Robust Policy Synthesis for Uncertain POMDPs via Convex Optimization

Marnix Suilen\textsuperscript{1}, Nils Jansen\textsuperscript{1}, Murat Cubuktepe\textsuperscript{2} and Ufuk Topcu\textsuperscript{2}

\textsuperscript{1}Department of Software Science, Radboud University, The Netherlands
\textsuperscript{2}Department of Aerospace Engineering and Engineering Mechanics, University of Texas at Austin, USA

marnix@marnixsuilen.nl, nils.jansen@ru.nl, \{mcubuktepe, utopcu\}@utexas.edu

Abstract

We study the problem of policy synthesis for uncertain partially observable Markov decision processes (uPOMDPs). The transition probability function of uPOMDPs is only known to belong to a so-called uncertainty set, for instance in the form of probability intervals. Such a model arises when, for example, an agent operates under information limitation due to imperfect knowledge about the accuracy of its sensors. The goal is to compute a policy for the agent that is robust against all possible probability distributions within the uncertainty set. In particular, we are interested in a policy that robustly ensures the satisfaction of temporal logic and expected reward specifications. We state the underlying optimization problem as a semi-infinite quadratically-constrained quadratic program (QCQP), which has finitely many variables and infinitely many constraints. Since QCQPs are non-convex in general and practically infeasible to solve, we resort to the so-called convex-concave procedure to convexify the QCQP. Even though convex, the resulting optimization problem still has infinitely many constraints and is NP-hard. For uncertainty sets that form convex polytopes, we provide a transformation of the problem to a convex QCQP with finitely many constraints. We demonstrate the feasibility of our approach by means of several case studies that highlight typical bottlenecks for our problem. In particular, we show that we are able to solve benchmarks with hundreds of thousands of states, hundreds of different observations, and we investigate the effect of different levels of uncertainty in the models.

1 Introduction

Partially observable Markov decision processes (POMDPs) model sequential decision-making problems under stochastic uncertainties and partial information [Kaelbling et al., 1998]. In particular, an agent that operates in an environment modeled by a POMDP receives observations according to which it tries to infer the likelihood, called the belief state, of the system being in a certain state. Based on this partial information about the environment, the agent chooses action whose outcome is stochastically determined.

The assumption that the transition and observation probabilities in POMDPs are explicitly given does often not hold. Unforeseeable events such as (unpredictable) structural damage to a system [Meuleu et al., 2010] or an imprecise sensor model [Bagnell et al., 2001] may necessitate to account for additional uncertainties in the value of the probabilities. Uncertain POMDPs (uPOMDPs) address this need by incorporating sets of uncertainties in the probabilities. The sets may be described as, for example, intervals [Givan et al., 2000] or more generally by likelihood functions [Nilim and Ghaoui, 2005]. For example, take a robust aircraft collision avoidance system that issues advisories to pilots [Kochenderfer, 2015]. Modeled as a uPOMDP, the actions relate to such advice and concern the flying altitude and the speed. Uncertainty enters the model via unreliable data of the reaction time of a pilot, and in uncertain probability of receiving false observations regarding the speed and altitude of other aircrafts.

We study the synthesis of policies in uPOMDPs. Specifically, we seek to compute a policy that satisfies temporal logic [Pnueli, 1977] or expected reward specifications against all possible probability distributions from the uncertainty set. For the aforementioned collision avoidance system, such a policy would minimize the acceleration due to fuel efficiency and ensure that the probability of not colliding with another aircraft is above a certain threshold.

The robust synthesis problem for uncertain MDPs, that is, with full observability, has been extensively studied. The existing approaches rely, for instance, on dynamic programming [Wolff et al., 2012], convex optimization [Puggelli et al., 2013], or value iteration [Hahn et al., 2017]. While the complexity of solving a standard MDP is polynomial in the number of states and actions, solving an uncertain MDP is NP-hard in general [Wiesemann et al., 2013]. The existing approaches for uPOMDPs rely on sampling [Burns and Brock, 2007] or robust value iteration [Ogoshi, 2015] on the belief space of the uPOMDP, but do not take temporal logic constraints into account. In general, the robust synthesis problem for uPOMDPs is hard. Already computing an optimal policy for POMDPs with no uncertainty is undecidable [Madani et al., 1999] and NP-hard [Vlassis et al., 2012] if policies do not take the execution history into account. In that case, policies are called memoryless.
We develop a novel solution for efficiently computing policies for uPOMDPs using robust convex optimization. We restrict the problem to memoryless policies, while the approach is readily applicable to finite-memory policies by explicitly encoding finite state controllers into the state space [Junges et al., 2018]. For brevity in this paper, we focus on uncertainty sets that are given by intervals, i.e., upper and lower bounds on probabilities. The approach, though, is applicable to all uncertainty sets that are represented as convex polytopes.

First, we encode the problem as a semi-infinite quadratically-constrained quadratic program (QCQP), which includes finitely many variables but infinitely (in fact uncountably) many constraints that capture the uncertainty set [Wiesemann et al., 2013]. The structure of the encoding is similar to the one for POMDPs without uncertainty [Amato et al., 2006]. This optimization problem is non-convex in general and thereby infeasible to solve in practice [Chen et al., 2017]. We use the so-called convex-concave procedure to convexify the problem [Lipp and Boyd, 2016]. The resulting convex QCQP provides a sound over-approximation of the original problem, yet, it still has infinitely many constraints and renders the application of the convex-concave procedure impractical.

Towards computational tractability for solving the semi-infinite convex QCQP, we restrict the uncertainty set to convex polytopes and gain two key advantages. First, a convex polytope represents the valid probability distributions exactly and avoids an unnecessarily coarse approximation of the uncertainty. Second, it suffices to enumerate over the finite set of vertices of these polytopes to retain optimal solutions [Löfberg, 2012, Section 5.2]. We exploit this property and transform the semi-infinite program to a finite convex QCQP which, integrated with the convex-concave procedure, provides an efficient solution to the robust synthesis problem.

Three complicating factors require special attention in the proposed solution. First, the iterative convex-concave procedure (CCP) may take exponentially many iterations in the number of its input variables [Park and Boyd, 2017]. The reason is that the standard stopping criterion of the CCP is conservative, and we observe in the numerical examples that it largely affects the runtime. We provide a dedicated version of the CCP that mitigates this problem by integrating a robust verification method [Benedikt et al., 2013], similar as in [Cubuktepe et al., 2018] for so-called parametric MDPs [Dehnert et al., 2015; Junges et al., 2019]. In particular, we compute the exact probability and expected cost values in intermediate candidate solutions delivered by the CCP. We check whether these solutions already satisfy the specifications, otherwise the exact values are used as input for the next iteration of the CCP.

Second, enumerating the vertices of the convex polytope causes an exponential number of constraints in the number of uncertain transitions. This number, however, depends here on the number of successor states of each state-action pair in the uPOMDP, and typical benchmarks, as available at http://pomdp.org or used in [Norman et al., 2017], are usually sparse, reducing the effect of this theoretical blowup.

The third complicating factor is the general hardness of problems with partial observability and particularly due to the number of observations. On the other hand, the size of the resulting convex optimization problems in the proposed solution is polynomial in the number of observations as well as states. Note that the range of the uncertainty sets, or more specifically the size of the intervals, does not affect the number of constraints. With our prototype implementation, we solve problems with hundreds of thousands of states and thousands of observations for several well-known case studies.

Related work. To the best of our knowledge, the proposed approach is the first that accounts for temporal logic specifications in the computation of policies for uPOMDPs. Beyond that, [Burns and Brock, 2007] relies on sampling and [Oso-gami, 2015] uses robust value iteration on the belief space of the uPOMDP. [Ahmadi et al., 2018] uses sum-of-squares optimization for verification of uPOMDPs, but the approach only scales to very small models. [Itoh and Nakamura, 2007; Cubuktepe et al., 2020] assume distributions over exact probability values. Robustness in [Chamie and Mostafa, 2018] is defined over fixed belief regions. [aras et al., 2007; Kumar et al., 2016] employ mixed-integer linear programming for POMDPs. An adaption to uPOMDPs would induce a rather coarse over-approximation of optimal robust policies.

2 Preliminaries

A probability distribution over a finite or countably infinite set $X$ is a function $\mu: X \to [0, 1] \subseteq \mathbb{R}$ with $\sum_{x \in X} \mu(x) = 1$. The set of all distributions on $X$ is denoted by $\text{Distr}(X)$. A convex polytope is an $n$-dimensional shape defined by linear inequalities $A \bar{x} \leq c$, where $A \in \mathbb{R}^{n \times m}$ and $c \in \mathbb{R}^n$.

Definition 1 (uMDP). An uncertain Markov decision process (uMDP) is a tuple $\mathcal{M} = (S, s_1, \text{Act}, \mathcal{P}, I)$ where $S$ is a set of states, $s_1 \in S$ is the initial state, Act is the set of actions, $I = \{(a, b) | a, b \in \{0, 1\} \text{ and } a \leq b\}$ is a set of probability intervals, such that $\mathcal{P}: S \times \text{Act} \times S \to I$ forms the uncertain transition function. A reward function $r: S \times \text{Act} \to \mathbb{R}_{\geq 0}$ assigns rewards to state action pairs.

For a uMDP $\mathcal{M}$ and a transition probability function $P: S \times \text{Act} \to \text{Distr}(S)$, we write $P \in \mathcal{P}$ if for all $s, s' \in S$ and $\alpha \in \text{Act}$ we have $P(s, \alpha, s') \in \mathcal{P}(s, \alpha, s')$. Intuitively, $P$ yields only values from the corresponding intervals of $\mathcal{P}$ for each state-action pair $s$, $\alpha$ and successor state $s'$, which is equivalent to having independent uncertainty between different actions. We restrict $P$ to only select values from the intervals that form valid probability distributions and discuss later how this is achieved algorithmically. A uMDP is instantiated by $P \in \mathcal{P}$, yielding a Markov decision process (MDP) $\mathcal{M}[P]$.

Remark 1. For the correctness of our method, we require the lower bounds of the intervals to be strictly larger than zero, that is, an instantiation cannot “eliminate” transitions. Put differently, either a transition exists in all instantiations of the uMDP, or in none. That assumption is standard and, for instance, also employed in [Wiesemann et al., 2013]. Moreover, the problem statement would be different and theoretically harder to solve, see [Winkler et al., 2019]. We allow the upper and lower bound of an interval to be the same, resulting in nominal transition probabilities.

Definition 2 (uPOMDP). An uncertain partially observable MDP (uPOMDP) is a tuple $\mathcal{M} = (M, Z, O)$, with $M = \ldots$
(S, s₁, Act, P, I) the underlying uMDP of M, Z a finite set of observations and O : S → Distr(Z) the observation function.

For ease of presentation, we often assume that the observation function is deterministic, that is, of the form O : S → Z. Note that deterministic observation functions can be derived from general stochastic functions by expanding the state space [Chatterjee et al., 2016]. Furthermore, we assume that all states have the same actions.

An observation-based policy σ : Z → Distr(Act) for a uPOMDP maps observations to distributions over actions. Note that such a policy is referred to as memoryless and randomized. More general (and powerful) types of policies take an (in)finite sequence of observations and actions into account. $M$ is the set of observation-based strategies for M. Applying $\sigma \in \Sigma M$ to M resolves all choices and partial observability and an induced (uncertain) Markov chain $M^\sigma$ results.

For a POMDP $M_L$ (without uncertainties) and a set of target states $T \subseteq S$, the reachability specification $\varphi_r = P_{\geq \lambda}(\sigma T)$ states that the probability of reaching $T$ shall be at least $\lambda$. Similarly, the expected cost specification $\varphi_c = \mathbb{E}_{\leq \kappa}(\sigma G)$ states that the expected cost of reaching the goal set $G \subseteq S$ shall be less than or equal to $\kappa$. A policy $\sigma \in \Sigma M$, satisfies $\varphi_r$ (or $\varphi_c$) if it is satisfied on the Markov chain $M^\sigma$, denoted by $\sigma \models \varphi_r$ (or $\sigma \models \varphi_c$). A policy for uPOMDPs takes all possible instantiations from the uncertainty sets into account.

Definition 3 (Robust Policy). For a uPOMDP M, the underlying uMDP $M = (S, s_1, Act, P, I)$, and a specification $\varphi$, an observation-based policy $\sigma \in \Sigma M$ robustly satisfies $\varphi$ for $M (\sigma \models \varphi)$ if for all $P \in P$ it holds that $M[P]^\sigma$ satisfies $\varphi$.

Intuitively, the policy needs to satisfy the specification for all instantiations from $M[P]$. If we have several (expected cost or reachability) specifications $\varphi_1, \ldots, \varphi_m$, we write $\sigma$ satisfies all specifications. Note that general temporal logic constraints can be reduced to reachability specifications [Baier and Katoen, 2008; Bouton et al., 2020], therefore we omit a detailed introduction to the underlying logic.

### 3 Formal Problem and Outline

We first state the central problem of this paper.

**Problem (Robust Synthesis for uPOMDPs).** Given an uPOMDP $M = (M, Z, O)$, and a specification $\varphi$, which is either a reachability specification $\varphi_r = P_{\geq \lambda}(\sigma T)$ or an expected cost specification $\varphi_c = \mathbb{E}_{\leq \kappa}(\sigma G)$, compute a randomized memoryless policy $\sigma \in \Sigma M$ for M such that $\sigma$ robustly satisfies the specification, that is, $\sigma \models \varphi$.

**Outline.** Figure 1 shows the outline of our approach. The input is a uPOMDP M and one or more specifications $\varphi$. We first state a semi-infinite optimization problem which defines the robust synthesis problem within this section. In Sect. 4, we show how this nonlinear problem can be convexified around an initial policy $\sigma \in \Sigma M$, followed by Sect. 5 which describes how a finite, efficiently solvable problem is obtained. This procedure is augmented by an efficient robust verification method.

### 3.1 Semi-Infinite Optimization Problem

We introduce the following variables for the optimization problem: $\{p_s \mid s \in S\}$ for the probability to reach the targets $T$ from state $s$, $\{c_s \mid s \in S\}$ for the expected cost to reach the goal set $G$ from $s$, and $\{c_{s,\alpha} \mid s \in S, \alpha \in Act\}$ to encode the randomized policy.

\[
\text{minimize } c_{s_1} \quad (1) \\
\text{subject to } p_{s_1} \geq \lambda, \quad c_{s_1} \leq \kappa, \quad (2) \\
\forall s \in T. \quad p_s = 1, \quad \forall s \in G. \quad c_s = 0, \quad (3) \\
\forall s \in S. \quad \sum_{\alpha \in Act} c_{s_1,\alpha} = 1, \quad (4) \\
\forall s \in S \setminus T. \quad \forall P \in P. \quad (5) \\
\forall s \in S \setminus G. \forall P \in P. \quad (6) \\
\forall s \in S \setminus G. \forall P \in P. \quad (7)
\]

The objective is to minimize $c_{s_1}$ for minimizing the expected cost at the initial state. The constraints in (2) encode a reachability and expected cost threshold, respectively. (3) defines the fixed reachability and cost values for states that belong to the respective target and goal set, and (4) encodes valid policy probabilities. (5) and (6) define the reachability and cost variables for all other states. Note that $p_s$ may at least be assigned the exact probability to reach $T$, ensuring the correct satisfaction of the specification $P_{\geq \lambda}(\sigma T)$. Finally, (7) defines policy variables from states with the same observation to have the same value, ensuring our policy is based on the observations instead of the states.

We will now take a closer look at the type of this optimization problem. First, the functions in (5) and (6) are quadratic. Essentially, the policy variables $c_{s_1,\alpha}$ are multiplied with the probability variables $p_s$ in constraint (5) and with the cost variables $c_s$ in (6). As the entries in the transition probability matrices $P(s, \alpha, s')$ for $s, s' \in S$ and $\alpha \in Act$ belong to continuous intervals, there are infinitely many of the constraints (5) and (6) over a finite set of variables. Note that

![Flowchart of the overall approach.](image-url)
we only consider policies where for all states \( s \) and actions \( \alpha \) it holds that \( \sigma_{s,\alpha} > 0 \), such that applying the policy to the uPOMDP does not exclude states or transitions.

4 Convexifying the Semi-Infinite QCQP

We discuss how we convexify the semi-infinite QCQP. We use the penalty convex-concave procedure (CCP) [Lipp and Boyd, 2016] which iteratively over-approximates a non-convex optimization problem via linearization. The resulting convex problem can then be solved efficiently, and the process is iterated until a suitable solution is found. Specifically, we rewrite the quadratic functions in (5) and (6) as a sum of convex and concave functions and compute upper bounds for the concave functions. We check the feasibility regarding the reachability and expected cost specifications using robust value iteration [Wiesemann et al., 2013]. Until such a feasible solution is found, the CCP seeks solutions in the vicinity of previous ones.

Remark 2. This section assumes we can effectively solve a semi-infinite convex optimization problem. How to do this in our setting is discussed in Section 5.

The CCP method starts with any (possibly infeasible) assignment \( p_s, c_s, \) and \( \sigma_{s,\alpha} \) to the variables \( p_s, c_s, \) and \( \sigma_{s,\alpha} \). Consider the bilinear function
\[
\hat{h}^c(s, \alpha, s', P) = P(s, \alpha, s') \cdot \sigma_{s,\alpha} \cdot c_s
\]
for any \( s, s' \in S, \alpha \in \mathcal{A} \) and \( P \in \mathcal{P} \) whose right-hand is part of the constraint (6) in the original QCQP. For simplicity, we set \( P(s, \alpha, s') = 2 \cdot \mathcal{d}, \sigma_{s,\alpha} = y, \) and \( c_s = z \) and get \( \hat{h}^c(s, \alpha, s', P) = 2 \cdot \mathcal{d} \cdot y \cdot z \). We rewrite \( 2 \cdot \mathcal{d} \cdot y \cdot z \) to \( 2 \cdot \mathcal{d} \cdot y \cdot z + d(y^2 + z^2) - d(y^2 + z^2) \). Then, we can write \( 2 \cdot \mathcal{d} \cdot y \cdot z + d(y^2 + z^2) \) as \( \hat{h}^c_{\text{cvx}}(s, \alpha, s', P) = d(y + z)^2 \), which is a quadratic convex function. Recalling (6), we add the cost function \( c(s, \alpha) \) and get the convex function \( \hat{h}^c_{\text{cvx}} = \hat{h}^c_{\text{cvx}} + y \cdot c(s, \alpha) \) as \( y \cdot c(s, \alpha) \) is affine.

The function \( \hat{h}^c_{\text{cvx}}(s, \alpha, s', P) = -d(y^2 + z^2) \), however, is concave, and we have to convexify it. In particular, we transform \( \hat{h}^c_{\text{cvx}}(s, \alpha, s', P) \) to \( \hat{h}^c_{\text{cvx}}(s, \alpha, s', P) = d(y^2 + z^2) + 2 \cdot d(y^2 + z^2 - y^2 - z^2) \), where \( y \) and \( z \) are any assignments to the policy and probability variables. \( \hat{h}^c_{\text{cvx}}(s, \alpha, s', P) \) is affine in \( y \) and \( z \) and therefore convex.

We convexify (5) analogously and replace the quadratic functions with \( \hat{h}^p_{\text{cvx}}(s, \alpha, s', P) \) and \( \hat{h}^c_{\text{cvx}}(s, \alpha, s', P) \).

After the convexification step, we replace (5) and (6) by
\[
\forall s \in S \setminus T. \forall P \in \mathcal{P}.
\]
\[
- p_s \geq \sum_{\alpha \in \mathcal{A} \cap s' \in S} \left( \hat{h}^p_{\text{cvx}}(s, \alpha, s', P) + \hat{h}^c_{\text{cvx}}(s, \alpha, s', P) \right),
\]
\[
\forall s \in S \setminus T. \forall P \in \mathcal{P}.
\]
\[
c_s \geq \sum_{\alpha \in \mathcal{A} \cap s' \in S} \left( \hat{h}^p_{\text{cvx}}(s, \alpha, s', P) + \hat{h}^c_{\text{cvx}}(s, \alpha, s', P) \right),
\]
which are semi-infinite convex constraints in \( \sigma_{s,\alpha}, p_s, c_s, \) and \( c_s, \).

We switch the sign of \( p_s \) as it was upper bounded before.

The resulting problem is convex (yet semi-infinite). As we over-approximate the quadratic functions, any feasible solution to the convex problem is also feasible for the original semi-infinite QCQP. However, due to the over-approximation, the resulting convex problem might be infeasible though the original one is not. To find a feasible assignment, we assign a so-called non-negative penalty variable \( k_s \) for each of the probability constraints in (8) for \( s \in S \setminus T \), and \( l_s \) for the cost constraints in (9). To find a solution that induces a minimal infeasibility, or minimal violations to the convexified constraints, we minimize the sum of the penalty variables. This gives us another semi-infinite convex problem:

\[
\text{minimize } c_s + \tau \left( \sum_{s \in S \setminus T} k_s + \sum_{s \in S \setminus G} l_s \right)
\]
subject to
\[
p_{s_1} \geq \lambda, \quad c_{s_{\text{init}}} \leq \kappa,
\]
\[
\forall s \in T. \quad p_s = 1, \quad \forall s \in \mathcal{G}. \quad c_s = 0,
\]
\[
\forall s \in S. \quad \sum_{\alpha \in \mathcal{A} \cap s} \sigma_{s,\alpha} = 1,
\]
\[
\forall s \in S \setminus T. \forall P \in \mathcal{P}.
\]
\[
k_s - p_s \geq \sum_{\alpha \in \mathcal{A} \cap s' \in S} \left( \hat{h}^p_{\text{cvx}}(s, \alpha, s', P) + \hat{h}^c_{\text{cvx}}(s, \alpha, s', P) \right),
\]
\[
\forall s \in S \setminus \mathcal{G}. \forall P \in \mathcal{P}.
\]
\[
l_s + c_s \geq \sum_{\alpha \in \mathcal{A} \cap s' \in S} \left( \hat{h}^p_{\text{cvx}}(s, \alpha, s', P) + \hat{h}^c_{\text{cvx}}(s, \alpha, s', P) \right),
\]
\[
\forall s, s' \in S. \forall \alpha \in \mathcal{A}. \quad O(s) = O(s') \rightarrow \sigma_{s,\alpha} = \sigma_{s',\alpha}.
\]

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

5 Derivation of the Convex QCQP

In this section, we describe how to transform the semi-infinite convex QCQP to a finite convex QCQP that is amenable to efficient solving techniques. That transformation largely depends on the type of uncertainty set that enters the problem. So far, we stated that we have to account for all concrete probability functions \( P \) within the uncertainty set \( \mathcal{P} \), see the constraints (5) and (6). We now describe this notion as a specific uncertainty set. In particular, we enumerate exactly all possible (valid) probability distributions from the uncertainty set that enter the problem in the form of probability intervals.

For a uPOMDP \( M = (S, Z, O) \) and its underlying mDPD \( M = (S, s_t, \mathcal{A}, P, \mathcal{I}) \), each state-action pair has a fixed number of associated probability intervals. For state \( s \in S \) and action \( \alpha \in \mathcal{A} \), we assume \( n \) intervals \([a_i, b_i] \in \mathbb{I}, 1 \leq i \leq n \).
For each transition probability function \( P \in \mathcal{P} \) at \((s, \alpha)\), we ensure that \( P \) is valid via \( \forall P \in \mathcal{P} . \sum_{x'} P(x', s, x') = 1 \). We define the set of all possible probability distributions formed by the intervals \([a_i, b_i]\), expressed by the following set of linear constraints that form a convex polytope:

\[
\forall i, 1 \leq i \leq n. \quad a_i \leq x_i \leq b_i, \quad \sum_{i=1}^{n} x_i = 1.
\]

We rewrite these constraints into their canonical form of values to the variables which induces a concrete policy a convex polytope can be exponential in the number of dimensions. However, in our specific case where we apply this description method [Fukuda and Prodon, 1996].

**Exact representation of distributions.** By construction, the polytope describes exactly the set of valid probability distributions from the intervals. Moreover, because the polytope is convex, it suffices to enumerate over the vertices [Löfberg, 2012] to capture all of these distributions. As we have these vertices now, we can simply replace the robust constraints (14) and (15) by a (finite) number of constraints in which the uncertainty is replaced by all possible combinations of values from the vertices. Effectively we enumerate all possible probabilities against which the policy needs to be robust. The resulting convex QCQP can directly be solved, for instance, by the QCQP solver Gurobi.

**Complexity of the convex QCQP.** Note that solving a robust convex QCQP with polytopic uncertainty is still NP-Hard [Bertsimas et al., 2011] as the number of vertices of a convex polytope can be exponential in the number of dimensions. However, in our specific case where we apply this method to uPOMDPs, the dimension of each polytope is determined by the number of successor states for each state-action pair. As mentioned before, we expect the number of successors to be low, and thus the dimension of each polytope and the number of vertices, to be manageable.

**Integrating robust verification.** In each iteration of the CCP as described in Sect.4, the QCQP solver assigns concrete values to the variables which induces a concrete policy \( \sigma \in \Sigma^M \). We apply this instantiation to the uPOMDP \( M \), resulting in an uncertain Markov chain \( M' \). For this model without partial observability and nondeterminism, we employ robust value iteration [Wiesemann et al., 2013] to check whether the specifications are already satisfied. Our numerical examples show that this additional verification step is a good heuristic for an earlier termination of the CCP when the penalty variables have not evaluated to zero yet. We also use the result to ensure that the probability and the cost variables are consistent with the policy variables for the next iteration of the CCP.

We evaluate our robust synthesis procedure on benchmark examples that are subject to either reachability or expected cost specifications. As part of a Python toolchain, we use the probabilistic model checker Storm [Dehnert et al., 2017] to extract an explicit state space representation of uPOMDPs.

The experiments were performed on a computer with an Intel Core i9-9900u 2.50 GHz processor and 64 GB of RAM with Gurobi 9.0 as the QCQP solver and our own implementation of a robust value iteration. We use a 1 hour time-out (TO). For all our examples we use a standard POMDP model so-called nominal probabilities, as well as two different sizes of probability intervals, namely a small one and a big one. The reason for these three options is that we want to showcase the effect of growing uncertainty on the runtime governed by the number of CCP iterations.

### 6 Numerical Examples

We consider the following POMDP case studies with added uncertainties in the form of intervals. **Grid-world robot** is based on the POMDP example in [Littman et al., 1995]. A robot is placed randomly into a grid and the goal is to safely reach the north-east corner, see Fig. 2(a). The robot may only reach intended states with a certain probability. We consider three variants for that probability: 0.98 (nominal), [0.95, 0.99] (small interval), and [0.50, 0.98] (big interval), yielding two distinct uPOMDPs and one POMDP. The reachability specification \( \mathbb{P}_{\geq 1} \) ensures to reach the target without visiting “traps”.

The second example is a maze setting, introduced in [McCullum, 1993], where a robot is to reach a target location in minimal time see Fig. 2(b). Again, we consider a “slippery” maze, similar to the previous example. We use the following probabilities for slipping: 0.97 (nominal), [0.94, 0.97] (small interval), and [0.50, 0.97] (big interval). We define an expected cost specification \( \mathbb{E}_{\leq 1} \) to reach the goal.

Our third example concerns scheduling wireless traffic, where at each time period a scheduler generates a new packet for each user [Yang et al., 2011]. The scheduler does not know the current states of the users, and has to schedule wireless traffic based on partial information with 9743 possible observations. We assume exact probabilities of the channel reliabilities are not known, and we have 0.9 (nominal), [0.875, 0.9] (small interval), and [0.8, 0.9] (big interval). The specification \( \mathbb{E}_{\leq 1} \) is to minimize the expected number of dropped packets for the scheduler.
Table 1: Numerical examples.

<table>
<thead>
<tr>
<th>Problem Type</th>
<th>States</th>
<th>Constraints</th>
<th>Iteration Time (s)</th>
<th>Problem Type</th>
<th>States</th>
<th>Constraints</th>
<th>Iteration Time (s)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Maze $E_{&lt;0.05}$</td>
<td>30</td>
<td>27</td>
<td>40</td>
<td>68</td>
<td>100</td>
<td>4.68</td>
<td>68</td>
</tr>
<tr>
<td>Maze $E_{&lt;0.05}$</td>
<td>30</td>
<td>27</td>
<td>42</td>
<td>68</td>
<td>101</td>
<td>4.90</td>
<td>68</td>
</tr>
<tr>
<td>Maze $E_{&lt;0.05}$</td>
<td>30</td>
<td>27</td>
<td>49</td>
<td>68</td>
<td>104</td>
<td>4.67</td>
<td>68</td>
</tr>
<tr>
<td>Grid $P_{&gt;0.84}$</td>
<td>18</td>
<td>14</td>
<td>8</td>
<td>56</td>
<td>8</td>
<td>0.20</td>
<td>56</td>
</tr>
<tr>
<td>Grid $P_{&gt;0.92}$</td>
<td>18</td>
<td>14</td>
<td>8</td>
<td>56</td>
<td>9</td>
<td>0.22</td>
<td>56</td>
</tr>
<tr>
<td>Aircraft $P_{&gt;0.80}$</td>
<td>175861</td>
<td>214448</td>
<td>2</td>
<td>399004</td>
<td>5</td>
<td>39.61</td>
<td>399004</td>
</tr>
<tr>
<td>Aircraft $P_{&gt;0.90}$</td>
<td>175861</td>
<td>214448</td>
<td>5</td>
<td>399004</td>
<td>26</td>
<td>215.94</td>
<td>399004</td>
</tr>
<tr>
<td>Aircraft $P_{&gt;0.95}$</td>
<td>175861</td>
<td>214448</td>
<td>40</td>
<td>399004</td>
<td>172</td>
<td>1475.70</td>
<td>399004</td>
</tr>
<tr>
<td>Aircraft $P_{&gt;0.97}$</td>
<td>175861</td>
<td>214448</td>
<td>50</td>
<td>399004</td>
<td>TO</td>
<td>TO</td>
<td>399004</td>
</tr>
<tr>
<td>Network $E_{&lt;0.90}$</td>
<td>38719</td>
<td>107068</td>
<td>8</td>
<td>187591</td>
<td>8</td>
<td>114.99</td>
<td>187591</td>
</tr>
<tr>
<td>Network $E_{&lt;0.50}$</td>
<td>38719</td>
<td>107068</td>
<td>10</td>
<td>187591</td>
<td>10</td>
<td>148.98</td>
<td>187591</td>
</tr>
<tr>
<td>Network $E_{&lt;0.5}$</td>
<td>38719</td>
<td>107068</td>
<td>12</td>
<td>187591</td>
<td>12</td>
<td>171.78</td>
<td>187591</td>
</tr>
</tbody>
</table>

6.2 Aircraft Collision Avoidance

In this more sophisticated example, we consider a robust aircraft collision avoidance problem [Kochenderfer, 2015]. The objective is to maximize the probability of avoiding a collision with an intruder aircraft while taking into account sensor errors and uncertainty in the future paths of the intruder. The problem is a POMDP with state variables (1) $h$, altitude of the intruder relative to the own aircraft, (2) $\dot{h}$, vertical rate of the intruder relative to the own aircraft, (3) $\tau$, time to potential collision, and (4) $s_{\text{res}}$, whether the pilot is responsive to requested commands.

We discretize the $h$ variable into 33 points over the range \pm 4000 feet, the $\dot{h}$ variable into 25 points between \pm 10,000 feet/minute, and $\tau$ to 40 points from 0 to 40 seconds. The 1905 possible observations give partial information of $h$ and $\dot{h}$. In the POMDP model, the probability of getting a correct observation is 0.95. Again, we assess the effect of the interval size by means of two intervals, namely [0.90, 0.95] (small interval) and [0.75, 0.95] (big interval). The specification $P_{>\lambda}$ is to maximize the probability of not having a collision with the intruder within 40 seconds. Similarly, we use different values of $\lambda$ to show the effect of different probability thresholds.

6.3 Discussion of the Results

In Table 1, we list the experimental results for different specification thresholds for each example. “States” denotes the number of states in the model, “Constraints” denotes the number of constraints in the convex QCQP, “Iterations” denotes the number of CCP iterations, and “Time (s)” denotes the time spent in Gurobi in seconds. We pick the specification thresholds such that one is near to the point where our procedure converges to an infeasible solution.

We remark that the number of constraints in each example increases by adding intervals (instead of concrete probabilities) to the model, due to the explicit enumeration of polytope vertices, see Section 5. However, the number of constraints does not depend on the size of the intervals. We also note that the solution time for each iteration for the problem with uncertainty (upOMDP) is larger than for the original model (POMDP) due to these additional constraints.

7 Conclusion

We presented a new approach to computing robust policies for uncertain POMDPs. The experiments showed that we are able to apply our method based on convex optimization on well-known benchmarks with varying levels of uncertainty. Future work will investigate finite memory policies, similar to [Junges et al., 2018], and recurrent neural networks as efficient representation for robust policies [Carr et al., 2019].

Acknowledgements

This work was supported by the grants ARL ACC-APG-RTP W911NF, DARPA D19AP00004, and NSF 1652113.
References


