Robust Markov Decision Processes: A Place Where AI and Formal Methods Meet

Marnix Suilen[1], Thom Badings[1], Eline M. Bovy[1], David Parker[2], and Nils Jansen[1,3]

[1] Radboud University, Nijmegen, The Netherlands
[2] University of Oxford, United Kingdom
[3] Ruhr-University Bochum, Germany

Abstract. Markov decision processes (MDPs) are a standard model for sequential decision-making problems and are widely used across many scientific areas, including formal methods and artificial intelligence (AI). MDPs do, however, come with the restrictive assumption that the transition probabilities need to be precisely known. *Robust* MDPs (RMDPs) overcome this assumption by instead defining the transition probabilities to belong to some *uncertainty set*. We present a gentle survey on RMDPs, providing a tutorial covering their fundamentals. In particular, we discuss RMDP semantics and how to solve them by extending standard MDP methods such as value iteration and policy iteration. We also discuss how RMDPs relate to other models and how they are used in several contexts, including reinforcement learning and abstraction techniques. We conclude with some challenges for future work on RMDPs.

Keywords: Robust Markov decision processes · Dynamic programming · Formal verification · Reinforcement learning.

1 Introduction

Markov decision processes (MDPs) are a fundamental model for tackling decision making under uncertainty across various areas, such as formal methods [72], operations research [99], and artificial intelligence [113]. At the core of MDPs is the assumption that the transition probabilities are precisely known, a requirement that is often prohibitive in practice [11]. For example, in data-driven applications in AI such as reinforcement learning (RL) [113], transition probabilities are unknown and can only be estimated from data. Furthermore, in formal verification problems, the state-explosion problem often prevents the model from being fully built [13, 14, 27]. As a remedy, sampling-based approaches that only estimate the transition probabilities, such as statistical model checking [6, 86], are used. Any sampling-based approach naturally carries the risk of statistical errors and hence, incorrect estimates of the probabilities.

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

choose. Tracing back to at least interval Markov chains in the formal methods community [68] and bounded-parameter MDPs in AI [46], RMDPs provide a general and flexible framework for modelling MDPs with uncertainty on the transition probabilities [64, 93, 118]. RMDPs provide a rigorous approach to quantifying the impact of data-driven methods for MDPs and, as such, represent an important topic in the intersection of AI and formal methods.

RMDPs in AI and formal methods. As RMDPs are studied across several research fields, results inevitably become scattered across the communities. While several recent algorithmic developments and applications of RMDPs stem from AI and operations research, see *e.g.*, [10, 47, 59] and many of the other works cited in this survey, tool support is arguably more mature in the formal methods community. While several of the major contributions to dynamic programming for RMDPs can be traced back more to the AI than the formal methods community, these algorithms have been implemented in probabilistic model checkers such as PRISM [80] and STORM [57], which are well-known within formal methods but less so in AI. Because work on RMDPs in formal methods and AI faces many of the same problems, we believe that research in both communities can benefit greatly from each other. In particular, theoretical contributions in one field may improve tool support in the other. Vice versa, improved tool support may lead to more advanced applications of RMDPs across both research areas.

Goal of this survey. The goal of this survey is to unify the views on RMDPs from the AI and formal methods communities. While the theory of RMDPs has made significant advances over the years, surveys summarizing these results are as of yet sparse. To the best of our knowledge, [95] is the only other survey on RMDPs available, with a primary focus on summarizing recent technical results. In contrast, this paper aims to provide an introduction to the theory of robust MDPs and a short review of its connections with other well-known models and applications in the areas of formal methods and AI.

Outline. This survey consists of two main parts. In the first part, we review the basics of MDPs in Section 2 and then lay out the theoretical foundations underpinning RMDPs in Section 3. These sections are meant to be accessible to readers with basic familiarity with MDPs. In the second part, we provide a gentle survey of the existing literature, in particular focusing on connections with other models (Section 4) and applications and tool support (Section 5). We conclude in Section 6 with some interesting directions for future work, both in theory and applications, for RMDPs.

2 Markov Decision Processes and How to Solve Them

For a set X, we write $|X|$ for its cardinality. Partial functions are denoted by $f\colon X \rightharpoonup Y$, and we write $\perp$ for *undefined*. A *discrete probability distribution* over

a finite set X is a function $\mu\colon X \to [0,1]$ such that $\sum_{x\in X} \mu(x) = 1$. The set of all probability distributions over X is denoted by $\mathcal{D}(X)$. A distribution is called Dirac if it assigns probability one to precisely one element and zero to all others.

2.1 Markov Decision Processes

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

Definition 1 (MDP). *A Markov decision process (MDP) is a tuple of the form (S, s_ι, A, P, R), where S is a finite set of states with $s_\iota \in S$ the initial state, A is a finite set of actions, $P\colon S \times A \rightharpoonup \mathcal{D}(S)$ is the probabilistic transition function, and $R\colon S \times A \rightharpoonup \mathbb{R}_{\geq 0}$ is the (non-negative) reward function.*

We focus on expected reward objectives and hence omit a labelling function from our MDPs. We use partial functions for the transition and reward functions to allow for enabled actions. An action is enabled if $P(s,a)$ is defined. We write $A(s) \subseteq A$ for the set of enabled actions at state s. We require that the transition and reward function are consistent with each other, that is, $P(s,a) = \bot \iff R(s,a) = \bot$. For convenience, we write $P(s,a,s')$ for the probability $P(s,a)(s')$.

A path in an MDP is an (in)finite sequence of successive states and actions: $\tau = (s_0, a_0, s_1, \dots) \in (S \times A)^* \times S$ where $s_0 = s_\iota$ and $\forall i \in \mathbb{N},\ P(s_i, a_i, s_{i+1}) > 0$. A path is finite if the sequence is finite, $\tau = (s_0, a_0, \dots, s_k)$, for which we write $\mathit{last}(\tau) = s_k$ for the last state. The set of all paths is denoted as Paths. The sequence of states in a path $\tau = (s_0, a_0, s_1, \dots)$ is $\mathit{states}(\tau) = (s_0, s_1, \dots)$.

A *discrete-time Markov chain* (DTMC) is an MDP with only one enabled action in each state: $\forall s \in S,\ |A(s)| = 1$. For DTMCs, we omit the actions altogether from the tuple and write (S, s_ι, P, R).

A *policy* (also called *scheduler* or *strategy*) is a function that maps paths to distributions over actions $\pi\colon \mathsf{Paths} \to \mathcal{D}(A)$. Such policies are called *history-based* and *randomized*. A policy is *deterministic* if it only maps to Dirac distributions over actions and *stationary* (also called *memoryless*) if it only considers finite paths of length one. Stationary deterministic (also called positional) policies are written as $\pi\colon S \to A$.

Given a policy π for an MDP M, the action choices in M are resolved, resulting in a DTMC.

Definition 2 (Induced DTMC). *Let $M = (S, s_\iota, A, P, R)$ be an MDP and $\pi\colon \mathsf{Paths} \to \mathcal{D}(A)$ a policy. The induced DTMC is a tuple $M_\pi = (S^*, s_\iota, P_\pi, R_\pi)$, where S^* is the (infinite) set of states, s_ι is the initial state, and the transition and reward functions are defined as*

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

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

The induced DTMC M_π has a unique probability measure $\mathbb{P}_{M_\pi}$ that follows from the standard cylinder set construction, see, *e.g.*, [14, 40]. For stationary policies, the set of states of the induced DTMC coincides with that of the MDP and is thus finite.

Objectives. The primary objective we consider in this paper is cumulative reward maximization until reaching a state in some target set $T \subseteq S$ [79]. We shall simply call this objective *reach-reward*. The goal is to compute the expected reward and an associated optimal policy for this objective in a given MDP M, which we formalize in the following subsection. Other common objectives include *reachability*, *discounted expected reward*, *reach-avoid*, and general temporal logic objectives expressed in (probabilistic) LTL [96] or CTL [52].

2.2 Classical Dynamic Programming

We review dynamic programming for MDPs with an infinite horizon undiscounted reward objective in preparation for our discussion of *robust* dynamic programming in Section 3. We define state and state-action value functions $V\colon S \to \mathbb{R}$ and $Q\colon S \times A \to \mathbb{R}$, respectively. Dynamic programming updates these value functions iteratively until the least fixed point is reached.

We preprocess the set of states S based on graph properties via the following standard procedure [14] and PRISM-semantics for reward objectives [41, 79]. Let $T \subseteq S$ be the target set of our objective, let $S^{\infty} \subseteq S$ be the set of states for which there exists a policy that does not reach T almost-surely, and denote the remaining states by $S^{?} = S \setminus (T \cup S^{\infty})$. For all $a \in A$ and target states $s \in T$, let $Q(s,a) = 0$. Similarly, for all $a \in A$ and $s \in S^{\infty}$, let $Q(s,a) = \infty$. For all other states $s \in S^{?}$ and $a \in A$, we initialize the state-action values as $Q(s,a) = 0$. We iteratively update the state and state-action values for all $(s,a) \in S^{?} \times A$ by:

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

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

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

The optimal value function V^* is the unique least fixed point of the *Bellman equation* in Equation (1).

For many objectives in MDPs, such as reach-reward maximization, optimal policies are stationary and deterministic, *i.e.*, of type $\pi\colon S \to A$ [99]. An optimal

stationary deterministic policy π^* that achieves value V^* can be extracted by performing the following one-step dynamic programming procedure:

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

Policy evaluation and improvement. As an alternative to value iteration, MDPs can also be solved through *policy iteration*. Policy iteration consists of two alternating steps: *policy evaluation* and *policy improvement*. Policy evaluation is the process of computing the value of an MDP for a given policy, which is also known as verifying or model checking the induced Markov chain from Definition 2 [14]. The value of a stationary policy $\pi \colon S \to \mathcal{D}(A)$ is computed by the following Bellman equation:

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

Alternatively, we may explicitly construct the induced DTMC $(S, s_\iota, P_\pi, R_\pi)$ from Definition 2, whose set of states coincides with that of the MDP (and is thus finite) as the policy π is stationary.

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

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

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

Modifications towards other objectives. Many other objectives, such as reachability and discounted reward, can be solved by straightforward modifications to the Bellman equation. For maximizing the reachability probability of a target set $T \subseteq S$, the reward function is removed, and the preprocessing step is changed to set $Q(s,a) = 1$ for all $(s,a) \in T \times A$, and $Q(s,a) = 0$ for all $(s,a) \in S^\infty \times A$. For discounted reward, the preprocessing is removed altogether, and all state-action pairs are initialized with $Q(s,a) = 0$. The Bellman equation from Equation (1) is modified for both cases, respectively:

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

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

These modifications can also be directly applied to the state-value function V from Equation (1).

Variations and other methods. Several variations to value iteration have been introduced to resolve issues with accuracy and convergence, such as *bounded value iteration* [18, 51], *interval iteration* [15] *optimistic value iteration* [54] and *sound value iteration* [101]. As an alternative to dynamic programming, MDPs can also be naturally encoded as a linear optimization problem which can be solved in polynomial time for many objectives (among which: reach-reward, discounted reward, reachability) [14]. An extensive experimental evaluation comparing several methods for solving MDPs can be found in [53].

3 Theory of Robust Markov Decision Processes

Having briefly recapped the basics of MDPs and dynamic programming, we now move to *robust* MDPs. In the following, let X be a set of variables. An *uncertainty set* $\mathcal{U}$ is a non-empty set of variable assignments subject to some constraints and is defined as $\mathcal{U} = \{f \colon X \to \mathbb{R} \mid \text{constraints on } f\}$.

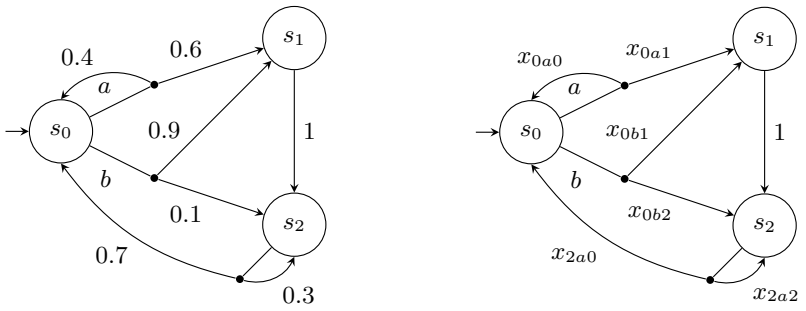
Definition 3 (RMDP). *A robust Markov decision process (RMDP) is a tuple $(S, s_\iota, A, \mathcal{P}, R)$, where the states S, initial state s_ι, actions A and reward function R are defined as in standard MDPs, and $\mathcal{P} \colon \mathcal{U} \to (S \times A \rightharpoonup \mathcal{D}(S))$ is the uncertain transition function.*

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

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

Example 1. Figure 1 depicts an MDP and an RMDP. Below are three possible uncertainty sets for this RMDP:

$$\begin{aligned} \mathcal{U}_1 &= \{x_{0a1} \in [0.1, 0.9] \land x_{0b1} \in [0.1, 0.9] \land x_{2a0} \in [0.1, 0.9]\}\,, \\ \mathcal{U}_2 &= \{x_{0a1} \in [0.1, 0.4] \land x_{0b1} = 2x_{0a1} \land x_{2a0} \in [0.1, 0.9]\}\,, \\ \mathcal{U}_3 &= \{x_{0a1} \in [0.1, 0.4] \land x_{0b1} = 2x_{0a1} \land x_{2a0} = x_{0a1}\}\,. \end{aligned} \tag{2}$$



(a) An MDP. (b) An RMDP.

Fig. 1: An MDP and RMDP for Example 1.

The agent can choose between action a and b in state s_0 and has singleton choices in the other states. An adversary, *nature*, chooses variable assignments for $x_{0a0}, x_{0a1}, x_{0b1}, x_{0b2}, x_{2a0}$, and x_{2a2}. As mentioned above, the restriction that each state-action pair is assigned a valid probability distribution is implied by the definition of the uncertain transition function $\mathcal{P}$. We can therefore focus purely on the choices of x_{0a0}, x_{0b1}, and x_{2a0}.

The uncertainty sets give restrictions on the possible variable assignments. In uncertainty set $\mathcal{U}_1$, variables x_{0a0}, x_{0b1}, and x_{2a0} can each be given any value in the interval $[0.1, 0.9]$. A possible variable assignment in $\mathcal{U}_1$ is $f^1 = \{x_{0a0} \mapsto 0.3, x_{0b1} \mapsto 0.1, x_{2a0} \mapsto 0.8\}$. This variable assignment is not possible in uncertainty sets $\mathcal{U}_2$ and $\mathcal{U}_3$ because of the dependencies between the variables. For example, in uncertainty set $\mathcal{U}_2$, variable x_{0b1} must be assigned twice the value of x_{0a0}, whose value must now be in the interval $[0.1, 0.4]$. A possible variable assignment in $\mathcal{U}_2$ is $f^2 = \{x_{0a0} \mapsto 0.3, x_{0b1} \mapsto 0.6, x_{2a0} \mapsto 0.8\}$. We further discuss the effect of dependencies in the uncertainty set in Section 3.1.

Objectives. For ease of presentation, we again focus on reach-reward maximization as an objective. However, as there is no single transition function, the goal is now to compute an optimal *robust* policy, meaning optimal against the worst-case probabilities in the model. What this worst-case exactly is, depends on the semantics of the RMDP.

3.1 RMDP Semantics and Structural Assumptions

RMDPs can be seen as a game between the *agent*, who aims to maximize their reward by selecting actions, and an adversarial *nature*, who aims to minimize the agent's reward by selecting variable assignments from the uncertainty set. Nature hence simulates the worst-case transition function that the agent should be robust against. This game interpretation can be fully formalized into a zero-sum stochastic game (SG), as we shall discuss further in Section 4.2.

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

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

Dependencies between the variables, or lack thereof, immediately follow from the constraints used to define the uncertainty set $\mathcal{U}$. *Independence* between states or state-action pairs is commonly referred to as *rectangularity*. Informally, an uncertainty set $\mathcal{U}$ is state-action or (s,a)-rectangular if there are no dependencies between the constraints on the variables at different state-action pairs, and state or s-rectangular if there are no dependencies between constraints on the variables at different states. More formally, following standard notation [118]:

Definition 4 (Rectangularity). *The uncertainty set $\mathcal{U}$ is* (s,a)-rectangular *if it can be split into lower dimensional uncertainty sets $\mathcal{U}_{(s,a)}$ that only relate to the variables at the respective state-action pair (s,a), such that their product forms the whole uncertainty set: $\mathcal{U} = \bigtimes_{(s,a)\in S\times A} \mathcal{U}_{(s,a)}$. Similarly, an uncertainty set $\mathcal{U}$ is* s-rectangular *if $\mathcal{U}$ can be split into lower dimensional uncertainty sets $\mathcal{U}_{(s)}$ that only relate to variables at state s, such that $\mathcal{U} = \bigtimes_{s\in S} \mathcal{U}_{(s)}$.*

Example 2. We revisit the RMDP in Figure 1b and the three possible uncertainty sets in Equation (2). The set $\mathcal{U}_1$ is an (s,a)-rectangular uncertainty set, as each variable influences the transition probabilities in only one state-action pair. In other words, there are no dependencies between constraints on the variables at different state-action pairs. In $\mathcal{U}_2$ the transition probabilities for state-action pairs $\langle s_0, a\rangle$ and $\langle s_0, b\rangle$ both depend on variable x_{0a1}. Therefore, $\mathcal{U}_2$ no longer has independence between actions but is still s-rectangular. The final uncertainty set, $\mathcal{U}_3$, has dependencies between all variables and is, therefore, *non-rectangular*.

The type of rectangularity has, together with whether the uncertainty set is convex or not, direct consequences for the computational complexity of policy evaluation, *i.e.*, computing the value for a given policy, and the type of policy that is sufficient to be optimal for discounted reward objectives under static uncertainty semantics. These results are due to Wiesemann et al. [118] and presented in Table 1.

Under s-rectangularity, an additional assumption is made that nature can no longer observe the last action of the agent. This assumption is mentioned explicitly in [58, 59, 118] but often left implicit. Note that this assumption on the

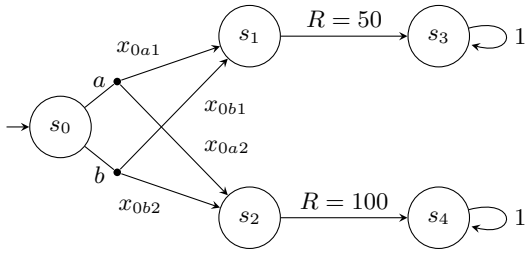


Fig. 2: An RMDP for Example 3.

last-action observability does influence the optimal policy and value, as demonstrated by Example 3. The question of last-action observability corresponds to the difference between *agent first* and *nature first* semantics in [17].

Example 3 (Last-action observability). Figure 2 depicts an RMDP. Below is an s-rectangular uncertainty set:

$$\mathcal{U} = \{x_{0a1} \in [0.1, 0.9] \wedge x_{0a1} = x_{0b2}\}\,.$$

Whether or not nature observed the agent's last action determines whether or not nature has to take the dependency between x_{0a1} and x_{0b2} into account. If nature observes the agent's last action, it can achieve an expected reward of 55 by choosing the maximal $x_{0a1} = x_{0b2} = 0.9$ when observing action a, and by choosing the minimal $x_{0a1} = x_{0b2} = 0.1$ when observing action b. If nature does not have this information, it has to account for both possible agent actions. The best course of action for nature is then to choose $x_{0a1} = x_{0b2} = 0.5$, leading to an expected reward of 75 regardless of the agent's choice.

Static and dynamic uncertainty semantics. The second point about RMDP semantics is whether nature's previous choice at a certain state-action pair should restrict its possible future choices. To that end, Iyengar [64] introduced the notions of *static* and *dynamic* uncertainty semantics. Static uncertainty semantics require nature to play a 'once-for-all' policy: if the state-action pair is revisited, nature is required to use the same variable assignment from the uncertainty set as before. In contrast, under dynamic uncertainty semantics nature plays 'memoryless' and is free to choose any variable assignment at every step. Simultaneous but independently, Nilim and El Ghaoui introduced these semantics as *time-stationary* and *time-varying* uncertainty models [93]. Note that these notions have only been introduced for (s, a)-rectangular RMDPs and are only of concern in cyclic, infinite horizon models. Interestingly, Iyengar [64] also shows that the distinction between static and dynamic uncertainty does not matter for reward maximization in (s, a)-rectangular RMDPs. A similar result was established for reachability in interval Markov chains in [26]. We state the result in general in the following lemma.

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

Table 1: Policy classes that are sufficient and computational complexity of policy evaluation for discounted reward RMDPs with various types of uncertainty sets, as identified by Wiesemann et al. [118] assuming *static* uncertainty semantics.

Lemma 1 (Static and dynamic uncertainty coincide [64]). *Consider an (s,a)-rectangular RMDP where both agent and nature are restricted to stationary policies,* i.e.*, policies of type $\pi\colon S \to \mathcal{D}(A)$. Let π^{st} be the optimal policy under static uncertainty, and π^{dy} be the optimal policy under dynamic uncertainty semantics. The robust values of these policies coincide,* i.e.*, $V_{\pi^{st}} = V_{\pi^{dy}}$.*

3.2 Robust Dynamic Programming

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

Remark 1 (Graph preservation). For computational tractability of robust dynamic programming, especially of objectives that rely on preprocessing the underlying graph, such as the reach-reward objective we consider, it is often assumed that the uncertainty set should be *graph preserving*. That is, all variable assignments in the uncertainty set $\mathcal{U}$ imply the same topology for the underlying graphs. Hence, if there exists some $P \in \mathcal{P}$ with $P(s,a,s') = 0$ for some transition, then all other $P' \in \mathcal{P}$ should also have $P'(s,a,s') = 0$.

Recall Equation (1), describing value iteration in a standard MDP. In an RMDP, we do not have access to a precisely defined transition function $P\colon S \times A \rightharpoonup \mathcal{D}(S)$. Instead, we have the uncertain transition function $\mathcal{P}$ that defines a set of such transition functions $P \in \mathcal{P}$.

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

$$\underline{V}^{(n+1)}(s) = \max_{a \in A} \left\{ R(s,a) + \inf_{P \in \mathcal{P}} \left\{ \sum_{s' \in S} P(s,a,s') \underline{V}^{(n)}(s') \right\} \right\}. \tag{3}$$

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

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

$$\underline{V}^{(n+1)}(s) = \max_{a\in A}\left\{R(s,a) + \inf_{P(s,a)\in\mathcal{P}(s,a)}\left\{\sum_{s'\in S} P(s,a,s')\underline{V}^{(n)}(s')\right\}\right\}. \quad (4)$$

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

As discussed in Section 3.1 and shown in Table 1, stationary deterministic policies are sufficient for optimality in (s,a)-rectangular RMDPs with convex uncertainty sets. Thus, an optimal *robust policy* $\underline{\pi}^*$ can again be extracted via

$$\underline{\pi}^*(s) = \operatorname*{argmax}_{a\in A}\left\{R(s,a) + \inf_{P(s,a)\in\mathcal{P}(s,a)}\left\{\sum_{s'\in S} P(s,a,s')\underline{V}^*(s')\right\}\right\}. \quad (5)$$

We underline the policy $\underline{\pi}^*$ to denote that it is an optimal *robust* policy, as we later also touch upon optimal *optimistic* policies, which we shall denote by $\overline{\pi}^*$.

Robust policy iteration [64, 76] extends standard policy iteration in a similar way. Policy evaluation, *i.e.*, model checking the induced *robust* Markov chain, is done by performing robust dynamic programming for some stationary policy π:

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

Here, we again use that our uncertainty set is (s,a)-rectangular and convex to ensure an efficiently solvable inner minimization problem. After convergence, we use the robust state values under the current policy $\underline{V}_\pi^*$ to compute the robust state-action values $\underline{Q}_\pi$:

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

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

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

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

Optimistic dynamic programming. Instead of assuming the worst-case from the uncertainty set, we may also assume the agent and nature play cooperatively. That is, both players attempt to maximize the agent's reward, and we instead obtain *optimistic* values $\overline{V}$ that are computed in the same way as the pessimistic values were, except that the inner minimization problem from Equation (3) is now replaced by an inner *maximization* problem:

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

The optimal *optimistic* policy $\overline{\pi}^*$ is again extracted by one final step of dynamic programming, as in Equation (5):

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

Convex optimization. Since dynamic programming approaches for MDPs extend with relative ease to RMDPs, especially in the case of (s,a)-rectangular uncertainty sets, a natural question to ask is whether the same goes for convex optimization approaches, and in particular the linear programming (LP) formulation for MDPs. As Iyengar [64] already notes, however, that is not the case, and the natural analogue of the LP of MDPs for RMDPs yields, in fact, a nonconvex optimization problem. In contrast, the optimistic setting does yield tractable LPs via standard dualization techniques, which have been applied to solve PCTL objectives in (s,a)-rectangular RMDPs with convex uncertainty sets [98].

Methods for *s*-rectangular RMDPs. For s-rectangular RMDPs, dynamic programming does not extend so straightforwardly, and a lot of research has focused on finding efficient Bellman operators for various types of uncertainty sets. Most notably, s-rectangular L_1-MDPs [58], but also s-rectangular uncertainty sets defined by an L_∞-norm [16] or ϕ-divergences [60]. In [59], a policy iteration algorithm was introduced, while [42, 77, 115] employ policy gradient techniques.

Other objectives. The RMDP literature primarily focuses on either finite horizon or discounted infinite horizon reward maximization. Adaptation to reachability objectives, such as the reach-reward maximization we consider, is usually straightforward, provided the graph preservation property of Remark 1 is met. Temporal logic objectives can be reduced to such reach-reward objectives via a product construction [119]. Finally, recent works study average reward (also known as mean payoff) and Blackwell optimality in RMDPs [25, 48, 49]. Average reward objectives consider the problem of maximizing the average reward collected in t time steps when $\lim_{t \to \infty}$, and Blackwell optimality balances the standard discounted reward objective by also accounting for long-term reward. A policy is Blackwell optimal if it is optimal for all discount factors sufficiently close to one, *i.e.*, all $\gamma \in [\gamma^*, 1)$ [99].

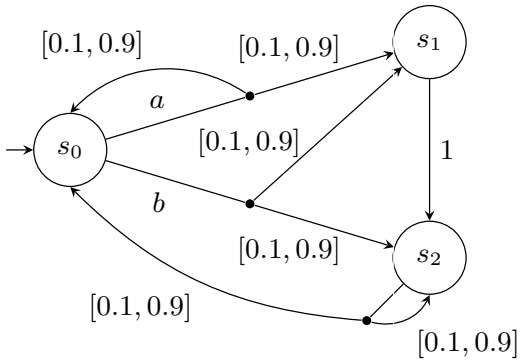


Fig. 3: An example IMDP.

3.3 Well-Known RMDP Instances

We review common types of RMDPs often used in formal verification and AI, namely interval MDPs, L_1-MDPs, and multi-environment MDPs. We provide a more commonly used tuple definition for each and explain how it fits the general RMDP framework.

Interval MDPs

Interval MDPs (IMDPs) [93], also referred to as bounded-parameter MDPs [46] or uncertain MDPs [111, 119], are a special instance of (s, a)-rectangular RMDPs.

Definition 5 (IMDP). *An interval MDP (IMDP) is a tuple* $(S, s_\iota, A, \check{P}, \hat{P}, R)$*, where* $\check{P} \colon S \times A \times S \rightharpoonup [0, 1]$ *and* $\hat{P} \colon S \times A \times S \rightharpoonup [0, 1]$ *are two transition functions that assign lower and upper bounds to each transition, respectively, such that* $\check{P} \leq \hat{P}$ *and for all transitions* $\check{P}(s, a, s') = 0 \iff \hat{P}(s, a, s') = 0$*.*

Our definition of an IMDP requires that a transition either does not exist (where both $\check{P}$ and $\hat{P}$ are zero) or is assigned an interval with a non-zero lower bound, thus ensuring graph preservation for tractable (standard) robust dynamic programming (Remark 1). For IMDPs, however, the statistical model checking literature offers solutions to circumvent the need for this requirement [6, 35].

IMDPs have a constraint that each state-action pair is required to have a valid probability distribution. An IMDP is an RMDP $(S, s_\iota, A, \mathcal{P}, R)$ where the uncertainty set is of the form $\mathcal{U} = \{f \colon X \to \mathbb{R} \mid \forall (s, a, s') \in S \times A \times S,\ f(x_{sas'}) \in [i, j]_{sas'} \subseteq [0, 1] \land \forall (s, a) \in S \times A,\ \sum_{s' \in S} f(x_{sas'}) = 1\}$. An example IMDP is depicted in Figure 3. Note that this IMDP is precisely the RMDP from Figure 1 with uncertainty set $\mathcal{U}^1$, see Example 1.

IMDPs have the nice property that their inner problem can be solved efficiently via a bisection algorithm [93], more explicitly given for interval DTMCs in [74] and presented in Algorithm 1. This algorithm sorts the successor states

Algorithm 1 Algorithm to solve the IMDP inner problem $\inf_{P \in \mathcal{P}(s,a)}$

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

$S' = \{s'_1, \ldots, s'_m\}$ of state-action pair (s,a) by the current value $V^{(n)}$ in ascending order. A variable *budget* indicates how much probability mass is still free to assign when we start with assigning the lower bounds to each successor state. Successor states occurring at low indices, *i.e.*, with low values $V^{(n)}$, will be assigned the upper bound of the transition leading to them until the budget runs out. One state will get a remaining *budget* added to its lower bound, which is the first state for which it is no longer possible to replace the lower bound by the upper bound, ensuring the transition probability lies within its interval. The remaining successor states, with high values, will be assigned the lower bounds, as the budget for replacement is now zero. As a result, transition function $P(s,a,\cdot)$ forms a valid probability distribution.

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

L_1-MDPs

L_1-MDPs [109] are another instance of (s,a)-rectangular RMDPs. Where IMDPs put an error margin around each individual transition probability, L_1-MDPs put an error margin around each probability distribution at a state-action pair.

Definition 6 (L_1-MDP). *An L_1-MDP is a tuple $(S, s_\iota, A, \tilde{P}, R, d)$, where S, $s_\iota \in S$, A and R are as for standard MDPs, $\tilde{P}\colon S \times A \rightharpoonup \mathcal{D}(S)$ is the centre transition function and $d\colon S \times A \to \mathbb{R}_{\geq 0}$ is a distance function assigning an error bound to each state-action pair.*

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

Similar to IMDPs, the inner optimization problem for L_1-MDPs can be solved efficiently, again by ordering the successor states along their current value and

Algorithm 2 Algorithm to solve the L_1-MDP inner problem $\inf_{P \in \mathcal{P}(s,a)}$

Sort $S' = \{s'_1, \ldots, s'_m\}$ according to $V^{(n)}$ ascending such that $V^{(n)}(s'_i) \leq V^{(n)}(s'_{i+1})$
$P(s, a, s'_m) \leftarrow \max\{0, \tilde{P}(s, a, s'_m) - {}^{d(s,a)}/_2\}$
$\forall s'_i \neq s'_m \in S'$: $P(s, a, s'_i) \leftarrow \tilde{P}(s, a, s'_i)$
$i \leftarrow 1$
while $\sum_{j=1}^{m} P(s, a, s'_j) < 1$ **do**
    $P(s, a, s'_i) \leftarrow \min\{1, 1 + \sum_{j \in \{1,\ldots,m\} \setminus \{i\}} P(s, a, s'_j)\}$
    $i \leftarrow i + 1$
end while
return $P(s, a, \cdot)$

assigning low-ranking states the most possible probability mass and high-ranking states the least probability mass. Specifically, the state with the highest value s'_m gets probability mass ${}^{d(s,a)}/_2$ subtracted from its estimate $\tilde{P}(s, a, s'_m)$, and the remaining states from low to high get added probability mass until a total of ${}^{d(s,a)}/_2$ has been added to these states, ensuring a valid probability distribution. This algorithm is the dual of the algorithm for computing the optimistic inner problem $\sup_{P \in \mathcal{P}(s,a)}$ of [109] and explicitly given in Algorithm 2.

Multi-Environment MDPs

Multi-environment MDPs (MEMDPs) [102] model *discrete* uncertainty. Specifically, a MEMDP is a finite set of MDPs that share the same states, actions, and reward function and only differ in their transition functions. Each MDP in a MEMDP is called an *environment*.

Definition 7 (MEMDP). *A multi-environment MDP (MEMDP) is a tuple* $(S, s_\iota, A, \{P_i\}_{i \in I}, R)$ *where* S, s_ι, A, R *are as for MDPs, and* $\{P_i \colon S \times A \rightharpoonup \mathcal{D}(S)\}_{i \in I}$ *is a set of* $I = \{1, \ldots, n\}$ *transition functions that are consistent with each other in terms of enabled actions:* $\forall (s, a) \in S \times A, \forall i, j \in I,\ P_i(s, a) = \bot \iff P_j(s, a) = \bot$.

A MEMDP is an RMDP $(S, s_\iota, A, \mathcal{P}, R)$ where the uncertainty set $\mathcal{U}$ is discrete: $|\mathcal{U}| \neq \infty$, and $\mathcal{P} = \{P_i\}_{i \in I}$. In general, MEMDPs are *non-rectangular* and follow *static* uncertainty semantics, as nature's choices at each state-action pair must be consistent with, and equivalent to, choosing a single $P_i \in \mathcal{P}$ at the start. As the uncertainty set is discrete, it is also nonconvex. Hence, optimal policies in MEMDPs need to be history-based and randomized; see Table 1.

MEMDPs have been studied in both formal methods and AI. In AI, MEMDPs have caught interest because of their applications in robotics, naturally modelling several possible worlds a robot may act in [104]. Besides being RMDPs, MEMDPs are also a subclass of *partially observable* MDPs (POMDPs), where the agent does not directly observe the states [71]. In particular, a MEMDP can be transformed into a POMDP by taking the disjoint union of all environments and using the partial observability to hide in which environment the agent is

playing [23]. As a result, quantitative objectives such as reward maximization may be solved by casting the MEMDP to a POMDP and using off-the-shelf POMDP methods.

In formal methods, emphasis has been given to complexity results, especially for *almost-sure* objectives, *i.e.*, objectives that need to be satisfied with probability one. In [102], it is shown that almost-sure parity objectives are in P for MEMDPs of two environments, while [114] shows that already for almost-sure reachability, an arbitrary number of environments leads to PSPACE-completeness. Recent work completes the complexity landscape for qualitative objectives in MEMDPs by establishing PSPACE-completeness for almost-sure parity and Rabin objectives [112]. In contrast, almost-sure reachability is already EXPTIME-complete for POMDPs [24], and almost-sure parity or Rabin objectives are undecidable [12], showing that MEMDPs are an interesting class worth investigating.

4 Connections to Other Models

In the following, we summarize the connections between RMDPs and some other commonly used models in formal methods and AI. In particular, we highlight the connections with parametric MDPs (Section 4.1), stochastic games (Section 4.2), robust POMDPs (Section 4.3), and a range of models that assume additional distributional information over the parameters (Section 4.4).

4.1 Parametric MDPs

Our general definition of RMDPs as given in Definition 3 closely resembles that of *parametric* MDPs (pMDPs) [67]. Indeed, both models assign variables (parameters) to the transitions instead of concrete probabilities, effectively defining a set of possible MDP models. Typically, pMDPs are defined more generally and allow for a rational function over two polynomials on the transitions, encoding dependencies between transitions directly by having two of these rationals share some of the parameters.

The *parameter synthesis* problem [37] typically considered in pMDPs is, however, different from the problem of computing robust policies and values in RMDPs. Parameter synthesis asks whether there exists a variable assignment such that *for all* policies, a certain (reachability) specification is met. That is, the quantifiers are reversed compared to RMDPs.

Common techniques for parameter synthesis in pMDPs or pMCs include convex optimization approaches [30, 31, 33], parameter lifting [100], or exact computation of the solution function [69]. Tool support for pMDPs can be found in, *e.g.*, Storm [57] or PROPhESY [36]. Parameter synthesis in pMDPs with memoryless deterministic policies is known to be NP-complete for a fixed number of parameters and ETR-complete for arbitrary numbers of parameters [70].

4.2 Stochastic Games

The connection between *stochastic games* (SG) and RMDPs has been noted many times in the RMDP literature. See, *e.g.*, [47, 49, 64, 93, 118, 121]. The most explicit game interpretations are given by [64, 93], which both link reward maximization in (s, a)-rectangular RMDPs to turn-based zero-sum stochastic games. This equivalent game is constructed by adding, in addition to the states S, states that correspond with tuples $\langle s, a\rangle$ of states $s \in S$ and actions $a \in A$. Then, the agent controls the original states, and nature controls the tuple states. In a state s, the agent chooses an action a, upon which the game transitions deterministically to the nature state $\langle s, a\rangle$. In this state $\langle s, a\rangle$, nature chooses a variable assignment. Given a nature state $\langle s, a\rangle$ and variable assignment $f \in \mathcal{U}$, the game transitions stochastically to an agent state s' according to $\mathcal{P}(f)(s, a, s')$. The reward function assigns the same value as the reward function of the RMDP in the agent states and zero in nature states.

Changes in the assumptions on nature or the objective require some changes in the translation to stochastic games. Paper [118] mentions that a similar construction follows for s-rectangular RMDPs, but does not describe the actual game. As s-rectangular RMDPs assume that nature chooses variable assignments without information of the agent's latest action, such games would have to either be partially observable or concurrent. An average reward objective in RMDPs can be linked to zero-sum mean pay-off games [49].

A key difference between RMDPs and SGs is that in RMDPs, it is typically assumed that nature plays memoryless, whereas in SGs, both players are allowed to use history. [49] shows that the assumption of playing against a memoryless nature is nonrestrictive for discounted reward maximization against a convex and compact s-rectangular uncertainty set.

4.3 Robust POMDPs

Partially observable MDPs (POMDPs) are an extension of MDPs where it is assumed that the agent cannot directly observe the state. Instead, the agent receives observations about the state and, optionally, the last action [71]. The same extension to the partially observable setting can be made for robust MDPs, resulting in *robust POMDPs* (RPOMDPs).

Computing optimal policies for POMDPs with infinite horizon objectives is undecidable [87]. Since standard POMDPs are trivially included in RPOMDPs, all decision problems for RPOMDPs are at least as hard as for POMDPs. Existing approaches for policy computation in RPOMDPs use convex optimization techniques [34, 110], robust versions of value iteration [94], or recurrent neural networks to learn policies [43]. Other approaches exist but either consider a different notion of optimal policy, such as optimal for one instance in the uncertainty set [63] or optimal given a pessimism level [106], or have additional assumptions on the uncertainty set, such as uncertainty in the observation function only [22] or existence of a distribution over the uncertainty set [92]. It is shown by [17] that an RPOMDP with an uncertain state-action-based observation function can be

transformed to an RPOMDP with a deterministic state-based observation function using a state space expansion. This transformation allows research to focus on RPOMDPs with uncertainty only in the transition function (and not in the observation function).

Paper [17] defines formal game semantics for RPOMDPs, linking RPOMDPs to turn-based zero-sum partially observable stochastic games (POSGs) [38]. They note that the existing literature on RPOMDPs makes implicit assumptions about uncertainty, which leads to semantically different POSGs and, hence, RPOMDPs with different optimal values. In particular, [17] shows that static and dynamic uncertainty semantics in RPOMDPs no longer coincide, even for (s, a)-rectangular uncertainty sets. For infinite horizon discounted reward maximization, [94] shows that static and dynamic uncertainty in RPOMDPs do still coincide under (s, a)-rectangularity and when nature plays stationary, *i.e.*, without memory. Finally, [17] shows that nature's ability to observe the agent's last-played action influences the optimal value in both RPOMDPs and RMDPs, see also Section 3.1.

4.4 Modelling Likelihoods of Transition Functions

Thus far, we have discussed models that capture *sets* of possible transition functions and the associated game behaviour of these models. However, what if certain transition functions are more likely than others? These likelihoods may result from different experts which value they would assume for, *e.g.*, transition probabilities in a given MDP [3]. The natural question is how we can incorporate this *prior knowledge* about the likelihood of different transition functions. One option is to neglect the likelihood completely and model the problem as an RMDP. However, the standard robust analysis of this RMDP (as discussed in section 3) may lead to overly conservative results. In this section, we explore approaches that aim to mitigate this conservatism and rigorously incorporate likelihoods over transition functions.

We can formalize the setting above by modelling the prior knowledge as a probability distribution over the transition function $\mathcal{P}$ of an RMDP. Although equivalent, it is often more convenient to model this setting as a pMDP together with a distribution over the parameter values. As such, these models have been named *uncertain parametric MDPs* (upMDPs) in the literature [7]. A common verification question is then to obtain a solution *"that is robust against (for example) at least a* 99% *probability mass of the distribution."* As a concrete application, consider flying a drone in an environment with uncertain weather conditions. We model this environment as a parametric MDP, where the parameter values are determined by the actual weather conditions. From historical weather data, we can derive a probability distribution for the weather conditions (and thus the transition probabilities) on a random day. A natural verification question is then: *"What is the probability that we can safely fly the drone without crashing on a random day?"*

One important question for upMDPs is whether policies can depend on the realization of the (uncertain) parameters. For the example above, one possible

assumption is that we can *first* observe the actual weather conditions, and *then* can compute an optimal policy based on this weather. This setting has first been investigated by [32] and in more detail by [7]. The other possible assumption is that we *cannot* observe the actual weather first and instead need to compute a *single* policy that is robust against all weather conditions. This setting has been considered by [103] for upMDPs, and also relates to so-called Bayes-adaptive MDPs which are commonly used in reinforcement learning [50, 29, 105]. A variant where parameter values cannot be observed and thus must be learned has been studied by [5]. The latter setting is arguably more difficult to solve due to dependencies between the policies for different weather conditions. However, which of the two assumptions is more appropriate depends on the context.

In practice, it may be unrealistic to have access to an explicit representation of the distributions over transition functions. For example, we may have prior knowledge in the form of a finite set of expert demonstrations [97], each of which leads to an MDP with different transition probabilities. These demonstrations can be interpreted as *samples* from an underlying distribution over the parameters. This setting has motivated sampling-based verification approaches for upMDPs, such as [32, 103]. A similar setting for continuous-time Markov chains is studied by [8, 103]. Generally, these approaches assume access to a finite set of parameter samples and aim to compute a solution with statistical (PAC-style) guarantees on its performance on yet another sample from the underlying distribution. For example, [7] obtains PAC guarantees by techniques from *scenario optimization*, which is a methodology to deal with stochastic convex optimization in a data-driven fashion [19, 20].

5 Robust MDPs in Practice: Applications and Tools

In this section, we review two key applications of RMDPs, namely in *learning* and *abstraction* methods, and discuss the current state of tool support.

5.1 Learning

A natural application of RMDPs in both formal methods and AI comes in learning MDPs from data. Naive estimation of the transition probabilities from a finite amount of observations introduces estimation errors. When following a path through a learned MDP, these errors may accumulate, leading to potentially significant differences in values (and possibly optimal policies) between the learned model and the true underlying MDP [47, 88]. To account for these errors, confidence intervals around the probabilities or distributions may be computed via, *e.g.*, Hoeffding's inequality [61] or the Weissman bound [116], and included in the learned model, yielding an RMDP. Resulting policies and values can then be given a *probably approximately correct* (PAC) guarantee.

Learning MDPs with PAC guarantees has been studied extensively in both formal methods and AI, namely in the form of statistical model checking (SMC) and reinforcement learning (RL). PAC-SMC is often applied when the original

MDP is too large to fit into memory and can mitigate the state-explosion problem (at the cost of precision) when verifying infinite or indefinite horizon objectives. These methods either build IMDPs by deriving confidence intervals through Hoeffding's inequality or construct lower and upper Bellman equations that are updated directly, implicitly performing robust dynamic programming [6, 35].

In contrast, RL is primarily concerned with finding an optimal policy that maximizes discounted or finite horizon reward objectives through efficient exploration [113]. RMDPs, and specifically L_1-MDPs based on the aforementioned Weissman bound, are used to achieve PAC guarantees on the learned model [108, 109] or efficient exploration through optimistic policies [66].

It should be noted that the PAC-MDP framework of [108] explicitly requires the sample efficiency of a learning algorithm to be polynomial in the input to be considered (efficiently) PAC. As a consequence, any non-finitary objective is not PAC-learnable following the PAC-MDP framework, and PAC-SMC methods are said to give anytime or best-effort guarantees [122]. Recent work investigates techniques to reduce the amount of data required to achieve PAC guarantees in both SMC and RL [90, 117].

Two subfields of RL that also commonly use RMDPs are the offline RL problem of *safe policy improvement* (SPI) and *robust RL*. In SPI, only a previously collected data set and the *behaviour* policy that collected it are given, and no further data collection is allowed. The SPI problem is to compute a new policy that outperforms the behaviour policy with a PAC-style guarantee. Solutions to this problem often construct (implicit) L_1-MDPs [44, 82, 107]. Robust RL is a broad field that considers RL under various kinds of uncertainty, disturbances, or structural changes perturbations, such as non-stationary environments [111]. We refer to the following recent survey for more on robust RL [91].

Aleatoric and epistemic uncertainty. Uncertainty that may be reduced by collecting more data, as encountered in these learning settings, is commonly referred to as *epistemic uncertainty* [11, 62]. Uncertainty that cannot be reduced but is known to be inherent to the system, such as the probability distributions in a standard MDP, is called *aleatoric uncertainty*. We emphasize that an RMDP's uncertainty set is not necessarily epistemic and that whether the uncertainty may be reduced by collecting more data is an additional assumption about the specific scenario in which the RMDP is used.

5.2 Abstraction

RMDPs are commonly used to model abstractions of more complex systems. The general idea is that states of a (non-robust) MDP can be aggregated by overapproximating the transition probabilities in the uncertainty set of an RMDP [65]. This idea at least dates back to [83] and has already been identified as an interesting application of RMDPs in [46]. Since then, such abstraction techniques have been used across areas, including formal methods, control theory, and AI.

First of all, game-based abstraction of MDPs in the form of IMDPs has been studied by [75, 78]. So-called 3-valued abstractions of Markov chains are

developed by [39], who abstract the system into an interval Markov chain whose labelling function has three possible values (true, false, or don't know). A similar approach for 3-valued abstraction of CTMCs is presented by [73]. Probabilistic bisimulation of IMDPs to reduce the number of states is considered by [56].

In control theory, models typically have continuous state and action spaces. A popular approach to synthesizing provably correct control policies is to generate a finite-state abstraction of the continuous model [1, 2, 81]. Under an appropriate simulation relation, satisfaction guarantees of temporal logic formulae carry over from the abstract to the continuous model [45]. Various papers generate IMDP abstractions of stochastic systems [9, 10, 28, 85], and tool support has been developed by, *e.g.*, [21, 120]. Similar to game-based abstraction, the general idea is to use the probability intervals to capture uncertainties and abstraction errors. For further details on abstractions in control, we refer to the survey [84].

5.3 Tool Support

General-purpose tool support for RMDPs is still relatively limited compared to other models. The probabilistic model checkers PRISM [80] and STORM [57] both support basic IMDP model checking, with PRISM's provision slightly more advanced, *e.g.*, in terms of user support for modelling. STORM, on the other hand, has more advanced support for pMDPs, and has been used as a back-end in several of the previously discussed works, *e.g.*, [7, 8, 33, 110]. Other IMDP tools have also begun to be developed, such as the Julia-based INTERVALMDP.JL [89].

6 Robust MDPs in the Future

We conclude this survey with some currently active directions for the development and application of RMDPs.

Tools, Benchmarks, and Evaluation

As mentioned in Section 5.3, tool support for RMDPs is still young, with a particular focus on IMDPs. Extending support to L_1-MDPs, a larger range of objectives including discounted reward and temporal logic objectives, and possibly s-rectangular RMDPs, would give further impulse to the development of new techniques exploiting the theory of RMDPs in, *e.g.*, learning or abstraction.

Alongside tools, we also see a clear need for a rich benchmark set to evaluate and compare RMDP algorithms and tools. Initiatives such as a rich comparison of algorithms [53] or extensive (tool) benchmarking on a standardized set of models [4, 55] could shed light on the differences between various methods in theory and practice.

Notably, and contrary to what has been experimentally verified for MDPs [53], robust policy iteration seems to be the preferred way to solve RMDPs [59, 76]. The argument is that it performs fewer inner minimization problems compared to robust value iteration, which, depending on the shape of the uncertainty set,

may become computationally expensive. An extensive experimental evaluation confirming this hypothesis is, to the best of our knowledge, missing as of yet.

Algorithmic and Theoretical Advances in Multi-Environment MDPs

In Section 3.3, we already identified MEMDPs as an interesting class of RMDPs as they naturally model a finite set of possible MDPs and may also be viewed as POMDPs with additional structure. Complexity results on almost-sure objectives show how this additional structure can be exploited to build computationally more efficient algorithms than their (general) POMDP counterpart [114]. We believe this direction of research should be continued and further investigated for quantitative objectives such as expected reward.

Uncertainty Assumptions in RMDPs

While the type of rectangularity assumptions about the uncertainty set are often made explicit, other assumptions, such as static and dynamic uncertainty and the type of nature policies considered, are mostly left implicit. In various cases, it is known that these assumptions do not influence the optimal policies and values and can, therefore, be ignored [49, 64], as also discussed in section Sections 3 and 4. However, many combinations of assumptions are yet to be investigated. [17] shows cases where assumptions on the uncertainty set influence the optimal policy and value for reward maximization in RPOMDPs. Therefore, richer (temporal logic) objectives for RMDPs may likely be subject to a similar influence. We believe expanding the knowledge of whether and when assumptions on the RMDP influence the optimality of policies and values is an interesting and necessary direction for future research.

7 Conclusion

We presented a short survey on robust MDPs, from the basic foundations, including semantics and solution methods, to common applications in formal methods and AI. We discussed RMDP semantics and how dynamic programming can be extended to *robust* dynamic programming in the (s, a)-rectangular case. Finally, we summarized the current and (possibly) future position RMDPs take within the formal methods and AI communities in terms of connections with other models and applications.

Acknowledgements

This work was supported by the ERC Starting Grant 101077178 (DEUCE) and the European Union's Horizon 2020 research and innovation programme (FUN2MODEL, grant agreement No. 834115), as well as the NWO grants OCENW.KLEIN.187 and NWA.1160.18.238 (PrimaVera).

References

1. Abate, A., Prandini, M., Lygeros, J., Sastry, S.: Probabilistic reachability and safety for controlled discrete time stochastic hybrid systems. Automatica **44**(11), 2724 – 2734 (2008)
2. Alur, R., Henzinger, T.A., Lafferriere, G., Pappas, G.J.: Discrete abstractions of hybrid systems. Proceedings of the IEEE **88**(7), 971–984 (2000)
3. Andrés, I., de Barros, L.N., Mauá, D.D., Simão, T.D.: When a robot reaches out for human help. In: IBERAMIA. Lecture Notes in Computer Science, vol. 11238, pp. 277–289. Springer (2018)
4. Andriushchenko, R., Bork, A., Budde, C.E., Ceska, M., Grover, K., Hahn, E.M., Hartmanns, A., Israelsen, B., Jansen, N., Jeppson, J., Junges, S., Köhl, M.A., Könighofer, B., Kretínský, J., Meggendorfer, T., Parker, D., Pranger, S., Quatmann, T., Ruijters, E., Taylor, L., Volk, M., Weininger, M., Zhang, Z.: Tools at the frontiers of quantitative verification. CoRR **abs/2405.13583** (2024)
5. Arming, S., Bartocci, E., Chatterjee, K., Katoen, J., Sokolova, A.: Parameter-independent strategies for pmdps via pomdps. In: QEST. Lecture Notes in Computer Science, vol. 11024, pp. 53–70. Springer (2018)
6. Ashok, P., Kretínský, J., Weininger, M.: PAC statistical model checking for markov decision processes and stochastic games. In: CAV (1). Lecture Notes in Computer Science, vol. 11561, pp. 497–519. Springer (2019)
7. Badings, T.S., Cubuktepe, M., Jansen, N., Junges, S., Katoen, J., Topcu, U.: Scenario-based verification of uncertain parametric mdps. Int. J. Softw. Tools Technol. Transf. **24**(5), 803–819 (2022)
8. Badings, T.S., Jansen, N., Junges, S., Stoelinga, M., Volk, M.: Sampling-based verification of ctmcs with uncertain rates. In: CAV (2). Lecture Notes in Computer Science, vol. 13372, pp. 26–47. Springer (2022)
9. Badings, T.S., Romao, L., Abate, A., Jansen, N.: Probabilities are not enough: Formal controller synthesis for stochastic dynamical models with epistemic uncertainty. In: AAAI. pp. 14701–14710. AAAI Press (2023)
10. Badings, T.S., Romao, L., Abate, A., Parker, D., Poonawala, H.A., Stoelinga, M., Jansen, N.: Robust control for dynamical systems with non-gaussian noise via formal abstractions. J. Artif. Intell. Res. **76**, 341–391 (2023)
11. Badings, T.S., Simão, T.D., Suilen, M., Jansen, N.: Decision-making under uncertainty: beyond probabilities. Int. J. Softw. Tools Technol. Transf. **25**(3), 375–391 (2023)
12. Baier, C., Bertrand, N., Größer, M.: On decision problems for probabilistic büchi automata. In: FoSSaCS. Lecture Notes in Computer Science, vol. 4962, pp. 287–301. Springer (2008)
13. Baier, C., Hermanns, H., Katoen, J.: The 10, 000 facets of MDP model checking. In: Computing and Software Science, Lecture Notes in Computer Science, vol. 10000, pp. 420–451. Springer (2019)
14. Baier, C., Katoen, J.: Principles of model checking. MIT Press (2008)
15. Baier, C., Klein, J., Leuschner, L., Parker, D., Wunderlich, S.: Ensuring the reliability of your model checker: Interval iteration for markov decision processes. In: CAV (1). Lecture Notes in Computer Science, vol. 10426, pp. 160–180. Springer (2017)
16. Behzadian, B., Petrik, M., Ho, C.P.: Fast algorithms for l_∞-constrained s-rectangular robust mdps. In: NeurIPS. pp. 25982–25992 (2021)

17. Bovy, E.M., Suilen, M., Junges, S., Jansen, N.: Imprecise probabilities meet partial observability: Game semantics for robust pomdps. CoRR **abs/2405.04941** (2024)
18. Brázdil, T., Chatterjee, K., Chmelik, M., Forejt, V., Kretínský, J., Kwiatkowska, M.Z., Parker, D., Ujma, M.: Verification of markov decision processes using learning algorithms. In: ATVA. Lecture Notes in Computer Science, vol. 8837, pp. 98–114. Springer (2014)
19. Campi, M.C., Carè, A., Garatti, S.: The scenario approach: A tool at the service of data-driven decision making. Annu. Rev. Control. **52**, 1–17 (2021)
20. Campi, M.C., Garatti, S.: The exact feasibility of randomized solutions of uncertain convex programs. SIAM J. Optim. **19**(3), 1211–1230 (2008)
21. Cauchi, N., Abate, A.: Stochy: Automated verification and synthesis of stochastic processes. In: TACAS (2). Lecture Notes in Computer Science, vol. 11428, pp. 247–264. Springer (2019)
22. Chamie, M.E., Mostafa, H.: Robust action selection in partially observable markov decision processes with model uncertainty. In: CDC. pp. 5586–5591. IEEE (2018)
23. Chatterjee, K., Chmelík, M., Karkhanis, D., Novotný, P., Royer, A.: Multiple-environment markov decision processes: Efficient analysis and applications. In: ICAPS. pp. 48–56. AAAI Press (2020)
24. Chatterjee, K., Doyen, L., Henzinger, T.A.: Qualitative analysis of partially-observable markov decision processes. In: MFCS. Lecture Notes in Computer Science, vol. 6281, pp. 258–269. Springer (2010)
25. Chatterjee, K., Goharshady, E.K., Karrabi, M., Novotný, P., Zikelic, D.: Solving long-run average reward robust mdps via stochastic games. CoRR **abs/2312.13912** (2023)
26. Chen, T., Han, T., Kwiatkowska, M.Z.: On the complexity of model checking interval-valued discrete time markov chains. Inf. Process. Lett. **113**(7), 210–216 (2013)
27. Clarke, E.M., Klieber, W., Novácek, M., Zuliani, P.: Model checking and the state explosion problem. In: LASER Summer School. Lecture Notes in Computer Science, vol. 7682, pp. 1–30. Springer (2011)
28. Coppola, R., Peruffo, A., Romao, L., Abate, A., Jr., M.M.: Data-driven interval MDP for robust control synthesis. CoRR **abs/2404.08344** (2024)
29. Costen, C., Rigter, M., Lacerda, B., Hawes, N.: Planning with hidden parameter polynomial mdps. In: AAAI. pp. 11963–11971. AAAI Press (2023)
30. Cubuktepe, M., Jansen, N., Junges, S., Katoen, J., Papusha, I., Poonawala, H.A., Topcu, U.: Sequential convex programming for the efficient verification of parametric mdps. In: TACAS (2). Lecture Notes in Computer Science, vol. 10206, pp. 133–150 (2017)
31. Cubuktepe, M., Jansen, N., Junges, S., Katoen, J., Topcu, U.: Synthesis in pmdps: A tale of 1001 parameters. In: ATVA. Lecture Notes in Computer Science, vol. 11138, pp. 160–176. Springer (2018)
32. Cubuktepe, M., Jansen, N., Junges, S., Katoen, J., Topcu, U.: Scenario-based verification of uncertain mdps. In: TACAS (1). Lecture Notes in Computer Science, vol. 12078, pp. 287–305. Springer (2020)
33. Cubuktepe, M., Jansen, N., Junges, S., Katoen, J., Topcu, U.: Convex optimization for parameter synthesis in mdps. IEEE Trans. Autom. Control. **67**(12), 6333–6348 (2022)
34. Cubuktepe, M., Jansen, N., Junges, S., Marandi, A., Suilen, M., Topcu, U.: Robust finite-state controllers for uncertain pomdps. In: AAAI. pp. 11792–11800. AAAI Press (2021)

35. Daca, P., Henzinger, T.A., Kretínský, J., Petrov, T.: Faster statistical model checking for unbounded temporal properties. In: TACAS. Lecture Notes in Computer Science, vol. 9636, pp. 112–129. Springer (2016)
36. Dehnert, C., Junges, S., Jansen, N., Corzilius, F., Volk, M., Bruintjes, H., Katoen, J., Ábrahám, E.: Prophesy: A probabilistic parameter synthesis tool. In: CAV (1). Lecture Notes in Computer Science, vol. 9206, pp. 214–231. Springer (2015)
37. Dehnert, C., Junges, S., Jansen, N., Corzilius, F., Volk, M., Katoen, J., Ábrahám, E., Bruintjes, H.: Parameter synthesis for probabilistic systems. In: MBMV. pp. 72–74. Albert-Ludwigs-Universität Freiburg (2016)
38. Delage, A., Buffet, O., Dibangoye, J.S., Saffidine, A.: HSVI can solve zero-sum partially observable stochastic games. Dynamic Games and Applications (2023)
39. Fecher, H., Leucker, M., Wolf, V.: *Don't Know* in probabilistic systems. In: SPIN. Lecture Notes in Computer Science, vol. 3925, pp. 71–88. Springer (2006)
40. Fijalkow, N., Bertrand, N., Bouyer-Decitre, P., Brenguier, R., Carayol, A., Fearnley, J., Gimbert, H., Horn, F., Ibsen-Jensen, R., Markey, N., Monmege, B., Novotný, P., Randour, M., Sankur, O., Schmitz, S., Serre, O., Skomra, M.: Games on graphs. CoRR **abs/2305.10546** (2023)
41. Forejt, V., Kwiatkowska, M., Norman, G., Parker, D.: Automated verification techniques for probabilistic systems. In: Bernardo, M., Issarny, V. (eds.) Formal Methods for Eternal Networked Software Systems (SFM'11). LNCS, vol. 6659, pp. 53–113. Springer (2011)
42. Gadot, U., Derman, E., Kumar, N., Elfatihi, M.M., Levy, K., Mannor, S.: Solving non-rectangular reward-robust mdps via frequency regularization. In: AAAI. pp. 21090–21098. AAAI Press (2024)
43. Galesloot, M.F.L., Suilen, M., Simão, T.D., Carr, S., Spaan, M.T.J., Topcu, U., Jansen, N.: Pessimistic iterative planning for robust pomdps (2024)
44. Ghavamzadeh, M., Petrik, M., Chow, Y.: Safe policy improvement by minimizing robust baseline regret. In: NIPS. pp. 2298–2306 (2016)
45. Girard, A., Pappas, G.J.: Approximation metrics for discrete and continuous systems. IEEE Trans. Autom. Control. **52**(5), 782–798 (2007)
46. Givan, R., Leach, S.M., Dean, T.L.: Bounded-parameter markov decision processes. Artif. Intell. **122**(1-2), 71–109 (2000)
47. Goyal, V., Grand-Clément, J.: Robust markov decision processes: Beyond rectangularity. Math. Oper. Res. **48**(1), 203–226 (2023)
48. Grand-Clément, J., Petrik, M.: Reducing blackwell and average optimality to discounted mdps via the blackwell discount factor. In: NeurIPS (2023)
49. Grand-Clément, J., Petrik, M., Vieille, N.: Beyond discounted returns: Robust markov decision processes with average and blackwell optimality. CoRR **abs/2312.03618** (2023)
50. Guez, A., Silver, D., Dayan, P.: Efficient bayes-adaptive reinforcement learning using sample-based search. In: NIPS. pp. 1034–1042 (2012)
51. Haddad, S., Monmege, B.: Reachability in mdps: Refining convergence of value iteration. In: RP. Lecture Notes in Computer Science, vol. 8762, pp. 125–137. Springer (2014)
52. Hansson, H., Jonsson, B.: A logic for reasoning about time and reliability. Formal Aspects Comput. **6**(5), 512–535 (1994)
53. Hartmanns, A., Junges, S., Quatmann, T., Weininger, M.: A practitioner's guide to MDP model checking algorithms. In: TACAS (1). Lecture Notes in Computer Science, vol. 13993, pp. 469–488. Springer (2023)
54. Hartmanns, A., Kaminski, B.L.: Optimistic value iteration. In: CAV (2). Lecture Notes in Computer Science, vol. 12225, pp. 488–511. Springer (2020)

55. Hartmanns, A., Klauck, M., Parker, D., Quatmann, T., Ruijters, E.: The quantitative verification benchmark set. In: Vojnar, T., Zhang, L. (eds.) Tools and Algorithms for the Construction and Analysis of Systems - 25th International Conference, TACAS 2019, Held as Part of the European Joint Conferences on Theory and Practice of Software, ETAPS 2019, Prague, Czech Republic, April 6-11, 2019, Proceedings, Part I. Lecture Notes in Computer Science, vol. 11427, pp. 344–350. Springer (2019). https://doi.org/10.1007/978-3-030-17462-0_20, https://doi.org/10.1007/978-3-030-17462-0_20
56. Hashemi, V., Hermanns, H., Song, L., Subramani, K., Turrini, A., Wojciechowski, P.: Compositional bisimulation minimization for interval markov decision processes. In: LATA. Lecture Notes in Computer Science, vol. 9618, pp. 114–126. Springer (2016)
57. Hensel, C., Junges, S., Katoen, J., Quatmann, T., Volk, M.: The probabilistic model checker storm. Int. J. Softw. Tools Technol. Transf. **24**(4), 589–610 (2022)
58. Ho, C.P., Petrik, M., Wiesemann, W.: Fast bellman updates for robust mdps. In: ICML. Proceedings of Machine Learning Research, vol. 80, pp. 1984–1993. PMLR (2018)
59. Ho, C.P., Petrik, M., Wiesemann, W.: Partial policy iteration for l1-robust markov decision processes. J. Mach. Learn. Res. **22**, 275:1–275:46 (2021)
60. Ho, C.P., Petrik, M., Wiesemann, W.: Robust ϕ-divergence mdps. In: NeurIPS (2022)
61. Hoeffding, W.: Probability inequalities for sums of bounded random variables. Journal of the American Statistical Association pp. 13–30 (1963)
62. Hüllermeier, E., Waegeman, W.: Aleatoric and epistemic uncertainty in machine learning: an introduction to concepts and methods. Mach. Learn. **110**(3), 457–506 (2021)
63. Itoh, H., Nakamura, K.: Partially observable markov decision processes with imprecise parameters. Artif. Intell. **171**(8-9), 453–490 (2007)
64. Iyengar, G.N.: Robust dynamic programming. Math. Oper. Res. **30**(2), 257–280 (2005)
65. Jaeger, M., Bacci, G., Bacci, G., Larsen, K.G., Jensen, P.G.: Approximating euclidean by imprecise markov decision processes. In: ISoLA (1). Lecture Notes in Computer Science, vol. 12476, pp. 275–289. Springer (2020)
66. Jaksch, T., Ortner, R., Auer, P.: Near-optimal regret bounds for reinforcement learning. J. Mach. Learn. Res. **11**, 1563–1600 (2010)
67. Jansen, N., Junges, S., Katoen, J.: Parameter synthesis in markov models: A gentle survey. In: Principles of Systems Design. Lecture Notes in Computer Science, vol. 13660, pp. 407–437. Springer (2022)
68. Jonsson, B., Larsen, K.G.: Specification and refinement of probabilistic processes. In: Proceedings of the Sixth Annual Symposium on Logic in Computer Science (LICS '91), Amsterdam, The Netherlands, July 15-18, 1991. pp. 266–277. IEEE Computer Society (1991). https://doi.org/10.1109/LICS.1991.151651, https://doi.org/10.1109/LICS.1991.151651
69. Junges, S., Ábrahám, E., Hensel, C., Jansen, N., Katoen, J., Quatmann, T., Volk, M.: Parameter synthesis for markov models: covering the parameter space. Formal Methods Syst. Des. **62**(1), 181–259 (2024)
70. Junges, S., Katoen, J., Pérez, G.A., Winkler, T.: The complexity of reachability in parametric markov decision processes. J. Comput. Syst. Sci. **119**, 183–210 (2021)
71. Kaelbling, L.P., Littman, M.L., Cassandra, A.R.: Planning and acting in partially observable stochastic domains. Artif. Intell. **101**(1-2), 99–134 (1998)

72. Katoen, J.: The probabilistic model checking landscape. In: LICS. pp. 31–45. ACM (2016)
73. Katoen, J., Klink, D., Leucker, M., Wolf, V.: Three-valued abstraction for continuous-time markov chains. In: CAV. Lecture Notes in Computer Science, vol. 4590, pp. 311–324. Springer (2007)
74. Katoen, J., Klink, D., Leucker, M., Wolf, V.: Three-valued abstraction for probabilistic systems. J. Log. Algebraic Methods Program. **81**(4), 356–389 (2012)
75. Kattenbelt, M., Kwiatkowska, M.Z., Norman, G., Parker, D.: A game-based abstraction-refinement framework for markov decision processes. Formal Methods Syst. Des. **36**(3), 246–280 (2010)
76. Kaufman, D.L., Schaefer, A.J.: Robust modified policy iteration. INFORMS J. Comput. **25**(3), 396–410 (2013)
77. Kumar, N., Derman, E., Geist, M., Levy, K.Y., Mannor, S.: Policy gradient for rectangular robust markov decision processes. In: NeurIPS (2023)
78. Kwiatkowska, M.Z., Norman, G., Parker, D.: Game-based abstraction for markov decision processes. In: QEST. pp. 157–166. IEEE Computer Society (2006)
79. Kwiatkowska, M.Z., Norman, G., Parker, D.: Stochastic model checking. In: SFM. Lecture Notes in Computer Science, vol. 4486, pp. 220–270. Springer (2007)
80. Kwiatkowska, M.Z., Norman, G., Parker, D.: PRISM 4.0: Verification of probabilistic real-time systems. In: CAV. Lecture Notes in Computer Science, vol. 6806, pp. 585–591. Springer (2011)
81. Lahijanian, M., Andersson, S.B., Belta, C.: Formal verification and synthesis for discrete-time stochastic systems. IEEE Trans. Autom. Control. **60**(8), 2031–2045 (2015)
82. Laroche, R., Trichelair, P., des Combes, R.T.: Safe policy improvement with baseline bootstrapping. In: ICML. Proceedings of Machine Learning Research, vol. 97, pp. 3652–3661. PMLR (2019)
83. Larsen, K.G., Skou, A.: Bisimulation through probabilistic testing. Inf. Comput. **94**(1), 1–28 (1991)
84. Lavaei, A., Soudjani, S., Abate, A., Zamani, M.: Automated verification and synthesis of stochastic hybrid systems: A survey. Autom. **146**, 110617 (2022)
85. Lavaei, A., Soudjani, S., Frazzoli, E., Zamani, M.: Constructing MDP abstractions using data with formal guarantees. IEEE Control. Syst. Lett. **7**, 460–465 (2023)
86. Legay, A., Delahaye, B., Bensalem, S.: Statistical model checking: An overview. In: Barringer, H., Falcone, Y., Finkbeiner, B., Havelund, K., Lee, I., Pace, G.J., Rosu, G., Sokolsky, O., Tillmann, N. (eds.) Runtime Verification - First International Conference, RV 2010, St. Julians, Malta, November 1-4, 2010. Proceedings. Lecture Notes in Computer Science, vol. 6418, pp. 122–135. Springer (2010). https://doi.org/10.1007/978-3-642-16612-9_11, https://doi.org/10.1007/978-3-642-16612-9_11
87. Madani, O., Hanks, S., Condon, A.: On the undecidability of probabilistic planning and related stochastic optimization problems. Artif. Intell. **147**(1-2), 5–34 (2003)
88. Mannor, S., Simester, D., Sun, P., Tsitsiklis, J.N.: Bias and variance approximation in value function estimates. Manag. Sci. **53**(2), 308–322 (2007)
89. Mathiesen, F.B., Lahijanian, M., Laurenti, L.: Intervalmdp.jl: Accelerated value iteration for interval markov decision processes. Tech. Rep. arXiv:2401.04068, arXiv (2024)
90. Meggendorfer, T., Weininger, M., Wienhöft, P.: What are the odds? improving the foundations of statistical model checking. CoRR **abs/2404.05424** (2024)

91. Moos, J., Hansel, K., Abdulsamad, H., Stark, S., Clever, D., Peters, J.: Robust reinforcement learning: A review of foundations and recent advances. Mach. Learn. Knowl. Extr. **4**(1), 276–315 (2022)
92. Nakao, H., Jiang, R., Shen, S.: Distributionally robust partially observable markov decision process with moment-based ambiguity. SIAM J. Optim. **31**(1), 461–488 (2021)
93. Nilim, A., Ghaoui, L.E.: Robust control of markov decision processes with uncertain transition matrices. Oper. Res. **53**(5), 780–798 (2005)
94. Osogami, T.: Robust partially observable markov decision process. In: ICML. JMLR Workshop and Conference Proceedings, vol. 37, pp. 106–115. JMLR.org (2015)
95. Ou, W., Bi, S.: Sequential decision-making under uncertainty: A robust mdps review. CoRR **abs/2305.10546** (2024)
96. Pnueli, A.: The temporal logic of programs. In: FOCS. pp. 46–57. IEEE Computer Society (1977)
97. Ponnambalam, C.T., Oliehoek, F.A., Spaan, M.T.J.: Abstraction-guided policy recovery from expert demonstrations. In: ICAPS. pp. 560–568. AAAI Press (2021)
98. Puggelli, A., Li, W., Sangiovanni-Vincentelli, A.L., Seshia, S.A.: Polynomial-time verification of PCTL properties of mdps with convex uncertainties. In: CAV. Lecture Notes in Computer Science, vol. 8044, pp. 527–542. Springer (2013)
99. Puterman, M.L.: Markov Decision Processes: Discrete Stochastic Dynamic Programming. Wiley Series in Probability and Statistics, Wiley (1994)
100. Quatmann, T., Dehnert, C., Jansen, N., Junges, S., Katoen, J.: Parameter synthesis for markov models: Faster than ever. In: ATVA. Lecture Notes in Computer Science, vol. 9938, pp. 50–67 (2016)
101. Quatmann, T., Katoen, J.: Sound value iteration. In: CAV (1). Lecture Notes in Computer Science, vol. 10981, pp. 643–661. Springer (2018)
102. Raskin, J., Sankur, O.: Multiple-environment markov decision processes. In: FSTTCS. LIPIcs, vol. 29, pp. 531–543. Schloss Dagstuhl - Leibniz-Zentrum für Informatik (2014)
103. Rickard, L., Abate, A., Margellos, K.: Learning robust policies for uncertain parametric markov decision processes. CoRR **abs/2312.06344** (2023)
104. Rigter, M., Lacerda, B., Hawes, N.: Minimax regret optimisation for robust planning in uncertain markov decision processes. In: AAAI. pp. 11930–11938. AAAI Press (2021)
105. Rigter, M., Lacerda, B., Hawes, N.: Risk-averse bayes-adaptive reinforcement learning. In: NeurIPS. pp. 1142–1154 (2021)
106. Saghafian, S.: Ambiguous partially observable markov decision processes: Structural results and applications. J. Econ. Theory **178**, 1–35 (2018)
107. Simão, T.D., Suilen, M., Jansen, N.: Safe policy improvement for pomdps via finite-state controllers. In: AAAI. pp. 15109–15117. AAAI Press (2023)
108. Strehl, A.L., Li, L., Littman, M.L.: Reinforcement learning in finite mdps: PAC analysis. J. Mach. Learn. Res. **10**, 2413–2444 (2009)
109. Strehl, A.L., Littman, M.L.: An analysis of model-based interval estimation for markov decision processes. J. Comput. Syst. Sci. **74**(8), 1309–1331 (2008)
110. Suilen, M., Jansen, N., Cubuktepe, M., Topcu, U.: Robust policy synthesis for uncertain pomdps via convex optimization. In: IJCAI. pp. 4113–4120. ijcai.org (2020)
111. Suilen, M., Simão, T.D., Parker, D., Jansen, N.: Robust anytime learning of markov decision processes. In: NeurIPS (2022)

112. Suilen, M., van der Vegt, M., Junges, S.: A pspace algorithm for almost-sure rabin objectives in multi-environment mdps. CoRR **abs/2407.07006** (2024)
113. Sutton, R.S., Barto, A.G.: Reinforcement learning - an introduction. Adaptive computation and machine learning, MIT Press (1998)
114. van der Vegt, M., Jansen, N., Junges, S.: Robust almost-sure reachability in multi-environment mdps. In: TACAS (1). Lecture Notes in Computer Science, vol. 13993, pp. 508–526. Springer (2023)
115. Wang, Q., Ho, C.P., Petrik, M.: Policy gradient in robust mdps with global convergence guarantee. In: ICML. Proceedings of Machine Learning Research, vol. 202, pp. 35763–35797. PMLR (2023)
116. Weissman, T., Ordentlich, E., Seroussi, G., Verdu, S., Weinberger, M.J.: Inequalities for the l1 deviation of the empirical distribution. Hewlett-Packard Labs, Tech. Rep (2003)
117. Wienhöft, P., Suilen, M., Simão, T.D., Dubslaff, C., Baier, C., Jansen, N.: More for less: Safe policy improvement with stronger performance guarantees. In: IJCAI. pp. 4406–4415. ijcai.org (2023)
118. Wiesemann, W., Kuhn, D., Rustem, B.: Robust markov decision processes. Math. Oper. Res. **38**(1), 153–183 (2013)
119. Wolff, E.M., Topcu, U., Murray, R.M.: Robust control of uncertain markov decision processes with temporal logic specifications. In: CDC. pp. 3372–3379. IEEE (2012)
120. Wooding, B., Lavaei, A.: Impact: Interval MDP parallel construction for controller synthesis of large-scale stochastic systems. CoRR **abs/2401.03555** (2024)
121. Xu, H., Mannor, S.: Distributionally robust markov decision processes. Math. Oper. Res. **37**(2), 288–300 (2012)
122. Yang, C., Littman, M.L., Carbin, M.: On the (in)tractability of reinforcement learning for LTL objectives. In: IJCAI. pp. 3650–3658. ijcai.org (2022)