Efficient Sensitivity Analysis
for Parametric Robust Markov Chains

Thom Badings[1(✉)], Sebastian Junges[1], Ahmadreza Marandi[2],
Ufuk Topcu[3], and Nils Jansen[1]

[1] Radboud University, Nijmegen, The Netherlands
`thom.badings@ru.nl`
[2] Eindhoven University of Technology, Eindhoven,
The Netherlands
[3] University of Texas at Austin, Austin, USA

Abstract. We provide a novel method for sensitivity analysis of parametric robust Markov chains. These models incorporate parameters and sets of probability distributions to alleviate the often unrealistic assumption that precise probabilities are available. We measure sensitivity in terms of partial derivatives with respect to the uncertain transition probabilities regarding measures such as the expected reward. As our main contribution, we present an efficient method to compute these partial derivatives. To scale our approach to models with thousands of parameters, we present an extension of this method that selects the subset of k parameters with the highest partial derivative. Our methods are based on linear programming and differentiating these programs around a given value for the parameters. The experiments show the applicability of our approach on models with over a million states and thousands of parameters. Moreover, we embed the results within an iterative learning scheme that profits from having access to a dedicated sensitivity analysis.

1 Introduction

Discrete-time Markov chains (MCs) are ubiquitous in stochastic systems modeling [8]. A classical assumption is that all probabilities of an MC are precisely known—an assumption that is difficult, if not impossible, to satisfy in practice [4]. Robust MCs (rMCs), or uncertain MCs, alleviate this assumption by using *sets of probability distributions*, e.g., intervals of probabilities in the simplest case [12,39]. A typical verification problem for rMCs is to compute upper or lower bounds on measures of interest, such as the expected cumulative reward, under *worst-case realizations* of these probabilities in the set of distributions [52,59]. Thus, verification results are *robust* against any selection of probabilities in these sets.

Where to improve my model? As a running example, consider a ground vehicle navigating toward a target location in an environment with different terrain types. On each terrain type, there is some probability that the vehicle will slip and fail to move. Assume that we obtain a sufficient number of *samples* to infer upper and lower bounds (i.e., intervals) on the slipping probability on each terrain. We use these probability intervals to model the grid world as an rMC. However, from the rMC, it is unclear how our model (and thus the measure of interest) will change if we obtain more samples. For instance, if we take one more sample for a particular terrain, some of the intervals of the rMC will change, but how can we expect the verification result to change? And if the verification result is unsatisfactory, for which terrain type should we obtain more samples?

Parametric Robust MCs. To reason about how additional samples will change our model and thus the verification result, we employ a sensitivity analysis [29]. To that end, we use parametric robust MCs (prMCs), which are rMCs whose sets of probability distributions are defined as a function of a set of *parameters* [26], e.g., intervals with parametric upper/lower bounds. With these functions over the parameters, we can describe dependencies between the model's states. The assignment of values to each of the parameters is called an *instantiation*. Applying an instantiation to a prMC induces an rMC by replacing each occurrence of the parameters with their assigned values. For this induced rMC, we compute a (robust) value for a given measure, and we call this verification result the *solution* for this instantiation. Thus, we can associate a prMC with a function, called the *solution function*, that maps parameter instantiations to values.

Differentation for prMCs. For our running example, we choose the parameters to represent the number of samples we have obtained for each terrain. Naturally, the *derivative of this solution function* with respect to each parameter (a.k.a. sample size) then corresponds to the expected change in the solution upon obtaining more samples. Such differentiation for parametric MCs (pMCs), where parameter instantiations yield one precise probability distribution, has been studied in [34]. For prMCs, however, it is unclear how to compute derivatives and under what conditions the derivative exists. We thus consider the following problem:

Problem 1 *(Computing derivatives).* Given a prMC and a parameter instantiation, compute the partial derivative of the solution function (evaluated at this instantiation) with respect to each of the parameters.

Our Approach. We compute derivatives for prMCs by solving a parameterized linear optimization problem. We build upon results from convex optimization theory for differentiating the optimal solution of this optimization problem [9,15]. We also present sufficient conditions for the derivative to exist.

Improving Efficiency. However, computing the derivative for every parameter explicitly does not scale to more realistic models with thousands of parameters. Instead, we observe that to determine for which parameter we should obtain more samples, we do not need to know *all partial derivatives explicitly.* Instead, it may suffice to know which parameters have *the highest* (or lowest, depending on the application) derivative. Thus, we also solve the following (related) problem:
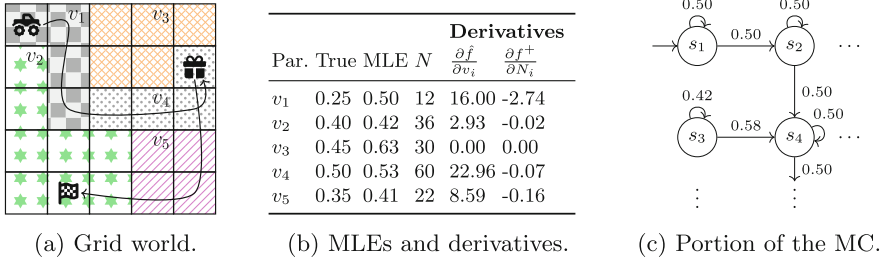
	Par.	True	MLE	N	**Derivatives** $\frac{\partial \hat{f}}{\partial v_i}$	$\frac{\partial f^+}{\partial N_i}$
	v_1	0.25	0.50	12	16.00	-2.74
	v_2	0.40	0.42	36	2.93	-0.02
	v_3	0.45	0.63	30	0.00	0.00
	v_4	0.50	0.53	60	22.96	-0.07
	v_5	0.35	0.41	22	8.59	-0.16

(a) Grid world.   (b) MLEs and derivatives.   (c) Portion of the MC.

Fig. 1. Grid world environment (a). The vehicle (🚚) must deliver the package (🎁) to the warehouse (🏁). We obtain the MLEs in (b), leading to the MC in (c).

Problem 2 *(k-highest derivatives).* Given a prMC with $|V|$ parameters, determine the $k < |V|$ parameters with the highest (or lowest) partial derivative.

We develop novel and efficient methods for solving Problem 2. Concretely, we design a linear program (LP) that finds the k parameters with the highest (or lowest) partial derivative without computing all derivatives explicitly. This LP constitutes a polynomial-time algorithm for Problem 2 and is, in practice, *orders of magnitude faster* than computing all derivatives explicitly, especially if the number of parameters is high. Moreover, if the concrete values for the partial derivatives are required, one can additionally solve Problem 1 for only the resulting k parameters. In our experiments, we show that we can compute derivatives for models with over a million states and thousands of parameters.

Learning Framework. Learning in stochastic environments is very data-intensive in general, and millions of samples may be required to obtain sufficiently tight bounds on measures of interest [43,47]. Several methods exist to obtain intervals on probabilities based on sampling, including statistical methods such as Hoeffding's inequality [14] and Bayesian methods that iteratively update intervals [57]. Motivated by this challenge of reducing the sample complexity of learning algorithms, we embed our methods in an iterative learning scheme that profits from having access to sensitivity values for the parameters. In our experiments, we show that derivative information can be used effectively to guide sampling when learning an unknown Markov chain with hundreds of parameters.

Contributions. Our contributions are threefold: (1) We present a first algorithm to compute partial derivatives for prMCs. (2) For both pMCs and prMCs, we develop an efficient method to determine a subset of parameters with the highest derivatives. (3) We apply our methods in an iterative learning scheme. We give an overview of our approach in Sect. 2 and formalize the problem statement in Sect. 3. In Sect. 4, we solve Problems (1) and (2) for pMCs, and in Sect. 5 for prMCs. Finally, the learning scheme and experiments are in Sect. 6.

2   Overview

We expand the example from Sect. 1 to illustrate our approach more concretely. The environment, shown in Fig. 1a, is partitioned into five regions of the same terrain type. The vehicle can move in the four cardinal directions. Recall that
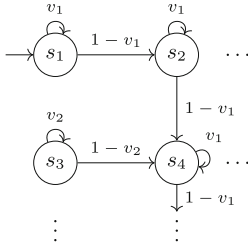
Fig. 2. Parametric MC.
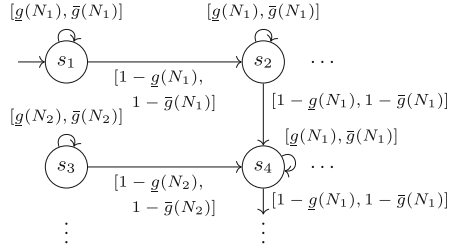
Fig. 3. Parametric robust MC.

the slipping probabilities are the same for all states with the same terrain. The vehicle follows a dedicated route to collect and deliver a package to a warehouse. Our goal is to estimate the expected number of steps $f^\star$ to complete the mission.

Estimating Probabilities. Classically, we would derive maximum likelihood estimates (MLEs) of the probabilities by sampling. Consider that, using N samples per slipping probability, we obtained the rough MLEs shown in Fig. 1b and thus the MC in Fig. 1c. Verifying the MC shows that the expected travel time (called the solution) under these estimates is $\hat{f} = 25.51$ steps, which is far from the travel time of $f^\star = 21.62$ steps under the true slipping probabilities. We want to close this *verification-to-real gap* by taking more samples for one of the terrain types. For which of the five terrain types should we obtain more samples?

Parametric Model. We can model the grid world as a pMC, i.e., an MC with symbolic probabilities. The solution function for this pMC is the travel time $\hat{f}$, being a function of these symbolic probabilities. We sketch four states of this pMC in Fig. 2. The most relevant parameter is then naturally defined as the parameter with the *largest partial derivative of the solution function.* As shown in Fig. 1B, parameter v_4 has the highest partial derivative of $\frac{\partial \hat{f}}{\partial v_4} = 22.96$, while the derivative of v_3 is zero as no states related to this parameter are ever visited.

Parametric Robust Model. The approach above does not account for the uncertainty in each MLE. Terrain type v_4 has the highest derivative but also the largest sample size, so sampling v_4 once more has likely less impact than for, e.g., v_1. So, is v_4 actually the best choice to obtain additional samples for? The prMC that allows us to answer this question is shown in Fig. 3, where we use (parametric) intervals as uncertainty sets. The parameters are the sample sizes $N_1, \ldots, N_5$ for all terrain types (contrary to the pMC, where parameters represent slipping probabilities). Now, if we obtain one additional sample for a particular terrain type, how can we expect the uncertainty sets to change?

Derivatives for prMCs. We use the prMC to compute an upper bound f^+ on the true solution $f^\star$. Obtaining one more sample for terrain type v_i (i.e., increasing N_i by one) shrinks the interval $[\underline{g}(N_i), \bar{g}(N_i)]$ on expectation, which in turn decreases our upper bound f^+. Here, $\underline{g}$ and $\bar{g}$ are functions mapping sample sizes to interval bounds. The partial derivatives $\frac{\partial f^+}{\partial N_i}$ for the prMC are also

shown in Fig. 1b and give a very different outcome than the derivatives for the pMC. In fact, sampling v_1 yields the biggest decrease in the upper bound f^+, so we ultimately decide to sample for terrain type v_1 instead of v_4.

Efficient Differentiation. We remark that we do not need to know all derivatives explicitly to determine where to obtain samples. Instead, it suffices to know *which parameter has the highest (or lowest) derivative.* In the rest of the paper, we develop efficient methods for computing either all or only the $k \in \mathbb{N}$ highest partial derivatives of the solution functions for pMCs and prMCs.

Supported Extensions. Our approaches are applicable to general pMCs and prMCs whose parameters can be shared between distributions (and thus capture dependencies, being a common advantage of parametric models in general [40]). Besides parameters in transition probabilities, we can handle parametric initial states, rewards, and policies. We could, e.g., use parameters to model the policy of a surveillance drone in our example and compute derivatives for these parameters.

3   Formal Problem Statement

Let $V = \{v_1, \ldots, v_\ell\}$, $v_i \in \mathbb{R}$ be a finite and ordered set of parameters. A parameter instantiation is a function $u \colon V \to \mathbb{R}$ that maps a parameter to a real valuation. The vector function $\mathbf{u}(v_1, \ldots, v_\ell) = [u(v_1), \ldots, u(v_\ell)]^\top \in \mathbb{R}^\ell$ denotes an ordered instantiation of all parameters in V through u. The set of polynomials over the parameters V is $\mathbb{Q}[V]$. A polynomial f can be interpreted as a function $f \colon \mathbb{R}^\ell \to \mathbb{R}$ where $f(\mathbf{u})$ is obtained by substituting each occurrence of v by $u(v)$. We denote these substitutions with $f[\mathbf{u}]$.

For any set X, let $pFun_V(X) = \{f \mid f \colon X \to \mathbb{Q}[V]\}$ be the set of functions that map from X to the polynomials over the parameters V. We denote by $pDist_V(X) \subset pFun_V(X)$ the set of *parametric probability distributions* over X, i.e., the functions $f \colon X \to \mathbb{Q}[V]$ such that $f(x)[\mathbf{u}] \in [0, 1]$ and $\sum_{x \in X} f(x)[\mathbf{u}] = 1$ for all parameter instantiations $\mathbf{u}$.

Parametric Markov Chain. We define a pMC as follows:

Definition 1 (pMC). *A pMC $\mathcal{M}$ is a tuple (S, s_I, V, P), where S is a finite set of states, $s_I \in Dist(S)$ a distribution over initial states, V a finite set of parameters, and $P \colon S \to pDist_V(S)$ a parametric transition function.*

Applying an instantiation $\mathbf{u}$ to a pMC yields an MC $\mathcal{M}[\mathbf{u}]$ by replacing each transition probability $f \in \mathbb{Q}[V]$ by $f[\mathbf{u}]$. We consider expected reward measures based on a state reward function $R \colon S \to \mathbb{R}$. Each parameter instantiation for a pMC yields an MC for which we can compute the solution for the expected reward measure [8]. We call the function that maps instantiations to a solution the *solution function.* The solution function is smooth over the set of graph-preserving instantiations [41]. Concretely, the solution function sol for the expected cumulative reward under instantiation $\mathbf{u}$ is written as follows:

$$\mathsf{sol}(\mathbf{u}) = \sum_{s \in S} \left(s_I(s) \sum_{\omega \in \Omega(s)} \mathrm{rew}(\omega) \cdot \Pr(\omega, \mathbf{u}) \right), \tag{1}$$

where $\Omega(s)$ is the set of paths starting in $s \in S$, $\mathrm{rew}(\omega) = R(s_0) + R(s_1) + \cdots$ is the cumulative reward over $\omega = s_0 s_1 \cdots$, and $\mathrm{Pr}(\omega, \mathbf{u})$ is the probability for a path $\omega \in \Omega(s)$. If a terminal (sink) state is reached from state $s \in S$ with probability one, the infinite sum over $\omega \in \Omega(s)$ in Eq. (1) exist [53].

Parametric Robust Markov Chains. The convex polytope $T_{A,b} \subseteq \mathbb{R}^n$ defined by matrix $A \in \mathbb{R}^{m \times n}$ and vector $b \in \mathbb{R}^m$ is the set $T_{A,b} = \{ p \in \mathbb{R}^n \mid Ap \leq b \}$. We denote by $\mathbb{T}_n$ the set of all convex polytopes of dimension n, i.e.,

$$\mathbb{T}_n = \{ T_{A,b} \mid A \in \mathbb{R}^{m \times n}, \, b \in \mathbb{R}^m, \, m \in \mathbb{N} \}. \tag{2}$$

A robust MC (rMC) [54, 58] is a tuple $(S, s_I, \mathcal{P})$, where S and s_I are defined as for pMCs and the uncertain transition function $\mathcal{P} \colon S \to \mathbb{T}_{|S|}$ maps states to convex polytopes $T \in \mathbb{T}_{|S|}$. Intuitively, an rMC is an MC with possibly infinite *sets of probability distributions*. To obtain robust bounds on the verification result for any of these MCs, an *adversary* nondeterministically chooses a precise transition function by fixing a probability distribution $\hat{P}(s) \in \mathcal{P}(s)$ for each $s \in S$.

We extend rMCs with polytopes whose halfspaces are defined by polynomials $\mathbb{Q}[V]$ over V. To this end, let $\mathbb{T}_n[V]$ be the set of all such *parametric polytopes*:

$$\mathbb{T}_n[V] = \{ T_{A,b} \mid A \in \mathbb{Q}[V]^{m \times n}, \, b \in \mathbb{Q}[V]^m, \, m \in \mathbb{N} \}. \tag{3}$$

An element $T \in \mathbb{T}_n[V]$ can be interpreted as a function $T \colon \mathbb{R}^\ell \to 2^{(\mathbb{R}^n)}$ that maps an instantiation $\mathbf{u}$ to a (possibly empty) convex polytopic subset of $\mathbb{R}^n$. The set $T[\mathbf{u}]$ is obtained by substituting each v_i in T by $u(v_i)$ for all $i = 1, \ldots, \ell$.

Example 1. The uncertainty set for state s_1 of the prMC in Fig. 3 is the parametric polytope $T \in \mathbb{T}_2[V]$ with singleton parameter set $V = \{N_1\}$, such that

$$\begin{aligned} T = \big\{ [p_{1,1}, p_{1,2}]^\top \in \mathbb{R}^2 \ \big| \ &\underline{g}_1(N_1) \leq p_{1,1} \leq \bar{g}_1(N_1), \\ &1 - \bar{g}_1(N_1) \leq p_{1,2} \leq 1 - \underline{g}_1(N_1), \ \ p_{1,2} + p_{1,2} = 1 \big\}. \end{aligned}$$

We use parametric convex polytopes to define prMCs:

Definition 2 (prMC). *A prMC $\mathcal{M}_R$ is a tuple $(S, s_I, V, \mathcal{P})$, where S, s_I, and V are defined as for pMCs (Def. 1), and where $\mathcal{P} \colon S \to \mathbb{T}_{|S|}[V]$ is a parametric and uncertain transition function that maps states to parametric convex polytopes.*

Applying an instantiation $\mathbf{u}$ to a prMC yields an rMC $\mathcal{M}_R[\mathbf{u}]$ by replacing each parametric polytope $T \in \mathbb{T}_{|S|}[V]$ by $T[\mathbf{u}]$, i.e., a polytope defined by a concrete matrix $A \in \mathbb{R}^{m \times n}$ and vector $b \in \mathbb{R}^m$. Without loss of generality, we consider adversaries minimizing the expected cumulative reward until reaching a set of terminal states $S_T \subseteq S$. This minimum expected cumulative reward $\mathsf{sol}_R(\mathbf{u})$, called the *robust solution* on the instantiated prMC $\mathcal{M}_R[\mathbf{u}]$, is defined as

$$\mathsf{sol}_R(\mathbf{u}) = \sum_{s \in S} \Big(s_I(s) \cdot \min_{P \in \mathcal{P}[\mathbf{u}]} \sum_{\omega \in \Omega(s)} \mathrm{rew}(\omega) \cdot \mathrm{Pr}(\omega, \mathbf{u}, P) \Big). \tag{4}$$

We refer to the function $\mathsf{sol}_R \colon \mathbb{R}^\ell \to \mathbb{R}$ as the *robust solution function*.

Assumptions on pMCs and prMCs. For both pMCs and prMCs, we assume that transitions cannot vanish under any instantiation (graph-preservation). That is, for every $s, s' \in S$, we have that $P(s)[\mathbf{u}](s')$ (for pMCs) and $\mathcal{P}(s)[\mathbf{u}](s')$ (for prMCs) are either zero or strictly positive for all instantiations $\mathbf{u}$.

Problem Statement. Let $f(q_1, \ldots, q_n) \in \mathbb{R}^m$ be a differentiable multivariate function with $m \in \mathbb{N}$. We denote the *partial derivative* of f with respect to q by $\frac{\partial x}{\partial q} \in \mathbb{R}^m$. The *gradient* of f combines all partial derivatives in a single vector as $\nabla_q f = [\frac{\partial f}{\partial q_1}, \ldots, \frac{\partial f}{\partial q_n}] \in \mathbb{R}^{m \times n}$. We only use gradients $\nabla_{\mathbf{u}} f$ with respect to the parameter instantiation $\mathbf{u}$, so we simply write ∇f in the remainder.

The gradient of the robust solution function evaluated at the instantiation $\mathbf{u}$ is $\nabla \mathsf{sol}_R[\mathbf{u}] = \left[\left(\frac{\partial \mathsf{sol}_R}{\partial u(v_1)} \right)[\mathbf{u}], \ldots, \left(\frac{\partial \mathsf{sol}_R}{\partial u(v_\ell)} \right)[\mathbf{u}] \right]$. We solve the following problem.

> *Problem 1. Given a prMC $\mathcal{M}_R$ and a parameter instantiation $\mathbf{u}$, compute the gradient $\nabla \mathsf{sol}_R[\mathbf{u}]$ of the robust solution function evaluated at $\mathbf{u}$.*

Solving Problem 1 is linear in the number of parameters, which may lead to significant overhead if the number of parameters is large. Typically, it suffices to only obtain the parameters with the highest derivatives:

> *Problem 2. Given a prMC $\mathcal{M}_R$, an instantiation $\mathbf{u}$, and a $k \leq |V|$, compute a subset $V^\star$ of k parameters for which the partial derivatives are maximal.*

For both problems, we present polynomial-time algorithms for pMCs (Sect. 4) and prMCs (Sect. 5). Section 6 defines problem variations that we study empirically.

4   Differentiating Solution Functions for pMCs

We can compute the solution of an MC $\mathcal{M}[\mathbf{u}]$ with instantiation $\mathbf{u}$ based on a system of $|S|$ linear equations; here for an expected reward measure [8]. Let $x = [x_{s_1}, \ldots, x_{s_{|S|}}]^\top$ and $r = [r_{s_1}, \ldots, r_{s_{|S|}}]^\top$ be variables for the expected cumulative reward and the instantaneous reward in each state $s \in S$, respectively. Then, for a set of terminal (*sink*) states $S_T \subset S$, we obtain the equation system

$$x_s = 0, \qquad\qquad \forall s \in S_T \tag{5a}$$

$$x_s = r_s + P(s)[\mathbf{u}]x, \quad \forall s \in S \backslash S_T. \tag{5b}$$

Let us set $P(s)[\mathbf{u}] = 0$ for all $s \in S_T$ and define the matrix $P[\mathbf{u}] \in \mathbb{R}^{|S| \times |S|}$ by stacking the rows $P(s)[\mathbf{u}]$ for all $s \in S$. Then, Eq. (5) is written in matrix form as $(I_{|S|} - P[\mathbf{u}])x = r$. The equation system in Eq. (5) can be efficiently solved by, e.g., Gaussian elimination or more advanced iterative equation solvers.

4.1   Computing Derivatives Explicitly

We differentiate the equation system in Eq. (5) with respect to an instantiation $u(v_i)$ for parameter $v_i \in V$, similar to, e.g., [34]. For all $s \in S_T$, the derivative

$\frac{\partial x_s}{\partial u(v_i)}$ is trivially zero. For all $s \in S \setminus S_T$, we obtain via the product rule that

$$\frac{\partial x_s}{\partial u(v_i)} = \frac{\partial P(s)x}{\partial u(v_i)}[\mathbf{u}] = (x^\star)^\top \frac{\partial P(s)^\top}{\partial u(v_i)}[\mathbf{u}] + P(s)[\mathbf{u}]\frac{\partial x}{\partial u(v_i)}, \tag{6}$$

where $x^\star \in \mathbb{R}^{|S|}$ is the solution to Eq. (5). In matrix form for all $s \in S$, this yields

$$\left(I_{|S|} - P[\mathbf{u}]\right)\frac{\partial x}{\partial u(v_i)} = \frac{\partial Px^\star}{\partial u(v_i)}[\mathbf{u}]. \tag{7}$$

The solution defined in Eq. (1) is computed as $\mathsf{sol}[\mathbf{u}] = s_I^\top x^\star$. Thus, the partial derivative of the solution function with respect to $u(v_i)$ in closed form is

$$\left(\frac{\partial \mathsf{sol}}{\partial u(v_i)}\right)[\mathbf{u}] = s_I^\top \frac{\partial x}{\partial u(v_i)} = s_I^\top \left(I_{|S|} - P[\mathbf{u}]\right)^{-1}\frac{\partial Px^\star}{\partial u(v_i)}[\mathbf{u}]. \tag{8}$$

Algorithm for Problem 1. Let us provide an algorithm to solve 1 for pMCs. 8 provides a closed-form expression for the partial derivative of the solution function, which is a function of the vector $x^\star$ in Eq. (5). However, due to the inversion of $(I_{|S|} - P[\mathbf{u}])$, it is generally more efficient to solve the system of equations in Eq. (7). Doing so, the partial derivative of the solution with respect to $u(v_i)$ is obtained by: (1) solving Eq. (5) with $\mathbf{u}$ to obtain $x^\star \in \mathbb{R}^{|S|}$, and (2) solving the equation system in Eq. (7) with $|S|$ unknowns for this vector $x^\star$. We repeat step 2 for all of the $|V|$ parameters. Thus, we can solve Problem 1 by solving $|V| + 1$ linear equation systems with $|S|$ unknowns each.

4.2 Computing k-Highest Derivatives

To solve Problem 2 for pMCs, we present a method to compute only the $k \leq \ell = |V|$ parameters with the highest (or lowest) partial derivative without computing all derivatives explicitly. Without loss of generality, we focus on the highest derivative. We can determine these parameters by solving a combinatorial optimization problem with binary variables $z_i \in \{0, 1\}$ for $i = 1, \ldots, \ell$. Our goal is to formulate this optimization problem such that an optimal value of $z_i^\star = 1$ implies that parameter $v_i \in V$ belongs to the set of k highest derivatives. Concretely, we formulate the following *mixed integer linear problem* (MILP) [60]:

$$\underset{y \in \mathbb{R}^{|S|}, z \in \{0,1\}^\ell}{\text{maximize}} \quad s_I^\top y \tag{9a}$$

$$\text{subject to } \left(I_{|S|} - P[\mathbf{u}]\right)y = \sum_{i=1}^{\ell} z_i \frac{\partial Px^\star}{\partial u(v_i)}[\mathbf{u}] \tag{9b}$$

$$z_1 + \cdots + z_\ell = k. \tag{9c}$$

Constraint (9c) ensures that any feasible solution to Eq. (9) has exactly k nonzero entries. Since matrix $(I_{|S|} - P[\mathbf{u}])$ is invertible by construction (see, e.g., [53]), Eq.

(9) has a unique solution in y for each choice of $z \in \{0,1\}^{\ell}$. Thus, the objective value $s_I^{\top} y$ is the sum of the derivatives for the parameters $v_i \in V$ for which $z_i = 1$. Since we maximize this objective, an optimal solution $y^{\star}, z^{\star}$ to Eq. (9) is guaranteed to correspond to the k parameters that maximize the derivative of the solution in Eq. (8). We state this correctness claim for the MILP:

Proposition 1. *Let $y^{\star}$, $z^{\star}$ be an optimal solution to Eq. (9). Then, the set $V^{\star} = \{v_i \in V \mid z_i^{\star} = 1\}$ is a subset of $k \leq \ell$ parameters with maximal derivatives.*

The set $V^{\star}$ may not be unique. However, to solve Problem 2, it suffices to obtain *a set* of k parameters for which the partial derivatives are maximal. Therefore, the set $V^{\star}$ provides a solution to Problem 2. We remark that, to solve Problem 2 for the k lowest derivatives, we change the objective in Eq. (9a) to minimize $s_I^{\top} y$.

Linear Relaxation. The MILP in Eq. (9) is computationally intractable for high values of ℓ and k. Instead, we compute the set $v^{\star}$ via a *linear relaxation* of the MILP. Specifically, we relax the binary variables $z \in \{0,1\}^{\ell}$ to continuous variables $z \in [0,1]^{\ell}$. As such, we obtain the following LP relaxation of Eq. (9):

$$\underset{y \in \mathbb{R}^{|S|},\, z \in \mathbb{R}^{\ell}}{\text{maximize}} \ s_I^{\top} y \tag{10a}$$

$$\text{subject to} \ \left(I_{|S|} - P[\mathbf{u}]\right) y = \sum_{i=1}^{\ell} z_i \frac{\partial P x^{\star}}{\partial u(v_i)}[\mathbf{u}] \tag{10b}$$

$$0 \leq z_i \leq 1, \quad \forall i = 1, \ldots, \ell \tag{10c}$$

$$z_1 + \cdots + z_{\ell} = k. \tag{10d}$$

Denote by y^+, z^+ the solution of the LP relaxation in Eq. (10). For details on such linear relaxations of integer problems, we refer to [36,46]. In our case, every optimal solution y^+, z^+ to the LP relaxation with only binary values $z_i^+ \in \{0,1\}$ is also optimal for the MILP, resulting in the following theorem.

Theorem 1. *The LP relaxation in Eq. (10) has an optimal solution y^+, z^+ with $z^+ \in \{0,1\}^{\ell}$ (i.e., every optimal variable z_i^+ is binary), and every such a solution is also an optimal solution of the MILP in Eq. (9).*

Proof. From invertibility of $\left(I_{|S|} - P[\mathbf{u}]\right)$, we know that Eq. (9) is equivalent to

$$\underset{z \in \{0,1\}^{\ell}}{\text{maximize}} \ \sum_{i=1}^{\ell} z_i \left(s_I^{\top} \left(I_{|S|} - P[\mathbf{u}]\right)^{-1} \frac{\partial P x^{\star}}{\partial u(v_i)}[\mathbf{u}] \right) \tag{11a}$$

$$\text{subject to } z_1 + \cdots + z_{\ell} = k. \tag{11b}$$

The linear relaxation of Eq. (11) is an LP whose feasible region has integer vertices (see, e.g., [37]). Therefore, both Eq. (11) and its relaxation Eq. (10) have an integer optimal solution z^+, which constructs $z^{\star}$ in Eq. (9). □

The binary solutions $z^+ \in \{0,1\}^\ell$ are the vertices of the feasible set of the LP in Eq. (10). A simplex-based LP solver can be set to return such a solution.[1]

Algorithm for Problem 2. We provide an algorithm to solve Problem 2 for pMCs consisting of two steps. First, for pMC $\mathcal{M}$ and parameter instantiation $\mathbf{u}$, we solve the linear equation system in Eq. (7) for $x^\star$ to obtain the solution $\mathsf{sol}[\mathbf{u}] = s_I^\top x^\star$. Second, we fix a number of parameters $k \leq \ell$ and solve the LP relaxation in Eq. (10). The set $V^\star$ of parameters with maximal derivatives is then obtained as defined in Proposition 1. The parameter set $V^\star$ is a solution to Proposition 2.

5   Differentiating Solution Functions for prMCs

We shift focus to prMCs. Recall that solutions $\mathsf{sol}_R[\mathbf{u}]$ are computed for the worst-case realization of the uncertainty, called the robust solution. We derive the following equation system, where, as for pMCs, $x \in \mathbb{R}^{|S|}$ represents the expected cumulative reward in each state.

$$x_s = 0, \qquad\qquad\qquad \forall s \in S_T \tag{12a}$$
$$x_s = r_s + \inf_{p \in \mathcal{P}(s)[\mathbf{u}]} \left(p^\top x\right), \qquad\qquad \forall s \in S \setminus S_T. \tag{12b}$$

Solving Eq. (12) directly corresponds to solving a system of nonlinear equations due to the inner infimum in Eq. (12b). The standard approach from robust optimization [12] is to leverage the dual problem for each inner infimum, e.g., as is done in [20,52]. For each $s \in S$, $\mathcal{P}(s)$ is a parametric convex polytope $T_{A,b}$ as defined in Eq. (3). The dimensionality of this polytope depends on the number of successor states, which is typically much lower than the total number of states. To make the number of successor states explicit, we denote by $\mathsf{post}(s) \subseteq S$ the successor states of $s \in S$ and define $T_{A,b} \in \mathbb{T}_{|\mathsf{post}(s)|}[V]$ with $A_s \in \mathbb{Q}^{m_s \times |\mathsf{post}(s)|}$ and $b_s[\mathbf{u}] \in \mathbb{Q}^{m_s}$ (recall m_s is the number of halfspaces of the polytope). Then, the infimum in Eq. (12b) for each $s \in S \setminus S_T$ is

$$\text{minimize } p^\top x \tag{13a}$$
$$\text{subject to } A_s[\mathbf{u}]p \leq b_s[\mathbf{u}] \tag{13b}$$
$$\mathbb{1}^\top p = 1, \tag{13c}$$

where $\mathbb{1}$ denotes a column vector of ones of appropriate size. Let $x_{\mathsf{post}(s)} = [x_s]_{s \in \mathsf{post}(s)}$ be the vector of decision variables corresponding to the (ordered) successor states in $\mathsf{post}(s)$. The dual problem of Eq. (13), with dual variables $\alpha \in \mathbb{R}^{m_s}$ and $\beta \in \mathbb{R}$ (see, e.g., [11] for details), is written as follows:

$$\text{maximize } -b_s[\mathbf{u}]^\top \alpha - \beta \tag{14a}$$
$$\text{subject to } A_s[\mathbf{u}]^\top \alpha + x_{\mathsf{post}(s)} + \beta\mathbb{1} = 0 \tag{14b}$$
$$\alpha \geq 0. \tag{14c}$$

---

[1] Even if a non-vertex solution y^+, z^+ is obtained, we can use an arbitrary tie-break rule on z^+, which forces each z_i^+ binary and preserves the sum in Eq. (10d).
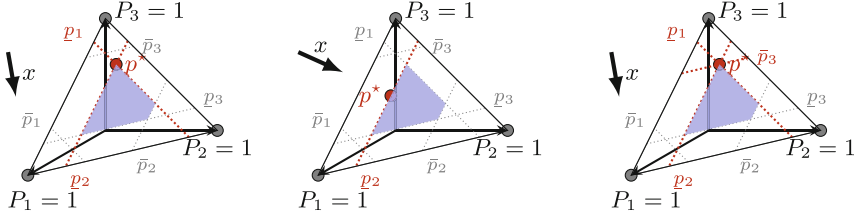
(a) Well-defined optimum. (b) Non-unique optimum. (c) Too many active constraints.

Fig. 4. Three polytopic uncertainty sets (blue shade), with the vector x, the worst-case points $p^\star$, and the active constraints shown in red. (Color figure online)

By using this dual problem in Eq. (12b), we obtain the following LP with decision variables $x \in \mathbb{R}^{|S|}$, and with $\alpha_s \in \mathbb{R}^{m_s}$ and $\beta_s \in \mathbb{R}$ for every $s \in S$:

$$\text{maximize } s_I^\top x \tag{15a}$$

$$\text{subject to } x_s = 0, \qquad\qquad\qquad\qquad \forall s \in S_T \tag{15b}$$

$$x_s = r_s - \left(b_s[\mathbf{u}]^\top \alpha_s + \beta_s \right), \qquad\qquad \forall s \in S \setminus S_T \tag{15c}$$

$$A_s[\mathbf{u}]^\top \alpha_s + x_{\mathsf{post}(s)} + \beta_s \mathbb{1} = 0, \quad \alpha_s \geq 0, \qquad \forall s \in S \setminus S_T. \tag{15d}$$

The reformulation of Eq. (12) to Eq. (15) requires that $s_I \geq 0$, which is trivially satisfied because s_I is a probability distribution. Denote by $x^\star, \alpha^\star, \beta^\star$ an optimal point of Eq. (15). The $x^\star$ element of this optimum is also an optimal solution of Eq. (12) [12]. Thus, the robust solution defined in Eq. (4) is $\mathsf{sol}_R[\mathbf{u}] = s_I^\top x^\star$.

5.1   Computing Derivatives via pMCs (and When It Does Not Work)

Toward solving Problem 1, we provide some intuition about computing robust solutions for prMCs. The infimum in Eq. (12) finds the *worst-case* point $p^\star$ in each set $\mathcal{P}(s)[\mathbf{u}]$ that minimizes $(p^\star)^\top x$. This minimization is visualized in Fig. 4a for an uncertainty set that captures three probability intervals $\underline{p}_i \leq p_i \leq \bar{p}_i$, $i = 1, 2, 3$. Given the optimization direction x (arrow in Fig. 4a), the point $p^\star$ (red dot) is attained at the vertex where the constraints $\underline{p}_1 \leq p_1$ and $\underline{p}_2 \leq p_2$ are active.[2] Thus, we obtain that the point in the polytope that minimizes $(p^\star)^\top x$ is $p^\star = [\underline{p}_1, \underline{p}_2, 1 - \underline{p}_1 - \underline{p}_2]^\top$. Using this procedure, we can obtain a worst-case point $p_s^\star$ for each state $s \in S$. We can use these points to convert the prMC into an induced pMC with transition function $P(s) = p_s^\star$ for each state $s \in S$.

For small changes in the parameters, the point $p^\star$ in Fig. 4a changes smoothly, and its closed-form expression (i.e., the functional form) remains the same. As such, it feels intuitive that we could apply the methods from Sect. 4 to compute partial derivatives on the induced pMC. However, this approach does not always work, as illustrated by the following two corner cases.

---

[2] An inequality constraint $gx \leq h$ is active under the optimal solution $x^\star$ if $gx^\star = h$ [15].

1. Consider Fig. 4b, where the optimization direction defined by x is parallel to one of the facets of the uncertainty set. In this case, the worst-case point $p^\star$ is not unique, but an infinitesimal change in the optimization direction x will force the point to one of the vertices again. Which point should we choose to obtain the induced pMC (and does this choice affect the derivative)?
2. Consider Fig. 4c with more than $|S| - 1$ active constraints at the point $p^\star$. Observe that decreasing $\bar{p}_3$ changes the point $p^\star$ while increasing $\bar{p}_3$ does not. In fact, the optimal point $p^\star$ changes *non-smoothly* with the halfspaces of the polytope. As a result, also the solution changes non-smoothly, and thus, the derivative is not defined. How do we deal with such a situation?

These examples show that computing derivatives via an induced pMC by obtaining each point $p_s^\star$ can be tricky or is, in some cases, not possible at all. In what follows, we present a method that directly derives a set of linear equations to obtain derivatives for prMCs (all or only the k highest) based on the solution to the LP in Eq. (15), which intrinsically identifies the corner cases above in which the derivative is not defined.

5.2 Computing Derivatives Explicitly

We now develop a dedicated method for identifying if the derivative of the solution function for a prMC exists, and if so, to compute this derivative. Observe from Fig. 4 that the point $p^\star$ is uniquely defined and has a smooth derivative only in Fig. 4a with two active constraints. For only one active constraint (Fig. 4b), the point is *underdetermined*, while for three active constraints (Fig. 4c), the derivative may *not be smooth*. In the general case, having exactly $n - 1$ active constraints (whose facets are nonparallel) is a sufficient condition for obtaining a unique and smoothly changing point $p^\star$ in the n-dimensional probability simplex.

Optimal Dual Variables. The optimal dual variables $\alpha_s^\star \geq 0$ for each $s \in S \setminus S_T$ in Eq. (15) indicate which constraints of the polytope $A_s[\mathbf{u}]p \leq b_s[\mathbf{u}]$ are active, i.e., for which rows $a_{s,i}[\mathbf{u}]$ of $A_s[\mathbf{u}]$ it holds that $a_{s,i}[\mathbf{u}]p^\star = b_s[\mathbf{u}]$. Specifically, a value of $\alpha_{s,i} > 0$ implies that the ith constraint is active, and $\alpha_{s,i} = 0$ indicates a nonactive constraint [15]. We define $E_s = [e_1, \ldots, e_{m_s}] \in \{0,1\}^{m_s}$ as a vector whose binary values $e_i \ \forall i \in \{1, \ldots, m_s\}$ are given as $e_i = [\![\alpha_{s,i}^\star > 0]\!]$.[3] Moreover, denote by $\mathbf{D}(E_s)$ the matrix with E_s on the diagonal and zeros elsewhere. We reduce the LP in Eq. (15) to a system of linear equations that encodes only the constraints that are active under the worst-case point $p_s^\star$ for each $s \in S \setminus S_T$:

$$x_s = 0, \qquad\qquad\qquad\qquad \forall s \in S_T \qquad (16a)$$

$$x_s = r_s - \left(b_s[\mathbf{u}]^\top \mathbf{D}(E_s)\alpha_s + \beta_s\right), \qquad \forall s \in S \setminus S_T \qquad (16b)$$

$$A_s[\mathbf{u}]^\top \mathbf{D}(E_s)\alpha_s + x_{\text{post}(s)} + \beta_s \mathbb{1} = 0, \quad \alpha_s \geq 0, \qquad \forall s \in S \setminus S_T. \qquad (16c)$$

Differentiation. However, when does Eq. (16) have a (unique) optimal solution? To provide some intuition, let us write the equation system in matrix form, i.e.,

---

[3] We use Iverson-brackets: $[\![x]\!] = 1$ if x is true and $[\![x]\!] = 0$ otherwise.

$C \begin{bmatrix} x & \alpha & \beta \end{bmatrix}^\top = d$, where we omit an explicit definition of matrix C and vector d for brevity. It is apparent that if matrix C is nonsingular, then Eq. (16) has a unique solution. This requires matrix C to be square, which is achieved if, for each $s \in S \setminus S_T$, we have $|\mathsf{post}(s)| = \sum E_s + 1$. In other words, the number of successor states of s is equal to the number of active constraints of the polytope plus one. This confirms our previous intuition from Sect. 5.1 on a polytope for $|\mathsf{post}(s)| = 3$ successor states, which required $\sum_{i=1}^{m_s} E_i = 2$ active constraints.

Let us formalize this intuition about computing derivatives for prMCs. We can compute the derivative of the solution $x^\star$ by differentiating the equation system in Eq. (16) through the product rule, in a very similar manner to the approach in Sect. 4. We state this key result in the following theorem.

Theorem 2. *Given a prMC $\mathcal{M}_R$ and an instantiation $\mathbf{u}$, compute $x^\star, \alpha^\star, \beta^\star$ for Eq. (15) and choose a parameter $v_i \in V$. The partial derivatives $\frac{\partial x}{\partial u(v_i)}, \frac{\partial \alpha}{\partial u(v_i)},$ and $\frac{\partial \beta}{\partial u(v_i)}$ are obtained as the solution to the linear equation system*

$$\frac{\partial x_s}{\partial u(v_i)} = 0, \qquad\qquad\qquad \forall s \in S_T \tag{17a}$$

$$\frac{\partial x_s}{\partial u(v_i)} + b_s[\mathbf{u}]^\top \mathbf{D}(E_s)\frac{\partial \alpha_s}{\partial u(v_i)} + \frac{\partial \beta_s}{\partial u(v_i)} \;= -(\alpha_s^\star)^\top \mathbf{D}(E_s)\frac{\partial b_s[\mathbf{u}]}{\partial u(v_i)}, \tag{17b}$$

$$\forall s \in S \setminus S_T$$

$$A_s[\mathbf{u}]^\top \mathbf{D}(E_s)\frac{\partial \alpha_s}{\partial u(v_i)} + \frac{\partial x_{\mathsf{post}(s)}}{\partial u(v_i)} + \frac{\partial \beta_s}{\partial u(v_i)}\mathbb{1} \;= -(\alpha_s^\star)^\top \mathbf{D}(E_s)\frac{\partial A_s[\mathbf{u}]}{\partial u(v_i)}, \tag{17c}$$

$$\forall s \in S \setminus S_T.$$

The proof follows from applying the product rule to Eq. (16) and is provided in [6, Appendix A.1]. To compute the derivative for a parameter $v_i \in V$, we thus solve a system of linear equations of size $|S| + \sum_{s \in S \setminus S_T} |\mathsf{post}(s)|$. Using Theorem 2, we obtain sufficient conditions for the solution function to be differentiable.

Lemma 1. *Write the linear equation system in Eq. (17) in matrix form, i.e.,*

$$C \left[\tfrac{\partial x}{\partial u(v_i)}, \tfrac{\partial \alpha}{\partial u(v_i)}, \tfrac{\partial \beta}{\partial u(v_i)} \right]^\top = d, \tag{18}$$

for $C \in \mathbb{R}^{q \times q}$ and $d \in \mathbb{R}^q$, $q = |S| + \sum_{s \in S \setminus S_T} |\mathsf{post}(s)|$, which are implicitly given by Eq. (17). The solution function $\mathsf{sol}_R[\mathbf{u}]$ is differentiable at instantiation $\mathbf{u}$ if matrix C is nonsingular, in which case we obtain $(\frac{\partial \mathsf{sol}_R}{\partial u(v_i)})[\mathbf{u}] = s_I^\top \frac{\partial x}{\partial u(v_i)}$.

Proof. The partial derivative of the solution function is $\frac{\partial \mathsf{sol}_R}{\partial u(v_i)}[\mathbf{u}] = s_I^\top \frac{\partial x^\star}{\partial u(v_i)}$, where $\frac{\partial x^\star}{\partial u(v_i}$ is (a part of) the solution to Eq. (16). Thus, the solution function is differentiable if there is a (unique) solution to Eq. (16), which is guaranteed if matrix C is nonsingular. Thus, the claim in Lemma 1 follows. □

Algorithm for Problem 1. We use Theorem 2 to solve Problem 1 for prMCs, similarly as for pMCs. Given a prMC $\mathcal{M}_R$ and an instantiation $\mathbf{u}$, we first solve

Eq. (15) to obtain $x^\star, \alpha^\star, \beta^\star$. Second, we use $\alpha_s^\star$ to compute the vector E_s of active constraints for each $s \in S \setminus S_T$. Third, for every parameter $v \in V$, we solve the equation system in Eq. (17). Thus, to compute the gradient of the solution function, we solve one LP and $|V|$ linear equation systems.

5.3   Computing k-Highest Derivatives

We directly apply the same procedure from Sect. 4.2 to compute the parameters with the $k \leq \ell$ highest derivatives. As for pMCs, we can compute the k highest derivatives by solving a MILP encoding the equation system in Eq. (17) for every parameter $v \in V$, which we present in [6, Appendix A.2] for brevity. This MILP has the same structure as Eq. (9), and thus we may apply the same linear relaxation to obtain an LP with the guarantees as stated in Theorem 1. In other words, solving the LP relaxation yields the set $V^\star$ of parameters with maximal derivatives as in Proposition 1. This set $V^\star$ is a solution to Problem 2 for prMCs.

6   Numerical Experiments

We perform experiments to answer the following questions about our approach:

1. Is it feasible (in terms of computational complexity and runtimes) to compute all derivatives, in particular compared to computing (robust) solutions?
2. How does computing only the k highest derivatives compare to computing all derivatives?
3. Can we apply our approach to effectively determine for which parameters to sample in a learning framework?

Let us briefly summarize the computations involved in answering these questions. First of all, computing the solution $\mathsf{sol}(\mathbf{u})$ for a pMC, which is defined in Eq. (1), means solving the linear equation system in Eq. (5). Similarly, computing the robust solution $\mathsf{sol}_R(\mathbf{u})$ for a prMC means solving the LP in Eq. (15). Then, solving Problem 1, i.e., computing all $|V|$ partial derivatives, amounts to solving a linear equation system for each parameter $v \in V$ (namely, Eq. (5) for a prMC and Eq. (17) for a prMC). In contrast, solving Problem 2, i.e., computing a subset $V^\star$ of parameters with maximal (or minimal) derivative, means for a pMC that we solve the LP in Eq. (10) (or the equivalent LP for a prMC) and thereafter extract the subset of $V^\star$ parameters using Proposition 1.

Problem 3: Computing the k-highest Derivatives. A solution to Problem 2 is a set $V^\star$ of k parameters but does not include the computation of the derivatives. However, it is straightforward to also obtain the actual derivatives $\left(\frac{\partial \mathsf{sol}}{\partial u(v)} \right) [\mathbf{u}]$ for each parameter $v \in V^\star$. Specifically, we solve Problem 1 for the k parameters in $V^\star$, such that we obtain the partial derivatives for all $v \in V^\star$. We remark that, for $k = 1$, the derivative follows directly from the optimal value $s_I^\top y^+$ of the LP in Eq. (10), so this additional step is not necessary. We will refer to computing the actual values of the k highest derivatives as *Problem 3*.

Setup. We implement our approach in Python 3.10, using Storm [35] to parse pMCs, Gurobi [31] to solve LPs, and the SciPy sparse solver to solve equation systems. All experiments run on a computer with a 4GHz Intel Core i9 CPU and 64 GB RAM, with a timeout of one hour. Our implementation is available at https://doi.org/10.5281/zenodo.7864260.

Grid World Benchmarks. We use scaled versions of the grid world from the example in Sect. 2 with over a million states and up to 10 000 terrain types. The vehicle only moves right or down, both with 50% probability (wrapping around when leaving the grid). Slipping only occurs when moving down and (slightly different from the example in Sect. 2) means that the vehicle moves *two cells instead of one*. We obtain between $N = 500$ and $1\,000$ samples of each slipping probability. For the pMCs, we use maximum likelihood estimation ($\frac{\bar{p}}{N}$, with $\bar{p}$ the sample mean) obtained from these samples as probabilities, whereas, for the prMCs, we infer probability intervals using Hoeffding's inequality (see Q3 for details).

Benchmarks from Literature. We also use several instances of parametric extensions of MCs and Markov decision processes (MDPs) from standard benchmark suits [33,44]. We also use pMC benchmarks from [5,23] as these models have more parameters than the traditional benchmarks. We extend these benchmarks to prMCs by constructing probability intervals around the pMC's probabilities.

Results. The results for all benchmarks are shown in [6, Appendix B, Tab. 2–3].

Q1. Computing Solutions vs. Derivatives

We investigate whether computing derivatives is feasible on p(r)MCs. In particular, we compare the computation times for computing derivatives on p(r)MCs (Problems 1 and 3) with the times for computing the solution for these models.
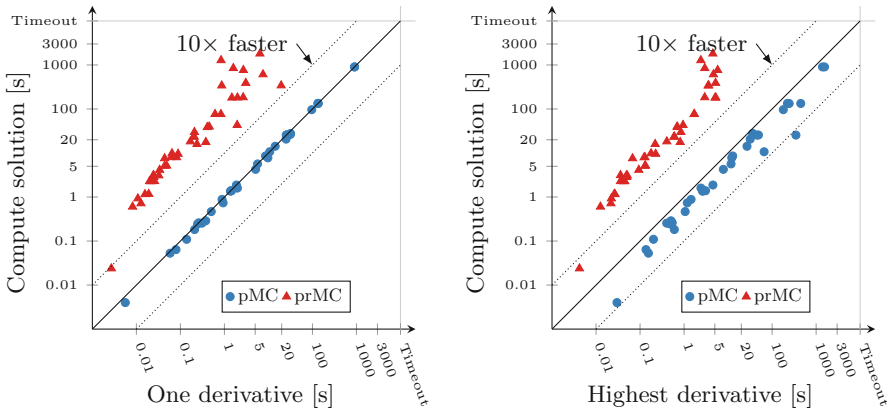

Fig. 5. Runtimes (log-scale) for computing a single derivative (left, Problem 1) or the highest derivative (right, Problem 3), vs. computing the solution $\mathsf{sol}[\mathbf{u}]/\mathsf{sol}_R[\mathbf{u}]$.

Table 1. Model sizes, runtimes, and derivatives for selection of grid world models.

Model statistics				Verifying		Problem 1	Problem 3		Derivatives	
Type	$|S|$	$|V|$	#trans	$\mathsf{sol}_{(R)}[\mathbf{u}]$	Time [s]	All derivs. [s]	$k=1$ [s]	$k=10[s]$	Highest	Error %
pMC	5000	50	14995	5.07	1.39	3.32	2.64	2.69	1.54e+00	0.0
pMC	5000	100	14995	5.05	1.36	4.17	2.63	2.66	1.28e+00	0.0
pMC	5000	921	14995	4.93	1.87	19.92	4.52	2.87	1.20e+00	0.0
pMC	80000	100	239995	8.01	25.54	98.47	45.18	46.87	1.95e+00	0.0
pMC	80000	1000	239995	8.01	25.64	612.97	48.92	58.20	2.08e+00	0.0
pMC	80000	9831	239995	7.93	25.52	5,650.25	347.76	1,343.59	2.10e+00	0.0
pMC	1280000	100	3839995	12.90	902.52	4,747.43	1,396.51	1,507.77	3.32e+00	0.0
pMC	1280000	1000	3839995	12.79	902.67	37,078.12	1,550.45	1,617.27	3.18e+00	0.0
pMC	1280000	10000	3839995	Timeout[b]	—	—	—	—	—	—
prMC	5000	100	14995	136.07	23.46	3.55	0.60	1.58	-1.26e-02	-0.0
prMC	5000	921	14995	138.74	29.82	25.23	0.85	1.09	-4.44e-03	-0.0
prMC	20000	100	59995	2,789.77	1,276.43	15.68	2.40	2.70	-4.96e-01	-0.1
prMC	20000	1000	59995	2,258.41	339.96	159.70	3.53	4.09	-9.51e-02	-0.0
prMC	80000	100	239995	Timeout[b]	—	—	—	—	—	—

[a]Extrapolated from the runtimes for 10 to all $|V|$ parameters.
[b]Timeout (1 h) occurred for verifying the p(r)MC, not for computing derivatives.

In Fig. 5, we show for all benchmarks the times for computing the solution (defined in Eqs. (1) and (4)), versus computing either a single derivative for Problem 1 (left) or the highest derivative of all parameters resulting from Problem 3 (right). A point (x, y) in the left plot means that computing a single derivative took x seconds while computing the solution took y seconds. A line above the (center) diagonal means we obtained a speed-up over the time for computing the solution; a point over the upper diagonal indicates a $10\times$ speed-up or larger.

One Derivative. The left plot in Fig. 5 shows that, for pMCs, the times for computing the solution and a single derivative are approximately the same. This is expected since both problems amount to solving a single equation system with $|S|$ unknowns. Recall that, for prMCs, computing the solution means solving the LP in Eq. (15), while for derivatives we solve an equation system. Thus, computing a derivative for a prMC is relatively cheap compared to computing the solution, which is confirmed by the results in Fig. 5.

Highest Derivative. The right plot in Fig. 5 shows that, for pMCs, computing the highest derivative is slightly slower than computing the solution (the LP to compute the highest derivative takes longer than the equation system to compute the solution). On the other hand, computing the highest derivative for a prMC is still cheap compared to computing the solution. Thus, if we are using a prMC anyways, computing the derivatives is relatively cheap.

Q2. Runtime Improvement of Computing only k Derivatives

We want to understand the computational benefits of solving Problem 3 over solving Problem 1. For Q2, we consider all models with $|V| \geq 10$ parameters.

An excerpt of results for the grid world benchmarks is presented in Table 1. Recall that, after obtaining the (robust) solution, solving Problem 1 amounts to solving $|V|$ linear equation systems, whereas Problem 3 involves solving a
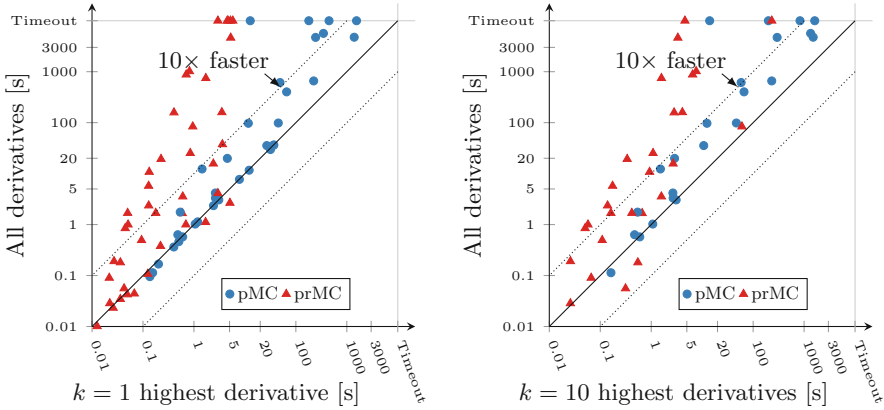
Fig. 6. Runtimes (log-scale) for computing the highest (left) or 10 highest (right) derivatives (Problem 3), versus computing all derivatives (Problem 1).

single LP and k equations systems. From Table 1, it is clear that computing k derivatives is orders of magnitudes faster than computing all $|V|$ derivatives, especially if the total number of parameters is high.

We compare the runtimes for computing all derivatives (Problem 1) with computing only the $k = 1$ or 10 highest derivatives (Problem 3). The left plot of Fig. 6 shows the runtimes for $k = 1$, and the right plot for the $k = 10$ highest derivatives. The interpretation for Fig. 6 is the same as for Fig. 5. From Fig. 6, we observe that computing only the k highest derivatives generally leads to significant speed-ups, often of more than 10 times (except for very small models). Moreover, the difference between $k = 1$ and $k = 10$ is minor, showing that retrieving the actual derivatives after solving Problem 2 is relatively cheap.

Numerical Stability. While our algorithm is exact, our implementation uses floating-point arithmetic for efficiency. To evaluate the numerical stability, we compare the highest derivatives (solving Problem 3 for $k = 1$) with an empirical approximation of the derivative obtained by perturbing the parameter by 1×10^3. The difference (column *'Error. %'* in Table 1 and [6, Appendix B, Table 2] between both is marginal, indicating that our implementation is sufficiently numerically stable to return accurate derivatives.

Q3. Application in a Learning Framework

Reducing the sample complexity is a key challenge in learning under uncertainty [43,47]. In particular, learning in stochastic environments is very data-intensive, and realistic applications tend to require millions of samples to provide tight bounds on measures of interest [16]. Motivated by this challenge, we apply our approach in a learning framework to investigate if derivatives can be used to effectively guide exploration, compared to alternative exploration strategies.
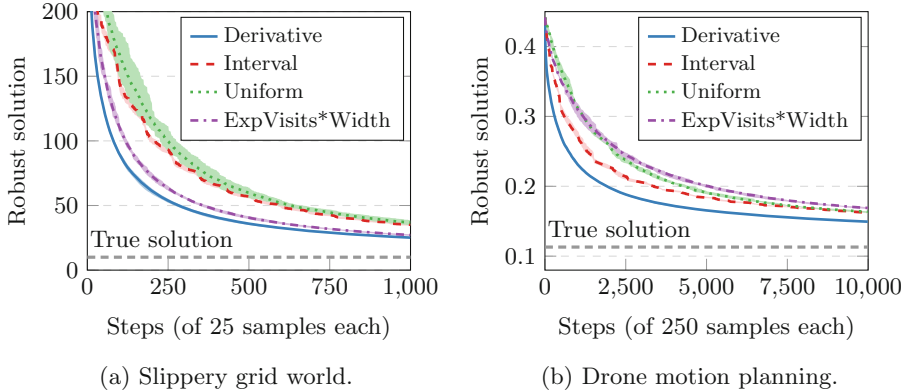
Fig. 7. Robust solutions for each sampling strategy in the learning framework for the grid world (a) and drone (b) benchmarks. Averages values of 10 (grid world) or 5 (drone) repetitions are shown, with shaded areas the min/max.

Models. We consider the problem of where to sample in 1) a slippery grid world with $|S| = 800$ and $|V| = 100$ terrain types, and 2) the drone benchmark from [23] with $|S| = 4\,179$ and $|V| = 1\,053$ parameters. As in the motivating example in Sect. 2, we learn a model of the unknown MC in the form of a prMC, where the parameters are the sample sizes for each parameter. We assume access to a model that can arbitrarily sample each parameter (i.e., the slipping probability in the case of the grid world). We use an initial sample size of $N_i = 100$ for each parameter $i \in \{1, \ldots, |V|\}$, from which we infer a $\beta = 0.9$ (90%) confidence interval using Hoeffding's inequality. The interval for parameter i is $[\hat{p}_i - \epsilon_i, \hat{p}_i + \epsilon_i]$, with $\hat{p}_i$ the sample mean and $\epsilon_i = \sqrt{\frac{\log 2 - \log(1-\beta)}{2N}}$ (see, e.g., [14] for details).

Learning Scheme. We iteratively choose for which parameter $v_i \in V$ to obtain 25 (for the grid world) or 250 (for the drone) additional samples. We compare four strategies for choosing the parameter v_i to sample for: 1) with highest derivative, i.e., solving Problem 3 for $k = 1$; 2) with biggest interval width ϵ_i; 3) uniformly; and 4) sampling according to the expected number of visits times the interval width (see [6, Appendix B.1] for details). After each step, we update the robust upper bound on the solution for the prMC with the additional samples.

Results. The upper bounds on the solution for each sampling strategy, as well as the solution for the MC with the true parameter values, are shown in Fig. 7. For both benchmarks, our derivative-guided sampling strategy converges to the true solution faster than the other strategies. Notably, our derivative-guided strategy accounts for both the uncertainty and importance of each parameter, which leads to a lower sample complexity required to approach the true solution.

7 Related Work

We discuss related work in three areas: pMCs, their extension to parametric interval Markov chains (piMCs), and general sensitivity analysis methods.

Parametric Markov Chains. pMCs [24, 45] have traditionally been studied in terms of computing the solution function [13, 25, 28, 29, 32]. Much recent literature considers synthesis (find a parameter valuation such that a specification is satisfied) or verification (prove that all valuations satisfy a specification). We refer to [38] for a recent overview. For our paper, particularly relevant are [55], which checks whether a derivative is positive (for all parameter valuations), and [34], which solves parameter synthesis via gradient descent. We note that all these problems are (co-)ETR complete [41] and that the solution function is exponentially large in the number of parameters [7], whereas we consider a polynomial-time algorithm. Furthermore, practical *verification* procedures for uncontrollable parameters (as we do) are limited to less than 10 parameters. Parametric verification is used in [51] to guide model refinement by detecting for which parameter values a specification is satisfied. In contrast, we consider slightly more conservative rMCs and aim to stepwise optimize an objective. Solution functions also provide an approach to compute and refine confidence intervals [17]; however, the size of the solution function hampers scalability.

Parametric interval Markov Chains (piMCs). While prMCs have, to the best of our knowledge, not been studied, their slightly more restricted version are piMCs. In particular, piMCs have interval-valued transitions with parametric bounds. Work on piMCs falls into two categories. First, *consistency* [27, 50]: is there a parameter instantiation such that the (reachable fragment of the) induced interval MC contains valid probability distributions? Second, parameter synthesis for quantitative and qualitative reachability in piMCs with up to 12 parameters [10].

Perturbation Analysis. Perturbation analysis considers the change in solution by any perturbation vector X for the parameter instantiation, whose norm is upper bounded by δ, i.e., $||X|| \leq \delta$ (or conversely, which δ ensures the solution perturbation is below a given maximum). Likewise, [21] uses the distance between two instantiations of a pMC (called augmented interval MC) to bound the change in reachability probability. Similar analyses exist for stationary distributions [1]. These problems are closely related to the verification problem in pMCs and are equally (in)tractable if there are dependencies over multiple parameters. To improve tractability, a follow-up [56] derives asymptotic bounds based on first or second-order Taylor expansions. Other approaches to perturbation analysis analyze individual paths of a system [18, 19, 30]. Sensitivity analysis in (parameter-free) imprecise MCs, a variation to rMCs, is thoroughly studied in [22].

Exploration in Learning. Similar to Q3 in Sect. 6, determining where to sample is relevant in many learning settings. Approaches such as probably approximately correct (PAC) statistical model checking [2,3] and model-based reinforcement learning [47] commonly use optimistic exploration policies [48]. By contrast, we guide exploration based on the sensitivity analysis of the solution function with respect to the parametric model.

8 Concluding Remarks

We have presented efficient methods to compute partial derivatives of the solution functions for pMCs and prMCs. For both models, we have shown how to compute these derivatives explicitly *for all parameters*, as well as how to compute only the *k highest derivatives*. Our experiments have shown that we can compute derivatives for models with over a million states and thousands of parameters. In particular, computing the k highest derivatives yields significant speed-ups compared to computing all derivatives explicitly and is feasible for prMCs which can be verified. In the future, we want to support nondeterminism in the models and apply our methods in (online) learning frameworks, in particular for settings where reducing the uncertainty is computationally expensive [42,49].

References

1. Abbas, K., Berkhout, J., Heidergott, B.: A critical account of perturbation analysis of markov chains. arXiv preprint arXiv:1609.04138 (2016)
2. Agarwal, C., Guha, S., Kretínský, J., Muruganandham, P.: PAC statistical model checking of mean payoff in discrete- and continuous-time MDP. In: CAV (2). Lecture Notes in Computer Science, vol. 13372, pp. 3–25. Springer (2022). https://doi.org/10.1007/978-3-031-13188-2_1
3. Ashok, P., Křetínský, J., Weininger, M.: PAC statistical model checking for markov decision processes and stochastic games. In: Dillig, I., Tasiran, S. (eds.) CAV 2019. LNCS, vol. 11561, pp. 497–519. Springer, Cham (2019). https://doi.org/10.1007/978-3-030-25540-4_29
4. Badings, T., Simão, T.D., Suilen, M., Jansen, N.: Decision-making under uncertainty: beyond probabilities. Int. J. Softw. Tools Technol. Transf. (2023)
5. 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)
6. Badings, T.S., Junges, S., Marandi, A., Topcu, U., Jansen, N.: Efficient sensitivity analysis for parametric robust markov chains (extended version). Tech. rep., CoRR, abs/2305.01473 (2023)
7. Baier, C., Hensel, C., Hutschenreiter, L., Junges, S., Katoen, J., Klein, J.: Parametric markov chains: PCTL complexity and fraction-free gaussian elimination. Inf. Comput. **272**, 104504 (2020)

8. Baier, C., Katoen, J.: Principles of model checking. MIT Press (2008)

9. Barratt, S.: On the differentiability of the solution to convex optimization problems. arXiv preprint arXiv:1804.05098 (2018)

10. Bart, A., Delahaye, B., Fournier, P., Lime, D., Monfroy, É., Truchet, C.: Reachability in parametric interval markov chains using constraints. Theor. Comput. Sci. **747**, 48–74 (2018)

11. Bazaraa, M.S., Jarvis, J.J., Sherali, H.D.: Linear programming and network flows. John Wiley & Sons (2011)

12. Ben-Tal, A., Ghaoui, L.E., Nemirovski, A.: Robust Optimization, Princeton Series in Applied Mathematics, vol. 28. Princeton University Press (2009)

13. Bortolussi, L., Milios, D., Sanguinetti, G.: Smoothed model checking for uncertain continuous-time markov chains. Inf. Comput. **247**, 235–253 (2016)

14. Boucheron, S., Lugosi, G., Massart, P.: Concentration Inequalities - A Nonasymptotic Theory of Independence. Oxford University Press, Oxford (2013)

15. Boyd, S.P., Vandenberghe, L.: Convex Optimization. Cambridge University Press, Cambridge (2014)

16. Buckman, J., Hafner, D., Tucker, G., Brevdo, E., Lee, H.: Sample-efficient reinforcement learning with stochastic ensemble value expansion. In: NeurIPS, pp. 8234–8244 (2018)

17. Calinescu, R., Ghezzi, C., Johnson, K., Pezzè, M., Rafiq, Y., Tamburrelli, G.: Formal verification with confidence intervals to establish quality of service properties of software systems. IEEE Trans. Reliab. **65**(1), 107–125 (2016)

18. Cao, X., Chen, H.: Perturbation realization, potentials, and sensitivity analysis of markov processes. IEEE Trans. Autom. Control **42**(10), 1382–1393 (1997)

19. Cao, X., Wan, Y.: Algorithms for sensitivity analysis of markov systems through potentials and perturbation realization. IEEE Trans. Control Syst. Technol. **6**(4), 482–494 (1998)

20. Chen, T., Feng, Y., Rosenblum, D.S., Su, G.: Perturbation analysis in verification of discrete-time markov chains. In: Baldan, P., Gorla, D. (eds.) CONCUR 2014. LNCS, vol. 8704, pp. 218–233. Springer, Heidelberg (2014). https://doi.org/10.1007/978-3-662-44584-6_16

21. Chonev, V.: Reachability in augmented interval markov chains. In: Filiot, E., Jungers, R., Potapov, I. (eds.) RP 2019. LNCS, vol. 11674, pp. 79–92. Springer, Cham (2019). https://doi.org/10.1007/978-3-030-30806-3_7

22. Cooman, G.D., Hermans, F., Quaeghebeur, E.: Sensitivity analysis for finite markov chains in discrete time. In: UAI, pp. 129–136. AUAI Press (2008)

23. 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)

24. Daws, C.: Symbolic and Parametric Model Checking of Discrete-Time Markov Chains. In: Liu, Z., Araki, K. (eds.) ICTAC 2004. LNCS, vol. 3407, pp. 280–294. Springer, Heidelberg (2005). https://doi.org/10.1007/978-3-540-31862-0_21

25. Dehnert, C., et al.: PROPhESY: a probabilistic parameter synthesis tool. In: Kroening, D., Păsăreanu, C.S. (eds.) CAV 2015. LNCS, vol. 9206, pp. 214–231. Springer, Cham (2015). https://doi.org/10.1007/978-3-319-21690-4_13

26. Delahaye, B.: Consistency for parametric interval markov chains. In: SynCoP. OASIcs, vol. 44, pp. 17–32. Schloss Dagstuhl - Leibniz-Zentrum für Informatik (2015)

27. Delahaye, B., Lime, D., Petrucci, L.: Parameter synthesis for parametric interval markov chains. In: Jobstmann, B., Leino, K.R.M. (eds.) VMCAI 2016. LNCS, vol. 9583, pp. 372–390. Springer, Heidelberg (2016). https://doi.org/10.1007/978-3-662-49122-5_18

28. Fang, X., Calinescu, R., Gerasimou, S., Alhwikem, F.: Fast parametric model checking through model fragmentation. In: ICSE, pp. 835–846. IEEE (2021)

29. Filieri, A., Tamburrelli, G., Ghezzi, C.: Supporting self-adaptation via quantitative verification and sensitivity analysis at run time. IEEE Trans. Softw. Eng. **42**(1), 75–99 (2016)

30. Fu, M.C., Hu, J.: Smoothed perturbation analysis derivative estimation for markov chains. Oper. Res. Lett. **15**(5), 241–251 (1994)

31. Gurobi Optimization, LLC: Gurobi optimizer reference manual (2023). https://www.gurobi.com

32. Hahn, E.M., Hermanns, H., Zhang, L.: Probabilistic reachability for parametric markov models. Int. J. Softw. Tools Technol. Transf. **13**(1), 3–19 (2011)

33. Hartmanns, A., Klauck, M., Parker, D., Quatmann, T., Ruijters, E.: The quantitative verification benchmark set. In: Vojnar, T., Zhang, L. (eds.) TACAS 2019. LNCS, vol. 11427, pp. 344–350. Springer, Cham (2019). https://doi.org/10.1007/978-3-030-17462-0_20

34. Heck, L., Spel, J., Junges, S., Moerman, J., Katoen, J.-P.: Gradient-descent for randomized controllers under partial observability. In: Finkbeiner, B., Wies, T. (eds.) VMCAI 2022. LNCS, vol. 13182, pp. 127–150. Springer, Cham (2022). https://doi.org/10.1007/978-3-030-94583-1_7

35. Hensel, C., Junges, S., Katoen, J.P., Quatmann, T., Volk, M.: The probabilistic model checker Storm. Softw. Tools Technol. Transf. (2021)

36. Hoffman, A.J., Kruskal, J.B.: Integral boundary points of convex polyhedra. In: Jünger, M., Liebling, T.M., Naddef, D., Nemhauser, G.L., Pulleyblank, W.R., Reinelt, G., Rinaldi, G., Wolsey, L.A. (eds.) 50 Years of Integer Programming 1958-2008, pp. 49–76. Springer, Heidelberg (2010). https://doi.org/10.1007/978-3-540-68279-0_3

37. Hoffman, A.J., Kruskal, J.B.: Integral boundary points of convex polyhedra. 50 Years of integer programming 1958–2008, p. 49 (2010)

38. 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). https://doi.org/10.1007/978-3-031-22337-2_20

39. Jonsson, B., Larsen, K.G.: Specification and refinement of probabilistic processes. In: LICS, pp. 266–277. IEEE Computer Society (1991)

40. Junges, S., et al.: Parameter synthesis for markov models. CoRR abs/1903.07993 (2019)

41. 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)
42. Junges, S., Spaan, M.T.J.: Abstraction-refinement for hierarchical probabilistic models. In: CAV (1). Lecture Notes in Computer Science, vol. 13371, pp. 102–123. Springer (2022). https://doi.org/10.1007/978-3-031-13185-1_6
43. Kakade, S.M.: On the sample complexity of reinforcement learning. Ph.D. thesis, University of London, University College London (United Kingdom) (2003)
44. Kwiatkowska, M., Norman, G., Parker, D.: PRISM 4.0: verification of probabilistic real-time systems. In: Gopalakrishnan, G., Qadeer, S. (eds.) CAV 2011. LNCS, vol. 6806, pp. 585–591. Springer, Heidelberg (2011). https://doi.org/10.1007/978-3-642-22110-1_47
45. Lanotte, R., Maggiolo-Schettini, A., Troina, A.: Parametric probabilistic transition systems for system design and analysis. Formal Aspects Comput. **19**(1), 93–109 (2007)
46. Matousek, J., Gärtner, B.: Integer Programming and LP Relaxation, pp. 29–40. Springer, Berlin Heidelberg (2007). https://doi.org/10.1007/978-3-540-30717-4_3
47. Moerland, T.M., Broekens, J., Jonker, C.M.: Model-based reinforcement learning: a survey. CoRR abs/2006.16712 (2020)
48. Munos, R.: From bandits to monte-carlo tree search: The optimistic principle applied to optimization and planning. Found. Trends Mach. Learn. **7**(1), 1–129 (2014)
49. Neary, C., Verginis, C.K., Cubuktepe, M., Topcu, U.: Verifiable and compositional reinforcement learning systems. In: ICAPS, pp. 615–623. AAAI Press (2022)
50. Petrucci, L., van de Pol, J.: Parameter synthesis algorithms for parametric interval markov chains. In: Baier, C., Caires, L. (eds.) FORTE 2018. LNCS, vol. 10854, pp. 121–140. Springer, Cham (2018). https://doi.org/10.1007/978-3-319-92612-4_7
51. Polgreen, E., Wijesuriya, V.B., Haesaert, S., Abate, A.: Automated experiment design for data-efficient verification of parametric markov decision processes. In: Bertrand, N., Bortolussi, L. (eds.) QEST 2017. LNCS, vol. 10503, pp. 259–274. Springer, Cham (2017). https://doi.org/10.1007/978-3-319-66335-7_16
52. Puggelli, A., Li, W., Sangiovanni-Vincentelli, A.L., Seshia, S.A.: Polynomial-time verification of PCTL properties of MDPs with convex uncertainties. In: Sharygina, N., Veith, H. (eds.) CAV 2013. LNCS, vol. 8044, pp. 527–542. Springer, Heidelberg (2013). https://doi.org/10.1007/978-3-642-39799-8_35
53. Puterman, M.L.: Markov Decision Processes: Discrete Stochastic Dynamic Programming. Wiley Series in Probability and Statistics, Wiley (1994)
54. Sen, K., Viswanathan, M., Agha, G.: Model-checking markov chains in the presence of uncertainties. In: Hermanns, H., Palsberg, J. (eds.) TACAS 2006. LNCS, vol. 3920, pp. 394–410. Springer, Heidelberg (2006). https://doi.org/10.1007/11691372_26
55. Spel, J., Junges, S., Katoen, J.-P.: Finding provably optimal markov chains. In: TACAS 2021. LNCS, vol. 12651, pp. 173–190. Springer, Cham (2021). https://doi.org/10.1007/978-3-030-72016-2_10
56. Su, G., Feng, Y., Chen, T., Rosenblum, D.S.: Asymptotic perturbation bounds for probabilistic model checking with empirically determined probability parameters. IEEE Trans. Software Eng. **42**(7), 623–639 (2016)

57. Suilen, M., Simão, T.D., Parker, D., Jansen, N.: Robust anytime learning of markov decision processes. In: NeurIPS, vol. 35, pp. 28790–28802. Curran Associates, Inc. (2022)
58. Wiesemann, W., Kuhn, D., Sim, M.: Distributionally robust convex optimization. Oper. Res. **62**(6), 1358–1376 (2014)
59. 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)
60. Wolsey, L.A.: Integer programming. John Wiley & Sons (2020)