Sequential Convex Programming for the Efficient Verification of Parametric MDPs*

Murat Cubuktepe[1], Nils Jansen[1], Sebastian Junges[2], Joost-Pieter Katoen[2], Ivan Papusha[1], Hasan A. Poonawala[1], Ufuk Topcu[1]

[1] The University of Texas at Austin, USA
[2] RWTH Aachen University, Germany

Abstract. Multi-objective verification problems of parametric Markov decision processes under optimality criteria can be naturally expressed as nonlinear programs. We observe that many of these computationally demanding problems belong to the subclass of signomial programs. This insight allows for a sequential optimization algorithm to efficiently compute sound but possibly suboptimal solutions. Each stage of this algorithm solves a geometric programming problem. These geometric programs are obtained by convexifying the nonconvex constraints of the original problem. Direct applications of the encodings as nonlinear programs are model repair and parameter synthesis. We demonstrate the scalability and quality of our approach by well-known benchmarks.

1 Introduction

We study the applicability of *convex optimization* to the formal verification of systems that exhibit randomness or stochastic uncertainties. Such systems are formally represented by so-called parametric Markov models.

In fact, many real-world systems exhibit random behavior and stochastic uncertainties. One major example is in the field of *robotics*, where the presence of measurement noise or input disturbances requires special controller synthesis techniques [39] that achieve robustness of robot actions against uncertainties in the robot model and the environment. On the other hand, formal verification offers methods for rigorously proving or disproving properties about the system behavior, and synthesizing strategies that satisfy these properties. In particular, *model checking* [36] is a well-studied technique that provides guarantees on appropriate behavior for all possible events and scenarios.

Model checking can be applied to systems with stochastic uncertainties, including discrete-time Markov chains (MCs), Markov decision processes (MDPs), and their continuous-time counterparts [31]. Probabilistic model checkers are able to verify reachability properties like "the probability of reaching a set of unsafe

* Partly funded by the awards AFRL # FA8650-15-C-2546, DARPA # W911NF-16-1-0001, ARO # W911NF-15-1-0592, ONR # N00014-15-IP-00052, ONR # N00014-16-1-3165, and NSF # 1550212. Also funded by the Excellence Initiative of the German federal and state government and the CDZ project CAP (GZ 1023).

states is $\leq 10\%$" and expected costs properties like "the expected cost of reaching a goal state is ≤ 20." A rich set of properties, specified by linear- and branching-time logics, reduces to such properties [31]. Tools like PRISM [15], STORM [29], and iscasMc [22] are probabilistic model checkers capable of handling a wide range of large-scale problems.

Key requirements for applying model checking are a reliable system model and formal specifications of desired or undesired behaviors. As a result, most approaches assume that models of the stochastic uncertainties are precisely given. For example, if a system description includes an environmental disturbance, the mean of that disturbance should be known *before* formal statements are made about expected system behavior. However, the desire to treat many applications where uncertainty measures (e.g., faultiness, reliability, reaction rates, packet loss ratio) are not exactly known at design time gives rise to *parametric* probabilistic models [1, 30]. Here, transition probabilities are expressed as functions over system parameters, i.e., *descriptions of uncertainties*. In this setting, *parameter synthesis* addresses the problem of computing parameter instantiations leading to satisfaction of system specifications. More precisely, parameters are mapped to concrete probabilities inducing the resulting *instantiated* model to satisfy specifications. A direct application is *model repair* [13], where a concrete model (without parameters) is changed (repaired) such that specifications *are* satisfied.

Dedicated tools like PARAM [11], PRISM [15], or PROPhESY [25] compute rational functions over parameters that express reachability probabilities or expected costs in a parametric Markov chain (pMC). These optimized tools work with millions of states but are restricted to a few parameters, as the necessary computation of greatest common divisors does not scale well with the number of parameters. Moreover, the resulting functions are inherently *nonlinear* and often of high degree. Evaluation by an SMT solver over nonlinear arithmetic such as Z3 [17] suffers from the fact that the solving procedures are *exponential in the degree of polynomials and the number of variables*.

This paper takes an alternative perspective. We discuss a general nonlinear programming formulation for the verification of parametric Markov decision processes (pMDPs). The powerful modeling capabilities of nonlinear programs (NLPs) enable incorporating multi-objective properties and penalties on the parameters of the pMDP. However, because of their generality, solving NLPs to find a global optimum is difficult. Even feasible solutions (satisfying the constraints) cannot always be computed efficiently [37, 5]. In contrast, for the class of NLPs called *convex optimization* problems, efficient methods to compute feasible solutions and global optima even for large-scale problems are available [38].

We therefore propose a novel automated method of utilizing convex optimization for pMDPs. Many NLP problems for pMDPs belong to the class of *signomial programs* (SGPs), a certain class of nonconvex optimization problems. For instance, all benchmarks available at the PARAM–webpage [26] belong to this class. Restricting the general pMDP problem accordingly yields a direct and efficient synthesis method—formulated as an NLP—for a large class of pMDP problems. We list the two main technical results of this paper:

1. We relax nonconvex constraints in SGPs and apply a simple transformation to the parameter functions. The resulting programs are *geometric programs* (GPs) [7], a class of *convex programs*. We show that a solution to the relaxed GP induces feasibility (satisfaction of all specifications) in the original pMDP problem. Note that solving GPs is *polynomial* in the number of variables.
2. Given an initial feasible solution, we use a technique called *sequential convex programming* [7] to improve a signomial objective. This local optimization method for nonconvex problems leverages convex optimization by solving a sequence of convex approximations (GPs) of the original SGP.

Sequential convex programming is known to efficiently find a feasible solution with good, though not necessarily globally optimal, objective values [7, 8]. We initialize the sequence with a feasible solution (obtained from the GP) of the original problem and compute a *trust region*. Inside this region, the optimal value of the approximation of the SGP is at least as good as the objective value at the feasible solution of the GP. The optimal solution of the approximation is then the initial point of the next iteration with a new trust region. This procedure is iterated to approximate a local optimum of the original problem.

Utilizing our results, we discuss the concrete problems of parameter synthesis and model repair for multiple specifications for pMDPs. Experimental results with a prototype implementation show the applicability of our optimization methods to benchmarks of up to 10^5 states. As solving GPs is polynomial in the number of variables, our approaches are relatively insensitive to the number of parameters in pMDPs. This is an improvement over state-of-the-art approaches that leverage SMT, which—for our class of problems—scale exponentially in variables and the degree of polynomials. This is substantiated by our experiments.

Related work. Several approaches exist for pMCs [11, 25, 12, 23] while the number of approaches for pMDPs [12, 33] is limited. Ceska *et al.* [21] synthesize rate parameters in stochastic biochemical networks. Multi-objective model checking of non-parametric MDPs [9] is a convex problem [14]. Bortolussi *et al.* [28] developed a Bayesian statistical algorithm for properties on stochastic population models. Convex uncertainties in MDPs without parameter dependencies are discussed in [20]. Parametric probabilistic models are used to rank patches in the repair of software [32] and to compute perturbation bounds [24, 34].

2 Preliminaries

A *probability distribution* over a finite or countably infinite set X is a function $\mu\colon X \to [0,\,1] \subseteq \mathbb{R}$ with $\sum_{x\in X} \mu(x) = 1$. The set of all distributions on X is denoted by $\mathit{Distr}(X)$.

Definition 1 (Monomial, Posynomial, Signomial). *Let $V = \{x_1, \ldots, x_n\}$ be a finite set of strictly positive real-valued* variables. *A* monomial *over V is an expression of the form*

$$g = c \cdot x_1^{a_1} \cdots x_n^{a_n} \ ,$$

where $c \in \mathbb{R}_{>0}$ is a positive coefficient, and $a_i \in \mathbb{R}$ are exponents for $1 \leq i \leq n$. A posynomial *over V is a sum of one or more monomials:*

$$f = \sum_{k=1}^{K} c_k \cdot x_1^{a_{1k}} \cdots x_n^{a_{nk}} \ . \tag{1}$$

If c_k is allowed to be a negative real number for any $1 \leq k \leq K$, then the expression (1) *is a* signomial. *The sets of all monomials, posynomials, and signomials over V are denoted by Mon_V, Pos_V, and Sig_V, respectively.*

This definition of monomials differs from the standard algebraic definition where exponents are positive integers with no restriction on the coefficient sign. A sum of monomials is then called a *polynomial*. Our definitions are consistent with [7].

Definition 2 (Valuation). *For a set of real-valued variables V, a* valuation u over V *is a function $u \colon V \to \mathbb{R}$. The set of all valuations over V is Val^V.*

Applying valuation u to monomial g over V yields a real number $g[u] \in \mathbb{R}$ by replacing each occurrence of variables $x \in V$ in g by $u(x)$; the procedure is analogous for posynomials and signomials using standard arithmetic operations.

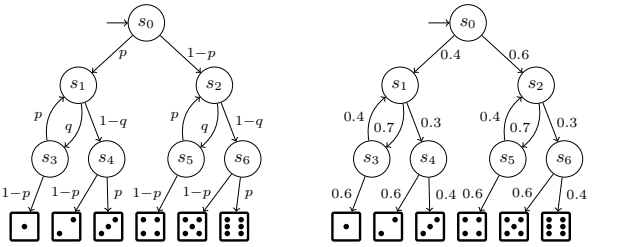
Definition 3 (pMDP and pMC). *A* parametric Markov decision process (pMDP) *is a tuple $\mathcal{M} = (S, s_I, Act, V, \mathcal{P})$ with a finite set S of states, an initial state $s_I \in S$, a finite set Act of actions, a finite set of real-valued variables V, and a transition function $\mathcal{P} \colon S \times Act \times S \to Sig_V$ satisfying for all $s \in S \colon Act(s) \neq \emptyset$, where $Act(s) = \{\alpha \in Act \mid \exists s' \in S. \mathcal{P}(s, \alpha, s') \neq 0\}$. If for all $s \in S$ it holds that $|Act(s)| = 1$, $\mathcal{M}$ is called a* parametric discrete-time Markov chain (pMC).

$Act(s)$ is the set of *enabled* actions at state s; as $Act(s) \neq \emptyset$, there are no deadlock states. *Costs* are defined using a state–action *cost function* $c \colon S \times Act \to \mathbb{R}_{\geq 0}$.

Remark 1. Largely due to algorithmic reasons, the transition probabilities in the literature [12, 25, 33] are polynomials or rational functions, i.e., fractions of polynomials. Our restriction to signomials is realistic; *all* benchmarks from the PARAM–webpage [26] contain only signomial transition probabilities.

A pMDP $\mathcal{M}$ is a *Markov decision process (MDP)* if the transition function is a valid probability distribution, i.e., $\mathcal{P} \colon S \times Act \times S \to [0, 1]$ and $\sum_{s' \in S} \mathcal{P}(s, \alpha, s') = 1$ for all $s \in S$ s.t. $\alpha \in Act(s)$. Analogously, a Markov chain (MC) is a special class of a pMC; a model is *parameter-free* if all probabilities are constant. Applying a *valuation* u to a pMDP, denoted $\mathcal{M}[u]$, replaces each signomial f in $\mathcal{M}$ by $f[u]$; we call $\mathcal{M}[u]$ the *instantiation* of $\mathcal{M}$ at u. The application of u is to replace the transition function f by the probability $f[u]$. A valuation u is *well-defined* for $\mathcal{M}$ if the replacement yields *probability distributions* at all states; the resulting model $\mathcal{M}[u]$ is an MDP or an MC.

Example 1 (pMC). Consider a variant of the Knuth–Yao model of a die [2], where a six-sided die is simulated by successive coin flips. We alternate flipping two biased coins, which result in *heads* with probabilities defined by the monomials



(a) pMC model (b) Instantiation using p=0.4 and q=0.7

Fig. 1. A variant of the Knuth–Yao die for unfair coins.

p and q, respectively. Consequently, the probability for *tails* is given by the signomials $1-p$ and $1-q$, respectively. The corresponding pMC is depicted in Fig. 1(a); and the *instantiated* MC for $p = 0.4$ and $q = 0.7$ is given in Fig. 1(b). Note that we omit actions, as the model is deterministic.

In order to define a probability measure and expected cost on MDPs, nondeterministic choices are resolved by so-called *schedulers*. For practical reasons we restrict ourselves to *memoryless* schedulers; details can be found in [36].

Definition 4 (Scheduler). *A (randomized)* scheduler *for an MDP $\mathcal{M}$ is a function $\sigma\colon S \to Distr(Act)$ such that $\sigma(s)(\alpha) > 0$ implies $\alpha \in Act(s)$. The set of all schedulers over $\mathcal{M}$ is denoted by $Sched^{\mathcal{M}}$.*

Applying a scheduler to an MDP yields a so-called *induced Markov chain*.

Definition 5 (Induced MC). *Let MDP $\mathcal{M} = (S, s_I, Act, \mathcal{P})$ and scheduler $\sigma \in Sched^{\mathcal{M}}$. The* MC induced by $\mathcal{M}$ and σ *is $\mathcal{M}^\sigma = (S, s_I, Act, \mathcal{P}^\sigma)$ where for all $s, s' \in S$,*

$$\mathcal{P}^\sigma(s, s') = \sum_{\alpha \in Act(s)} \sigma(s)(\alpha) \cdot \mathcal{P}(s, \alpha, s').$$

We consider *reachability properties* and *expected cost properties*. For MC $\mathcal{D}$ with states S, let $\Pr_s^{\mathcal{D}}(\Diamond T)$ denote the probability of reaching a set of *target states* $T \subseteq S$ from state $s \in S$; simply $\Pr^{\mathcal{D}}(\Diamond T)$ denotes the probability for initial state s_I. We use the standard probability measure as in [36, Ch. 10]. For threshold $\lambda \in [0, 1]$, the *reachability property* asserting that a target state is to be reached with probability at most λ is denoted $\varphi = \mathbb{P}_{\leq\lambda}(\Diamond T)$. The property is satisfied by $\mathcal{D}$, written $\mathcal{D} \models \varphi$, iff $\Pr^{\mathcal{D}}(\Diamond T) \leq \lambda$.

The cost of a path through MC $\mathcal{D}$ until a set of *goal states* $G \subseteq S$ is the sum of action costs visited along the path. The expected cost of a finite path is the product of its probability and its cost. For $\Pr^{\mathcal{D}}(\Diamond G) = 1$, the expected cost of reaching G is the sum of expected costs of all paths leading to G. An expected cost property $\mathrm{EC}_{\leq\kappa}(\Diamond G)$ is satisfied if the expected cost of reaching T is bounded by a threshold $\kappa \in \mathbb{R}$. Formal definitions are given in e.g., [36].

If multiple specifications $\varphi_1, \ldots, \varphi_q$ are given, which are either reachability properties or expected cost properties of the aforementioned forms, we write the satisfaction of all specifications $\varphi_1, \ldots, \varphi_q$ for an MC $\mathcal{D}$ as $\mathcal{D} \models \varphi_1 \wedge \ldots \wedge \varphi_q$.

An MDP $\mathcal{M}$ satisfies the specifications $\varphi_1, \ldots, \varphi_q$, iff *for all* schedulers $\sigma \in \mathit{Sched}^{\mathcal{M}}$ it holds that $\mathcal{M}^\sigma \models \varphi_1 \wedge \ldots \wedge \varphi_q$. The verification of multiple specifications is also referred to as *multi-objective model checking* [9, 16]. We are also interested in the so-called scheduler *synthesis problem*, where the aim is to find a scheduler σ such that the specifications are satisfied (although other schedulers may not satisfy the specifications).

3 Nonlinear programming for pMDPs

In this section we formally state a general pMDP parameter synthesis problem and describe how it can be formulated using nonlinear programming.

3.1 Formal problem statement

Problem 1. Given a pMDP $\mathcal{M} = (S, s_I, \mathit{Act}, V, \mathcal{P})$, specifications $\varphi_1, \ldots, \varphi_q$ that are either probabilistic reachability properties or expected cost properties, and an objective function $f \colon V \to \mathbb{R}$ over the variables V, compute a well-defined valuation $u \in \mathit{Val}^V$ for $\mathcal{M}$, and a (randomized) scheduler $\sigma \in \mathit{Sched}^{\mathcal{M}}$ such that the following conditions hold:

(a) *Feasibility*: the Markov chain $\mathcal{M}^\sigma[u]$ induced by scheduler σ and instantiated by valuation u satisfies the specifications, i.e., $\mathcal{M}^\sigma[u] \models \varphi_1 \wedge \ldots \wedge \varphi_q$.
(b) *Optimality*: the objective f is minimized.

Intuitively, we wish to compute a parameter valuation and a scheduler such that all specifications are satisfied, and the objective is globally minimized. We refer to a valuation–scheduler pair (u, σ) that satisfies condition (a), i.e., only guarantees satisfaction of the specifications but does not necessarily minimize the objective f, as a *feasible* solution to the pMDP synthesis problem. If both (a) and (b) are satisfied, the pair is an *optimal* solution to the pMDP synthesis problem.

3.2 Nonlinear encoding

We now provide an NLP encoding of Problem 1. A general NLP over a set of real-valued variables $\mathcal{V}$ can be written as

$$\begin{aligned} &\text{minimize} && f && (2)\\ &\text{subject to} && && \\ &\forall i.\, 1 \le i \le m && g_i \le 0, && (3)\\ &\forall j.\, 1 \le i \le p && h_j = 0, && (4) \end{aligned}$$

where f, g_i, and h_j are arbitrary functions over $\mathcal{V}$, and m and p are the number of inequality and equality constraints of the program respectively. Tools like IPOPT [10] solve small instances of such problems.

Consider a pMDP $\mathcal{M} = (S, s_I, Act, V, \mathcal{P})$ with specifications $\varphi_1 = \mathbb{P}_{\leq\lambda}(\lozenge T)$ and $\varphi_2 = \mathrm{EC}_{\leq\kappa}(\lozenge G)$. We will discuss how additional specifications of either type can be encoded. The set $\mathcal{V} = V \cup W$ of variables of the NLP consists of the variables V that occur in the pMDP as well as a set W of additional variables:

- $\{\sigma^{s,\alpha} \mid s \in S, \alpha \in Act(s)\}$, which define the randomized scheduler σ by $\sigma(s)(\alpha) = \sigma^{s,\alpha}$.
- $\{p_s \mid s \in S\}$, where p_s is the probability of reaching the target set $T \subseteq S$ from state s under scheduler σ, and
- $\{c_s \mid s \in S\}$, where c_s is the expected cost to reach $G \subseteq S$ from s under σ.

A valuation over $\mathcal{V}$ consists of a valuation $u \in Val^V$ over the pMDP variables and a valuation $w \in Val^W$ over the additional variables.

$$\begin{aligned}
&\text{minimize} && f && (5)\\
&\text{subject to} &&&&\\
& && p_{s_I} \leq \lambda, && (6)\\
& && c_{s_I} \leq \kappa, && (7)\\
&\forall s \in S. && \sum_{\alpha \in Act(s)} \sigma^{s,\alpha} = 1, && (8)\\
&\forall s \in S\ \forall \alpha \in Act(s). && 0 \leq \sigma^{s,\alpha} \leq 1, && (9)\\
&\forall s \in S\ \forall \alpha \in Act(s). && \sum_{s' \in S} \mathcal{P}(s,\alpha,s') = 1, && (10)\\
&\forall s, s' \in S\ \forall \alpha \in Act(s). && 0 \leq \mathcal{P}(s,\alpha,s') \leq 1, && (11)\\
&\forall s \in T. && p_s = 1, && (12)\\
&\forall s \in S \setminus T. && p_s = \sum_{\alpha \in Act(s)} \sigma^{s,\alpha} \cdot \sum_{s' \in S} \mathcal{P}(s,\alpha,s') \cdot p_{s'}, && (13)\\
&\forall s \in G. && c_s = 0, && (14)\\
&\forall s \in S \setminus G. && c_s = \sum_{\alpha \in Act(s)} \sigma^{s,\alpha} \cdot \Big(c(s,\alpha) + \sum_{s' \in S} \mathcal{P}(s,\alpha,s') \cdot c_{s'} \Big). && (15)
\end{aligned}$$

The NLP (5)–(15) encodes Problem 1 in the following way. The objective function f in (5) is any real-valued function over the variables $\mathcal{V}$. The constraints (6) and (7) encode the specifications φ_1 and φ_2, respectively. The constraints (8)–(9) ensure that the scheduler obtained is well-defined by requiring that the scheduler variables at each state sum to unity. Similarly, the constraints (10)–(11) ensure that for all states, parameters from V are instantiated such that probabilities sum up to one. (These constraints are included if not all probabilities at a state are constant.) The probability of reaching the target for all states in the target set is set to one using (12). The reachability probabilities in each state depend on the reachability of the successor states and the transition probabilities to those states through (13). Analogously to the reachability probabilities, the cost for

each goal state $G \subseteq S$ must be zero, thereby precluding the collection of infinite cost at absorbing states, as enforced by (14). Finally, the expected cost for all states except target states is given by the equation (15), where according to the strategy σ the cost of each action is added to the expected cost of the successors.

We can readily extend the NLP to include more specifications. If another reachability property $\varphi' = \mathbb{P}_{\leq \lambda'}(\Diamond T')$ is given, we add the set of probability variables $\{p'_s \mid s \in S\}$ to W, and duplicate the constraints (12)–(13) accordingly. To ensure satisfaction of φ', we also add the constraint $p'_{s_I} \leq \lambda'$. The procedure is similar for additional expected cost properties. By construction, we have the following result relating the NLP encoding and Problem 1.

Theorem 1. *The NLP* (5)–(15) *is sound and complete with respect to Problem 1.*

We refer to soundness in the sense that each variable assignment that satisfies the constraints induces a scheduler and a valuation of parameters such that a feasible solution of the problem is induced. Moreover, any optimal solution to the NLP induces an optimal solution of the problem. Completeness means that all possible solutions of the problem can be encoded by this NLP; while unsatisfiability means that no such solution exists, making the problem *infeasible*.

Signomial programs. By Def. 1 and 3, all constraints in the NLP consist of signomial functions. A special class of NLPs known as *signomial programs* (SGPs) is of the form (2)–(4) where f, g_i and h_j are signomials over $\mathcal{V}$, see Def. 1. Therefore, we observe that the NLP (5)–(15) is an SGP. We will refer to the NLP as an SGP in what follows.

SGPs with equality constraints consisting of functions that are *not affine* are not *convex* in general. In particular, the SGP (5)–(15) is not necessarily convex. Consider a simple pMC only having transition probabilities of the form p and $1 - p$, as in Example 1. The function in the equality constraint (13) of the corresponding SGP encoding is not affine in parameter p and the probability variable p_s for some state $s \in S$. More generally, the equality constraints (10), (13), and (15) involving $\mathcal{P}$ are not necessarily affine, and thus the SGP may not be a convex program [38]. Whereas for convex programs *global optimal solutions* can be found efficiently [38], such guarantees are not given for SGPs. However, we can efficiently obtain local optimal solutions for SGPs in our setting, as shown in the following sections.

4 Convexification

We investigate how to transform the SGP (5)–(15) into a convex program by relaxing equality constraints and a lifting of variables of the SGP. A certain subclass of SGPs called *geometric programs* (GPs) can be transformed into convex programs [7, §2.5] and solved efficiently. A GP is an SGP of the form (2)–(4) where $f, g_i \in \mathit{Pos}_{\mathcal{V}}$ and $h_j \in \mathit{Mon}_{\mathcal{V}}$. We will refer to a constraint with posynomial or monomial function as a posynomial or monomial constraint, respectively.

4.1 Transformation and relaxation of equality constraints

As discussed before, the SGP (5)–(15) is not convex because of the presence of non-affine equality constraints. First observe the following transformation [7]:

$$f \leq h \iff \frac{f}{h} \leq 1, \tag{16}$$

for $f \in Pos_{\mathcal{V}}$ and $h \in Mon_{\mathcal{V}}$. Note that monomials are strictly positive (Def. 1). This *(division-)transformation* of $f \leq h$ yields a *posynomial inequality constraint*.

We *relax* all equality constraints of SGP (5)–(15) that are not monomials to inequalities, then we apply the division-transformation wherever possible. Constraints (6), (7), (8), (10), (13), and (15) are transformed to

$$\frac{p_{s_I}}{\lambda} \leq 1, \tag{17}$$

$$\frac{c_{s_I}}{\kappa} \leq 1, \tag{18}$$

$$\forall s \in S. \quad \sum_{\alpha \in Act(s)} \sigma^{s,\alpha} \leq 1, \tag{19}$$

$$\forall s \in S \, \forall \alpha \in Act(s). \quad \sum_{s' \in S} \mathcal{P}(s, \alpha, s') \leq 1, \tag{20}$$

$$\forall s \in S \setminus T. \quad \frac{\sum_{\alpha \in Act(s)} \sigma^{s,\alpha} \cdot \sum_{s' \in S} \mathcal{P}(s, \alpha, s') \cdot p_{s'}}{p_s} \leq 1, \tag{21}$$

$$\forall s \in S \setminus G. \quad \frac{\sum_{\alpha \in Act(s)} \sigma^{s,\alpha} \cdot \Big(c(s, \alpha) + \sum_{s' \in S} \mathcal{P}(s, \alpha, s') \cdot c_{s'} \Big)}{c_s} \leq 1. \tag{22}$$

These constraints are not necessarily posynomial inequality constraints because (as in Def. 3) we allow signomial expressions in the transition probability function $\mathcal{P}$. Therefore, replacing (6), (7), (8), (10), (13), and (15) in the SGP with (17)–(22) does not by itself convert the SGP to a GP.

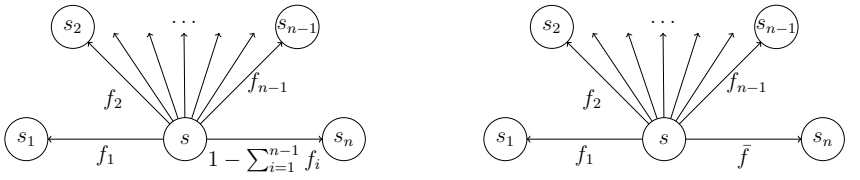
4.2 Convexification by lifting

The relaxed equality constraints (20)–(22) involving $\mathcal{P}$ are signomial, rather than posynomial, because the parameters enter Problem 1 in signomial form. Specifically, consider the relaxed equality constraint (21) at s_0 in Example 1,

$$\frac{p \cdot p_{s_1} + (1 - p) \cdot p_{s_2}}{p_{s_0}} \leq 1. \tag{23}$$

The term $(1 - p) \cdot p_{s_2}$ is signomial in p and p_{s_2}. We *lift* by introducing a new variable $\bar{p} = 1 - p$, and rewrite (23) as a posynomial inequality constraint and an equality constraint in the lifted variables:

$$\frac{p \cdot p_{s_1} + \bar{p} \cdot p_{s_2}}{p_{s_0}} \leq 1, \quad \bar{p} = 1 - p. \tag{24}$$



(a) signomial transition functions (b) posynomial transition functions

Fig. 2. Lifting of signomial transition probability function.

We relax the (non-monomial) equality constraint to $p + \bar{p} \leq 1$. More generally, we restrict the way parameters occur in $\mathcal{P}$ as follows. Refer to Fig. 2(a). For every state $s \in S$ and every action $\alpha \in Act(s)$ we require that there exists at most one state $\bar{s} \in S$ such that $\mathcal{P}(s, \alpha, \bar{s}) \in Sig_V$ and $\mathcal{P}(s, \alpha, s') \in Pos_V$ for all $s' \in S \setminus \{\bar{s}\}$. In particular, we require that

$$\underbrace{\mathcal{P}(s, \alpha, \bar{s})}_{\in Sig_V} = 1 - \sum_{s' \in S \setminus \{\bar{s}\}} \underbrace{\mathcal{P}(s, \alpha, s')}_{\in Pos_V} \ .$$

This requirement is met by all benchmarks available at the PARAM–webpage [26]. In general, we lift by introducing a new variable $\bar{p}_{s,\alpha,\bar{s}} = \mathcal{P}(s, \alpha, \bar{s})$ for each such state $s \in S$; refer to Fig. 2(b). We denote this set of *lifting variables* by L. Lifting as explained above then creates a new transition probability function $\bar{\mathcal{P}}$ where for every $s, s' \in S$ and $\alpha \in Act$ we have $\bar{\mathcal{P}}(s, \alpha, s') \in Pos_{V \cup L}$.

We call the set of constraints obtained through transformation, relaxation, and lifting of every constraint of the SGP (6)–(15) as shown above the *convexified constraints*. Any posynomial objective subject to the convexified constraints forms by construction a GP over the pMDP parameters V, the SGP additional variables W, and the lifting variables L.

4.3 Tightening the constraints

A solution of the GP as obtained in the previous section does not have a direct relation to the original SGP (5)–(15). In particular, a solution to the GP may not have the relaxed constraints satisfied with equality. For (19) and (20), the induced parameter valuation and the scheduler are not well-defined, i.e., the probabilities may not sum to one. We need to relate the relaxed and lifted GP to Problem 1. By defining a *regularization function* F over all parameter and scheduler variables, we ensure that the constraints are satisfied with equality; enforcing well-defined probability distributions.

$$F = \sum_{p \in V} \frac{1}{p} + \sum_{\bar{p} \in L} \frac{1}{\bar{p}} + \sum_{s \in S, \alpha \in Act(s)} \frac{1}{\sigma_{s,\alpha}} \ . \tag{25}$$

The function F is monotone in all its variables. We discard the original objective f in (5) and form a GP with the regularization objective F (25):

$$\begin{aligned}
&\text{minimize} && F && (26)\\
&\text{subject to}\\
& && \frac{p_{s_I}}{\lambda} \leq 1, && (27)\\
& && \frac{c_{s_I}}{\kappa} \leq 1, && (28)\\
&\forall s \in S. && \sum_{\alpha \in Act(s)} \sigma^{s,\alpha} \leq 1, && (29)\\
&\forall s \in S\, \forall \alpha \in Act(s). && \sigma^{s,\alpha} \leq 1, && (30)\\
&\forall s \in S\, \forall \alpha \in Act(s). && \sum_{s' \in S} \bar{\mathcal{P}}(s,\alpha,s') \leq 1, && (31)\\
&\forall s, s' \in S\, \forall \alpha \in Act(s). && \bar{\mathcal{P}}(s,\alpha,s') \leq 1, && (32)\\
&\forall s \in T. && p_s = 1, && (33)\\
&\forall s \in S \setminus T. && \frac{\sum_{\alpha \in Act(s)} \sigma^{s,\alpha} \cdot \sum_{s' \in S} \bar{\mathcal{P}}(s,\alpha,s') \cdot p_{s'}}{p_s} \leq 1, && (34)\\
&\forall s \in S \setminus G. && \frac{\sum_{\alpha \in Act(s)} \sigma^{s,\alpha} \cdot \left(c(s,\alpha) + \sum_{s' \in S} \bar{\mathcal{P}}(s,\alpha,s') \cdot c_{s'} \right)}{c_s} \leq 1. && (35)
\end{aligned}$$

Since the objective F (25) and the inequality constraints (29) and (31) are monotone in V, L, and the scheduler variables, each optimal solution for a feasible problem satisfies them with equality. We obtain a well-defined scheduler σ and a valuation u as in Problem 1. Note that variables from (14) are explicitly excluded from the GP by treating them as constants.

The reachability probability constraints (34) and cost constraints (35) need not be satisfied with equality. However, (34) is equivalent to

$$p_s \geq \sum_{\alpha \in Act(s)} \sigma^{s,\alpha} \cdot \sum_{s' \in S} \bar{\mathcal{P}}(s,\alpha,s') \cdot p_{s'}$$

for all $s \in S \setminus T$ and $\alpha \in Act$. The probability variables p_s are assigned upper bounds on the actual probability to reach the target states T under scheduler σ and valuation u. Put differently, the p_s variables cannot be assigned values that are lower than the actual probability; ensuring that σ and u induce satisfaction of the specification given by (27) if the problem is feasible and σ and u are well-defined. An analogous reasoning applies to the expected cost computation (35). A solution consisting of a scheduler or valuation that are not well-defined occurs only if Problem 1 itself is infeasible. Identifying that such a solution has been obtained is easy. These facts allow us to state the main result of this section.

Theorem 2. *A solution of the GP* (26)–(35) *inducing well-defined scheduler* σ *and valuation* u *is a feasible solution to Problem 1.*

Note that the actual probabilities induced by σ and u for the given pMDP $\mathcal{M}$ are given by the MC $\mathcal{M}^\sigma[u]$ induced by σ and instantiated by u. Since all variables are implicitly positive in a GP, no transition probability function will be instantiated to probability zero. The case of a scheduler variable being zero to induce the optimum can be excluded by a previous graph analysis.

5 Sequential Geometric Programming

We showed how to efficiently obtain a feasible solution for Problem 1 by solving GP (26)–(35). We propose a *sequential convex programming* trust-region method to compute a local optimum of the SGP (5)–(15), following [7, §9.1], solving a sequence of GPs. We obtain each GP by replacing signomial functions in equality constraints of the SGP (5)–(15) with *monomial approximations* of the functions.

Definition 6 (Monomial approximation). *Given a posynomial* $f \in \mathit{Sig}_\mathcal{V}$*, variables* $\mathcal{V} = \{x_1, \ldots, x_n\}$*, and a valuation* $u \in \mathit{Val}^\mathcal{V}$*, a* monomial approximation $\hat{f} \in \mathit{Mon}_\mathcal{V}$ for f near u *is*

$$\forall i.1 \leq i \leq n \quad \hat{f} = f[u] \prod_{i=1}^{n} \left(\frac{x_i}{u(x_i)} \right)^{a_i}, \quad \text{where } a_i = \frac{u(x_i)}{f[u]} \frac{\partial f}{\partial x_i}[u].$$

Intuitively, we compute a *linearization* $\hat{f}$ of $f \in \mathit{Sig}_\mathcal{V}$ around a fixed valuation u. We enforce the fidelity of monomial approximation $\hat{f}$ of $f \in \mathit{Sig}_\mathcal{V}$ by restricting valuations to remain within a set known as *trust region*. We define the following constraints on the variables $\mathcal{V}$ with $t > 1$ determining the size of the trust region:

$$\forall i.1 \leq i \leq n \quad (1/t) \cdot u(x_i) \leq x_i \leq t \cdot u(x_i) \tag{36}$$

For a given valuation u, we approximate the SGP (5)–(15) to obtain a *local GP* as follows. First, we apply a *lifting* procedure (Section 4.2) to the SGP ensuring that all constraints consist of posynomial functions. The thus obtained posynomial inequality constraints are included in the local GP. After replacing posynomials in every equality constraint by their monomial approximations near u, the resulting monomial equality constraints are also included. Finally, we add trust region constraints (36) for scheduler and parameter variables. The objective function is the same as for the SGP. The optimal solution of the local GP is not necessarily a feasible solution to the SGP. Therefore, we first normalize the scheduler and parameter values to obtain well-defined probability distributions. These normalized values are used to compute precise probabilities and expected cost using PRISM. The steps above provide a feasible solution of the SGP.

We use such approximations to obtain a sequence of feasible solutions to the SGP approaching a local optimum of the SGP. First, we compute a feasible solution $u^{(0)}$ for Problem 1 (Section 4), forming the initial point of a sequence

of solutions $u^{(0)}, \ldots, u^{(N)}, N \in \mathbb{N}$. The solution $u^{(k)}$ for $0 \leq k \leq N$ is obtained from a local GP defined using $u^{(k-1)}$ as explained above.

The parameter t for each iteration k is determined based on its value for the previous iteration, and the ratio of $f\left[u^{(k-1)}\right]$ to $f\left[u^{(k-2)}\right]$, where f is the objective function in (5). The iterations are stopped when $\left|f\left[u^{(k)}\right] - f\left[u^{(k-1)}\right]\right| < \epsilon$. Intuitively, ϵ defines the required improvement on the objective value for each iteration; once there is not enough improvement the process terminates.

6 Applications

We discuss two applications and their restrictions for the general SGP (5)–(15).

Model repair. For MC $\mathcal{D}$ and specification φ with $\mathcal{D} \not\models \varphi$, the *model repair* problem [13] is to transform $\mathcal{D}$ to $\mathcal{D}'$ such that $\mathcal{D}' \models \varphi$. The transformation involves a change of transition probabilities. Additionally, a cost function measures the change of probabilities. The natural underlying model is a pMC where parameters are added to probabilities. The cost function is minimized subject to constraints that induce satisfaction of φ. In [13], the problem is given as NLP. Heuristic [27] and simulation-based methods [19] (for MDPs) were presented.

Leveraging our results, one can readily encode model repair problems for MDPs, multiple objectives, and restrictions on probability or cost changes directly as NLPs. The encoding as in [13] is handled by our method in Section 5 as it involves signomial constraints. We now propose a more efficient approach, which encodes the change of probabilities using monomial functions. Consider an MDP $\mathcal{M} = (S, s_I, Act, \mathcal{P})$ and specifications $\varphi_1, \ldots, \varphi_q$ with $\mathcal{M} \not\models \varphi_1 \wedge \ldots \wedge \varphi_q$. For each probability $\mathcal{P}(s, \alpha, s') = a \in \mathbb{R}$ that may be changed, introduce a parameter p, forming the parameter set V. We define a parametric transition probability function by $\mathcal{P}'(s, \alpha, s') = p \cdot a \in Mon_V$. The quadratic cost function is for instance $f = \sum_{p \in V} p^2 \in Pos_V$. By minimizing the sum of squares of the parameters (with some regularization), the change of probabilities is minimized.

By incorporating these modifications into SGP (5)–(15), our approach is directly applicable. Either we restrict the cost function f to an upper bound, and efficiently solve a feasibility problem (Section 4), or we compute a local minimum of the cost function (Section 5). In contrast to [13], our approach works for MDPs and has an efficient solution. While [19] uses fast simulation techniques, we can directly incorporate multiple objectives and restrictions on the results while offering an efficient numerical solution of the problem.

Parameter space partitioning. For pMDPs, tools like PRISM [15] or PROPhESY [25] aim at partitioning the parameter space into regions with respect to a specification. A *parameter region* is given by a convex polytope defined by linear inequalities over the parameters, restricting valuations to a region. Now, for pMDP $\mathcal{M}$ a region is *safe* regarding a specification φ, if no valuation u inside this region and no scheduler σ induce $\mathcal{M}^\sigma[u] \not\models \varphi$. Vice versa, a region is unsafe, if there is no valuation and scheduler such that the specification is satisfied. In [25],

this certification is performed using SMT solving. More efficiency is achieved by using an approximation method [33].

Certifying regions to be unsafe is directly possible using our approach. Assume pMDP $\mathcal{M}$, specifications $\varphi_1, \ldots, \varphi_q$, and a region candidate defined by a set of linear inequalities. We incorporate the inequalities in the NLP (5)–(15). If the feasibility problem (Section 4) has no solution, the region is unsafe. This yields the *first efficient numerical method* for this problem of which we are aware. Proving that a region is safe is more involved. Given one specification $\varphi = \mathbb{P}_{\leq\lambda}(\Diamond T)$, we maximize the probability to reach T. If this probability is at most λ, the region is safe. For using our method from Section 5, one needs domain specific knowledge to show that a local optimum is a global optimum.

7 Experiments

We implemented a prototype using the Python interfaces of the probabilistic model checker STORM [29] and the optimization solver MOSEK [35]. All experiments were run on a 2.6 GHz machine with 32 GB RAM. We used PRISM [15] to correct approximation errors as explained before. We evaluated our approaches using mainly examples from the PARAM–webpage [26] and from PRISM [18]. We considered several parametric instances of the *Bounded Retransmission Protocol* (BRP) [4], *NAND Multiplexing* [6], and the *Consensus* protocol (CONS) [3]. For BRP, we have a pMC and a pMDP version, NAND is a pMC, and CONS is a pMDP. For obtaining feasibility solutions, we compare to the SMT solver Z3 [17]. For additional optimality criteria, there is no comparison to another tool possible as IPOPT [10] already fails for the smallest instances we consider.

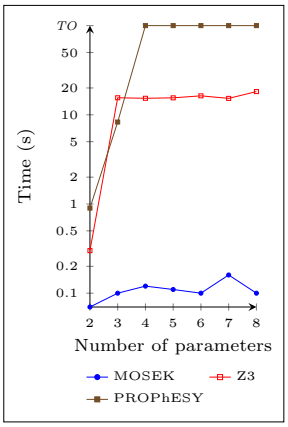
Fig. 3(a) states for each benchmark instance the number of states (#states) and the number of parameters (#par). We defined two specifications consisting of a expected cost property (EC) and a reachability property ($\mathbb{P}$). For some benchmarks, we also maximized the probability to reach a set of "good states" ($*$). We list the times taken by MOSEK; for optimality problems we also list the times PRISM took to compute precise probabilities or costs (Section 5). For feasibility problems we list the times of Z3. The timeout (TO) is 90 minutes.

We observe that both for feasibility with optimality criteria we can handle most benchmarks of up to 10^5 states within the timeout, while we ran into a timeout for CONS. The number of iterations N in the sequential convex programming is less than 12 for all benchmarks with $\epsilon = 10^{-3}$. As expected, simply solving feasibility problems is faster by at least one order of magnitude. Raising the number of parameters from 2 to 4 for BRP does not cause a major performance hit, contrary to existing tools. For all benchmarks except NAND, Z3 only delivered results for the smallest instances within the timeout.

To demonstrate the insensitivity of our approach to the number of parameters, we considered a pMC of rolling multiple Knuth–Yao dice with 156 states, 522 transitions and considered instances with up to 8 different parameters. The timeout is 100 seconds. In Fig. 3(b) we compare our encoding in MOSEK for this benchmark to the mere computation of a rational function using PROPhESY [25]

Benchmark	#states	#par	specs	MOSEK (s)	Z3
BRP (pMC)	5382	2	EC, $\mathbb{P}, *$	23.17 (6.48)	–
	112646	2	EC, $\mathbb{P}, *$	3541.59 (463.74)	–
	112646	4	EC, $\mathbb{P}, *$	4173.33 (568.79)	–
	5382	2	EC, $\mathbb{P}$	3.61	904.11
	112646	2	EC, $\mathbb{P}$	479.08	*TO*
NAND (pMC)	4122	2	EC, $\mathbb{P}, *$	14.67 (2.51)	–
	35122	2	EC, $\mathbb{P}, *$	1182.41 (95.19)	–
	4122	2	EC, $\mathbb{P}$	1.25	1.14
	35122	2	EC, $\mathbb{P}$	106.40	11.49
BRP (pMDP)	5466	2	EC, $\mathbb{P}, *$	31.04 (8.11)	–
	112846	2	EC, $\mathbb{P}, *$	4319.16 (512.20)	–
	5466	2	EC, $\mathbb{P}$	4.93	1174.20
	112846	2	EC, $\mathbb{P}$	711.50	*TO*
CONS (pMDP)	4112	2	EC, $\mathbb{P}, *$	102.93 (1.14)	–
	65552	2	EC, $\mathbb{P}, *$	*TO*	–
	4112	2	EC, $\mathbb{P}$	6.13	*TO*
	65552	2	EC, $\mathbb{P}$	1361.96	*TO*

(a) Benchmark results



(b) Sensitivity to #par

Fig. 3. Experiments.

and again to Z3. PROPhESY already runs into a timeout for 4 parameters[3]. Z3 needs around 15 seconds for most of the tests. Using GPs with MOSEK proves far more efficient as it needs less than one second for all instances.

In addition, we test model repair (Section 6) on a BRP instance with 17415 states for $\varphi = \mathbb{P}_{\leq 0.9}(\Diamond T)$. The initial parameter instantiation violates φ. We performed model repair towards satisfaction of φ. The probability of reaching T results in 0.79 and the associated cost is 0.013. The computation time is 21.93 seconds. We compare our result to an implementation of [19], where the probability of reaching T is 0.58 and the associated cost is 0.064. However, the time for the simulation-based method is only 2.4 seconds, highlighting the expected trade-off between optimality and computation times for the two methods.

Finally, we encode model repair for the small pMC from Example 1 in IPOPT, see [13]. For $\psi = \mathbb{P}_{\leq 0.125}(\Diamond T)$ where T represents the outcome of the die being 2, the initial instantiation induces probability 1/6. With our method, the probability of satisfying ψ is 0.1248 and the cost is 0.0128. With IPOPT, the probability is 0.125 with cost 0.1025, showing that our result is nearly optimal.

8 Conclusion and future work

We presented a way to use convex optimization in the field of parameter synthesis for parametric Markov models. Using our results, many NLP encodings of related problems now have a direct and efficient solution.

Future work will concern the integration of these methods into mature tools like PRISM or PROPhESY to enable large-scale benchmarking by state space reduction techniques and advanced data structures. Moreover, we will explore extensions to richer models like continuous-time Markov chains [31].

[3] Due to the costly computation of greatest common divisors employed in PROPhESY.

References

1. Jay K. Satia and Roy E. Lave Jr. Markovian decision processes with uncertain transition probabilities. *Operations Research*, 21(3):728–740, 1973.
2. Donald E. Knuth and Andrew C. Yao. The complexity of nonuniform random number generation. In Joseph F. Traub, editor, *Algorithms and Complexity: New Directions and Recent Results*, page 375. Academic Press, 1976.
3. James Aspnes and Maurice Herlihy. Fast randomized consensus using shared memory. *Journal of Algorithms*, 15(1):441–460, 1990.
4. L. Helmink, M. Sellink, and F. Vaandrager. Proof-checking a data link protocol. In *TYPES*, volume 806 of *LNCS*, pages 127–165. Springer, 1994.
5. Jean B. Lasserre. Global optimization with polynomials and the problem of moments. *SIAM Journal on Optimization*, 11(3):796–817, 2001.
6. Jie Han and Pieter Jonker. A system architecture solution for unreliable nanoelectronic devices. *IEEE Transactions on Nanotechnology*, 1:201–208, 2002.
7. Stephen Boyd, Seung-Jean Kim, Lieven Vandenberghe, and Arash Hassibi. A tutorial on geometric programming. *Optimization and Engineering*, 8(1), 2007.
8. Stephen Boyd. Sequential convex programming. Lecture Notes, 2008.
9. Kousha Etessami, Marta Kwiatkowska, Moshe Y. Vardi, and Mihalis Yannakakis. Multi-objective model checking of Markov decision processes. *LMCS*, 4(4), 2008.
10. Lorenz T. Biegler and Victor M. Zavala. Large-scale nonlinear programming using IPOPT: An integrating framework for enterprise-wide dynamic optimization. *Computers & Chemical Engineering*, 33(3):575–582, 2009.
11. Ernst Moritz Hahn, Holger Hermanns, Björn Wachter, and Lijun Zhang. PARAM: A model checker for parametric Markov models. In *CAV*, volume 6174 of *LNCS*, pages 660–664. Springer, 2010.
12. Ernst Moritz Hahn, Holger Hermanns, and Lijun Zhang. Probabilistic reachability for parametric Markov models. *STTT*, 13(1):3–19, 2010.
13. Ezio Bartocci, Radu Grosu, Panagiotis Katsaros, CR Ramakrishnan, and Scott A Smolka. Model repair for probabilistic systems. In *TACAS*, volume 6605 of *LNCS*, pages 326–340. Springer, 2011.
14. Vojtech Forejt, Marta Kwiatkowska, Gethin Norman, David Parker, and Hongyang Qu. Quantitative multi-objective verification for probabilistic systems. In *TACAS*, volume 6605 of *LNCS*, pages 112–127. Springer, 2011.
15. Marta Kwiatkowska, Gethin Norman, and David Parker. PRISM 4.0: Verification of probabilistic real-time systems. In *CAV*, volume 6806 of *LNCS*, pages 585–591. Springer, 2011.
16. Vojtech Forejt, Marta Kwiatkowska, and David Parker. Pareto curves for probabilistic model checking. In *ATVA*, volume 7561 of *LNCS*, pages 317–332. Springer, 2012.
17. Dejan Jovanovic and Leonardo Mendonça de Moura. Solving non-linear arithmetic. In *IJCAR*, volume 7364 of *LNCS*, pages 339–354. Springer, 2012.
18. Marta Kwiatkowska, Gethin Norman, and David Parker. The PRISM benchmark suite. In *QEST*, pages 203–204. IEEE CS, 2012.
19. Taolue Chen, Ernst Moritz Hahn, Tingting Han, Marta Kwiatkowska, Hongyang Qu, and Lijun Zhang. Model repair for Markov decision processes. In *TASE*, pages 85–92. IEEE CS, 2013.
20. Alberto Puggelli, Wenchao Li, Alberto L. Sangiovanni-Vincentelli, and Sanjit A. Seshia. Polynomial-time verification of PCTL properties of MDPs with convex uncertainties. In *CAV*, volume 8044 of *LNCS*, pages 527–542. Springer, 2013.

21. Milan Ceska, Frits Dannenberg, Marta Kwiatkowska, and Nicola Paoletti. Precise parameter synthesis for stochastic biochemical systems. In *CMSB*, volume 8859 of *LNCS*, pages 86–98. LNCS, 2014.
22. Ernst Moritz Hahn, Yong Li, Sven Schewe, Andrea Turrini, and Lijun Zhang. iscasMc: A web-based probabilistic model checker. In *FM*, volume 8442 of *LNCS*, pages 312–317. Springer, 2014.
23. Nils Jansen, Florian Corzilius, Matthias Volk, Ralf Wimmer, Erika Ábrahám, Joost-Pieter Katoen, and Bernd Becker. Accelerating parametric probabilistic verification. In *QEST*, volume 8657 of *LNCS*, pages 404–420. Springer, 2014.
24. Guoxin Su and David S. Rosenblum. Nested reachability approximation for discrete-time Markov chains with univariate parameters. In *ATVA*, volume 8837 of *LNCS*, pages 364–379. Springer, 2014.
25. Christian Dehnert, Sebastian Junges, Nils Jansen, Florian Corzilius, Matthias Volk, Harold Bruintjes, Joost-Pieter Katoen, and Erika Ábrahám. Prophesy: A probabilistic parameter synthesis tool. In *CAV (1)*, volume 9206 of *LNCS*, pages 214–231. Springer, 2015.
26. PARAM Website, 2015. `http://depend.cs.uni-sb.de/tools/param/`.
27. Shashank Pathak, Erika Ábrahám, Nils Jansen, Armando Tacchella, and Joost-Pieter Katoen. A greedy approach for the efficient repair of stochastic models. In *NFM*, volume 9058 of *LNCS*, pages 295–309. Springer, 2015.
28. Luca Bortolussi, Dimitrios Milios, and Guido Sanguinetti. Smoothed model checking for uncertain continuous-time Markov chains. *Inf. Comput.*, 247:235–253, 2016.
29. Christian Dehnert, Sebastian Junges, Joost-Pieter Katoen, and Matthias Volk. The probabilistic model checker storm (extended abstract). *CoRR*, abs/1610.08713, 2016.
30. Karina Valdivia Delgado, Leliane N. de Barros, Daniel B. Dias, and Scott Sanner. Real-time dynamic programming for Markov decision processes with imprecise probabilities. *Artif. Intell.*, 230:192–223, 2016.
31. Joost-Pieter Katoen. The probabilistic model checking landscape. In *IEEE Symposium on Logic In Computer Science (LICS)*. ACM, 2016.
32. Fan Long and Martin Rinard. Automatic patch generation by learning correct code. In *POPL*, pages 298–312. ACM, 2016.
33. Tim Quatmann, Christian Dehnert, Nils Jansen, Sebastian Junges, and Joost-Pieter Katoen. Parameter synthesis for Markov models: Faster than ever. In *ATVA*, volume 9938 of *LNCS*, pages 50–67, 2016.
34. Guoxin Su, David S. Rosenblum, and Giordano Tamburrelli. Reliability of run-time qos evaluation using parametric model checking. In *ICSE*. ACM, 2016. to appear.
35. MOSEK ApS. *The MOSEK optimization toolbox for PYTHON. Version 7.1 (Revision 60)*, 2015.
36. Christel Baier and Joost-Pieter Katoen. *Principles of Model Checking*. MIT Press, 2008.
37. Dimitri P. Bertsekas. *Nonlinear Programming*. Athena Scientific Belmont, 1999.
38. Stephen Boyd and Lieven Vandenberghe. *Convex Optimization*. Cambridge University Press, New York, NY, USA, 2004.
39. Sebastian Thrun, Wolfram Burgard, and Dieter Fox. *Probabilistic Robotics*. MIT Press, 2005.