by a wall. Once the mouse reaches the cheese in square 13, it will stay there.

The maze is slippery, and the chance of successfully reaching a state is uncertain. Whenever the mouse tries to move, the chance of reaching the intended square lies between 85% and 95%, and the chance of staying in the same square lies between 5% and 15%.

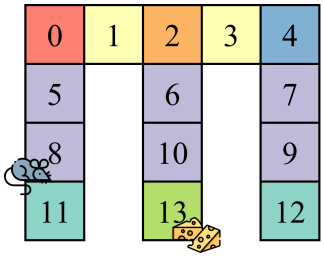
The mouse does not know its initial position. It believes with 80% that it is in square 8, but also with 10% that it is in square 9 or 10.

[1]The figures accompanying the problem descriptions were made using images from `flaticon.com`

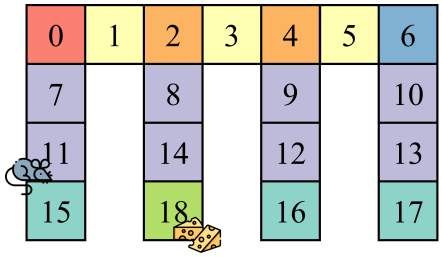
Below, we shortly discuss the size alterations made to the above-described base version of the cheese maze problem. Figure 7.1 depicts the three size versions. Table 7.1 gives an overview of the seven versions of the cheese maze problem we use for the numerical experiments.

Size

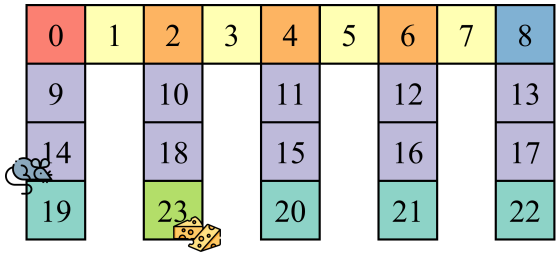
We make two variants of the cheese maze problem with an increased size. We keep the uncertainty intervals at size 0.1 for all size versions. We add one or two extra arches to the maze. The uncertain observation and transition functions follow the same rules as in the base version. The initial belief is again divided over all lower vertical halls, with a 10% chance of being in all but the left-most lower vertical hall, which has the remaining belief.



(a) Base version.



(b) Size 2 version.



(c) Size 3 version.

Figure 7.1: Size versions of the cheese maze problem.

Problem	Unc.	Size	$\|S\|$	$\|A\|$	$\|Z\|$
Cheese maze	0.1	base	14	5	7
Cheese maze	0.2	base	14	5	7
Cheese maze	0.3	base	14	5	7
Cheese maze	0.4	base	14	5	7
Cheese maze	0.5	base	14	5	7
Cheese maze	0.1	s2	19	5	7
Cheese maze	0.1	s3	24	5	7

Table 7.1: Overview of the cheese maze problem versions.

7.2.2 Tiger problem

In the tiger problem, the agent stands in front of two doors. Behind one door is a tiger, representing a big negative reward, and behind the other door is money, representing a medium positive reward. There is an equal chance of the tiger being behind the left and right doors.

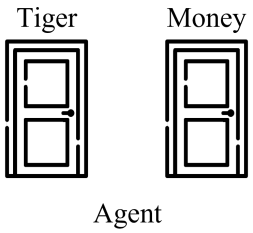
The agent must choose between opening one of the doors or listing to gain more information about the tiger's location. If the agent chooses to open a door, it receives the corresponding reward, and the problem resets. If the agent chooses to listen, it observes the tiger either left or right. This observation is not flawless: the chance of hearing the tiger correctly lies between 80% and 90%, meaning the chance of mishearing it lies between 10% and 20%. The agent can listen as often as it wants, but every time it does, it receives a small negative reward.

The agent starts with the initial belief that there is an equal chance of the tiger being behind the left and right doors.

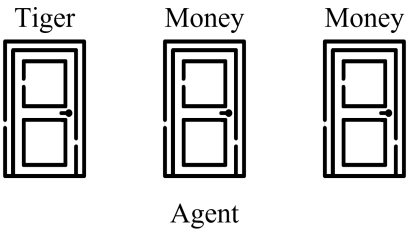
Below, we shortly discuss the size alterations made to the above-described base version of the tiger problem. Figure 7.2 depicts the three size versions. Table 7.2 gives an overview of the seven versions of the tiger problem we use for the numerical experiments.

Size

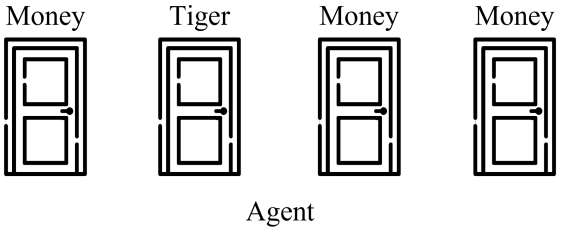
We make two variants of the tiger problem with an increased size. We keep the uncertainty intervals at size 0.1 for all size versions. We add one or two extra doors, hence one or two additional observations and actions. There is still only one tiger. All other doors have rewards behind them. At the start and after each reset, there is an equal chance of the tiger being behind any of the doors. The agent's belief also changes accordingly. For the size 2 version, we use uncertainty intervals of $[0.3, 0.4]$ as rounding $\frac{1}{3}$ would require an uneven probability distribution over the doors.



(a) Base version.



(b) Size 2 version.



(c) Size 3 version.

Figure 7.2: Size versions of the tiger problem.

Problem	Unc.	Size	$\|S\|$	$\|A\|$	$\|Z\|$
Tiger problem	0.1	base	6	3	3
Tiger problem	0.2	base	6	3	3
Tiger problem	0.3	base	6	3	3
Tiger problem	0.4	base	6	3	3
Tiger problem	0.5	base	6	3	3
Tiger problem	0.1	s2	12	4	4
Tiger problem	0.1	s3	20	5	5

Table 7.2: Overview of the tiger problem versions.

7.2.3 Grid-world robot

In the grid-world robot problem, a robot must move through a grid world and reach the southeast corner. The grid world is a 3×3 square and hence consists of 9 squares. The robot can only observe the number of walls surrounding its current square and does not know its true position. Additionally, the robot can observe when it has reached its goal.

As long as the robot has not reached its goal, it can move in any of the four cardinal directions. If the robot chooses to move towards a wall, he will remain in the same square. If the robot chooses to move toward another square, there is a chance of deviating to one of the diagonally adjacent squares. The chance of deviating to diagonally adjacent squares always lies between 5% and 15%. The chance of reaching the intended square depends on whether there are one or two unblocked diagonally adjacent squares. If the robot can only deviate to

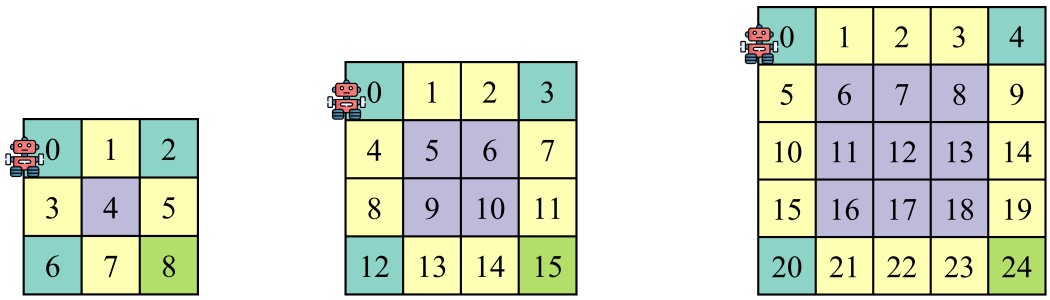
one square, the chance of reaching the intended square lies between 85% and 95%. If it can deviate to two squares, the chance lies between 75% and 85%.

At the start, the robot believes with 80% that it is in the northwest corner, and with 10% that it is in the northeast or southwest corner.

Below, we shortly discuss the size alterations made to the above-described base version of the grid-world robot problem. Figure 7.3 depicts the three size versions. Table 7.3 gives an overview of the seven versions of the grid-world robot problem we use for the numerical experiments.

Size

We make two variants of the grid-world robot problem with an increased size. We keep the uncertainty intervals at size 0.1 for all size versions. We increase the width and height of the grid by one or two. The uncertain observation and transition functions follow the same rules as in the base problem. The initial belief is also still distributed over the non-goal corners.



(a) Base version. (b) Size 2 version. (c) Size 3 version.

Figure 7.3: Size versions of the grid-world robot problem.

Problem	Unc.	Size	$\|S\|$	$\|A\|$	$\|Z\|$
Grid-world robot	0.1	base	9	5	4
Grid-world robot	0.2	base	9	5	4
Grid-world robot	0.3	base	9	5	4
Grid-world robot	0.4	base	9	5	4
Grid-world robot	0.5	base	9	5	4
Grid-world robot	0.1	s2	16	5	4
Grid-world robot	0.1	s3	25	5	4

Table 7.3: Overview of the grid-world robot problem versions.

7.2.4 Aircraft

In the aircraft problem, a pilot must reach a target while avoiding collision with an intruder aircraft in a two-dimensional space by adjusting the acceleration of its aircraft. The states are based on the relative distance and speed difference in the x and y directions between the

aircraft and the intruder aircraft. The pilot of the aircraft cannot determine the exact distance to the intruder aircraft but instead observes an area of possible distances.

Whenever the pilot needs to make an acceleration choice, there is only a 70% to 90% chance that the pilot is responsive, resulting in a 10% to 30% chance of continuing with the same acceleration. The acceleration of the intruder aircraft is determined probabilistically, having a 20% to 80% chance of it being increased and a 20% to 80% chance of it being decreased in each direction.

The pilot starts with a 100% belief that it is in the area one diagonal removed from the southeast corner of the two-dimensional space.

Problem	$\|S\|$	$\|A\|$	$\|Z\|$
Aircraft	10159	8	35

Table 7.4: Overview of the aircraft problem.

7.3 Experiments

For each of the 22 problem versions, we compute six belief uMDPs: two for the partial decoupling methods and four for the random sampling method, varying with the `spud` and `factor` parameters. This means in total, we compute 132 different belief uMDPs. We introduce the abbreviations used to describe the methods below, also recapping the methods briefly.

PD Charnes-Cooper
Partial decoupling with the Charnes-Cooper transformation, solving the resulting linear program with Gurobi.

PD Gurobi
Partial decoupling, solving the resulting linear fractional program with Gurobi.

Sampling `spud`, `ffactor`
Pregenerating random uncertain transition function samples with the number of samples based on `spud` and `factor`, as explained in Section 6.6. Then, for each successor uncertain belief state, generating random uncertain belief state samples with the number of samples based on `spud`, applying the function on all samples and taking the minimum and maximum. The four variants of the random sampling method are formed by the combinations of `spud` $\in \{100, 1000\}$ and `factor` $\in \{2, 5\}$.

We use a time limit of 60 minutes for the aircraft problem and a time limit of 5 minutes for the 21 smaller problems. This time limit includes reading the input and any required precomputation or pregeneration. We run the experiments on a computer with an Intel Core i7-7500u 2.70 GHz processor and 8 GB of RAM. We use Gurobi version 10.0.

Using these experiments, we want to compare the approximation methods based on:

1. The speed of computing a finite horizon belief uMDP.

2. The approximation quality of the computed finite horizon belief uMDPs.

Additionally, we want to investigate the effect of:

1. Increased uncertainty.
2. Increased size.

On those same points.

7.4 Results

We summarize the results in tables. The headers have the following meanings:

Problem	The name of the problem.
Method	The approximation method used.
Size	The size version of the problem
Unc.	The size of the uncertainty intervals.
Min.	The timelimit in minutes.
Expl.	The uncertain belief states fully explored within the time limit. Fully explored means all successor uncertain belief states have been computed.
Found	The uncertain belief states found within the time limit. Found means the uncertain belief state has been computed, but its successor uncertain belief states not necessarily.
Hor.	The largest completed horizon.

7.4.1 Base versions

Table 7.5 summarizes the results of the experiments performed on the base versions of the problems. Below, we discuss the number of explored uncertain belief states and the belief uncertainty intervals.

Problem	Method	Min.	Expl.	Found	Hor.
Aircraft	PD Charnes-Cooper	60	5734	7966	17
Aircraft	PD Gurobi	60	3321	5553	16
Aircraft	Sampling 100, f2	60	22	28	7
Aircraft	Sampling 100, f5	60	22	28	7
Aircraft	Sampling 1000, f2	60	17	22	7
Aircraft	Sampling 1000, f5	60	4	4	3
Cheese maze	PD Charnes-Cooper	5	8923	37497	7
Cheese maze	PD Gurobi	5	3395	14134	6
Cheese maze	Sampling 100, f2	5	1033	4177	5
Cheese maze	Sampling 100, f5	5	1061	4299	6
Cheese maze	Sampling 1000, f2	5	55	216	3
Cheese maze	Sampling 1000, f5	5	55	216	3
Tiger problem	PD Charnes-Cooper	5	1433	5733	6
Tiger problem	PD Gurobi	5	1275	5101	5
Tiger problem	Sampling 100, f2	5	1228	4915	5
Tiger problem	Sampling 100, f5	5	1264	5058	5
Tiger problem	Sampling 1000, f2	5	143	575	4
Tiger problem	Sampling 1000, f5	5	143	573	4
Grid-world robot	PD Charnes-Cooper	5	6199	44107	5
Grid-world robot	PD Gurobi	5	4229	30144	4
Grid-world robot	Sampling 100, f2	5	2041	14591	4
Grid-world robot	Sampling 100, f5	5	2094	14966	4
Grid-world robot	Sampling 1000, f2	5	236	1713	3
Grid-world robot	Sampling 1000, f5	5	239	1733	3

Table 7.5: Overview of the results of the experiments on the base versions.

Explored uncertain belief states

Looking at the results, we see that the two partial decoupling methods evidently outperform the sampling methods regarding the number of explored uncertain belief states. Especially PD Charnes-Cooper, which explores the highest number of uncertain belief states in every experiment we performed. With this method, we can unfold the belief uMDP of a reasonably large and realistic problem, the aircraft problem, up to horizon 17 within one hour.

As the sampling methods require more precomputation than the partial decoupling methods, the big gap between the number of explored uncertain belief states can be partially ascribed to the pregeneration of the uncertain transition function samples. In Figure 7.4, we plot the total passed time against the number of explored uncertain belief states, excluding the precomputation. This figure shows that the partial decoupling methods are also substantially faster with exploring uncertain belief states.

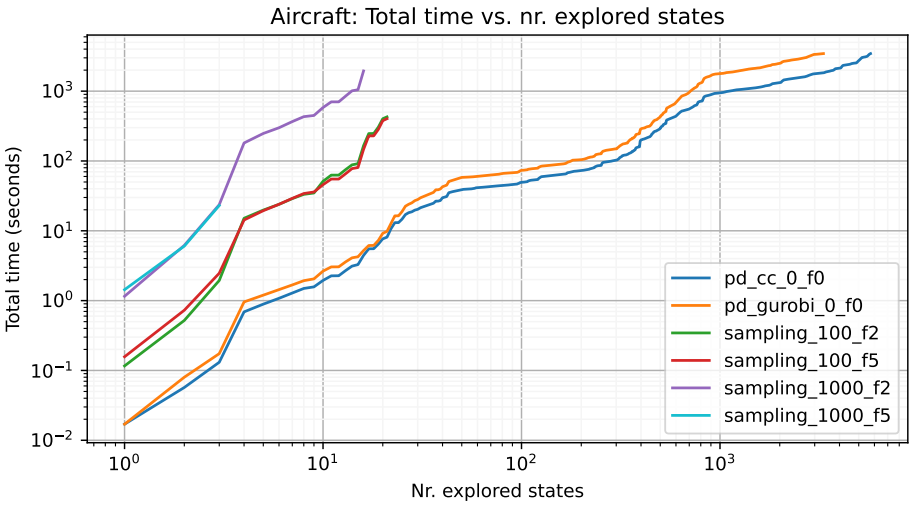


Figure 7.4: Total time vs. the number of explored states in the aircraft problem.

For the aircraft problem, PD Carnes-Cooper outperforms PD Gurobi from the start. We see in some of the other problems that occasionally, PD Gurobi starts faster. However, PD Charnes-Cooper always overtakes PD Gurobi before the time limit.

As expected, the sampling methods with a larger `spud` value explore less uncertain belief states and are slower in their exploration. A larger `factor` value seems to have little effect on the number of explored uncertain belief states, despite requiring a longer precomputation. This could be because generating samples for the uncertain transition function is relatively easy compared to generating those for an uncertain belief state since the uncertain transition function samples are not restricted by additional constraints. Hence, generating uncertain belief state samples might overshadow the additional precomputation time.

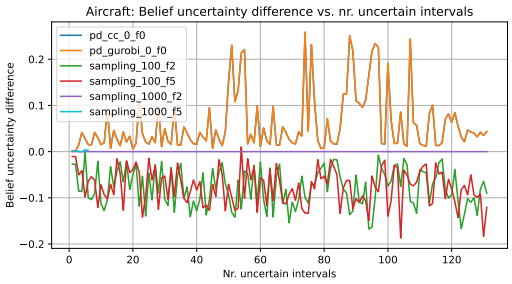
Belief uncertainty intervals

The first thing to note when looking at the computed belief uncertainty intervals is that, as expected, the two partial decoupling method return almost exactly the same values. Not only for the belief uncertainty intervals but also for the additional distribution constraints and the uncertain transitions and rewards. The largest value difference between these two methods for the aircraft problem is 0.000006. These differences are likely caused by tiny differences in the optimal values returned by Gurobi, which stack by continued exploration.

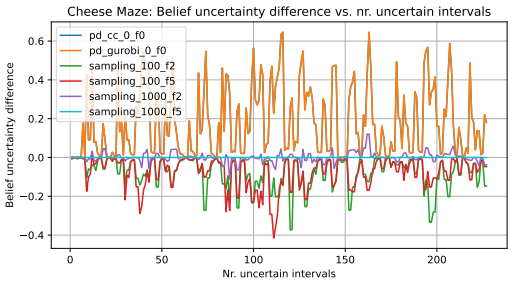
It is difficult to say anything concrete about the quality of the approximations since we do not know the true values of the optimization problems. However, as we consider both under- and over-approximation methods, we can compute some upper and lower bounds on the errors in the belief uncertainty intervals. Note that we talk about the error with respect to the optimization problem here and not with respect to the true uncertain beliefs that these optimization problems over approximate.

We take one of the sampling methods with `spud` value 1000, which in general gives the closest under-approximation, and compute the differences in uncertainty interval size with the other methods. Figure 7.5 plots the computed differences for each base version problem. Note

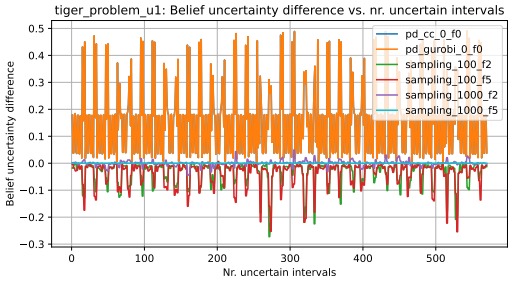
that these plots are limited by the relatively few data points we have for the sampling methods with `spud` value 1000.



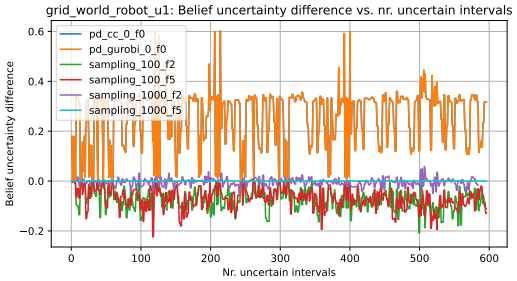
(a) Aircraft.



(b) Cheese maze.



(c) Tiger problem.



(d) Grid-world Robot.

Figure 7.5: Difference in belief uncertainty intervals compared to sampling methods with `spud` value 1000.

Since we know that the correct belief uncertainty intervals are greater than or equal to those computed by the random sampling methods with `spud` value 1000, the negative differences with other sampling methods are lower bounds on the errors for those uncertainty intervals. Similarly, we know that the positive differences with the partial decoupling methods are upper bounds on the errors for those belief uncertainty intervals. The upper and lower bounds on the errors differ strongly between the uPOMDP problems.

As expected, the sampling methods with the larger `spud` value, in general, have larger belief uncertainty intervals, meaning a closer approximation, than the sampling methods with the smaller `spud` value. The larger `factor` value seems to have little effect on the size of the belief uncertainty intervals.

7.4.2 The effect of increased uncertainty

Next, we look at the effect of increased uncertainty in the smaller uPOMDP problems. Tables 7.6 to 7.8 summarize the results of the experiments performed on the increased uncertainty versions of the problems. We grouped the results per method. Below, we again discuss the number of explored uncertain belief states and the belief uncertainty intervals.

Problem	Method	Unc.	Min.	Expl.	Found	Hor.
Cheese maze	PD Charnes-Cooper	0.1	5	8923	37497	7
Cheese maze	PD Charnes-Cooper	0.2	5	6341	26556	7
Cheese maze	PD Charnes-Cooper	0.3	5	7353	30771	7
Cheese maze	PD Charnes-Cooper	0.4	5	8618	36110	7
Cheese maze	PD Charnes-Cooper	0.5	5	4922	20436	7
Cheese maze	PD Gurobi	0.1	5	3395	14134	6
Cheese maze	PD Gurobi	0.2	5	2970	12421	6
Cheese maze	PD Gurobi	0.3	5	2688	11276	6
Cheese maze	PD Gurobi	0.4	5	2673	11224	6
Cheese maze	PD Gurobi	0.5	5	1061	4300	6
Cheese maze	Sampling 100, f2	0.1	5	1033	4177	5
Cheese maze	Sampling 100, f2	0.2	5	657	2697	5
Cheese maze	Sampling 100, f2	0.3	5	287	1127	5
Cheese maze	Sampling 100, f2	0.4	5	253	1002	4
Cheese maze	Sampling 100, f2	0.5	5	217	860	4
Cheese maze	Sampling 100, f5	0.1	5	1061	4299	6
Cheese maze	Sampling 100, f5	0.2	5	649	2671	5
Cheese maze	Sampling 100, f5	0.3	5	287	1132	5
Cheese maze	Sampling 100, f5	0.4	5	257	1015	4
Cheese maze	Sampling 100, f5	0.5	5	212	842	4
Cheese maze	Sampling 1000, f2	0.1	5	55	216	3
Cheese maze	Sampling 1000, f2	0.2	5	24	92	3
Cheese maze	Sampling 1000, f2	0.3	5	19	69	2
Cheese maze	Sampling 1000, f2	0.4	5	14	59	2
Cheese maze	Sampling 1000, f2	0.5	5	14	54	2
Cheese maze	Sampling 1000, f5	0.1	5	55	216	3
Cheese maze	Sampling 1000, f5	0.2	5	24	92	3
Cheese maze	Sampling 1000, f5	0.3	5	19	69	2
Cheese maze	Sampling 1000, f5	0.4	5	14	59	2
Cheese maze	Sampling 1000, f5	0.5	5	14	54	2

Table 7.6: Overview of the results of the experiments on the increased uncertainty versions of the cheese maze problem.

Problem	Method	Unc.	Min.	Expl.	Found	Hor.
Tiger problem	PD Charnes-Cooper	0.1	5	1433	5733	6
Tiger problem	PD Charnes-Cooper	0.2	5	1686	6744	6
Tiger problem	PD Charnes-Cooper	0.3	5	1680	6722	6
Tiger problem	PD Charnes-Cooper	0.4	5	1668	6672	6
Tiger problem	PD Charnes-Cooper	0.5	5	1688	6752	6
Tiger problem	PD Gurobi	0.1	5	1275	5101	5
Tiger problem	PD Gurobi	0.2	5	1296	5186	5
Tiger problem	PD Gurobi	0.3	5	1225	4901	5
Tiger problem	PD Gurobi	0.4	5	1210	4842	5
Tiger problem	PD Gurobi	0.5	5	1204	4818	5
Tiger problem	Sampling 100, f2	0.1	5	1228	4915	5
Tiger problem	Sampling 100, f2	0.2	5	693	2774	5
Tiger problem	Sampling 100, f2	0.3	5	499	1998	5
Tiger problem	Sampling 100, f2	0.4	5	325	1303	4
Tiger problem	Sampling 100, f2	0.5	5	328	1313	4
Tiger problem	Sampling 100, f5	0.1	5	1264	5058	5
Tiger problem	Sampling 100, f5	0.2	5	685	2742	5
Tiger problem	Sampling 100, f5	0.3	5	495	1983	5
Tiger problem	Sampling 100, f5	0.4	5	351	1406	5
Tiger problem	Sampling 100, f5	0.5	5	324	1297	4
Tiger problem	Sampling 1000, f2	0.1	5	143	575	4
Tiger problem	Sampling 1000, f2	0.2	5	73	295	3
Tiger problem	Sampling 1000, f2	0.3	5	54	218	3
Tiger problem	Sampling 1000, f2	0.4	5	44	177	3
Tiger problem	Sampling 1000, f2	0.5	5	36	147	3
Tiger problem	Sampling 1000, f5	0.1	5	143	573	4
Tiger problem	Sampling 1000, f5	0.2	5	72	289	3
Tiger problem	Sampling 1000, f5	0.3	5	53	215	3
Tiger problem	Sampling 1000, f5	0.4	5	44	178	3
Tiger problem	Sampling 1000, f5	0.5	5	36	146	3

Table 7.7: Overview of the results of the experiments on the increased uncertainty versions of the tiger problem.

Problem	Method	Unc.	Min.	Expl.	Found	Hor.
Grid-world robot	PD Charnes-Cooper	0.1	5	6199	44107	5
Grid-world robot	PD Charnes-Cooper	0.2	5	7659	54459	5
Grid-world robot	PD Charnes-Cooper	0.3	5	6836	48646	5
Grid-world robot	PD Charnes-Cooper	0.4	5	6139	43686	5
Grid-world robot	PD Charnes-Cooper	0.5	5	6052	43049	4
Grid-world robot	PD Gurobi	0.1	5	4229	30144	4
Grid-world robot	PD Gurobi	0.2	5	4562	32494	4
Grid-world robot	PD Gurobi	0.3	5	4175	29744	4
Grid-world robot	PD Gurobi	0.4	5	3449	24591	4
Grid-world robot	PD Gurobi	0.5	5	3965	28260	4
Grid-world robot	Sampling 100, f2	0.1	5	2041	14591	4
Grid-world robot	Sampling 100, f2	0.2	5	1306	9387	4
Grid-world robot	Sampling 100, f2	0.3	5	994	7137	4
Grid-world robot	Sampling 100, f2	0.4	5	823	5932	3
Grid-world robot	Sampling 100, f2	0.5	5	613	4419	3
Grid-world robot	Sampling 100, f5	0.1	5	2094	14966	4
Grid-world robot	Sampling 100, f5	0.2	5	1153	8297	4
Grid-world robot	Sampling 100, f5	0.3	5	838	6041	3
Grid-world robot	Sampling 100, f5	0.4	5	819	5901	3
Grid-world robot	Sampling 100, f5	0.5	5	555	4008	3
Grid-world robot	Sampling 1000, f2	0.1	5	236	1713	3
Grid-world robot	Sampling 1000, f2	0.2	5	148	1090	3
Grid-world robot	Sampling 1000, f2	0.3	5	105	786	2
Grid-world robot	Sampling 1000, f2	0.4	5	79	604	2
Grid-world robot	Sampling 1000, f2	0.5	5	72	543	2
Grid-world robot	Sampling 1000, f5	0.1	5	239	1733	3
Grid-world robot	Sampling 1000, f5	0.2	5	149	1095	3
Grid-world robot	Sampling 1000, f5	0.3	5	101	761	2
Grid-world robot	Sampling 1000, f5	0.4	5	77	582	2
Grid-world robot	Sampling 1000, f5	0.5	5	72	543	2

Table 7.8: Overview of the results of the experiments on the increased uncertainty versions of the grid-world robot problem.

Explored uncertain belief states

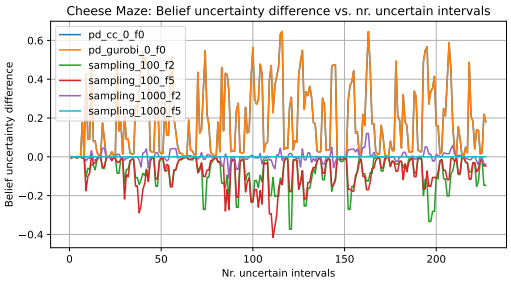
As the highlighted rows indicate, the uncertainty increase does not necessarily lead to a slow-down for the partial decoupling methods. The larger uncertainty intervals might lead to linear programs that are relatively easier to solve for Gurobi. We do not see this effect in the sampling methods, but this might be overshadowed by the time required to (pre)generate samples.

Another option is that it is caused by small differences in the computer usage on which the experiments ran. Although the computer was not occupied with other tasks while running the experiments, activity when starting the experiments and irregular checks to see whether the

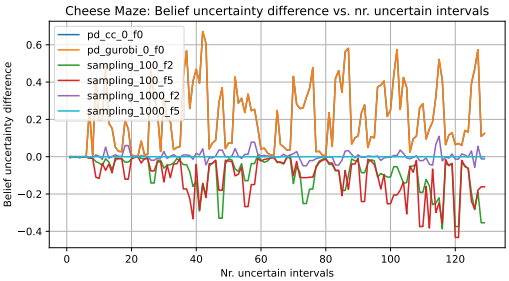
code was still running might have caused a difference in available resources. We unintentionally reran the base version problems for the increased size version comparison and, as can be seen in Tables 7.9 to 7.11, the number of explored uncertain belief states is higher for most of the problems. This difference supports our second theory. We highlighted the rows in Tables 7.9 to 7.11 where the rerun resulted in a larger number of explored uncertain belief states.

Belief uncertainty intervals

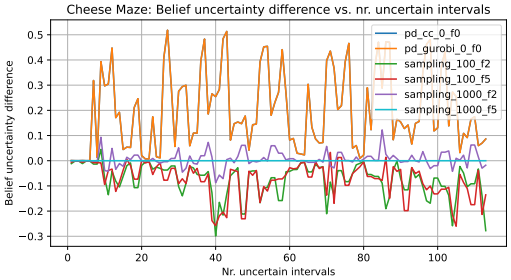
Comparing the belief uncertainty intervals for the same method on multiple uncertainty interval versions makes little sense, as the intervals are supposed to have different sizes. We can, however, try to compare the differences in belief uncertainty intervals compared to a sampling method with `spud` value 1000. Figures 7.6 to 7.8 show the differences in the belief uncertainty intervals for the increased uncertainty versions of the three smaller uPOMDP problems.



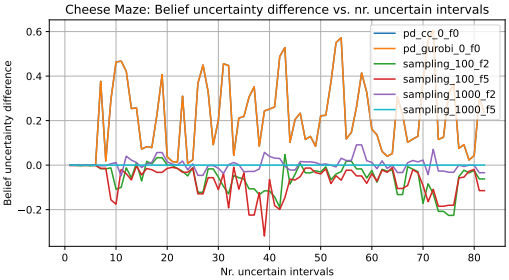
(a) Uncertainty interval 0.1.



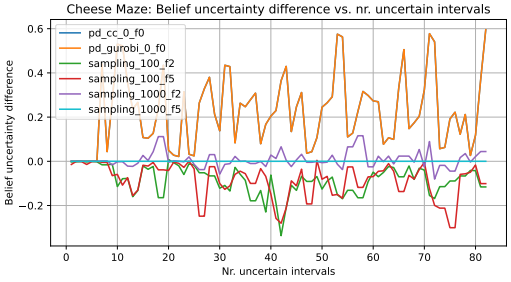
(b) Uncertainty interval 0.2.



(c) Uncertainty interval 0.3.

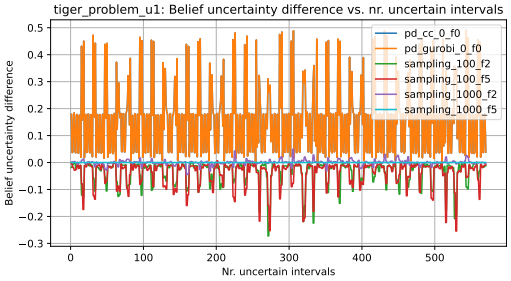


(d) Uncertainty interval 0.4.

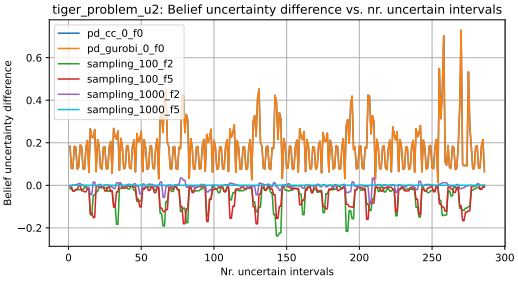


(e) Uncertainty interval 0.5.

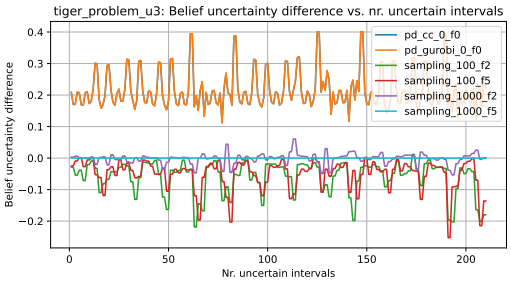
Figure 7.6: Difference in belief uncertainty intervals compared to sampling methods with `spud` value 1000 of the increased uncertainty versions of the cheese maze problem.



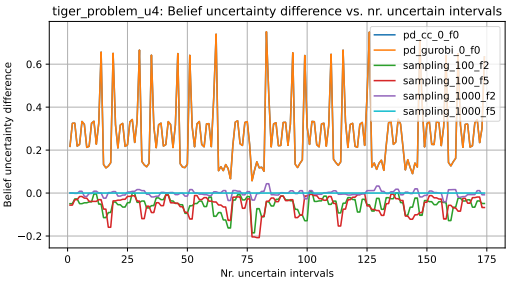
(a) Uncertainty interval 0.1.



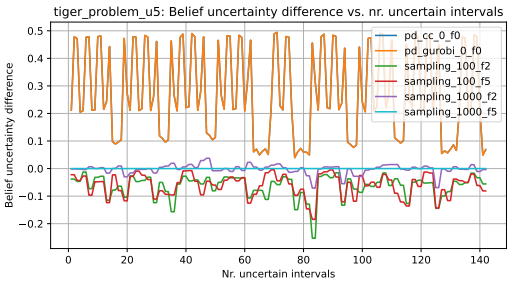
(b) Uncertainty interval 0.2.



(c) Uncertainty interval 0.3.

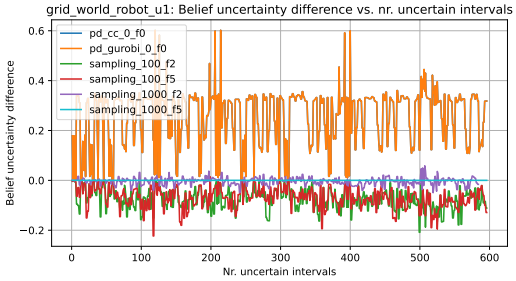


(d) Uncertainty interval 0.4.

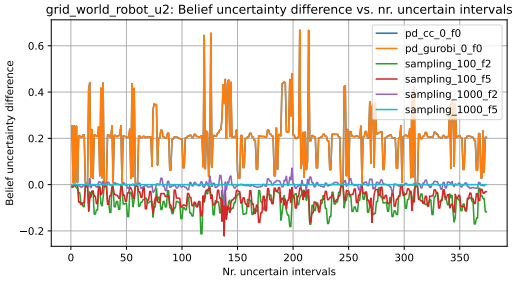


(e) Uncertainty interval 0.5.

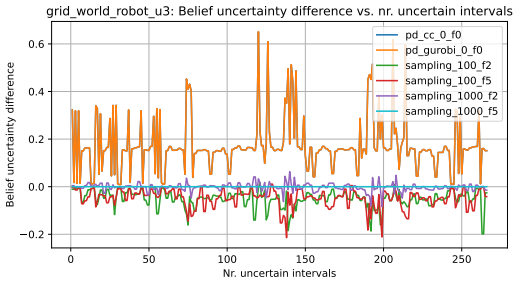
Figure 7.7: Difference in belief uncertainty intervals compared to sampling methods with `spud` value 1000 of the increased uncertainty versions of the tiger problem.



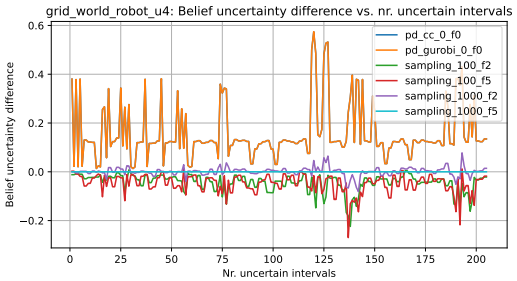
(a) Uncertainty interval 0.1.



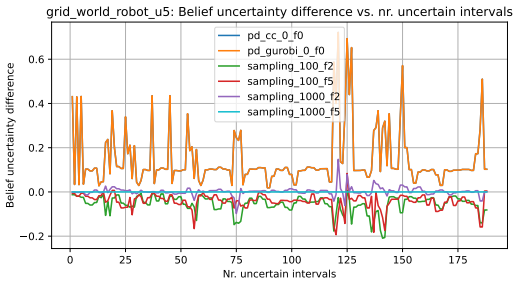
(b) Uncertainty interval 0.2.



(c) Uncertainty interval 0.3.



(d) Uncertainty interval 0.4.



(e) Uncertainty interval 0.5.

Figure 7.8: Difference in belief uncertainty intervals compared to sampling methods with `spud` value 1000 of the increased uncertainty versions of the grid-world robot problem.

Although we can identify some shifted patterns in the plotted differences of the partial decoupling methods of the tiger and grid-world robot problems, it is difficult to explain the effect of the increased uncertainty on the belief uncertainty intervals in general.

The pattern shift in the tiger problem versions towards greater differences could be attributed to the reset of the belief after the agent opens a door. On the other hand, the pattern shift in the grid-world robot problem towards smaller differences might be caused by the correct uncertainty interval becoming so large that there is limited space for over-approximation.

7.4.3 The effect of increased size

Finally, we look at the effect of increased size on the smaller uPOMDP problems. Tables 7.9 to 7.11 summarize the results of the experiments performed on the increased uncertainty versions of the problems. We grouped the results per method.

Problem	Method	Size	Min.	Expl.	Found	Hor.
Cheese maze	PD Charnes-Cooper	base	5	8276	34628	7
Cheese maze	PD Charnes-Cooper	s2	5	2883	12777	6
Cheese maze	PD Charnes-Cooper	s3	5	1393	5939	6
Cheese maze	PD Gurobi	base	5	3408	14180	6
Cheese maze	PD Gurobi	s2	5	1099	4657	6
Cheese maze	PD Gurobi	s3	5	829	3556	5
Cheese maze	Sampling 100, f2	base	5	1027	4168	5
Cheese maze	Sampling 100, f2	s2	5	251	1028	4
Cheese maze	Sampling 100, f2	s3	5	146	591	4
Cheese maze	Sampling 100, f5	base	5	1025	4164	5
Cheese maze	Sampling 100, f5	s2	5	260	1064	4
Cheese maze	Sampling 100, f5	s3	5	150	611	4
Cheese maze	Sampling 1000, f2	base	5	55	216	3
Cheese maze	Sampling 1000, f2	s2	5	19	69	2
Cheese maze	Sampling 1000, f2	s3	5	14	54	2
Cheese maze	Sampling 1000, f5	base	5	55	216	3
Cheese maze	Sampling 1000, f5	s2	5	19	69	2
Cheese maze	Sampling 1000, f5	s3	5	14	54	2

Table 7.9: Overview of the results of the experiments on the increased size versions of the cheese maze problem.

Problem	Method	Size	Min.	Expl.	Found	Hor.
Tiger problem	PD Charnes-Cooper	base	5	1760	7040	6
Tiger problem	PD Charnes-Cooper	s2	5	628	3768	4
Tiger problem	PD Charnes-Cooper	s3	5	271	2175	3
Tiger problem	PD Gurobi	base	5	1426	5706	6
Tiger problem	PD Gurobi	s2	5	446	2678	4
Tiger problem	PD Gurobi	s3	5	231	1852	3
Tiger problem	Sampling 100, f2	base	5	1474	5899	6
Tiger problem	Sampling 100, f2	s2	5	244	1467	3
Tiger problem	Sampling 100, f2	s3	5	114	916	3
Tiger problem	Sampling 100, f5	base	5	1436	5747	6
Tiger problem	Sampling 100, f5	s2	5	245	1472	3
Tiger problem	Sampling 100, f5	s3	5	111	893	3
Tiger problem	Sampling 1000, f2	base	5	151	606	4
Tiger problem	Sampling 1000, f2	s2	5	24	146	2
Tiger problem	Sampling 1000, f2	s3	5	14	117	2
Tiger problem	Sampling 1000, f5	base	5	151	607	4
Tiger problem	Sampling 1000, f5	s2	5	23	139	2
Tiger problem	Sampling 1000, f5	s3	5	14	114	2

Table 7.10: Overview of the results of the experiments on the increased uncertainty size of the tiger problem.

Problem	Method	Size	Min.	Expl.	Found	Hor.
Grid-world robot	PD Charnes-Cooper	base	5	8003	56948	5
Grid-world robot	PD Charnes-Cooper	s2	5	2428	19010	4
Grid-world robot	PD Charnes-Cooper	s3	5	1606	12431	4
Grid-world robot	PD Gurobi	base	5	5394	38414	4
Grid-world robot	PD Gurobi	s2	5	1435	11336	4
Grid-world robot	PD Gurobi	s3	5	1058	8169	4
Grid-world robot	Sampling 100, f2	base	5	2156	15413	4
Grid-world robot	Sampling 100, f2	s2	5	808	6433	3
Grid-world robot	Sampling 100, f2	s3	5	563	4354	3
Grid-world robot	Sampling 100, f5	base	5	2261	16128	4
Grid-world robot	Sampling 100, f5	s2	5	752	6011	3
Grid-world robot	Sampling 100, f5	s3	5	554	4275	3
Grid-world robot	Sampling 1000, f2	base	5	241	1751	3
Grid-world robot	Sampling 1000, f2	s2	5	78	655	2
Grid-world robot	Sampling 1000, f2	s3	5	74	573	2
Grid-world robot	Sampling 1000, f5	base	5	241	1747	3
Grid-world robot	Sampling 1000, f5	s2	5	78	653	2
Grid-world robot	Sampling 1000, f5	s3	5	72	560	2

Table 7.11: Overview of the results of the experiments on the increased size versions of the grid-world robot problem.

We do not see anything unexpected in the number of explored uncertain belief states. The factor with which this number decreases varies quite a lot between the different problems and different increases, but this can be attributed to the ways the problems are extended in combination with the time limit.

We cannot fairly compare the belief uncertainty intervals of the increased size versions, as the uncertain belief states now concern different numbers of states.

Chapter 8

Discussion

In this chapter, we discuss some flaws of the project. We first discuss some suboptimal choices made during the implementation and the consequences thereof. Next, we discuss the lack of guarantees we can give about the quality of the various approximations made.

Implementation

Over the course of the project, our objectives for the implementation changed multiple times due to changes in the definition of the uncertain belief state and which approximation methods seemed most promising. Due to this, some choices that made sense in an earlier implementation turned into flaws in the final implementation. Furthermore, as the initial implementation was made with only the smaller uPOMDPs in mind, we implemented several parts of the code in a naive fashion.

One of those naive parts of code is the sample generation, which is a clear bottleneck for the random sampling approximation methods. Especially when the additional distribution constraints on an uncertain belief state are tight, our simple implementation results in many rejected samples and hence requires a lot of additional computation to generate the desired number of samples. Replacing the sample generation with a proper sampling algorithm, such as [Kannan and Narayanan, 2012, Chen et al., 2018], would make for a fairer comparison time-wise with the other approximation methods.

Another thing we implemented rather naively is the way we explore the uncertain belief states. Although, in an earlier version, we looked for duplicate uncertain belief states to prevent recomputation, we removed this when we added the additional distribution constraints. Especially in a model that resets after certain choices, such as the tiger problem, recomputing the successor uncertain belief states of a duplicate state wastes a lot of time.

A final flaw in the implementation is the precision parameter. We introduced this to compare the uncertain belief states to see which are sufficiently close to be considered the same uncertain belief state. In the current version of the code, which we used for the numerical experiments, we do not check for duplicates anymore, so this has become superfluous. Furthermore, we could check for duplicates by rounding only for this comparison. However, this requires either a lot of repeated computation or additional memory to store a rounded copy of each uncertain belief state.

Due to the precision parameter, we also had to add a lower bound on the computed uncertainty intervals to ensure graph preservation. Both the precision parameter and the resulting lower bound cause unintended and unnecessary errors in the model.

Approximation errors

As explained in the Introduction, the belief uMDP we defined in this thesis is an over-approximation of the true set of belief MDPs. On top of this approximation, we approximate various optimization problems. Furthermore, unintentional error can enter the model via the use of Gurobi and the precision parameter. We elaborate on the unintentional errors below.

According to Gurobi's documentation [Gurobi Optimization, LLC, 2023], the `MIPgap` parameter is used as the acceptable gap between the upper and lower bounds on the optimal solution when looking at, among others, non-convex quadratic models. This parameter has a default value of 0.0001. As mentioned in Section 6.5, Gurobi can compute solutions for the linear fractional programs by transforming them into quadratically constrained linear programs. During the implementation, Gurobi could only find solutions for these models with the non-convex flag set, meaning this gap might have been involved. This could also explain the tiny differences we found between the computed values in the uncertain belief states of the two partial decoupling methods.

Regarding the precision parameter, error can enter the model both via the rounding of computed values and the consequently necessary lower bound on the belief uncertainty intervals. Initially, this error is bounded by the precision parameter, but this quickly increases because of the successive computations in the belief uMDP.

Apart from the `MIPgap`, we have no information about the amount of error that seeps into the model by the various approximations and discrepancies. Due to this, we cannot give any concrete guarantees about the quality of belief uMDP or finite horizon policies computed on it. By removing the precision parameter, we do know that the finite horizon belief uMDPs computed with partial decoupling with the Charnes-Cooper transformation is an over-approximation of the true belief uMDP.

Chapter 9

Conclusions

In this thesis, we define a computationally tractable over-approximation of the underlying belief model of the uncertain POMDP, referred to as the belief uMDP.

We define the uncertain belief state, a convex polytope over the belief in each state, which over-approximates the true possible belief. Based on this definition, we define the belief uMDP, following the blueprint of the belief MDP and adjusting it to work with probability intervals. We add a second step to the uncertain version of the belief update function to deal with information loss.

Our definition of the belief uMDP gives rise to various optimization problems. We use the property of independence between transitions from different states, which follows from our (s,a)-rectangularity assumption, to transform some parts of the optimization problems into (precomputable) linear programs. Still, we need to deal with various non-convex quadratic fractional programs, which require approximations to become computationally tractable. We discuss various ways of approximating the uncertainty intervals, the most promising being under-approximation via random sampling on the uncertainty set and over-approximations by partial decoupling of the links between the transition variables in the numerator and denominator functions. The partial decoupling method can be extended with the Charnes-Cooper transformation.

We implement finite horizon computation of the belief uMDP with the three most promising approximation methods for the non-convex quadratic fractional programs: the random sampling approach and the partial decoupling approach with and without the Charnes-Cooper transformation. We use the mathematical optimization solver Gurobi to solve all linear (fractional) programs.

We compare the approximation methods based on the number of explored uncertain belief states within a given time limit and the sizes of the probability intervals of the computed uncertain belief states. Furthermore, we investigate the effect of increased uncertainty and increased model size. Most notably, with the partial decoupling approach with the Charnes-Cooper transformation, we can explore the belief uMDP of a relatively large uPOMDP up to horizon 17 within one hour.

Future research

There are many interesting future research directions concerning both improvement and extensions of the current work. We summarize them below:

- As explained in the Discussion, the final implementation that we used for the numerical experiments contains some flaws that should be ironed out. Improving these should lead to a smaller approximation error, a fairer comparison with the sampling approximation methods of the non-convex quadratic fractional programs, and a more efficient exploration strategy.
- Also discussed in the Discussion, another valuable area of improvement is the available information on the approximation error of the general approximation of the true belief uMDP, as well as of the various approximations of the non-convex quadratic fractional programs. Although it might not be possible to establish anything concrete, some time should be spent exploring this.
- We do not have any formal proofs to support that the belief uMDP we defined in this thesis over-approximates the true belief uMDP. Although we argue, for each function in the belief uMDP, how it ensures that all true probabilities are contained, it would be good if we could support those claims with proofs.
- During this thesis, we chose to use the interval version of uncertainty. To make the belief uMDP framework more general, it could be interesting to extend the definition to other forms of uncertainty, such as the point-set uncertainty of [Itoh and Nakamura, 2007].
- We often mentioned the possibility of applying finite horizon uMDP algorithms to the belief uMDP but have not yet done so in this thesis. Future research in this direction would allow for performance comparisons with other uPOMDP policy algorithms.
- As mentioned in Section 5.8, the uncertain belief update function has some additional possible applications. Both lifting POMDP policy computation algorithms that rely on the belief update function to the uncertain setting [Walraven and Spaan, 2019, Shani et al., 2013] and robust reinforcement learning for POMDPs [Kaelbling et al., 1996] are interesting future research directions.
- Testing the belief uMDP on a real-life case study could provide interesting insights into the other benefits of having the underlying belief model, namely the repeated finite horizon policy computation without full recomputation and the insight into the effect of the uncertainty.

Bibliography

[Ahmadi et al., 2018] Ahmadi, M., Cubuktepe, M., Jansen, N., and Topcu, U. (2018). Verification of uncertain pomdps using barrier certificates. In Allerton, pages 115–122. IEEE.

[Baier and Katoen, 2008] Baier, C. and Katoen, J. (2008). Principles of model checking. MIT Press.

[Beck and Teboulle, 2010] Beck, A. and Teboulle, M. (2010). On minimizing quadratically constrained ratio of two quadratic functions. Journal of Convex Analysis, 17(3):789–804.

[Bentobache et al., 2022] Bentobache, M., Telli, M., and Mokhtari, A. (2022). New lp-based local and global algorithms for continuous and mixed-integer nonconvex quadratic programming. J. Glob. Optim., 82(4):659–689.

[Boyd and Vandenberghe, 2014] Boyd, S. P. and Vandenberghe, L. (2014). Convex Optimization. Cambridge University Press.

[Chamie and Mostafa, 2018] Chamie, M. E. and Mostafa, H. (2018). Robust action selection in partially observable markov decision processes with model uncertainty. In CDC, pages 5586–5591. IEEE.

[Charnes and Cooper, 1962] Charnes, A. and Cooper, W. W. (1962). Programming with linear fractional functionals. Nav. Res. Logist. Q., 9(3-4):181–186.

[Chatterjee et al., 2016] Chatterjee, K., Chmelik, M., Gupta, R., and Kanodia, A. (2016). Optimal cost almost-sure reachability in pomdps. Artif. Intell., 234:26–48.

[Chen et al., 2018] Chen, Y., Dwivedi, R., Wainwright, M. J., and Yu, B. (2018). Fast MCMC sampling algorithms on polytopes. J. Mach. Learn. Res., 19:55:1–55:86.

[Chong and Żak, 2013] Chong, E. K. P. and Żak, S. H. (2013). An Introduction to Optimization. Wiley Series in Discrete Mathematics and Optimization. Wiley, fourth edition.

[Cubuktepe et al., 2021] Cubuktepe, M., Jansen, N., Junges, S., Marandi, A., Suilen, M., and Topcu, U. (2021). Robust finite-state controllers for uncertain pomdps. In AAAI, pages 11792–11800. AAAI Press.

[Givan et al., 2000] Givan, R., Leach, S. M., and Dean, T. L. (2000). Bounded-parameter markov decision processes. Artif. Intell., 122(1-2):71–109.

[Gurobi Optimization, LLC, 2023] Gurobi Optimization, LLC (2023). Gurobi Optimizer Reference Manual.

[Itoh and Nakamura, 2007] Itoh, H. and Nakamura, K. (2007). Partially observable markov decision processes with imprecise parameters. Artif. Intell., 171(8-9):453–490.

[Iyengar, 2005] Iyengar, G. N. (2005). Robust dynamic programming. Math. Oper. Res., 30(2):257–280.

[Kaelbling et al., 1998] Kaelbling, L. P., Littman, M. L., and Cassandra, A. R. (1998). Planning and acting in partially observable stochastic domains. Artif. Intell., 101(1-2):99–134.

[Kaelbling et al., 1996] Kaelbling, L. P., Littman, M. L., and Moore, A. W. (1996). Reinforcement learning: A survey. J. Artif. Intell. Res., 4:237–285.

[Kannan and Narayanan, 2012] Kannan, R. and Narayanan, H. (2012). Random walks on polytopes and an affine interior point method for linear programming. Mathematics of Operations Research, 37(1):1–20.

[Kochenderfer, 2015] Kochenderfer, M. J. (2015). Decision Making Under Uncertainty. MIT Lincoln Laboratory Series. MIT Press.

[Kochenderfer and Wheeler, 2019] Kochenderfer, M. J. and Wheeler, T. A. (2019). Algorithms for Optimization. MIT Press.

[Kwiatkowska et al., 2011] Kwiatkowska, M., Norman, G., and Parker, D. (2011). PRISM 4.0: Verification of probabilistic real-time systems. In Gopalakrishnan, G. and Qadeer, S., editors, Proc. 23rd International Conference on Computer Aided Verification (CAV'11), volume 6806 of LNCS, pages 585–591. Springer.

[Liberzon, 2003] Liberzon, D. (2003). Switching in Systems and Control. Systems & Control: Foundations & Applications. Birkhäuser.

[Lipp and Boyd, 2016] Lipp, T. and Boyd, S. (2016). Variations and extension of the convex-concave procedure. Optim. Eng., 17(2):263–287.

[Littman et al., 1995] Littman, M. L., Cassandra, A. R., and Kaelbling, L. P. (1995). Learning policies for partially observable environments: Scaling up. In ICML, pages 362–370. Morgan Kaufmann.

[Madani et al., 2003] Madani, O., Hanks, S., and Condon, A. (2003). On the undecidability of probabilistic planning and related stochastic optimization problems. Artif. Intell., 147(1-2):5–34.

[McCallum, 1993] McCallum, A. (1993). Overcoming incomplete perception with utile distinction memory. In ICML, pages 190–196. Morgan Kaufmann.

[Meuleau et al., 1999] Meuleau, N., Kim, K., Kaelbling, L. P., and Cassandra, A. R. (1999). Solving pomdps by searching the space of finite policies. In UAI, pages 417–426. Morgan Kaufmann.

[Nguyen et al., 2014] Nguyen, V.-B., Sheu, R.-L., and Xia, Y. (2014). An sdp approach for solving quadratic fractional programming problems.

[Nilim and Ghaoui, 2005] Nilim, A. and Ghaoui, L. E. (2005). Robust control of markov decision processes with uncertain transition matrices. Oper. Res., 53(5):780–798.

[Osogami, 2015] Osogami, T. (2015). Robust partially observable markov decision process. In ICML, volume 37 of JMLR Workshop and Conference Proceedings, pages 106–115. JMLR.org.

[Prajna et al., 2007] Prajna, S., Jadbabaie, A., and Pappas, G. J. (2007). A framework for worst-case and stochastic safety verification using barrier certificates. IEEE Trans. Autom. Control., 52(8):1415–1428.

[Puterman, 1994] Puterman, M. L. (1994). Markov Decision Processes: Discrete Stochastic Dynamic Programming. Wiley Series in Probability and Statistics. Wiley.

[Sahni, 1974] Sahni, S. (1974). Computationally related problems. SIAM J. Comput., 3(4):262–279.

[Shani et al., 2013] Shani, G., Pineau, J., and Kaplow, R. (2013). A survey of point-based POMDP solvers. Auton. Agents Multi Agent Syst., 27(1):1–51.

[Suilen et al., 2020] Suilen, M., Jansen, N., Cubuktepe, M., and Topcu, U. (2020). Robust policy synthesis for uncertain pomdps via convex optimization. In IJCAI, pages 4113–4120. ijcai.org.

[Suilen et al., 2022] Suilen, M., Simão, T. D., Parker, D., and Jansen, N. (2022). Robust anytime learning of markov decision processes. In NeurIPS.

[Walraven and Spaan, 2019] Walraven, E. and Spaan, M. T. J. (2019). Point-based value iteration for finite-horizon pomdps. J. Artif. Intell. Res., 65:307–341.

[Wiesemann et al., 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 CONCUR, volume 140 of LIPIcs, pages 14:1–14:17. Schloss Dagstuhl - Leibniz-Zentrum für Informatik.

[Wolff et al., 2012] Wolff, E. M., Topcu, U., and Murray, R. M. (2012). Robust control of uncertain markov decision processes with temporal logic specifications. In CDC, pages 3372–3379. IEEE.

[Wu and Koutsoukos, 2008] Wu, D. and Koutsoukos, X. D. (2008). Reachability analysis of uncertain systems using bounded-parameter markov decision processes. Artif. Intell., 172(8-9):945–954.

[Xu and Zhou, 2021] Xu, Z. and Zhou, J. (2021). A global optimization algorithm for solving linearly constrained quadratic fractional problems. Mathematics, 9(22).

[Young et al., 2010] Young, S. J., Gasic, M., Keizer, S., Mairesse, F., Schatzmann, J., Thomson, B., and Yu, K. (2010). The hidden information state model: A practical framework for pomdp-based spoken dialogue management. Comput. Speech Lang., 24(2):150–174.

Appendix A

Proofs of non-convexity

A.1 Non-convexity of quadratic uncertain transition function

Given optimization variable $\vec{x} = \begin{bmatrix} x_{1,1} & \dots & x_{n,n} & y_1 & \dots & y_n \end{bmatrix}$ and observation $o \in Z$ with $\exists s_i \in S : \boldsymbol{O}(s_i, o) = 1$, we proof the following lemma:

Lemma A.1.1:

$$f(\vec{x}) = \sum_{s' \in S} \boldsymbol{O}(s', o) \sum_{s \in S} x_{s,s'} y_s \text{ is non-convex}$$

Proof.
We begin by writing the quadratic function in canonical form:

$$f(\vec{x}) = \tfrac{1}{2} \vec{x}^T Q \vec{x} + \vec{b}^T \vec{x}$$

As the function has no linear part, $\vec{b}$ is a zero vector. Q has the following shape, where $q_i = \boldsymbol{O}(s_i, o)$. We highlight the non-zero parts.

$$Q = \begin{bmatrix}
0 & 0 & \dots & 0 & 0 & 0 & \dots & 0 & \dots & 0 & 0 & \dots & 0 & q_1 & 0 & 0 & \dots & 0 \\
0 & 0 & \dots & 0 & 0 & 0 & \dots & 0 & \dots & 0 & 0 & \dots & 0 & q_2 & 0 & 0 & \dots & 0 \\
\vdots & \vdots & \ddots & \vdots & \vdots & \vdots & \ddots & \vdots & \ddots & \vdots & \vdots & \ddots & \vdots & \vdots & \vdots & \vdots & \ddots & \vdots \\
0 & 0 & \dots & 0 & 0 & 0 & \dots & 0 & \dots & 0 & 0 & \dots & 0 & q_n & 0 & 0 & \dots & 0 \\
0 & 0 & \dots & 0 & 0 & 0 & \dots & 0 & \dots & 0 & 0 & \dots & 0 & 0 & q_1 & 0 & \dots & 0 \\
0 & 0 & \dots & 0 & 0 & 0 & \dots & 0 & \dots & 0 & 0 & \dots & 0 & 0 & q_2 & 0 & \dots & 0 \\
\vdots & \vdots & \ddots & \vdots & \vdots & \vdots & \ddots & \vdots & \ddots & \vdots & \vdots & \ddots & \vdots & \vdots & \vdots & \vdots & \ddots & \vdots \\
0 & 0 & \dots & 0 & 0 & 0 & \dots & 0 & \dots & 0 & 0 & \dots & 0 & 0 & q_n & 0 & \dots & 0 \\
\vdots & \vdots & \ddots & \vdots & \vdots & \vdots & \ddots & \vdots & \ddots & \vdots & \vdots & \ddots & \vdots & \vdots & \vdots & \vdots & \ddots & \vdots \\
0 & 0 & \dots & 0 & 0 & 0 & \dots & 0 & \dots & 0 & 0 & \dots & 0 & 0 & 0 & 0 & \dots & q_1 \\
0 & 0 & \dots & 0 & 0 & 0 & \dots & 0 & \dots & 0 & 0 & \dots & 0 & 0 & 0 & 0 & \dots & q_2 \\
\vdots & \vdots & \ddots & \vdots & \vdots & \vdots & \ddots & \vdots & \ddots & \vdots & \vdots & \ddots & \vdots & \vdots & \vdots & \vdots & \ddots & \vdots \\
0 & 0 & \dots & 0 & 0 & 0 & \dots & 0 & \dots & 0 & 0 & \dots & 0 & 0 & 0 & 0 & \dots & q_n \\
q_1 & q_2 & \dots & q_n & 0 & 0 & \dots & 0 & \dots & 0 & 0 & \dots & 0 & 0 & 0 & 0 & \dots & 0 \\
0 & 0 & \dots & 0 & q_1 & q_2 & \dots & q_n & \dots & 0 & 0 & \dots & 0 & 0 & 0 & 0 & \dots & 0 \\
0 & 0 & \dots & 0 & 0 & 0 & \dots & 0 & \dots & 0 & 0 & \dots & 0 & 0 & 0 & 0 & \dots & 0 \\
\vdots & \vdots & \ddots & \vdots & \vdots & \vdots & \ddots & \vdots & \ddots & \vdots & \vdots & \ddots & \vdots & \vdots & \vdots & \vdots & \ddots & \vdots \\
0 & 0 & \dots & 0 & 0 & 0 & \dots & 0 & \dots & q_1 & q_2 & \dots & q_n & 0 & 0 & 0 & \dots & 0
\end{bmatrix}$$

(n^2 columns; n columns; n^2 rows; n rows)

As explained in Section 3.3.1 of the Prelimiaries, the Hessian of a quadratic function is its quadratic part: $\nabla^2 f(\vec{x}) = Q$.

Let $s_i \in S$ be a state with $\boldsymbol{O}(s_i, o) = 1$. Let $\vec{r} \in \mathbb{R}^{n^2+n}$ be a vector with all zeros except for elements $r_{i \cdot n+i}$ and r_{n^2+i}. Then we get:

$$\vec{r}^T Q \vec{r} = \vec{r}^T \cdot \begin{bmatrix}
0 \\ \vdots \\ 0 \\
q_1 \cdot r_{n^2+i} \\
q_2 \cdot r_{n^2+i} \\
\vdots \\
q_n \cdot r_{n^2+i} \\
0 \\ \vdots \\ 0 \\
q_i \cdot r_{i \cdot n+i} \\
0 \\ \vdots \\ 0
\end{bmatrix}$$

($i \cdot n$; $(n-i-1) \cdot n + i$; $n-i-1$)

$$
\begin{aligned}
&= r_{i\cdot n+i} \cdot q_i \cdot r_{n^2+i} + r_{n^2+i} \cdot q_i \cdot r_{i\cdot n+i} \\
&= 2 \cdot \boldsymbol{O}(s_i, o) \cdot r_{i\cdot n+i} \cdot r_{n^2+i} \\
&= 2 \cdot r_{i\cdot n+i} \cdot r_{n^2+i}
\end{aligned}
$$

Now, if we set $r_{i\cdot n+i} = 1$ and $r_{n^2+i} = -1$, we get:

$$
\begin{aligned}
\vec{r}^T Q \vec{r} &= -2 \\
&< 0
\end{aligned}
$$

And if we set $r_{i\cdot n+i} = 1$ and $r_{n^2+i} = 1$, we get:

$$
\begin{aligned}
\vec{r}^T Q \vec{r} &= 2 \\
&> 0
\end{aligned}
$$

Hence Q is indefinite and therefore $f(\vec{x})$ is non-convex

□

A.2 Non-convexity of quadratic fractional belief uncertainty interval function

Given optimization variable $\vec{x} = \begin{bmatrix} x_{1,1} & \dots & x_{n,n} & y_1 & \dots & y_n \end{bmatrix}$, state $s_j \in S$ with $\exists s \in S : x_{s,s_j} y_s \neq 0$, and observation $o \in Z$ with $\boldsymbol{O}(s_j, o) = 1$ and $\exists s' \in S \setminus \{s_j\} : \boldsymbol{O}(s', o) = 1$, we proof the following lemma. Note that the functions we exclude with the additional conditions always result in the constant values 0 or 1 and are therefore convex and concave.

Lemma A.2.1:

$$
f(\vec{x}) = \frac{\boldsymbol{O}(s_j, o) \sum_{s \in S} x_{s,s_j} y_s}{\sum_{s' \in S} \boldsymbol{O}(s', o) \sum_{s \in S} x_{s,s'} y_s} \text{ is non-convex}
$$

Proof.

We being by looking at the Hessian of f:

$$
\nabla^2 f(\vec{x}) = \begin{bmatrix}
\frac{\partial^2 f}{\partial x_{1,1}^2}(\vec{x}) & \dots & \frac{\partial^2 f}{\partial x_{1,1} \partial x_{n,n}}(\vec{x}) & \frac{\partial^2 f}{\partial x_{1,1} \partial y_1}(\vec{x}) & \dots & \frac{\partial^2 f}{\partial x_{1,1} \partial y_n}(\vec{x}) \\
\vdots & \ddots & \vdots & \vdots & \ddots & \vdots \\
\frac{\partial^2 f}{\partial x_{n,n} \partial x_{1,1}}(\vec{x}) & \dots & \frac{\partial^2 f}{\partial x_{n,n}^2}(\vec{x}) & \frac{\partial^2 f}{\partial x_{n,n} \partial y_1}(\vec{x}) & \dots & \frac{\partial^2 f}{\partial x_{n,n} \partial y_n}(\vec{x}) \\
\frac{\partial^2 f}{\partial y_1}(\vec{x}) & \dots & \frac{\partial^2 f}{\partial y_1 \partial x_{n,n}}(\vec{x}) & \frac{\partial^2 f}{\partial y_1^2}(\vec{x}) & \dots & \frac{\partial^2 f}{\partial y_1 \partial y_n}(\vec{x}) \\
\vdots & \ddots & \vdots & \vdots & \ddots & \vdots \\
\frac{\partial^2 f}{\partial y_n \partial x_{1,1}}(\vec{x}) & \dots & \frac{\partial^2 f}{\partial y_n \partial x_{n,n}}(\vec{x}) & \frac{\partial^2 f}{\partial y_n \partial y_1}(\vec{x}) & \dots & \frac{\partial^2 f}{\partial y_n^2}(\vec{x})
\end{bmatrix}
$$

This square matrix is in general not symmetric. As mentioned in Section 3.3, we can rewrite any non-symmetric square matrix to its symmetric part without changing the outcome. However, as we only need the values on the diagonal for this proof, we shall omit this step.

We get two different types of values on the first n^2 elements of the diagonal depending on whether the second parameter of the x variables is state s_j or not. We get for all states $s_i \in S$:

$$\begin{aligned}
\frac{\partial f}{\partial x_{s_i,s_j}} &= \frac{(\boldsymbol{O}(s_j,o)\sum_{s\in S} x_{s,s_j}y_s)'(\sum_{s'\in S}\boldsymbol{O}(s',o)\sum_{s\in S}x_{s,s'}y_s)}{(\sum_{s'\in S}\boldsymbol{O}(s',o)\sum_{s\in S}x_{s,s'}y_s)^2} \\
&\quad - \frac{(\boldsymbol{O}(s_j,o)\sum_{s\in S} x_{s,s_j}y_s)(\sum_{s'\in S}\boldsymbol{O}(s',o)\sum_{s\in S}x_{s,s'}y_s)'}{(\sum_{s'\in S}\boldsymbol{O}(s',o)\sum_{s\in S}x_{s,s'}y_s)^2} \\
&= \frac{(\boldsymbol{O}(s_j,o)y_{s_i})(\sum_{s'\in S}\boldsymbol{O}(s',o)\sum_{s\in S}x_{s,s'}y_s)}{(\sum_{s'\in S}\boldsymbol{O}(s',o)\sum_{s\in S}x_{s,s'}y_s)^2} - \frac{(\boldsymbol{O}(s_j,o)\sum_{s\in S} x_{s,s_j}y_s)(\boldsymbol{O}(s_j,o)y_{s_i})}{(\sum_{s'\in S}\boldsymbol{O}(s',o)\sum_{s\in S}x_{s,s'}y_s)^2} \\
&= \frac{\boldsymbol{O}(s_j,o)y_{s_i}}{\sum_{s'\in S}\boldsymbol{O}(s',o)\sum_{s\in S}x_{s,s'}y_s} - \frac{(\boldsymbol{O}(s_j,o)\sum_{s\in S} x_{s,s_j}y_s)(\boldsymbol{O}(s_j,o)y_{s_i})}{(\sum_{s'\in S}\boldsymbol{O}(s',o)\sum_{s\in S}x_{s,s'}y_s)^2} \\
&= \frac{\boldsymbol{O}(s_j,o)y_{s_i}}{\sum_{s'\in S}\boldsymbol{O}(s',o)\sum_{s\in S}x_{s,s'}y_s} - \frac{\boldsymbol{O}(s_j,o)^2 y_{s_i}\sum_{s\in S} x_{s,s_j}y_s}{(\sum_{s'\in S}\boldsymbol{O}(s',o)\sum_{s\in S}x_{s,s'}y_s)^2}
\end{aligned}$$

$$\begin{aligned}
\frac{\partial^2 f}{\partial x^2_{s_i,s_j}} &= \frac{(\boldsymbol{O}(s_j,o)y_{s_i})'(\sum_{s'\in S}\boldsymbol{O}(s',o)\sum_{s\in S}x_{s,s'}y_s)}{(\sum_{s'\in S}\boldsymbol{O}(s',o)\sum_{s\in S}x_{s,s'}y_s)^2} - \frac{(\boldsymbol{O}(s_j,o)y_{s_i})(\sum_{s'\in S}\boldsymbol{O}(s',o)\sum_{s\in S}x_{s,s'}y_s)'}{(\sum_{s'\in S}\boldsymbol{O}(s',o)\sum_{s\in S}x_{s,s'}y_s)^2} \\
&\quad - \frac{(\boldsymbol{O}(s_j,o)^2 y_{s_i}\sum_{s\in S} x_{s,s_j}y_s)'(\sum_{s'\in S}\boldsymbol{O}(s',o)\sum_{s\in S}x_{s,s'}y_s)^2}{(\sum_{s'\in S}\boldsymbol{O}(s',o)\sum_{s\in S}x_{s,s'}y_s)^4} \\
&\quad + \frac{(\boldsymbol{O}(s_j,o)^2 y_{s_i}\sum_{s\in S} x_{s,s_j}y_s)((\sum_{s'\in S}\boldsymbol{O}(s',o)\sum_{s\in S}x_{s,s'}y_s)^2)'}{(\sum_{s'\in S}\boldsymbol{O}(s',o)\sum_{s\in S}x_{s,s'}y_s)^4} \\
&= \frac{0\cdot(\sum_{s'\in S}\boldsymbol{O}(s',o)\sum_{s\in S}x_{s,s'}y_s)}{(\sum_{s'\in S}\boldsymbol{O}(s',o)\sum_{s\in S}x_{s,s'}y_s)^2} - \frac{(\boldsymbol{O}(s_j,o)y_{s_i})(\boldsymbol{O}(s_j,o)y_{s_i})}{(\sum_{s'\in S}\boldsymbol{O}(s',o)\sum_{s\in S}x_{s,s'}y_s)^2} \\
&\quad - \frac{(\boldsymbol{O}(s_j,o)^2 y_{s_i}y_{s_i})}{(\sum_{s'\in S}\boldsymbol{O}(s',o)\sum_{s\in S}x_{s,s'}y_s)^2} \\
&\quad + \frac{(\boldsymbol{O}(s_j,o)^2 y_{s_i}\sum_{s\in S} x_{s,s_j}y_s)\cdot 2\cdot(\sum_{s'\in S}\boldsymbol{O}(s',o)\sum_{s\in S}x_{s,s'}y_s)\cdot(\boldsymbol{O}(s_j,o)y_{s_i})}{(\sum_{s'\in S}\boldsymbol{O}(s',o)\sum_{s\in S}x_{s,s'}y_s)^4} \\
&= \frac{-2\cdot\boldsymbol{O}(s_j,o)^2 y^2_{s_i}}{(\sum_{s'\in S}\boldsymbol{O}(s',o)\sum_{s\in S}x_{s,s'}y_s)^2} + \frac{2\cdot\boldsymbol{O}(s_j,o)^3 y^2_{s_i}\sum_{s\in S}x_{s,s_j}y_s}{(\sum_{s'\in S}\boldsymbol{O}(s',o)\sum_{s\in S}x_{s,s'}y_s)^3} \\
&= \frac{-2\cdot\boldsymbol{O}(s_j,o)^2 y^2_{s_i}\cdot(\sum_{s'\in S}\boldsymbol{O}(s',o)\sum_{s\in S}x_{s,s'}y_s) + 2\cdot\boldsymbol{O}(s_j,o)^3 y^2_{s_i}\sum_{s\in S}x_{s,s_j}y_s}{(\sum_{s'\in S}\boldsymbol{O}(s',o)\sum_{s\in S}x_{s,s'}y_s)^3} \\
&= \frac{2\cdot\boldsymbol{O}(s_j,o)^2 y^2_{s_i}\cdot(\boldsymbol{O}(s_j,o)\sum_{s\in S}x_{s,s_j}y_s - \sum_{s'\in S}\boldsymbol{O}(s',o)\sum_{s\in S}x_{s,s'}y_s)}{(\sum_{s'\in S}\boldsymbol{O}(s',o)\sum_{s\in S}x_{s,s'}y_s)^3} \\
&= \frac{-2\cdot\boldsymbol{O}(s_j,o)^2 y^2_{s_i}\cdot\sum_{s'\in S\setminus\{s_j\}}\boldsymbol{O}(s',o)\sum_{s\in S}x_{s,s'}y_s}{(\sum_{s'\in S}\boldsymbol{O}(s',o)\sum_{s\in S}x_{s,s'}y_s)^3}
\end{aligned}$$

Since all variables in $\vec{x}$ can only take probability values and uncertain observation function $\boldsymbol{O}$ returns either 0 or 1, we now know:

$$\frac{\partial^2 f}{\partial x^2_{s_i,s_j}} \leq 0$$

Furthermore, if s_i is a state for which $x_{s_i,s_j} y_{s_i} \neq 0$, we know:

$$\frac{\partial^2 f}{\partial x^2_{s_i,s_j}} < 0$$

Next we look at the diagonal elements with a different second parameter. We get for all states $s_i \in S$ and $s_k \in S \setminus \{s_j\}$:

$$\begin{aligned}
\frac{\partial f}{\partial x_{s_i,s_k}} &= \frac{(\boldsymbol{O}(s_j,o)\sum_{s\in S} x_{s,s_j} y_s)'(\sum_{s'\in S}\boldsymbol{O}(s',o)\sum_{s\in S} x_{s,s'} y_s)}{(\sum_{s'\in S}\boldsymbol{O}(s',o)\sum_{s\in S} x_{s,s'} y_s)^2} \\
&\quad - \frac{(\boldsymbol{O}(s_j,o)\sum_{s\in S} x_{s,s_j} y_s)(\sum_{s'\in S}\boldsymbol{O}(s',o)\sum_{s\in S} x_{s,s'} y_s)'}{(\sum_{s'\in S}\boldsymbol{O}(s',o)\sum_{s\in S} x_{s,s'} y_s)^2} \\
&= \frac{0\cdot(\sum_{s'\in S}\boldsymbol{O}(s',o)\sum_{s\in S} x_{s,s'} y_s)}{(\sum_{s'\in S}\boldsymbol{O}(s',o)\sum_{s\in S} x_{s,s'} y_s)^2} - \frac{(\boldsymbol{O}(s_j,o)\sum_{s\in S} y_s)(\boldsymbol{O}(s_k,o) x_{s_i,s_k} y_{s_i})}{(\sum_{s'\in S}\boldsymbol{O}(s',o)\sum_{s\in S} x_{s,s'} y_s)^2} \\
&= \frac{-(\boldsymbol{O}(s_j,o)\sum_{s\in S} x_{s,s_j} y_s)(\boldsymbol{O}(s_k,o) y_{s_i})}{(\sum_{s'\in S}\boldsymbol{O}(s',o)\sum_{s\in S} x_{s,s'} y_s)^2}
\end{aligned}$$

$$\begin{aligned}
\frac{\partial^2 f}{\partial x^2_{s_i,s_k}} &= \frac{(-(\boldsymbol{O}(s_j,o)\sum_{s\in S} x_{s,s_j} y_s)(\boldsymbol{O}(s_k,o) y_{s_i}))'(\sum_{s'\in S}\boldsymbol{O}(s',o)\sum_{s\in S} x_{s,s'} y_s)^2}{(\sum_{s'\in S}\boldsymbol{O}(s',o)\sum_{s\in S} x_{s,s'} y_s)^4} \\
&\quad + \frac{(\boldsymbol{O}(s_j,o)\sum_{s\in S} x_{s,s_j} y_s)(\boldsymbol{O}(s_k,o) y_{s_i})((\sum_{s'\in S}\boldsymbol{O}(s',o)\sum_{s\in S} x_{s,s'} y_s)^2)'}{(\sum_{s'\in S}\boldsymbol{O}(s',o)\sum_{s\in S} x_{s,s'} y_s)^4} \\
&= \frac{0}{(\sum_{s'\in S}\boldsymbol{O}(s',o)\sum_{s\in S} x_{s,s'} y_s)^2} \\
&\quad + \frac{(\boldsymbol{O}(s_j,o)\sum_{s\in S} x_{s,s_j} y_s)(\boldsymbol{O}(s_k,o) y_{s_i})\cdot 2\cdot(\sum_{s'\in S}\boldsymbol{O}(s',o)\sum_{s\in S} x_{s,s'} y_s)\cdot(\boldsymbol{O}(s_k,o) y_{s_i})}{(\sum_{s'\in S}\boldsymbol{O}(s',o)\sum_{s\in S} x_{s,s'} y_s)^4} \\
&= \frac{2\cdot\boldsymbol{O}(s_k,o)^2 y^2_{s_i}\cdot\boldsymbol{O}(s_j,o)\sum_{s\in S} x_{s,s_j} y_s}{(\sum_{s'\in S}\boldsymbol{O}(s',o)\sum_{s\in S} x_{s,s'} y_s)^3}
\end{aligned}$$

Since all variables in $\vec{x}$ can only take probability values and uncertain observation function $\boldsymbol{O}$ returns either 0 or 1, we now know:

$$\frac{\partial^2 f}{\partial x^2_{s_i,s_k}} \geq 0$$

Furthermore, if s_i is a state for which $x_{s_i,s_j} y_{s_i} \neq 0$ and s_k is a state for which $\boldsymbol{O}(s_k,o) = 1$, we know:

$$\frac{\partial^2 f}{\partial x^2_{s_i,s_k}} > 0$$

Let $\vec{r} \in \mathbb{R}^{n^2+n}$ be a vector with all zeros except for element $r_{i \cdot n+j} = 1$, and let $\vec{t} \in \mathbb{R}^{n^2+n}$ be a vector with all zeros except for element $t_{i \cdot n+k} = 1$. Then we get:

$$\vec{r}^T(\nabla^2 f(\vec{x}))\vec{r} < 0$$
$$\vec{t}^T(\nabla^2 f(\vec{x}))\vec{t} > 0$$

Hence $\nabla^2 f(\vec{x})$ is indefinite and therefore $f(\vec{x})$ is non-convex

□

A.3 Non-convexity of quadratic fractional belief contribution function

Given optimization variable $\vec{x} = \begin{bmatrix} x_{1,1} & \dots & x_{n,n} & y_1 & \dots & y_n \end{bmatrix}$, states $s_i, s_j \in S$ with $x_{s_i,s_j} y_{s_i} \neq 0$ and $\exists s \in S \setminus \{s_i\} : x_{s,s_j} y_s \neq 0$, and observation $o \in Z$ with $\boldsymbol{O}(s_j, o) = 1$, we proof the following lemma. Note that the functions we exclude with the additional conditions always result in the constant values 0 or 1 and are therefore convex and concave.

Lemma A.3.1:

$$f(\vec{x}) = \frac{\boldsymbol{O}(s_j, o) x_{s_i,s_j} y_{s_i}}{\boldsymbol{O}(s_j, o) \sum_{s \in S} x_{s,s_j} y_s} \text{ is non-convex}$$

Proof.
We being by looking at the Hessian of f:

$$\nabla^2 f(\vec{x}) = \begin{bmatrix} \frac{\partial^2 f}{\partial x_{1,1}^2}(\vec{x}) & \dots & \frac{\partial^2 f}{\partial x_{1,1} \partial x_{n,n}}(\vec{x}) & \frac{\partial^2 f}{\partial x_{1,1} \partial y_1}(\vec{x}) & \dots & \frac{\partial^2 f}{\partial x_{1,1} \partial y_n}(\vec{x}) \\ \vdots & \ddots & \vdots & \vdots & \ddots & \vdots \\ \frac{\partial^2 f}{\partial x_{n,n} \partial x_{1,1}}(\vec{x}) & \dots & \frac{\partial^2 f}{\partial x_{n,n}^2}(\vec{x}) & \frac{\partial^2 f}{\partial x_{n,n} \partial y_1}(\vec{x}) & \dots & \frac{\partial^2 f}{\partial x_{n,n} \partial y_n}(\vec{x}) \\ \frac{\partial^2 f}{\partial y_1}(\vec{x}) & \dots & \frac{\partial^2 f}{\partial y_1 \partial x_{n,n}}(\vec{x}) & \frac{\partial^2 f}{\partial y_1^2}(\vec{x}) & \dots & \frac{\partial^2 f}{\partial y_1 \partial y_n}(\vec{x}) \\ \vdots & \ddots & \vdots & \vdots & \ddots & \vdots \\ \frac{\partial^2 f}{\partial y_n \partial x_{1,1}}(\vec{x}) & \dots & \frac{\partial^2 f}{\partial y_n \partial x_{n,n}}(\vec{x}) & \frac{\partial^2 f}{\partial y_n \partial y_1}(\vec{x}) & \dots & \frac{\partial^2 f}{\partial y_n^2}(\vec{x}) \end{bmatrix}$$

This square matrix is in general not symmetric. As mentioned in Section 3.3, we can rewrite any non-symmetric square matrix to its symmetric part without changing the outcome. However, as we only need the values on the diagonal for this proof, we shall omit this step.

We get two different types of values on the first n^2 elements of the diagonal depending on whether the first parameter of the x variables is states s_i or not:

$$\begin{aligned}\frac{\partial f}{\partial x_{s_i,s_j}} &= \frac{(\boldsymbol{O}(s_j,o)x_{s_i,s_j}y_{s_i})'(\boldsymbol{O}(s_j,o)\sum_{s\in S}x_{s,s_j}y_s) - (\boldsymbol{O}(s_j,o)x_{s_i,s_j}y_{s_i})(\boldsymbol{O}(s_j,o)\sum_{s\in S}x_{s,s_j}y_s)'}{(\boldsymbol{O}(s_j,o)\sum_{s\in S}x_{s,s_j}y_s)^2} \\ &= \frac{(\boldsymbol{O}(s_j,o)y_{s_i})(\boldsymbol{O}(s_j,o)\sum_{s\in S}x_{s,s_j}y_s) - (\boldsymbol{O}(s_j,o)x_{s_i,s_j}y_{s_i})(\boldsymbol{O}(s_j,o)y_{s_i})}{(\boldsymbol{O}(s_j,o)\sum_{s\in S}x_{s,s_j}y_s)^2} \\ &= \frac{\boldsymbol{O}(s_j,o)^2 y_{s_i}\cdot(\sum_{s\in S\setminus\{s_i\}}x_{s,s_j}y_s)}{(\boldsymbol{O}(s_j,o)\sum_{s\in S}x_{s,s_j}y_s)^2}\end{aligned}$$

$$\begin{aligned}\frac{\partial^2 f}{\partial x^2_{s_i,s_j}} &= \frac{(\boldsymbol{O}(s_j,o)^2 y_{s_i}\cdot(\sum_{s\in S\setminus\{s_i\}}x_{s,s_j}y_s))'(\boldsymbol{O}(s_j,o)\sum_{s\in S}x_{s,s_j}y_s)^2}{(\boldsymbol{O}(s_j,o)\sum_{s\in S}x_{s,s_j}y_s)^4} \\ &\quad - \frac{(\boldsymbol{O}(s_j,o)^2 y_{s_i}\cdot(\sum_{s\in S\setminus\{s_i\}}x_{s,s_j}y_s))((\boldsymbol{O}(s_j,o)\sum_{s\in S}x_{s,s_j}y_s)^2)'}{(\boldsymbol{O}(s_j,o)\sum_{s\in S}x_{s,s_j}y_s)^4} \\ &= \frac{0}{(\boldsymbol{O}(s_j,o)\sum_{s\in S}x_{s,s_j}y_s)^2} \\ &\quad - \frac{(\boldsymbol{O}(s_j,o)^2 y_{s_i}\cdot(\sum_{s\in S\setminus\{s_i\}}x_{s,s_j}y_s))\cdot 2\cdot(\boldsymbol{O}(s_j,o)\sum_{s\in S}x_{s,s_j}y_s)\cdot(\boldsymbol{O}(s_j,o)y_{s_i})}{(\boldsymbol{O}(s_j,o)\sum_{s\in S}x_{s,s_j}y_s)^4} \\ &= \frac{-2\cdot\boldsymbol{O}(s_j,o)^3 y_{s_i}^2\cdot(\sum_{s\in S\setminus\{s_i\}}x_{s,s_j}y_s))}{(\boldsymbol{O}(s_j,o)\sum_{s\in S}x_{s,s_j}y_s)^3}\end{aligned}$$

Since all variables in $\vec{x}$ can only take probability values and $\boldsymbol{O}(s_j,o)=1$, we now know:

$$\frac{\partial^2 f}{\partial x^2_{s_i,s_j}} \leq 0$$

Furthermore, since we assumed $\exists s\in S\setminus\{s_i\} : x_{s,s_j}y_s \neq 0$, we know:

$$\frac{\partial^2 f}{\partial x^2_{s_i,s_j}} < 0$$

Next we look at the diagonal elements with a different first parameter. We get for all states $s_k \in S\setminus\{s_i\}$:

$$\begin{aligned}\frac{\partial f}{\partial x_{s_k,s_j}} &= \frac{(\boldsymbol{O}(s_j,o)x_{s_i,s_j}y_{s_i})'(\boldsymbol{O}(s_j,o)\sum_{s\in S}x_{s,s_j}y_s) - (\boldsymbol{O}(s_j,o)x_{s_i,s_j}y_{s_i})(\boldsymbol{O}(s_j,o)\sum_{s\in S}x_{s,s_j}y_s)'}{(\boldsymbol{O}(s_j,o)\sum_{s\in S}x_{s,s_j}y_s)^2} \\ &= \frac{0\cdot(\boldsymbol{O}(s_j,o)\sum_{s\in S}x_{s,s_j}y_s) - (\boldsymbol{O}(s_j,o)x_{s_i,s_j}y_{s_i})(\boldsymbol{O}(s_j,o)y_{s_k})}{(\boldsymbol{O}(s_j,o)\sum_{s\in S}x_{s,s_j}y_s)^2} \\ &= \frac{-(\boldsymbol{O}(s_j,o)x_{s_i,s_j}y_{s_i})(\boldsymbol{O}(s_j,o)y_{s_k})}{(\boldsymbol{O}(s_j,o)\sum_{s\in S}x_{s,s_j}y_s)^2}\end{aligned}$$

$$
\begin{aligned}
\frac{\partial^2 f}{\partial x^2_{s_k,s_j}} &= \frac{(-(\boldsymbol{O}(s_j,o)x_{s_i,s_j}y_{s_i})(\boldsymbol{O}(s_j,o)y_{s_k}))'(\boldsymbol{O}(s_j,o)\sum_{s\in S}x_{s,s_j}y_s)^2}{(\boldsymbol{O}(s_j,o)\sum_{s\in S}x_{s,s_j}y_s)^4} \\
&\quad + \frac{(\boldsymbol{O}(s_j,o)x_{s_i,s_j}y_{s_i})(\boldsymbol{O}(s_j,o)y_{s_k})((\boldsymbol{O}(s_j,o)\sum_{s\in S}x_{s,s_j}y_s)^2)'}{(\boldsymbol{O}(s_j,o)\sum_{s\in S}x_{s,s_j}y_s)^4} \\
&= \frac{0}{(\boldsymbol{O}(s_j,o)\sum_{s\in S}x_{s,s_j}y_s)^2} \\
&\quad + \frac{(\boldsymbol{O}(s_j,o)x_{s_i,s_j}y_{s_i})(\boldsymbol{O}(s_j,o)y_{s_k})\cdot 2\cdot(\boldsymbol{O}(s_j,o)\sum_{s\in S}x_{s,s_j}y_s)\cdot(\boldsymbol{O}(s_j,o)y_{s_k})}{(\boldsymbol{O}(s_j,o)\sum_{s\in S}x_{s,s_j}y_s)^4} \\
&= \frac{2\cdot\boldsymbol{O}(s_j,o)x_{s_i,s_j}y_{s_i}\boldsymbol{O}(s_j,o)^2y^2_{s_k}}{(\boldsymbol{O}(s_j,o)\sum_{s\in S}x_{s,s_j}y_s)^3}
\end{aligned}
$$

Since all variables in $\vec{x}$ can only take probability values and $\boldsymbol{O}(s_j,o)=1$, we now know:

$$\frac{\partial^2 f}{\partial x^2_{s_k,s_j}} \geq 0$$

Furthermore, if s_k is a state for which $x_{s_k,s_j}y_{s_k}\neq 0$, we know:

$$\frac{\partial^2 f}{\partial x^2_{s_k,s_j}} > 0$$

Let $\vec{r}\in\mathbb{R}^{n^2+n}$ be a vector with all zeros except for element $r_{i\cdot n+j}=1$, and let $\vec{t}\in\mathbb{R}^{n^2+n}$ be a vector with all zeros except for element $t_{k\cdot n+j}=1$. Then we get:

$$\vec{r}^T(\nabla^2 f(\vec{x}))\vec{r} < 0$$
$$\vec{t}^T(\nabla^2 f(\vec{x}))\vec{t} > 0$$

Hence $\nabla^2 f(\vec{x})$ is indefinite and therefore $f(\vec{x})$ is non-convex

□

A.4 Non-convexity of quadratic fractional belief normalization function

Given optimization variable $\vec{x}=\begin{bmatrix}x_{1,1} & \dots & x_{n,n} & y_1 & \dots & y_n\end{bmatrix}$, state $s_i\in S$, and observation $o\in Z$ with $\exists s'\in S:\boldsymbol{O}(s',o)x_{s_i,s'}y_{s_i}\neq 0$ and $\exists s\in S\setminus\{s_i\},\exists s'\in S:\boldsymbol{O}(s',o)x_{s,s'}y_s\neq 0$, we proof the following lemma. Note that the functions we exclude with the additional conditions always result in the constant values 0 or 1 and are therefore convex and concave.

Lemma A.4.1:

$$f(\vec{x})=\frac{\sum_{s'\in S}\boldsymbol{O}(s',o)x_{s_i,s'}y_{s_i}}{\sum_{s'\in S}\boldsymbol{O}(s',o)\sum_{s\in S}x_{s,s'}y_s} \text{ is non-convex}$$

Proof.
We being by looking at the Hessian of f:

$$\nabla^2 f(\vec{x}) = \begin{bmatrix} \frac{\partial^2 f}{\partial x_{1,1}^2}(\vec{x}) & \dots & \frac{\partial^2 f}{\partial x_{1,1}\partial x_{n,n}}(\vec{x}) & \frac{\partial^2 f}{\partial x_{1,1}\partial y_1}(\vec{x}) & \dots & \frac{\partial^2 f}{\partial x_{1,1}\partial y_n}(\vec{x}) \\ \vdots & \ddots & \vdots & \vdots & \ddots & \vdots \\ \frac{\partial^2 f}{\partial x_{n,n}\partial x_{1,1}}(\vec{x}) & \dots & \frac{\partial^2 f}{\partial x_{n,n}^2}(\vec{x}) & \frac{\partial^2 f}{\partial x_{n,n}\partial y_1}(\vec{x}) & \dots & \frac{\partial^2 f}{\partial x_{n,n}\partial y_n}(\vec{x}) \\ \frac{\partial^2 f}{\partial y_1}(\vec{x}) & \dots & \frac{\partial^2 f}{\partial y_1\partial x_{n,n}}(\vec{x}) & \frac{\partial^2 f}{\partial y_1^2}(\vec{x}) & \dots & \frac{\partial^2 f}{\partial y_1\partial y_n}(\vec{x}) \\ \vdots & \ddots & \vdots & \vdots & \ddots & \vdots \\ \frac{\partial^2 f}{\partial y_n\partial x_{1,1}}(\vec{x}) & \dots & \frac{\partial^2 f}{\partial y_n\partial x_{n,n}}(\vec{x}) & \frac{\partial^2 f}{\partial y_n\partial y_1}(\vec{x}) & \dots & \frac{\partial^2 f}{\partial y_n^2}(\vec{x}) \end{bmatrix}$$

This square matrix is in general not symmetric. As mentioned in Section 3.3, we can rewrite any non-symmetric square matrix to its symmetric part without changing the outcome. However, as we only need the values on the diagonal for this proof, we shall omit this step.

We get two different types of values on the first n^2 elements of the diagonal depending on whether the first parameter of the x variables is state s_i or not. We get for all states $s_j \in S$:

$$\begin{aligned}
\frac{\partial f}{\partial x_{s_i,s_j}} &= \frac{(\sum_{s'\in S} \boldsymbol{O}(s',o)x_{s_i,s'}y_{s_i})'(\sum_{s'\in S} \boldsymbol{O}(s',o)\sum_{s\in S} x_{s,s'}y_s)}{(\sum_{s'\in S} \boldsymbol{O}(s',o)\sum_{s\in S} x_{s,s'}y_s)^2} \\
&\quad - \frac{(\sum_{s'\in S} \boldsymbol{O}(s',o)x_{s_i,s'}y_{s_i})(\sum_{s'\in S} \boldsymbol{O}(s',o)\sum_{s\in S} x_{s,s'}y_s)'}{(\sum_{s'\in S} \boldsymbol{O}(s',o)\sum_{s\in S} x_{s,s'}y_s)^2} \\
&= \frac{(\boldsymbol{O}(s_j,o)y_{s_i})(\sum_{s'\in S} \boldsymbol{O}(s',o)\sum_{s\in S} x_{s,s'}y_s) - (\sum_{s'\in S} \boldsymbol{O}(s',o)x_{s_i,s'}y_{s_i})\cdot \boldsymbol{O}(s_j,o)y_{s_i}}{(\sum_{s'\in S} \boldsymbol{O}(s',o)\sum_{s\in S} x_{s,s'}y_s)^2} \\
&= \frac{(\boldsymbol{O}(s_j,o)y_{s_i})(\sum_{s'\in S} \boldsymbol{O}(s',o)\sum_{s\in S\setminus\{s_i\}} x_{s,s'}y_s)}{(\sum_{s'\in S} \boldsymbol{O}(s',o)\sum_{s\in S} x_{s,s'}y_s)^2}
\end{aligned}$$

$$\begin{aligned}
\frac{\partial^2 f}{\partial x_{s_i,s_j}^2} &= \frac{((\boldsymbol{O}(s_j,o)y_{s_i})(\sum_{s'\in S} \boldsymbol{O}(s',o)\sum_{s\in S\setminus\{s_i\}} x_{s,s'}y_s))'(\sum_{s'\in S} \boldsymbol{O}(s',o)\sum_{s\in S} x_{s,s'}y_s)^2}{(\sum_{s'\in S} \boldsymbol{O}(s',o)\sum_{s\in S} x_{s,s'}y_s)^4} \\
&\quad - \frac{(\boldsymbol{O}(s_j,o)y_{s_i})(\sum_{s'\in S} \boldsymbol{O}(s',o)\sum_{s\in S\setminus\{s_i\}} x_{s,s'}y_s)((\sum_{s'\in S} \boldsymbol{O}(s',o)\sum_{s\in S} x_{s,s'}y_s)^2)'}{(\sum_{s'\in S} \boldsymbol{O}(s',o)\sum_{s\in S} x_{s,s'}y_s)^4} \\
&= \frac{0}{(\sum_{s'\in S} \boldsymbol{O}(s',o)\sum_{s\in S} x_{s,s'}y_s)^2} \\
&\quad - \frac{(\boldsymbol{O}(s_j,o)y_{s_i})(\sum_{s'\in S} \boldsymbol{O}(s',o)\sum_{s\in S\setminus\{s_i\}} x_{s,s'}y_s)\cdot 2\cdot(\sum_{s'\in S} \boldsymbol{O}(s',o)\sum_{s\in S} x_{s,s'}y_s)(\boldsymbol{O}(s_j,o)y_{s_i})}{(\sum_{s'\in S} \boldsymbol{O}(s',o)\sum_{s\in S} x_{s,s'}y_s)^4} \\
&= \frac{-2\cdot \boldsymbol{O}(s_j,o)^2 y_{s_i}^2\cdot(\sum_{s'\in S} \boldsymbol{O}(s',o)\sum_{s\in S\setminus\{s_i\}} x_{s,s'}y_s)}{(\sum_{s'\in S} \boldsymbol{O}(s',o)\sum_{s\in S} x_{s,s'}y_s)^3}
\end{aligned}$$

Since all variables in $\vec{x}$ can only take probability values and uncertain observation function $\boldsymbol{O}$ returns either 0 or 1, we now know:

$$\frac{\partial^2 f}{\partial x_{s_i,s_j}^2} \leq 0$$

Furthermore, if s_j is a state for which $\boldsymbol{O}(s_j, o)x_{s_i,s_j}y_{s_i} \neq 0$, so $\boldsymbol{O}(s_j, o) = 1$, and since we assumed $\exists s \in S \setminus \{s_i\}, \exists s' \in S : \boldsymbol{O}(s', o)x_{s,s'}y_s \neq 0$, we know:

$$\frac{\partial^2 f}{\partial x_{s_i,s_j}^2} < 0$$

Next we look at the diagonal elements with a different first parameter. We get for all states $s_k \in S \setminus \{s_i\}$ and $s_j \in S$:

$$\begin{aligned}
\frac{\partial f}{\partial x_{s_k,s_j}} &= \frac{(\sum_{s' \in S} \boldsymbol{O}(s', o)x_{s_i,s'}y_{s_i})'(\sum_{s' \in S} \boldsymbol{O}(s', o) \sum_{s \in S} x_{s,s'}y_s)}{(\sum_{s' \in S} \boldsymbol{O}(s', o) \sum_{s \in S} x_{s,s'}y_s)^2} \\
&\quad - \frac{(\sum_{s' \in S} \boldsymbol{O}(s', o)x_{s_i,s'}y_{s_i})(\sum_{s' \in S} \boldsymbol{O}(s', o) \sum_{s \in S} x_{s,s'}y_s)'}{(\sum_{s' \in S} \boldsymbol{O}(s', o) \sum_{s \in S} x_{s,s'}y_s)^2} \\
&= \frac{0 \cdot (\sum_{s' \in S} \boldsymbol{O}(s', o) \sum_{s \in S} x_{s,s'}y_s) - (\sum_{s' \in S} \boldsymbol{O}(s', o)x_{s_i,s'}y_{s_i})(\boldsymbol{O}(s_j, o)y_{s_k})}{(\sum_{s' \in S} \boldsymbol{O}(s', o) \sum_{s \in S} x_{s,s'}y_s)^2} \\
&= \frac{-(\sum_{s' \in S} \boldsymbol{O}(s', o)x_{s_i,s'}y_{s_i})(\boldsymbol{O}(s_j, o)y_{s_k})}{(\sum_{s' \in S} \boldsymbol{O}(s', o) \sum_{s \in S} x_{s,s'}y_s)^2}
\end{aligned}$$

$$\begin{aligned}
\frac{\partial^2 f}{\partial x_{s_k,s_j}^2} &= \frac{(-(\sum_{s' \in S} \boldsymbol{O}(s', o)x_{s_i,s'}y_{s_i})(\boldsymbol{O}(s_j, o)y_{s_k}))'(\sum_{s' \in S} \boldsymbol{O}(s', o) \sum_{s \in S} x_{s,s'}y_s)^2}{(\sum_{s' \in S} \boldsymbol{O}(s', o) \sum_{s \in S} x_{s,s'}y_s)^4} \\
&\quad + \frac{(\sum_{s' \in S} \boldsymbol{O}(s', o)x_{s_i,s'}y_{s_i})(\boldsymbol{O}(s_j, o)y_{s_k})((\sum_{s' \in S} \boldsymbol{O}(s', o) \sum_{s \in S} x_{s,s'}y_s)^2)}{(\sum_{s' \in S} \boldsymbol{O}(s', o) \sum_{s \in S} x_{s,s'}y_s)^4} \\
&= \frac{0}{(\sum_{s' \in S} \boldsymbol{O}(s', o) \sum_{s \in S} x_{s,s'}y_s)^2} \\
&\quad + \frac{(\sum_{s' \in S} \boldsymbol{O}(s', o)x_{s_i,s'}y_{s_i})(\boldsymbol{O}(s_j, o)y_{s_k}) \cdot 2 \cdot (\sum_{s' \in S} \boldsymbol{O}(s', o) \sum_{s \in S} x_{s,s'}y_s)(\boldsymbol{O}(s_j, o)y_{s_k})}{(\sum_{s' \in S} \boldsymbol{O}(s', o) \sum_{s \in S} x_{s,s'}y_s)^4} \\
&= \frac{2 \cdot \boldsymbol{O}(s_j, o)^2 y_{s_k}^2 \cdot (\sum_{s' \in S} \boldsymbol{O}(s', o)x_{s_i,s'}y_{s_i})}{(\sum_{s' \in S} \boldsymbol{O}(s', o) \sum_{s \in S} x_{s,s'}y_s)^3}
\end{aligned}$$

Since all variables in $\vec{x}$ can only take probability values and uncertain observation function $\boldsymbol{O}$ returns either 0 or 1, we now know:

$$\frac{\partial^2 f}{\partial x_{s_k,s_j}^2} \geq 0$$

Furthermore, if s_j is a state for which $\boldsymbol{O}(s_j, o)x_{s_i,s_j}y_{s_i} \neq 0$, so $\boldsymbol{O}(s_j, o) = 1$, and s_k is a state for which $\exists s' \in S : \boldsymbol{O}(s', o)x_{s_k,s'}y_{s_k} \neq 0$, so $y_{s_k} \neq 0$, we know:

$$\frac{\partial^2 f}{\partial x^2_{s_k,s_j}} > 0$$

Let $\vec{r} \in \mathbb{R}^{n^2+n}$ be a vector with all zeros except for element $r_{i \cdot n+j} = 1$, and let $\vec{t} \in \mathbb{R}^{n^2+n}$ be a vector with all zeros except for element $t_{k \cdot n+j} = 1$. Then we get:

$$\vec{r}^T(\nabla^2 f(\vec{x}))\vec{r} < 0$$
$$\vec{t}^T(\nabla^2 f(\vec{x}))\vec{t} > 0$$

Hence $\nabla^2 f(\vec{x})$ is indefinite and therefore $f(\vec{x})$ is non-convex

□

A.5 Non-convexity of quadratic belief uncertainty interval sub-functions

A.5.1 Numerator

Given optimization variable $\vec{x} = \begin{bmatrix} x_{1,1} & \dots & x_{n,n} & y_1 & \dots & y_n \end{bmatrix}$, state $s_i \in S$ and observation $o \in Z$ with $\boldsymbol{O}(s_i, o) = 1$, we proof the following lemma:

Lemma A.5.1:

$$f(\vec{x}) = \boldsymbol{O}(s_i, o) \sum_{s \in S} x_{s,s_i} y_s \text{ is non-convex}$$

Proof.
We begin by writing the quadratic function in canonical form:

$$f(\vec{x}) = \tfrac{1}{2}\vec{x}^T Q \vec{x} + \vec{b}^T \vec{x}$$

As the function has no linear part, $\vec{b}$ is a zero vector. Q has the following shape, where $q = \boldsymbol{O}(s_i, o)$. We highlight the non-zero parts.

$$
Q = \begin{bmatrix}
0 & \cdots & 0 & 0 & 0 & \cdots & 0 & \cdots & 0 & \cdots & 0 & 0 & 0 & \cdots & 0 & 0 & 0 & \cdots & 0 \\
\vdots & \ddots & \vdots & \vdots & \vdots & \ddots & \vdots & \ddots & \vdots & \ddots & \vdots & \vdots & \vdots & \ddots & \vdots & \vdots & \vdots & \ddots & \vdots \\
0 & \cdots & 0 & 0 & 0 & \cdots & 0 & \cdots & 0 & \cdots & 0 & 0 & 0 & \cdots & 0 & 0 & 0 & \cdots & 0 \\
0 & \cdots & 0 & 0 & 0 & \cdots & 0 & \cdots & 0 & \cdots & 0 & 0 & 0 & \cdots & 0 & q & 0 & \cdots & 0 \\
0 & \cdots & 0 & 0 & 0 & \cdots & 0 & \cdots & 0 & \cdots & 0 & 0 & 0 & \cdots & 0 & 0 & 0 & \cdots & 0 \\
\vdots & \ddots & \vdots & \vdots & \vdots & \ddots & \vdots & \ddots & \vdots & \ddots & \vdots & \vdots & \vdots & \ddots & \vdots & \vdots & \vdots & \ddots & \vdots \\
0 & \cdots & 0 & 0 & 0 & \cdots & 0 & \cdots & 0 & \cdots & 0 & 0 & 0 & \cdots & 0 & 0 & 0 & \cdots & 0 \\
\vdots & \ddots & \vdots & \vdots & \vdots & \ddots & \vdots & \ddots & \vdots & \ddots & \vdots & \vdots & \vdots & \ddots & \vdots & \vdots & \vdots & \ddots & \vdots \\
0 & \cdots & 0 & 0 & 0 & \cdots & 0 & \cdots & 0 & \cdots & 0 & 0 & 0 & \cdots & 0 & 0 & 0 & \cdots & 0 \\
\vdots & \ddots & \vdots & \vdots & \vdots & \ddots & \vdots & \ddots & \vdots & \ddots & \vdots & \vdots & \vdots & \ddots & \vdots & \vdots & \vdots & \ddots & \vdots \\
0 & \cdots & 0 & 0 & 0 & \cdots & 0 & \cdots & 0 & \cdots & 0 & 0 & 0 & \cdots & 0 & 0 & 0 & \cdots & 0 \\
0 & \cdots & 0 & 0 & 0 & \cdots & 0 & \cdots & 0 & \cdots & 0 & 0 & 0 & \cdots & 0 & 0 & 0 & \cdots & q \\
0 & \cdots & 0 & 0 & 0 & \cdots & 0 & \cdots & 0 & \cdots & 0 & 0 & 0 & \cdots & 0 & 0 & 0 & \cdots & 0 \\
\vdots & \ddots & \vdots & \vdots & \vdots & \ddots & \vdots & \ddots & \vdots & \ddots & \vdots & \vdots & \vdots & \ddots & \vdots & \vdots & \vdots & \ddots & \vdots \\
0 & \cdots & 0 & 0 & 0 & \cdots & 0 & \cdots & 0 & \cdots & 0 & 0 & 0 & \cdots & 0 & 0 & 0 & \cdots & 0 \\
0 & \cdots & 0 & q & 0 & \cdots & 0 & \cdots & 0 & \cdots & 0 & 0 & 0 & \cdots & 0 & 0 & 0 & \cdots & 0 \\
0 & \cdots & 0 & 0 & 0 & \cdots & 0 & \cdots & 0 & \cdots & 0 & 0 & 0 & \cdots & 0 & 0 & 0 & \cdots & 0 \\
\vdots & \ddots & \vdots & \vdots & \vdots & \ddots & \vdots & \ddots & \vdots & \ddots & \vdots & \vdots & \vdots & \ddots & \vdots & \vdots & \vdots & \ddots & \vdots \\
0 & \cdots & 0 & 0 & 0 & \cdots & 0 & \cdots & 0 & \cdots & 0 & q & 0 & \cdots & 0 & 0 & 0 & \cdots & 0
\end{bmatrix}
$$

(Column groups: n^2 columns split as i, $n-i-1$, …, i, $n-i-1$, followed by n columns; row groups: i, $n-i-1$, i, $n-i-1$ within n^2, followed by n.)

As explained in Section 3.3.1 of the Prelimiaries, the Hessian of a quadratic function is its quadratic part: $\nabla^2 f(\vec{x}) = Q$.

Let $\vec{r} \in \mathbb{R}^{n^2+n}$ be a vector with all zeros except for elements $r_{i \cdot n+i}$ and r_{n^2+i}. Then we get:

$$
\vec{r}^T Q \vec{r} = \vec{r}^T \cdot \begin{bmatrix} 0 \\ \vdots \\ 0 \\ q \cdot r_{n^2+i} \\ 0 \\ \vdots \\ 0 \\ q \cdot r_{i \cdot n+i} \\ 0 \\ \vdots \\ 0 \end{bmatrix}
\begin{matrix} \left.\vphantom{\begin{matrix}0\\\vdots\\0\end{matrix}}\right\} i \cdot n + i \\ \\ \left.\vphantom{\begin{matrix}0\\\vdots\\0\end{matrix}}\right\} (n-i-1) + (n-i-1) \cdot n + i \\ \\ \left.\vphantom{\begin{matrix}0\\\vdots\\0\end{matrix}}\right\} n-i-1 \end{matrix}
$$

$$
\begin{aligned}
&= r_{i \cdot n+i} \cdot q \cdot r_{n^2+i} + r_{n^2+i} \cdot q \cdot r_{i \cdot n+i} \\
&= 2 \cdot \boldsymbol{O}(s_i, o) \cdot r_{i \cdot n+i} \cdot r_{n^2+i} \\
&= 2 \cdot r_{i \cdot n+i} \cdot r_{n^2+i}
\end{aligned}
$$

Now, if we set $r_{i \cdot n+i} = 1$ and $r_{n^2+i} = -1$, we get:

$$\begin{aligned} \vec{r}^T Q \vec{r} &= -2 \\ &< 0 \end{aligned}$$

And if we set $r_{i \cdot n+i} = 1$ and $r_{n^2+i} = 1$, we get:

$$\begin{aligned} \vec{r}^T Q \vec{r} &= 2 \\ &> 0 \end{aligned}$$

Hence Q is indefinite and therefore $f(\vec{x})$ is non-convex

□

A.5.2 Denominator

The denominator sub-function of the belief uncertainty interval function is the same as the uncertain transition function. We proof the non-convexity of this function in Appendix A.1.

A.6 Non-convexity of quadratic belief contribution sub-functions

A.6.1 Numerator

Given optimization variable $\vec{x} = \begin{bmatrix} x_{1,1} & \dots & x_{n,n} & y_1 & \dots & y_n \end{bmatrix}$, states $s_i, s_j \in S$ and observation $o \in Z$ with $\boldsymbol{O}(s_j, o) = 1$, we proof the following lemma:

Lemma A.6.1:

$$f(\vec{x}) = \boldsymbol{O}(s_j, o) x_{s_i, s_j} y_{s_i} \text{ is non-convex}$$

Proof.
We begin by writing the quadratic function in canonical form:

$$f(\vec{x}) = \tfrac{1}{2} \vec{x}^T Q \vec{x} + \vec{b}^T \vec{x}$$

As the function has no linear part, $\vec{b}$ is a zero vector. Q has the following shape, where $q = \boldsymbol{O}(s_i, o)$. We highlight the non-zero parts.

$$
Q = \begin{array}{c}
\overbrace{}^{n^2}\ \overbrace{}^{n} \\
\overbrace{0\ \dots\ 0}^{i\cdot n+j}\ 0\ \overbrace{0\ \dots\ 0}^{(n-j-1)+(n-i-1)\cdot n}\ \overbrace{0\ \dots\ 0}^{i}\ 0\ \overbrace{0\ \dots\ 0}^{n-i-1} \\
\begin{bmatrix}
0 & \dots & 0 & 0 & 0 & \dots & 0 & 0 & \dots & 0 & 0 & 0 & \dots & 0 \\
\vdots & \ddots & \vdots & \vdots & \vdots & \ddots & \vdots & \vdots & \ddots & \vdots & \vdots & \vdots & \ddots & \vdots \\
0 & \dots & 0 & 0 & 0 & \dots & 0 & 0 & \dots & 0 & 0 & 0 & \dots & 0 \\
0 & \dots & 0 & 0 & 0 & \dots & 0 & 0 & \dots & 0 & q & 0 & \dots & 0 \\
0 & \dots & 0 & 0 & 0 & \dots & 0 & 0 & \dots & 0 & 0 & 0 & \dots & 0 \\
\vdots & \ddots & \vdots & \vdots & \vdots & \ddots & \vdots & \vdots & \ddots & \vdots & \vdots & \vdots & \ddots & \vdots \\
0 & \dots & 0 & 0 & 0 & \dots & 0 & 0 & \dots & 0 & 0 & 0 & \dots & 0 \\
0 & \dots & 0 & 0 & 0 & \dots & 0 & 0 & \dots & 0 & 0 & 0 & \dots & 0 \\
\vdots & \ddots & \vdots & \vdots & \vdots & \ddots & \vdots & \vdots & \ddots & \vdots & \vdots & \vdots & \ddots & \vdots \\
0 & \dots & 0 & 0 & 0 & \dots & 0 & 0 & \dots & 0 & 0 & 0 & \dots & 0 \\
0 & \dots & 0 & q & 0 & \dots & 0 & 0 & \dots & 0 & 0 & 0 & \dots & 0 \\
0 & \dots & 0 & 0 & 0 & \dots & 0 & 0 & \dots & 0 & 0 & 0 & \dots & 0 \\
\vdots & \ddots & \vdots & \vdots & \vdots & \ddots & \vdots & \vdots & \ddots & \vdots & \vdots & \vdots & \ddots & \vdots \\
0 & \dots & 0 & 0 & 0 & \dots & 0 & 0 & \dots & 0 & 0 & 0 & \dots & 0
\end{bmatrix}
\begin{array}{l}
\left.\phantom{\begin{matrix}0\\\vdots\\0\end{matrix}}\right\} i\cdot n+j \\
\left.\phantom{\begin{matrix}0\\\vdots\\0\end{matrix}}\right\} (n-j-1)+(n-i-1)\cdot n \\
\left.\phantom{\begin{matrix}0\\\vdots\\0\end{matrix}}\right\} i \\
\left.\phantom{\begin{matrix}0\\\vdots\\0\end{matrix}}\right\} n-i-1
\end{array}
\begin{array}{l}
\left.\phantom{\begin{matrix}0\\\vdots\\0\end{matrix}}\right\} n^2 \\
\left.\phantom{\begin{matrix}0\\\vdots\\0\end{matrix}}\right\} n
\end{array}
\end{array}
$$

As explained in Section 3.3.1 of the Prelimiaries, the Hessian of a quadratic function is its quadratic part: $\nabla^2 f(\vec{x}) = Q$.

Let $\vec{r} \in \mathbb{R}^{n^2+n}$ be a vector with all zeros except for elements $r_{i\cdot n+j}$ and r_{n^2+i}. Then we get:

$$
\begin{aligned}
\vec{r}^T Q \vec{r} &= \vec{r}^T \cdot \begin{bmatrix}
0 \\ \vdots \\ 0 \\ q \cdot r_{n^2+i} \\ 0 \\ \vdots \\ 0 \\ q \cdot r_{i\cdot n+j} \\ 0 \\ \vdots \\ 0
\end{bmatrix}
\begin{array}{l}
\left.\phantom{\begin{matrix}0\\\vdots\\0\end{matrix}}\right\} i\cdot n+j \\
\left.\phantom{\begin{matrix}0\\\vdots\\0\end{matrix}}\right\} (n-j-1)+(n-i-1)\cdot n+i \\
\left.\phantom{\begin{matrix}0\\\vdots\\0\end{matrix}}\right\} n-i-1
\end{array} \\
&= r_{i\cdot n+j} \cdot q \cdot r_{n^2+i} + r_{n^2+i} \cdot q \cdot r_{i\cdot n+j} \\
&= 2 \cdot \boldsymbol{O}(s_j, o) \cdot r_{i\cdot n+j} \cdot r_{n^2+i} \\
&= 2 \cdot r_{i\cdot n+j} \cdot r_{n^2+i}
\end{aligned}
$$

Now, if we set $r_{i\cdot n+j} = 1$ and $r_{n^2+i} = -1$, we get:

$$
\begin{aligned}
\vec{r}^T Q \vec{r} &= -2 \\
&< 0
\end{aligned}
$$

And if we set $r_{i \cdot n+j} = 1$ and $r_{n^2+i} = 1$, we get:

$$\begin{aligned} \vec{r}^T Q \vec{r} &= 2 \\ &> 0 \end{aligned}$$

Hence Q is indefinite and therefore $f(\vec{x})$ is non-convex

□

A.6.2 Denominator

The denominator sub-function of the belief contribution function is the same as the numerator sub-function of the belief uncertainty interval function. We proof the non-convexity of this function in Appendix A.5.

A.7 Non-convexity of quadratic belief normalization sub-functions

A.7.1 Numerator

Given optimization variable $\vec{x} = \begin{bmatrix} x_{1,1} & \dots & x_{n,n} & y_1 & \dots & y_n \end{bmatrix}$, state $s_i \in S$ and observation $o \in Z$ with $\exists j \in [n] : \boldsymbol{O}(s_j, o) = 1$, we proof the following lemma:

Lemma A.7.1:

$$f(\vec{x}) = \sum_{s' \in S} \boldsymbol{O}(s', o) x_{s_i, s'} y_{s_i} \text{ is non-convex}$$

Proof.
We begin by writing the quadratic function in canonical form:

$$f(\vec{x}) = \tfrac{1}{2} \vec{x}^T Q \vec{x} + \vec{b}^T \vec{x}$$

As the function has no linear part, $\vec{b}$ is a zero vector. Q has the following shape, where $q_j = \boldsymbol{O}(s_j, o)$. We highlight the non-zero parts.

$$
Q = \begin{array}{c}
\overbrace{}^{n^2}\ \overbrace{}^{n} \\
\overbrace{}^{i\cdot n}\ \overbrace{}^{(n-i-1)\cdot n}\ \overbrace{}^{i}\ \overbrace{}^{n-i-1} \\
\begin{bmatrix}
0 & \dots & 0 & 0 & 0 & \dots & 0 & 0 & \dots & 0 & 0 & \dots & 0 & 0 & 0 & \dots & 0 \\
\vdots & \ddots & \vdots & \vdots & \vdots & \ddots & \vdots & \vdots & \ddots & \vdots & \vdots & \ddots & \vdots & \vdots & \vdots & \ddots & \vdots \\
0 & \dots & 0 & 0 & 0 & \dots & 0 & 0 & \dots & 0 & 0 & \dots & 0 & 0 & 0 & \dots & 0 \\
0 & \dots & 0 & 0 & 0 & \dots & 0 & 0 & \dots & 0 & 0 & \dots & 0 & q_1 & 0 & \dots & 0 \\
0 & \dots & 0 & 0 & 0 & \dots & 0 & 0 & \dots & 0 & 0 & \dots & 0 & q_2 & 0 & \dots & 0 \\
\vdots & \ddots & \vdots & \vdots & \vdots & \ddots & \vdots & \vdots & \ddots & \vdots & \vdots & \ddots & \vdots & \vdots & \vdots & \ddots & \vdots \\
0 & \dots & 0 & 0 & 0 & \dots & 0 & 0 & \dots & 0 & 0 & \dots & 0 & q_n & 0 & \dots & 0 \\
0 & \dots & 0 & 0 & 0 & \dots & 0 & 0 & \dots & 0 & 0 & \dots & 0 & 0 & 0 & \dots & 0 \\
\vdots & \ddots & \vdots & \vdots & \vdots & \ddots & \vdots & \vdots & \ddots & \vdots & \vdots & \ddots & \vdots & \vdots & \vdots & \ddots & \vdots \\
0 & \dots & 0 & 0 & 0 & \dots & 0 & 0 & \dots & 0 & 0 & \dots & 0 & 0 & 0 & \dots & 0 \\
0 & \dots & 0 & 0 & 0 & \dots & 0 & 0 & \dots & 0 & 0 & \dots & 0 & 0 & 0 & \dots & 0 \\
\vdots & \ddots & \vdots & \vdots & \vdots & \ddots & \vdots & \vdots & \ddots & \vdots & \vdots & \ddots & \vdots & \vdots & \vdots & \ddots & \vdots \\
0 & \dots & 0 & q_1 & q_2 & \dots & q_n & 0 & \dots & 0 & 0 & \dots & 0 & 0 & 0 & \dots & 0 \\
0 & \dots & 0 & 0 & 0 & \dots & 0 & 0 & \dots & 0 & 0 & \dots & 0 & 0 & 0 & \dots & 0 \\
\vdots & \ddots & \vdots & \vdots & \vdots & \ddots & \vdots & \vdots & \ddots & \vdots & \vdots & \ddots & \vdots & \vdots & \vdots & \ddots & \vdots \\
0 & \dots & 0 & 0 & 0 & \dots & 0 & 0 & \dots & 0 & 0 & \dots & 0 & 0 & 0 & \dots & 0
\end{bmatrix}
\end{array}
\begin{array}{l}
\left.\begin{matrix}\\\\\\\end{matrix}\right\} i\cdot n \\
\left.\begin{matrix}\\\\\\\\\\\\\\\end{matrix}\right\} n^2 \\
\left.\begin{matrix}\\\\\\\end{matrix}\right\} (n-i-1)\cdot n \\
\left.\begin{matrix}\\\\\\\end{matrix}\right\} i \\
\left.\begin{matrix}\\\\\\\\\\\\\\\end{matrix}\right\} n \\
\left.\begin{matrix}\\\\\\\end{matrix}\right\} n-i-1
\end{array}
$$

As explained in Section 3.3.1 of the Prelimiaries, the Hessian of a quadratic function is its quadratic part: $\nabla^2 f(\vec{x}) = Q$.

Let $s_j \in S$ be a state with $\boldsymbol{O}(s_j, o) = 1$. Let $\vec{r} \in \mathbb{R}^{n^2+n}$ be a vector with all zeros except for elements $r_{i\cdot n+j}$ and r_{n^2+i}. Then we get:

$$
\begin{aligned}
\vec{r}^T Q \vec{r} = \vec{r}^T \cdot & \begin{array}{l}
\left.\begin{bmatrix} 0 \\ \vdots \\ 0 \end{bmatrix}\right\} i\cdot n \\
\left.\begin{bmatrix} q_1 \cdot r_{n^2+i} \\ q_2 \cdot r_{n^2+i} \\ \vdots \\ q_n \cdot r_{n^2+i} \\ 0 \\ \vdots \\ 0 \end{bmatrix}\right\} (n-i-1)\cdot n + i \\
\left.\begin{bmatrix} q_i \cdot r_{i\cdot n+j} \\ 0 \\ \vdots \\ 0 \end{bmatrix}\right\} n-i-1
\end{array} \\
& = r_{i\cdot n+j} \cdot q_j \cdot r_{n^2+i} + r_{n^2+i} \cdot q \cdot r_{i\cdot n+j}
\end{aligned}
$$

$$
\begin{aligned}
&= 2 \cdot \boldsymbol{O}(s_j, o) \cdot r_{i \cdot n+j} \cdot r_{n^2+i} \\
&= 2 \cdot r_{i \cdot n+j} \cdot r_{n^2+i}
\end{aligned}
$$

Now, if we set $r_{i \cdot n+j} = 1$ and $r_{n^2+i} = -1$, we get:

$$
\begin{aligned}
\vec{r}^T Q \vec{r} &= -2 \\
&< 0
\end{aligned}
$$

And if we set $r_{i \cdot n+j} = 1$ and $r_{n^2+i} = 1$, we get:

$$
\begin{aligned}
\vec{r}^T Q \vec{r} &= 2 \\
&> 0
\end{aligned}
$$

Hence Q is indefinite and therefore $f(\vec{x})$ is non-convex

□

A.7.2 Denominator

The denominator sub-function of the belief normalization function is the same as the uncertain transition function. We proof the non-convexity of this function in Appendix A.1.

Appendix B

Problem data

In this appendix, we give the concrete data of the base versions of the three uPOMDPs as described in Section 7.2.

B.1 Cheese maze

$$
\begin{aligned}
S &= \{s0, s1, s2, s3, s4, s5, s6, s7, s8, s9, s10, s11, s12, s13\} \\
A &= \{\text{North, East, South, West, Nothing}\} \\
Z &= \{\text{NW, NS, N, NE, EW, ESW, C}\}
\end{aligned}
$$

$$
R(s,a) = \begin{cases} 1 & \text{if } s = s13 \text{ and } a = \text{Nothing} \\ 0 & \text{otherwise} \end{cases}
$$

$$
\boldsymbol{O}(s) = \begin{cases} \text{NW} & \text{if } s = s0 \\ \text{NS} & \text{if } s \in \{s1, s3\} \\ \text{N} & \text{if } s = s2 \\ \text{NE} & \text{if } s = s4 \\ \text{EW} & \text{if } s \in \{s5, s6, s7, s8, s9, s10\} \\ \text{ESW} & \text{if } s \in \{s11, s12\} \\ \text{C} & \text{if } s = s13 \end{cases}
$$

We give the uncertain state transition function $\boldsymbol{P}$ as a transition matrix per action. We abbreviate the uncertainty intervals as follows:

$$
\begin{aligned}
[0, 0] &\mapsto \text{-} \\
[0.85, 0.95] &\mapsto \text{suc} \\
[0.05, 0.15] &\mapsto \text{fail} \\
[1, 1] &\mapsto \text{cer}
\end{aligned}
$$

	$s0$	$s1$	$s2$	$s3$	$s4$	$s5$	$s6$	$s7$	$s8$	$s9$	$s10$	$s11$	$s12$	$s13$
$s0$	-	-	-	-	-	-	-	-	-	-	-	-	-	-
$s1$	-	-	-	-	-	-	-	-	-	-	-	-	-	-
$s2$	-	-	-	-	-	-	-	-	-	-	-	-	-	-
$s3$	-	-	-	-	-	-	-	-	-	-	-	-	-	-
$s4$	-	-	-	-	-	-	-	-	-	-	-	-	-	-
$s5$	suc	-	-	-	-	fail	-	-	-	-	-	-	-	-
$s6$	-	-	suc	-	-	-	fail	-	-	-	-	-	-	-
$s7$	-	-	-	-	suc	-	-	fail	-	-	-	-	-	-
$s8$	-	-	-	suc	-	-	fail	-	-	-	-	-	-	-
$s9$	-	-	-	-	-	-	-	suc	-	fail	-	-	-	-
$s10$	-	-	-	-	-	suc	-	-	-	fail	-	-	-	-
$s11$	-	-	-	-	-	-	-	-	suc	-	fail	-	-	-
$s12$	-	-	-	-	-	-	-	-	suc	-	-	fail	-	-
$s13$	-	-	-	-	-	-	-	-	-	-	-	-	-	-

Table B.1: Transition matrix North.

	$s0$	$s1$	$s2$	$s3$	$s4$	$s5$	$s6$	$s7$	$s8$	$s9$	$s10$	$s11$	$s12$	$s13$
$s0$	fail	suc	-	-	-	-	-	-	-	-	-	-	-	-
$s1$	-	fail	suc	-	-	-	-	-	-	-	-	-	-	-
$s2$	-	-	fail	suc	-	-	-	-	-	-	-	-	-	-
$s3$	-	-	-	fail	suc	-	-	-	-	-	-	-	-	-
$s4$	-	-	-	-	-	-	-	-	-	-	-	-	-	-
$s5$	-	-	-	-	-	-	-	-	-	-	-	-	-	-
$s6$	-	-	-	-	-	-	-	-	-	-	-	-	-	-
$s7$	-	-	-	-	-	-	-	-	-	-	-	-	-	-
$s8$	-	-	-	-	-	-	-	-	-	-	-	-	-	-
$s9$	-	-	-	-	-	-	-	-	-	-	-	-	-	-
$s10$	-	-	-	-	-	-	-	-	-	-	-	-	-	-
$s11$	-	-	-	-	-	-	-	-	-	-	-	-	-	-
$s12$	-	-	-	-	-	-	-	-	-	-	-	-	-	-
$s13$	-	-	-	-	-	-	-	-	-	-	-	-	-	-

Table B.2: Transition matrix East.

	$s0$	$s1$	$s2$	$s3$	$s4$	$s5$	$s6$	$s7$	$s8$	$s9$	$s10$	$s11$	$s12$	$s13$
$s0$	fail	-	-	-	-	suc	-	-	-	-	-	-	-	-
$s1$	-	-	-	-	-	-	-	-	-	-	-	-	-	-
$s2$	-	-	fail	-	-	-	suc	-	-	-	-	-	-	-
$s3$	-	-	-	-	-	-	-	-	-	-	-	-	-	-
$s4$	-	-	-	-	fail	-	-	suc	-	-	-	-	-	-
$s5$	-	-	-	-	-	fail	-	-	suc	-	-	-	-	-
$s6$	-	-	-	-	-	-	fail	-	-	-	suc	-	-	-
$s7$	-	-	-	-	-	-	-	fail	-	suc	-	-	-	-
$s8$	-	-	-	-	-	-	-	-	fail	-	-	suc	-	-
$s9$	-	-	-	-	-	-	-	-	-	fail	-	-	suc	-
$s10$	-	-	-	-	-	-	-	-	-	-	fail	-	-	suc
$s11$	-	-	-	-	-	-	-	-	-	-	-	-	-	-
$s12$	-	-	-	-	-	-	-	-	-	-	-	-	-	-
$s13$	-	-	-	-	-	-	-	-	-	-	-	-	-	-

Table B.3: Transition matrix South.

	$s0$	$s1$	$s2$	$s3$	$s4$	$s5$	$s6$	$s7$	$s8$	$s9$	$s10$	$s11$	$s12$	$s13$
$s0$	-	-	-	-	-	-	-	-	-	-	-	-	-	-
$s1$	suc	fail	-	-	-	-	-	-	-	-	-	-	-	-
$s2$	-	suc	fail	-	-	-	-	-	-	-	-	-	-	-
$s3$	-	-	suc	fail	-	-	-	-	-	-	-	-	-	-
$s4$	-	-	-	suc	fail	-	-	-	-	-	-	-	-	-
$s5$	-	-	-	-	-	-	-	-	-	-	-	-	-	-
$s6$	-	-	-	-	-	-	-	-	-	-	-	-	-	-
$s7$	-	-	-	-	-	-	-	-	-	-	-	-	-	-
$s8$	-	-	-	-	-	-	-	-	-	-	-	-	-	-
$s9$	-	-	-	-	-	-	-	-	-	-	-	-	-	-
$s10$	-	-	-	-	-	-	-	-	-	-	-	-	-	-
$s11$	-	-	-	-	-	-	-	-	-	-	-	-	-	-
$s12$	-	-	-	-	-	-	-	-	-	-	-	-	-	-
$s13$	-	-	-	-	-	-	-	-	-	-	-	-	-	-

Table B.4: Transition matrix West.

	$s0$	$s1$	$s2$	$s3$	$s4$	$s5$	$s6$	$s7$	$s8$	$s9$	$s10$	$s11$	$s12$	$s13$
$s0$	-	-	-	-	-	-	-	-	-	-	-	-	-	-
$s1$	-	-	-	-	-	-	-	-	-	-	-	-	-	-
$s2$	-	-	-	-	-	-	-	-	-	-	-	-	-	-
$s3$	-	-	-	-	-	-	-	-	-	-	-	-	-	-
$s4$	-	-	-	-	-	-	-	-	-	-	-	-	-	-
$s5$	-	-	-	-	-	-	-	-	-	-	-	-	-	-
$s6$	-	-	-	-	-	-	-	-	-	-	-	-	-	-
$s7$	-	-	-	-	-	-	-	-	-	-	-	-	-	-
$s8$	-	-	-	-	-	-	-	-	-	-	-	-	-	-
$s9$	-	-	-	-	-	-	-	-	-	-	-	-	-	-
$s10$	-	-	-	-	-	-	-	-	-	-	-	-	-	-
$s11$	-	-	-	-	-	-	-	-	-	-	-	-	-	-
$s12$	-	-	-	-	-	-	-	-	-	-	-	-	-	-
$s13$	-	-	-	-	-	-	-	-	-	-	-	-	-	cer

Table B.5: Transition matrix Nothing.

Initial belief:

$$b0\colon \{s8 \mapsto [0.8, 0.8], s9 \mapsto [0.1, 0.1], s10 \mapsto [0.1, 0.1]\}$$

B.2 Tiger problem

$$S = \{\text{LoN, RoN, LoL, RoL, LoR, RoR}\}$$
$$A = \{\text{Left, Right, Listen}\}$$
$$Z = \{\text{Nothing, TigerLeft, TigerRight}\}$$
$$R(s,a) = \begin{cases} 10 & \text{if } s \in \{\text{LoN, LoL, LoR}\} \text{ and } a = \text{Left} \\ 10 & \text{if } s \in \{\text{RoN, RoL, RoR}\} \text{ and } a = \text{Right} \\ -100 & \text{if } s \in \{\text{LoN, LoL, LoR}\} \text{ and } a = \text{Right} \\ -100 & \text{if } s \in \{\text{RoN, RoL, RoR}\} \text{ and } a = \text{Left} \\ -1 & \text{if } a = \text{Listen} \end{cases}$$
$$\boldsymbol{O}(s) = \begin{cases} \text{Nothing} & \text{if } s \in \{\text{LoN, RoN}\} \\ \text{TigerLeft} & \text{if } s \in \{\text{LoL, RoL}\} \\ \text{TigerRight} & \text{if } s \in \{\text{LoR, RoR}\} \end{cases}$$

We give the uncertain state transition function $\boldsymbol{P}$ as a transition matrix per action. We abbreviate the uncertainty intervals as follows:

$$[0,0] \mapsto \text{-}$$
$$[0.8,0.9] \mapsto \text{suc}$$
$$[0.1,0.2] \mapsto \text{fail}$$
$$[0.5,0.5] \mapsto \text{res}$$

	LoN	RoN	LoL	RoL	LoR	RoR
LoN	res	res	-	-	-	-
RoN	res	res	-	-	-	-
LoL	res	res	-	-	-	-
RoL	res	res	-	-	-	-
LoR	res	res	-	-	-	-
RoR	res	res	-	-	-	-

Table B.6: Transition matrix Left.

	LoN	RoN	LoL	RoL	LoR	RoR
LoN	res	res	-	-	-	-
RoN	res	res	-	-	-	-
LoL	res	res	-	-	-	-
RoL	res	res	-	-	-	-
LoR	res	res	-	-	-	-
RoR	res	res	-	-	-	-

Table B.7: Transition matrix right.

	LoN	RoN	LoL	RoL	LoR	RoR
LoN	-	-	suc	-	fail	-
RoN	-	-	-	fail	-	suc
LoL	-	-	suc	-	fail	-
RoL	-	-	-	fail	-	suc
LoR	-	-	suc	-	fail	-
RoR	-	-	-	fail	-	suc

Table B.8: Transition matrix Listen.

Initial belief:

$$b0\colon \{\text{LoN} \mapsto [0.5,0.5], \text{RoN} \mapsto [0.5,0.5]\}$$

B.3 Grid-world robot

$$
\begin{aligned}
S &= \{s0, s1, s2, s3, s4, s5, s6, s7, s8\} \\
A &= \{\text{North, East, South, West, Nothing}\} \\
Z &= \{\text{Corner, Wall, Free, Goal}\}
\end{aligned}
$$

$$
R(s,a) = \begin{cases} 1 & \text{if } s = s8 \text{ and } a = \text{Nothing} \\ 0 & \text{otherwise} \end{cases}
$$

$$
\boldsymbol{O}(s) = \begin{cases} \text{Corner} & \text{if } s \in \{s0, s2, s6\} \\ \text{Wall} & \text{if } s \in \{s1, s3, s5, s7\} \\ \text{Free} & \text{if } s = s5 \\ \text{Goal} & \text{if } s = s8 \end{cases}
$$

We give the uncertain state transition function $\boldsymbol{P}$ as a transition matrix per action. We abbreviate the uncertainty intervals as follows:

$$
\begin{aligned}
[0, 0] &\mapsto \text{-} \\
[0.85, 0.95] &\mapsto \text{suc1} \\
[0.75, 0.85] &\mapsto \text{suc2} \\
[0.05, 0.15] &\mapsto \text{dev} \\
[1, 1] &\mapsto \text{cer}
\end{aligned}
$$

	$s0$	$s1$	$s2$	$s3$	$s4$	$s5$	$s6$	$s7$	$s8$
$s0$	cer	-	-	-	-	-	-	-	-
$s1$	-	cer	-	-	-	-	-	-	-
$s2$	-	-	cer	-	-	-	-	-	-
$s3$	suc1	dev	-	-	-	-	-	-	-
$s4$	dev	suc2	dev	-	-	-	-	-	-
$s5$	-	dev	suc1	-	-	-	-	-	-
$s6$	-	-	-	suc1	dev	-	-	-	-
$s7$	-	-	-	dev	suc2	dev	-	-	-
$s8$	-	-	-	-	-	-	-	-	-

Table B.9: Transition matrix North.

	$s0$	$s1$	$s2$	$s3$	$s4$	$s5$	$s6$	$s7$	$s8$
$s0$	-	suc1	-	-	dev	-	-	-	-
$s1$	-	-	suc1	-	-	dev	-	-	-
$s2$	-	-	cer	-	-	-	-	-	-
$s3$	-	dev	-	-	suc2	-	-	dev	-
$s4$	-	-	dev	-	-	suc2	-	-	dev
$s5$	-	-	-	-	-	cer	-	-	-
$s6$	-	-	-	-	dev	-	-	suc1	-
$s7$	-	-	-	-	-	dev	-	-	suc1
$s8$	-	-	-	-	-	-	-	-	-

Table B.10: Transition matrix East.

	$s0$	$s1$	$s2$	$s3$	$s4$	$s5$	$s6$	$s7$	$s8$
$s0$	-	-	-	suc1	dev	-	-	-	-
$s1$	-	-	-	dev	suc2	dev	-	-	-
$s2$	-	-	-	-	dev	suc1	-	-	-
$s3$	-	-	-	-	-	-	suc1	dev	-
$s4$	-	-	-	-	-	-	dev	suc2	dev
$s5$	-	-	-	-	-	-	-	dev	suc1
$s6$	-	-	-	-	-	-	cer	-	-
$s7$	-	-	-	-	-	-	-	cer	-
$s8$	-	-	-	-	-	-	-	-	-

Table B.11: Transition matrix South.

	$s0$	$s1$	$s2$	$s3$	$s4$	$s5$	$s6$	$s7$	$s8$
$s0$	cer	-	-	-	-	-	-	-	-
$s1$	suc1	-	-	dev	-	-	-	-	-
$s2$	-	suc1	-	-	dev	-	-	-	-
$s3$	-	-	-	cer	-	-	-	-	-
$s4$	dev	-	-	suc2	-	-	dev	-	-
$s5$	-	dev	-	-	suc2	-	-	dev	-
$s6$	-	-	-	-	-	-	cer	-	-
$s7$	-	-	-	dev	-	-	suc1	-	-
$s8$	-	-	-	-	-	-	-	-	-

Table B.12: Transition matrix West.

	$s0$	$s1$	$s2$	$s3$	$s4$	$s5$	$s6$	$s7$	$s8$
$s0$	-	-	-	-	-	-	-	-	-
$s1$	-	-	-	-	-	-	-	-	-
$s2$	-	-	-	-	-	-	-	-	-
$s3$	-	-	-	-	-	-	-	-	-
$s4$	-	-	-	-	-	-	-	-	-
$s5$	-	-	-	-	-	-	-	-	-
$s6$	-	-	-	-	-	-	-	-	-
$s7$	-	-	-	-	-	-	-	-	-
$s8$	-	-	-	-	-	-	-	-	cer

Table B.13: Transition matrix Nothing.

Initial belief:

$$b0\colon \{s0 \mapsto [0.8, 0.8], s2 \mapsto [0.1, 0.1], s6 \mapsto [0.1, 0.1]\}$$