# Synthesis in pMDPs: A Tale of 1001 Parameters

Murat Cubuktepe[1], Nils Jansen[2], Sebastian Junges[3],
Joost-Pieter Katoen[3], Ufuk Topcu[1]

[1] The University of Texas at Austin, Austin, TX, USA*
[2] Radboud University, Nijmegen, The Netherlands
[3] RWTH Aachen University, Aachen, Germany**

**Abstract.** This paper considers parametric Markov decision processes (pMDPs) whose transitions are equipped with affine functions over a finite set of parameters. The synthesis problem is to find a parameter valuation such that the instantiated pMDP satisfies a (temporal logic) specification under all strategies. We show that this problem can be formulated as a quadratically-constrained quadratic program (QCQP) and is non-convex in general. To deal with the NP-hardness of such problems, we exploit a convex-concave procedure (CCP) to iteratively obtain local optima. An appropriate interplay between CCP solvers and probabilistic model checkers creates a procedure — realized in the tool PROPheSY — that solves the synthesis problem for models with thousands of parameters.

## 1 Introduction

*The parameter synthesis problem.* Probabilistic model checking concerns the automatic verification of models such as Markov decision processes (MDPs). Unremitting improvements in algorithms and efficient tool implementations [14,22, 26] have opened up a wide variety of applications, most notably in dependability, security, and performance analysis as well as systems biology. However, at early development stages, certain system quantities such as fault or reaction rates are often not fully known. This lack of information gives rise to parametric models where transitions are functions over real-valued parameters [12, 21, 27], forming symbolic descriptions of (uncountable) families of concrete MDPs. The *parameter synthesis problem* is: Given a finite-state parametric MDP, find a parameter instantiation such that the induced concrete model satisfies a given specification. An inherent problem is *model repair*, where probabilities are changed ("repaired") with respect to parameters such that a model satisfies a specification [5]. Concrete applications include adaptive software systems [9], sensitivity analysis [33], and optimizing randomized distributed algorithms [1].

*State-of-the-art.* First approaches to parameter synthesis compute a rational function over the parameters to symbolically express reachability probabilities [12, 18, 21]. Equivalently, [17, 23] employ Gaussian elimination for matrices over

* Supported by the grants ONR N000141613165, NASA NNX17AD04G and AFRL FA8650-15-C-2546
** Supported by the CDZ project CAP (GZ 1023), and the DFG RTG 2236 "UnRAVeL".

the field of rational functions. Solving the (potentially very large, high-degree) functions is naturally a SAT-modulo theories (SMT) problem over non-linear arithmetic, or a nonlinear program (NLP) [5, 11]. However, solving such SMT problems is *exponential in the degree of functions and the number of variables* [23], and solving NLPs is *NP-hard in general* [28]. Specific approaches to model repair rely on NLP [5] or particle-swarm optimization (PSO) [10].

Finally, parameter synthesis is equivalent to computing finite-memory strategies for partially observable MDPs (POMDPs) [25]. Such strategies may be obtained, for instance, by employing sequential quadratic programming (SQP) [3]. Exploiting this approach is not practical, though, because SQP for our setting already requires a (feasible) solution satisfying the given specification. Overall, efficient implementations in tools like PARAM [21], PRISM [26], and PROPhESY [13] can handle thousands of states but only a *handful of parameters*.

*Our approach.* We overcome the restriction to few parameters by employing convex optimization [7]. This direction is not new; [11] describes a convexification of the NLP into a geometric program [6], which can still only handle up to about ten parameters. We take a different approach. First, we transform the NLP formulation [5, 11] into a *quadratically-constrained quadratic program* (QCQP). As such an optimization problem is nonconvex in general, we cannot resort to polynomial-time algorithms for convex QCQPs [2]. Instead, to solve our NP-hard problem, we massage the QCQP formulation into a *difference-of-convex* (DC) problem. The convex-concave procedure (CCP) [29] yields local optima of a DC problem by a convexification towards a convex quadratic program, which is amenable for state-of-the-art solvers such as Gurobi [19].

Yet, blackbox CCP solvers [31, 32] suffer from severe numerical issues and can only solve very small problems. We integrate the procedure with a probabilistic model checker, creating a method that — realized in the open-source tool PROPheSY [13] — yields (a) an improvement of multiple orders of magnitude compared to just using CCP as a black box and (b) ensures the correctness of the solution. In particular, we use probabilistic model checking to:

- rule out feasible solutions that may be spurious due to numerical errors,
- check if intermediate solutions are already feasible for earlier termination,
- compute concrete probabilities from intermediate parameter instantiations to avoid potential numerical instabilities.

An extensive empirical evaluation on a large range of benchmarks shows that our approach can solve the parameter synthesis problem for models with large state spaces and up to thousands of parameters, and is superior to all existing parameter synthesis tools [13, 20, 26], geometric programming [11], and an efficient re-implementation of PSO [10] that we create to deliver a better comparison. Contrary to the geometric programming approach in [11], we compute solutions that hold for all possible (adversarial) schedulers for parametric MDPs. Traditionally, model checking delivers results for such adversarial schedulers [4], which are for instance useful when the nondeterminism is not controllable and induced by the environment, which is the case in the example below.

*An illustrative example.* Consider the Carrier Sense Multiple Access/Collision Detection (CSMA/CD) protocol in Ethernet networks, which was subject to probabilistic model checking [16]. When two stations simultaneously attempt sending a packet (giving rise to a collision), a so-called randomized exponential back-off mechanism is used to avoid the collision. Until the $k$-th attempt, a delay out of $2^k$ possibilities is randomly drawn from a uniform distribution. An interesting question is if a uniform distribution is optimal, where optimality refers to the minimal expected time until all packets have been sent. A bias for small delays seems beneficial, but raises the collision probability. Using our novel techniques, within a minute we synthesize a different distribution, which induces less expected time compared to the uniform distribution. The used model has about $10^5$ states and 26 parameters. We are not aware of any other parameter-synthesis approach being able to generate such a result within reasonable time.

## 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 $\mathit{Distr}(X)$. Let $V = \{x_1, \ldots, x_n\}$ be a finite set of *variables* over the real numbers $\mathbb{R}$. The set of multivariate polynomials over $V$ is $\mathbb{Q}[V]$. An *instantiation* for $V$ is a function $u\colon V \to \mathbb{R}$.

A function $f\colon \mathbb{R}^n \to \mathbb{R}$ is *affine* if $f(\boldsymbol{x}) = a^T\boldsymbol{x} + b$ with $a \in \mathbb{R}^n$ and $b \in \mathbb{R}$, and $f\colon \mathbb{R}^n \to \mathbb{R}$ is *quadratic* if $f(\boldsymbol{x}) = \boldsymbol{x}^T P\boldsymbol{x} + a^T\boldsymbol{x} + b$ with $a, b$ as before and $P \in \mathbb{R}^{n\times n}$. A symmetric matrix $P \in \mathbb{R}^{n\times n}$ is *positive semidefinite* (PSD) if $\boldsymbol{x}^T P\boldsymbol{x} \geq 0 \;\; \forall \boldsymbol{x} \in \mathbb{R}^n$, or equivalently, if all eigenvalues of $P$ are nonnegative.

**Definition 1 ((Affine) pMDP).** *A* parametric Markov decision process (pMDP) *is a tuple* $\mathcal{M} = (S,\, s_I, \mathit{Act}, V, \mathcal{P})$ *with a finite set* $S$ *of* states*, an* initial state $s_I \in S$*, a finite set* $\mathit{Act}$ *of* actions*, a finite set* $V$ *of real-valued variables* (parameters) *and a* transition function $\mathcal{P}\colon S \times \mathit{Act} \times S \to \mathbb{Q}[V]$*. A pMDP is* affine *if* $\mathcal{P}(s, \alpha, s')$ *is affine for every* $s, s' \in S$ *and* $\alpha \in \mathit{Act}$*.*

For $s \in S$, $A(s) = \{\alpha \in \mathit{Act} \mid \exists s' \in S.\, \mathcal{P}(s,\, \alpha,\, s') \neq 0\}$ is the set of *enabled* actions at $s$. Without loss of generality, we require $A(s) \neq \emptyset$ for $s \in S$. If $|A(s)| = 1$ for all $s \in S$, $\mathcal{M}$ is a *parametric discrete-time Markov chain (pMC)*. We denote the transition function for pMCs by $\mathcal{P}(s, s')$. MDPs can be equipped with a state–action *cost function* $c\colon S \times \mathit{Act} \to \mathbb{R}_{\geq 0}$.

A pMDP $\mathcal{M}$ is a *Markov decision process (MDP)* if the transition function yields *well-defined* probability distributions, i.e., $\mathcal{P}\colon S \times \mathit{Act} \times S \to [0, 1]$ and $\sum_{s'\in S} \mathcal{P}(s, \alpha, s') = 1$ for all $s \in S$ and $\alpha \in A(s)$. Applying an *instantiation* $u\colon V \to \mathbb{R}$ to a pMDP $\mathcal{M}$ yields $\mathcal{M}[u]$ by replacing each $f \in \mathbb{Q}[V]$ in $\mathcal{M}$ by $f[u]$. An instantiation $u$ is *well-defined* for $\mathcal{M}$ if the resulting model $\mathcal{M}[u]$ is an MDP.

To define measures on MDPs, nondeterministic choices are resolved by a so-called *strategy* $\sigma\colon S \to \mathit{Act}$ with $\sigma(s) \in A(s)$. The set of all strategies over $\mathcal{M}$ is $\mathit{Str}^{\mathcal{M}}$. For the measures in this paper, memoryless deterministic strategies

suffice [4]. Applying a strategy to an MDP yields an *induced Markov chain* where all nondeterminism is resolved.

For an MC $\mathcal{D}$, the *reachability specification* $\varphi_r = \mathbb{P}_{\leq\lambda}(\Diamond T)$ asserts that a set $T \subseteq S$ of *target states* is reached with probability at most $\lambda \in [0,1]$. If $\varphi_r$ holds for $\mathcal{D}$, we write $\mathcal{D} \models \varphi_r$. Accordingly, for an *expected cost specification* $\varphi_c = \mathbb{E}_{\leq\kappa}(\Diamond G)$ it holds that $\mathcal{D} \models \varphi_c$ if and only if the expected cost of reaching a set $G \subseteq S$ is bounded by $\kappa \in \mathbb{R}$. We use standard measures and definitions as in [4, Ch. 10]. An MDP $\mathcal{M}$ satisfies a specification $\varphi$, written $\mathcal{M} \models \varphi$, if and only if *for all* strategies $\sigma \in \mathit{Str}^{\mathcal{M}}$ it holds that $\mathcal{M}^\sigma \models \varphi$.

## 3 Formal Problem Statement

*Problem 1 (pMDP synthesis problem).* Given a pMDP $\mathcal{M} = (S, s_I, \mathit{Act}, V, \mathcal{P})$, and a reachability specification $\varphi_r = \mathbb{P}_{\leq\lambda}(\Diamond T)$, compute a well-defined instantiation $u\colon V \to \mathbb{R}$ for $\mathcal{M}$ such that $\mathcal{M}[u] \models \varphi_r$.

Intuitively, we seek for an instantiation of parameters $u$ that satisfies $\varphi_r$ for all possible strategies for the instantiated MDP. We show necessary adaptions for an expected cost specification $\varphi_c = \mathbb{E}_{\leq\kappa}(\Diamond T)$ later.

For a given instantiation $u$, Problem 1 boils down to verifying if $\mathcal{M}[u] \models \varphi_r$. The standard formulation uses a linear program (LP) to minimize the probability $p_{s_I}$ of reaching the target set $T$ from the initial state $s_I$, while ensuring that this probability is realizable under any strategy [4, Ch. 10]. The straightforward extension of this approach to pMDPs in order to *compute* a suitable instantiation $u$ yields the following nonlinear program (NLP):

$$\begin{aligned}
&\text{minimize} && p_{s_I} && (1)\\
&\text{subject to} && && \\
&\forall s \in T. && p_s = 1 && (2)\\
&\forall s, s' \in S. \forall \alpha \in \mathit{Act}. && \mathcal{P}(s,\alpha,s') \geq 0 && (3)\\
&\forall s \in S. \forall \alpha \in \mathit{Act}. && \sum_{s' \in S} \mathcal{P}(s,\alpha,s') = 1 && (4)\\
& && \lambda \geq p_{s_I} && (5)\\
&\forall s \in S \setminus T. \forall \alpha \in \mathit{Act}. && p_s \geq \sum_{s' \in S} \mathcal{P}(s,\alpha,s') \cdot p_{s'}. && (6)
\end{aligned}$$

For $s \in S$, the *probability variable* $p_s \in [0,1]$ represents an upper bound of the probability of reaching target set $T \subseteq S$, and the *parameters* in set $V$ enter the NLP as part of the functions from $\mathbb{Q}[V]$ in the transition function $\mathcal{P}$.

**Proposition 1.** *The NLP in* (1) – (6) *computes the* minimal probability *of reaching $T$ under a* maximizing *strategy.*

The probability to reach a state in $T$ from $T$ is one (2). The constraints (3) and (4) ensure *well-defined* transition probabilities. Constraint (5) is optional

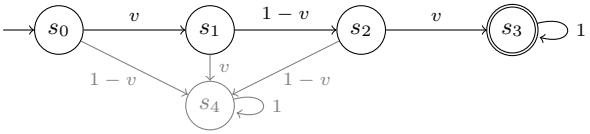


**Fig. 1.** A pMC with parameter $v$.

but necessary later, and ensures that the probability of reaching $T$ is below the threshold $\lambda$. For each non-target state $s \in S$ and any action $\alpha \in Act$, the probability induced by the *maximizing scheduler* is a lower bound to the probability variables $p_s$ (6). To assign probability variables the minimal values with respect to the parameters from $V$, $p_{s_I}$ is minimized in the objective (1).

*Remark 1 (Graph-preserving instantiations).* In the LP formulation for MDPs, states with probability 0 to reach $T$ are determined via a preprocessing on the underlying graph, and their probability variables are set to zero, to avoid an underdetermined equation system. For the same reason, we preserve the underlying graph of the pMDP, as in [13, 21]. We thus exclude valuations $u$ with $f[u] = 0$ for $f \in \mathcal{P}(s, \alpha, s')$ for all $s, s' \in S$ and $\alpha \in Act$. We replace (3) by

$$\forall s, s' \in S. \forall \alpha \in Act. \quad \mathcal{P}(s, \alpha, s') \geq \varepsilon_{\text{graph}}. \tag{7}$$

where $\varepsilon_{\text{graph}} > 0$ is a small constant.

*Example 1.* Consider the pMC in Fig. 1 with parameter set $V = \{v\}$, initial state $s_0$, and target set $T = \{s_3\}$. Let $\lambda$ be an arbitrary constant. The NLP in (8) – (13) minimizes the probability of reaching $s_3$ from the initial state:

$$\begin{aligned}
&\text{minimize} && p_{s_0} && (8)\\
&\text{subject to} && && \\
& && p_{s_3} = 1 && (9)\\
& && \lambda \geq p_{s_0} \geq v \cdot p_{s_1} && (10)\\
& && p_{s_1} \geq (1-v) \cdot p_{s_2} && (11)\\
& && p_{s_2} \geq v \cdot p_{s_3} && (12)\\
& && 1 - \varepsilon_{\text{graph}} \geq v \geq \varepsilon_{\text{graph}}. && (13)
\end{aligned}$$

*Expected cost specifications.* The NLP in (1) – (7) considers reachability probabilities. If we have instead an expected cost specification $\varphi_c = \mathbb{E}_{\leq \kappa}(\Diamond G)$, we replace (2), (5), and (6) in the NLP by the following constraints:

$$\forall s \in G. \quad p_s = 0, \tag{14}$$

$$\forall s \in S \setminus G. \forall \alpha \in A(s). \quad p_s \geq c(s, \alpha) + \sum_{s' \in S} \mathcal{P}(s, \alpha, s') \cdot p_{s'} \tag{15}$$

$$\kappa \geq p_{s_I}. \tag{16}$$

We have $p_s \in \mathbb{R}$, as these variables represent the expected cost to reach $G$. At $G$, the expected cost is set to zero (14), and the actual expected cost for other states is a lower bound to $p_s$ (15). Finally, $p_{s_I}$ is bounded by the threshold $\kappa$.

## 4 QCQP Reformulation of the pMDP Synthesis Problem

For the remainder of this paper, we restrict pMDPs to be affine, see Def. 1. For an affine pMDP $\mathcal{M}$, the functions in the resulting NLP (1) – (7) for pMDP synthesis from the previous section are affine in $V$. However, the functions in (6) are *quadratic*, as a result of multiplying affine functions occurring in $\mathcal{P}$ with the probability variables $p_{s'}$. We rewrite the NLP as a standard form of a *quadratically-constrained quadratic program* (QCQP) [7]. Afterwards, we examine this QCQP in detail and show that it is nonconvex.

In general, a QCQP is an optimization problem with a quadratic objective function and $m$ quadratic constraints, written as

$$\begin{aligned} &\text{minimize} && \boldsymbol{x}^T P_0 \boldsymbol{x} + q_0^T \boldsymbol{x} + r_0 && (17)\\ &\text{subject to} && \\ &\forall i \in \{1, \ldots, m\} && \boldsymbol{x}^T P_i \boldsymbol{x} + q_i^T \boldsymbol{x} + r_i \leq 0, && (18) \end{aligned}$$

where $\boldsymbol{x}$ is a vector of *variables*, and the coefficients are $P_i \in \mathbb{R}^{n \times n}$, $g_i \in \mathbb{R}^n$, $r_i \in \mathbb{R}$ for $0 \leq i \leq m$. We assume $P_0, \ldots, P_m$ are symmetric without loss of generality. Constraints of the form $\boldsymbol{x}^T P_i \boldsymbol{x} + q_i^T \boldsymbol{x} + r_i = 0$ are encoded by

$$\boldsymbol{x}^T P_i \boldsymbol{x} + q_i^T \boldsymbol{x} + r_i \leq 0 \text{ and } -\boldsymbol{x}^T P_i \boldsymbol{x} - q_i^T \boldsymbol{x} - r_i \leq 0 \ .$$

*Properties of QCQPs.* We discuss properties of all matrices $P_i$ for $0 \leq i \leq m$. If all $P_i = 0$, the function $q_i^T \boldsymbol{x} + r_i$ is *affine*, and the QCQP is in fact an LP. If every $P_i$ is PSD, the function $\boldsymbol{x}^T P_i \boldsymbol{x} + q_i^T \boldsymbol{x} + r_i$ is *convex*, and the QCQP is a *convex optimization problem*, that can be solved in polynomial time [2]. If any $P_i$ is not PSD, the resulting QCQP is nonconvex. The problem of finding a feasible solution in a nonconvex QCQP is NP-hard [8].

To ease the presentation, we transform the quadratic constraints in the NLP in (1) – (7) to the standard QCQP form in (17) – (18):

$$\begin{aligned} &\text{minimize} && p_{s_I} && (19)\\ &\text{subject to} && && \\ &\forall s \in T. && p_s = 1 && (20)\\ &\forall s, s' \in S. \forall \alpha \in A(s). && \mathcal{P}(s, \alpha, s') \geq \varepsilon_{\text{graph}} && (21)\\ &\forall s \in S. \forall \alpha \in A(s). && \sum_{s' \in S} \mathcal{P}(s, \alpha, s') = 1 && (22)\\ & && \lambda \geq p_{s_I} && (23)\\ &\forall s \in S \setminus T. \forall \alpha \in A(s). && p_s \geq \boldsymbol{x}^T P_{s,\alpha} \boldsymbol{x} + q_{s,\alpha}^T \boldsymbol{x}, && (24) \end{aligned}$$

where $\boldsymbol{x}$ is a vector consisting of the probability variables $p_s$ for all $s \in S$ and the pMDP parameters from $V$, i.e., $\boldsymbol{x}$ has $|S| + |V|$ rows. Furthermore, $P_{s,\alpha} \in \mathbb{R}^{(|S|+|V|)\times(|S|+|V|)}$ is a symmetric matrix, and $q_{s,\alpha} \in \mathbb{R}^{(|S|+|V|)}$.

*Construction of the QCQP.* We use the matrix $P_{s,\alpha}$ to capture the *quadratic* part and the vector $q_{s,\alpha}$ to capture the *affine* part in (24). More precisely, consider an affine function $\mathcal{P}(s, \alpha, s') = a \cdot v + b$ with $a, b \in \mathbb{R}$. The function occurs in the constraint (6) as part of the function $(a \cdot v + b) \cdot p_{s'}$. The quadratic part thus occurs as $a \cdot v \cdot p_{s'}$ and the affine part as $b \cdot p_{s'}$.

We first consider the product $\boldsymbol{x}^T P_{s,\alpha} \boldsymbol{x}$, which denotes the sum over all products of entries in $\boldsymbol{x}$. Thus, in $P_{s,\alpha}$, each row or column corresponds either to a probability variable $p_s$ for a state $s \in S$ or to a parameter $v \in V$. In fact, the cells indexed $(v, p_{s'})$ and $(p_{s'}, v)$ correspond to the product of these variables. These two entries are summed in $\boldsymbol{x}^T P_{s,\alpha} \boldsymbol{x}$. In $P_{s,\alpha}$, the sum is reflected by two entries $a/2$ in the cells $(v, p_{s'})$ and $(p_{s'}, v)$. Then $P_{s,\alpha}$ is a symmetric matrix, as required. Similarly, we construct $q_{s,\alpha}$; the entry corresponding to $p_{s'}$ is set to $b$.

We do not modify the affine functions in (20) – (23) for the QCQP form.

*Example 2.* Recall Example 1. We reformulate the NLP in (8) – (13) as a QCQP in the form of (19) – (24) using the same variables.

$$
\begin{aligned}
\text{minimize} \quad & p_{s_0} \\
\text{subject to} \quad & \\
& p_{s_3} = 1 \\
& \lambda \geq p_{s_0} \geq \begin{bmatrix} v \\ p_{s_1} \end{bmatrix}^T P_{s_0} \begin{bmatrix} v \\ p_{s_1} \end{bmatrix} = \begin{bmatrix} v \\ p_{s_1} \end{bmatrix}^T \begin{bmatrix} 0 & 0.5 \\ 0.5 & 0 \end{bmatrix} \begin{bmatrix} v \\ p_{s_1} \end{bmatrix} \\
& p_{s_1} \geq \begin{bmatrix} v \\ p_{s_2} \end{bmatrix}^T P_{s_1} \begin{bmatrix} v \\ p_{s_2} \end{bmatrix} = \begin{bmatrix} v \\ p_{s_2} \end{bmatrix}^T \begin{bmatrix} 0 & -0.5 \\ -0.5 & 0 \end{bmatrix} \begin{bmatrix} v \\ p_{s_2} \end{bmatrix} + p_{s_2} \\
& p_{s_2} \geq \begin{bmatrix} v \\ p_{s_3} \end{bmatrix}^T P_{s_2} \begin{bmatrix} v \\ p_{s_3} \end{bmatrix} = \begin{bmatrix} v \\ p_{s_3} \end{bmatrix}^T \begin{bmatrix} 0 & 0.5 \\ 0.5 & 0 \end{bmatrix} \begin{bmatrix} v \\ p_{s_3} \end{bmatrix} \\
& 1 - \varepsilon_{\text{graph}} \geq v \geq \varepsilon_{\text{graph}}.
\end{aligned}
$$

**Theorem 1.** *The QCQP in* (19) – (23) *is nonconvex in general.*

*Proof.* The matrices $P_{s_0}, P_{s_1}, P_{s_2}$ in Example 2 have an eigenvalue of $-0.5$ and are not PSD. Thus, the constraints and the resulting QCQP are nonconvex. □

## 5 Efficient pMDP Synthesis via Convexification

The QCQP in (19) – (23) is nonconvex and hard to solve in general. We provide a solution by employing a heuristic called the *convex-concave procedure* (CCP) [29], which relies on the ability to efficiently solve convex optimization problems.

The CCP computes a *local optimum* of a non-convex *difference-of-convex* (DC) problem. A DC problem has the form

$$\begin{aligned} &\text{minimize} && f_0(\boldsymbol{x}) - g_0(\boldsymbol{x}) && (25)\\ &\text{subject to} && && \\ &\forall i = 1, \dots, m. && f_i(\boldsymbol{x}) - g_i(\boldsymbol{x}) \leq 0, && (26) \end{aligned}$$

where for $i = 0, 1, \dots, m$, $f_i(\boldsymbol{x})\colon \mathbb{R}^n \to \mathbb{R}$ and $g_i(\boldsymbol{x})\colon \mathbb{R}^n \to \mathbb{R}$ are convex. The functions $-g_i(\boldsymbol{x})$ are *concave*. Every quadratic function can be rewritten as a DC function. Consider the indefinite quadratic function $\boldsymbol{x}^T P_{s,\alpha} \boldsymbol{x} + q_{s,\alpha}^T \boldsymbol{x}$ from (24). We decompose the matrix $P_{s,\alpha}$ into the difference of two matrices

$$P_{s,\alpha} = P_{s,\alpha}^+ - P_{s,\alpha}^- \text{ with } P_{s,\alpha}^+ = P_{s,\alpha} + tI \text{ and } P_{s,\alpha}^- = tI,$$

where $I$ is the identity matrix, and $t \in \mathbb{R}_+$ is sufficiently large to render $P_{s,\alpha}^+$ PSD, e.g., larger than the largest eigenvalue of $P_{s,\alpha}$. Then, we rewrite $\boldsymbol{x}^T P_{s,\alpha} \boldsymbol{x} + q_{s,\alpha}^T \boldsymbol{x}$ as $\left(\boldsymbol{x}^T P_{s,\alpha}^+ \boldsymbol{x} + q_{s,\alpha}^T \boldsymbol{x}\right) - \boldsymbol{x}^T P_{s,\alpha}^- \boldsymbol{x}$, which is in the form of (26).

*Example 3.* Recall the pMC in Fig. 1 and the QCQP from Example 2. All matrices $P_s$ of the QCQP are not PSD. We construct a DC problem with $t = 1$ for all $P_s$:

$$\begin{aligned} &\text{minimize} && p_{s_0} \\ &\text{subject to} && \\ &&& p_{s_3} = 1 \\ &&& \lambda \geq p_{s_0} \geq \begin{bmatrix} v \\ p_{s_1} \end{bmatrix}^T \begin{bmatrix} 1 & 0.5 \\ 0.5 & 1 \end{bmatrix} \begin{bmatrix} v \\ p_{s_1} \end{bmatrix} - \begin{bmatrix} v \\ p_{s_1} \end{bmatrix}^T \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix} \begin{bmatrix} v \\ p_{s_1} \end{bmatrix} \\ &&& p_{s_1} \geq \begin{bmatrix} v \\ p_{s_2} \end{bmatrix}^T \begin{bmatrix} 1 & -0.5 \\ -0.5 & 1 \end{bmatrix} \begin{bmatrix} v \\ p_{s_2} \end{bmatrix} - \begin{bmatrix} v \\ p_{s_2} \end{bmatrix}^T \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix} \begin{bmatrix} v \\ p_{s_2} \end{bmatrix} + p_{s_2} \\ &&& p_{s_2} \geq \begin{bmatrix} v \\ p_{s_3} \end{bmatrix}^T \begin{bmatrix} 1 & 0.5 \\ 0.5 & 1 \end{bmatrix} \begin{bmatrix} v \\ p_{s_3} \end{bmatrix} - \begin{bmatrix} v \\ p_{s_3} \end{bmatrix}^T \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix} \begin{bmatrix} v \\ p_{s_3} \end{bmatrix} \\ &&& 1 - \varepsilon_{\text{graph}} \geq v \geq \varepsilon_{\text{graph}}. \end{aligned}$$

We have $\boldsymbol{x} = (p_{s_1}, p_{s_2}, p_{s_3}, v)$ and an initial assignment $\hat{x} = (\hat{p}_{s_1}, \hat{p}_{s_2}, \hat{p}_{s_3}, \hat{v})$.

*CCP approach.* For the resulting DC problem, we consider the iterative *penalty CCP method* [29]. The procedure is initialized with any initial assignment $\hat{x}$ of the variables $\boldsymbol{x}$. In the *convexification* stage, we compute affine approximations in form of a linearization of $g_i(\boldsymbol{x})$ around $\hat{x}$:

$$\bar{g}_i(\boldsymbol{x}) = g_i(\hat{x}) + \nabla g_i(\hat{x})^T (\boldsymbol{x} - \hat{x}),$$

where $\nabla g_i$ is the gradient of the functions $g_i(\boldsymbol{x})$ at $\hat{x}$. Then, we replace the DC function $f_i(\boldsymbol{x}) - g_i(\boldsymbol{x})$ by $f_i(\boldsymbol{x}) - \bar{g}_i(\boldsymbol{x})$, which is a *convex over-approximation* of

the original function. A feasible assignment for the resulting over-approximated and *convex* DC problem is also feasible for the original DC problem.

To find such a feasible assignment, a *penalty variable* $k_{s,\alpha}$ for all $s \in S \setminus T$ and $\alpha \in Act$ is added to all convexified constraints. Solving the resulting problem then seeks to minimize the violation of the original DC constraints by minimizing the sum of the penalty variables. The resulting convex problem is written as

$$\text{minimize} \quad p_{s_I} + \tau \sum_{\forall s \in S \setminus T} \sum_{\forall \alpha \in Act} k_{s,\alpha} \tag{27}$$

subject to

$$\forall s \in T. \quad p_s = 1 \tag{28}$$

$$\forall s, s' \in S. \forall \alpha \in A(s). \quad \mathcal{P}(s, \alpha, s') \geq \varepsilon_{\text{graph}} \tag{29}$$

$$\forall s \in S. \forall \alpha \in A(s). \quad \sum_{s' \in S} \mathcal{P}(s, \alpha, s') = 1 \tag{30}$$

$$\lambda \geq p_{s_I} \tag{31}$$

$$\forall s \in S \setminus T. \forall \alpha \in A(s) \quad k_{s,\alpha} + p_s \geq \boldsymbol{x}^T P^+_{s,\alpha} \boldsymbol{x} + q^T_{s,\alpha} \boldsymbol{x} - \hat{x}^T P^-_{s,\alpha} (2\boldsymbol{x} - \hat{x}) \tag{32}$$

$$\forall s \in S \setminus T. \forall \alpha \in A(s) \quad k_{s,\alpha} \geq 0, \tag{33}$$

where $\tau > 0$ is a fixed *penalty parameter*, and the gradient of $\boldsymbol{x}^T P^-_{s,\alpha} \boldsymbol{x}$ is $2 \cdot P^-_{s,\alpha} \hat{x}$. This convexified DC problem is in fact a convex QCQP. The changed objective now makes the constraint (31) important.

*Example 4.* Recall the pMC in Fig. 1 and the DC problem from Example 3. We introduce the penalty variables $k_{s_i}$ and assume a fixed $\tau$. We linearize around $\hat{x}$. The resulting convex problem is:

$$\begin{aligned}
&\text{minimize} \quad p_{s_0} + \tau \sum_{i=0}^{2} k_{s_i} \\
&\text{subject to} \\
&\quad p_{s_3} = 1 \\
&\quad \lambda \geq p_{s_0} \\
&\quad k_{s_0} + p_{s_0} \geq \begin{bmatrix} v \\ p_{s_1} \end{bmatrix}^T \begin{bmatrix} 1 & 0.5 \\ 0.5 & 1 \end{bmatrix} \begin{bmatrix} v \\ p_{s_1} \end{bmatrix} - \begin{bmatrix} \hat{v} \\ \hat{p}_{s_1} \end{bmatrix}^T \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix} \begin{bmatrix} 2 \cdot v - \hat{v} \\ 2 \cdot p_{s_1} - \hat{p}_{s_1} \end{bmatrix} \\
&\quad k_{s_1} + p_{s_1} \geq \begin{bmatrix} v \\ p_{s_1} \end{bmatrix}^T \begin{bmatrix} 1 & -0.5 \\ -0.5 & 1 \end{bmatrix} \begin{bmatrix} v \\ p_{s_2} \end{bmatrix} - \begin{bmatrix} \hat{v} \\ \hat{p}_{s_2} \end{bmatrix}^T \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix} \begin{bmatrix} 2 \cdot v - \hat{v} \\ 2 \cdot p_{s_2} - \hat{p}_{s_2} \end{bmatrix} + p_{s_2} \\
&\quad k_{s_2} + p_{s_2} \geq \begin{bmatrix} v \\ p_{s_3} \end{bmatrix}^T \begin{bmatrix} 1 & 0.5 \\ 0.5 & 1 \end{bmatrix} \begin{bmatrix} v \\ p_{s_3} \end{bmatrix} - \begin{bmatrix} \hat{v} \\ \hat{p}_{s_3} \end{bmatrix}^T \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix} \begin{bmatrix} 2 \cdot v - \hat{v} \\ 2 \cdot p_{s_3} - \hat{p}_{s_3} \end{bmatrix} \\
&\quad 1 - \varepsilon_{\text{graph}} \geq v \geq \varepsilon_{\text{graph}} \\
&\quad k_{s_0} \geq 0, k_{s_1} \geq 0, k_{s_2} \geq 0.
\end{aligned}$$

If all penalty variables are assigned to zero, we can terminate the algorithm immediately, for the proof see Appendix A.

**Theorem 2.** *A satisfying assignment of the convex DC problem in* (27) – (33)

$$\textit{with} \qquad \tau \sum_{\forall s \in S \setminus T} \sum_{\forall \alpha \in Act} k_{s,\alpha} = 0$$

*is a feasible solution to Problem 1.*

If any of the penalty variables are assigned to a positive value, we update the penalty parameter $\tau$ by $\mu + \tau$ for a $\mu > 0$, until an upper limit for $\tau_{max}$ is reached to avoid numerical problems. Then, we compute a linearization of the $g_i$ functions around the current (not feasible) solution and solve the resulting problem. This procedure is repeated until we find a feasible solution. If the procedure converges to an infeasible solution, it may be restarted with an adapted initial $\hat{x}$.

### Efficiency Improvements in the Convex-Concave Procedure

*Better convexification.* We can use the previous transformation to perform CCP, but it involves expensive matrix operations, including computing the numerous eigenvalues. Observe that the matrices $P_{s,\alpha}$ and vectors $q_{s,\alpha}$ are sparse. Then, the eigenvalue method introduces more occurrences of the variables in every constraint, and thereby increases the approximation error during convexification.

We use an alternative convexification: Consider the bilinear function $h = 2c \cdot yz$, where $y$ and $z$ are variables, and $c \in \mathbb{R}_+$. We rewrite $h$ equivalently to $h + c(y^2 + z^2) - c(y^2 + z^2)$. Then, we rewrite $h + c(y^2 + z^2)$ as $c(y + z)^2$. We obtain $h = c(y + z)^2 - c(y^2 + z^2)$. The function $c(y + z)^2$ is a quadratic convex function, and we convexify the function $-c(y^2 + z^2)$ as $-c(\hat{y}^2 + \hat{z}^2) + 2c(\hat{y}^2 + \hat{z}^2 - y\hat{y} - z\hat{z})$, where $\hat{y}$ and $\hat{z}$ are the assignments as before. We convexify the bilinear function $h = 2c \cdot yz$ with $c \in \mathbb{R}_-$ analogously. Consequently, we reduce the occurrences of variables for sparse matrices compared to the eigenvalue method.

*Integrating model checking with CCP.* In each iteration of the CCP, we obtain values $\hat{v}$ which give rise to a parameter instantiation. Model checking at these instantiations is a good heuristic to allow for *early termination*. We check whether the values $\hat{v}$ already induce a feasible solution to the original NLP, even though the penalty variables have not converged to zero.

Additionally, instead of instantiating the initial probability values $\hat{p}_s$ in iteration $i + 1$, we may use the model checking result of the MDP instantiated at $\hat{v}$ from iteration $i$. Model checking ensures that the probability variables are consistent with the parameter variables, i.e., that the constraints describing the transition relation in the original NLP are all met. Using the model checking results overcomes problems with local optima. Small violations in (32), i.e., small $k_{s,\alpha}$ values can lead to big differences in the actual probability valuations. Then, the CCP may be trapped in poor local optima, where the sum of constraint violations is small, but the violation for the probability threshold is too large.

*Algorithmic improvements.* We list three key improvements that we make as opposed to a naive implementation of the approaches. (1) We efficiently precompute the states $s \in S$ that reach target states with probability 0 or 1. Then, we simplify the NLP in (1) – (6) accordingly. (2) Often, all instantiations with admissible parameter values yield well-defined MDPs. We verify this property via an easy preprocessing. Then, we omit the constraints for the well-definedness. (3) Parts of the encoding are untouched over multiple CCP iterations. Instead of rebuilding the encoding, we only update constraints which contain iteration-dependent values. The update is based on a preprocessed representation of the model. The improvement is two-fold: We spend less time constructing the encoding, and the solver reuses previous results, making solving up to three times faster.

## 6 Experiments

### 6.1 Implementation

We implement the CCP with the discussed efficiency improvements from Sect. 5 in the parameter synthesis framework PROPhESY [13]. We use the probabilistic model checker Storm [14] to extract an explicit representation of an pMDP. We keep the pMDP in memory, and update the parameter instantiations using an efficient data structure to enable efficient repetitive model checking. To solve convex QCQPs, we use Gurobi 7.5 [19], configured for numerical stability.

*Tuning constants.* Optimally, we would initialize the CCP procedure, i.e., $\hat{v}$ (for the parameters) and $\hat{p}_s$ (for the probability variables), with a feasible point, but that would require to already solve Problem 1. Instead, we instantiate $\hat{v}$ as the center of the parameter space, and thereby minimize the worst-case distance to a feasible solution. For $\hat{p}_s$, we use the threshold $\lambda$ from the specification $\mathbb{P}_{\leq\lambda}(\Diamond T)$ to initialize the probability variables, and analogously for expected cost. We initialize the penalty parameter $\tau = 0.05$ for reachability, and $\tau = 5$ for expected cost, a conservative number in the same order of magnitude as the values $\hat{p}_s$. As expected cost evaluations have wider ranges than probability evaluations, a larger $\tau$ is sensible. We pick $\mu = \max_{s \in S \setminus T} \hat{p}_s$. We update $\tau$ by adding $\mu$ after each iteration. Empirically, increasing $\tau$ with bigger steps is beneficial for the run time, but induces more numerical instability. In contrast, in the literature, the update parameter $\mu$ is frequently used as a constant, i.e., it is not updated between the iterations. In, e.g, [29], $\tau$ is multiplied by $\mu$ after each iteration.

### 6.2 Evaluation

*Set-up.* We evaluate on a HP BL685C G7 with 48 2 GHz cores, a 32 GB memory limit, and 1800 seconds time limit; the implementation only using a single thread. The task is to find feasible parameter valuation for pMCs and pMDPs with non-trivial upper/lower thresholds on probabilities/costs in the specifications, as in Problem 1. We ask for a well-defined valuation of the parameters, with

$\varepsilon_{\text{graph}} = 10^{-5}$. We run all the approaches with the exact same configuration of Storm. For pMCs, we enable weak bisimulation, which is beneficial for all presented examples. We do not use bisimulation for pMDPs.

We compare runtimes with a particle-swarm optimization (PSO) and two SMT-based approaches. PSO is a heuristic sampling approach which searches the parameter space, inspired by [10]. For each valuation, PSO performs model checking *without* rebuilding the model, rather it adapts the matrix from previous valuations. As PSO is a randomized procedure, we run it with random seeds 0–19. The PSO implementation requires the well-defined parameter regions to constitute a hyper-rectangle, as proper sampling from polygons is a non-trivial task. The first SMT approach directly solves the NLP (2) – (7) using the SMT solver Z3 [24]. The second SMT approach preprocesses the NLP using state elimination [12] as implemented in, e.g., PARAM, PRISM and Storm.

We additionally compare against a prototype of the geometric programming (GP) approach [11] based on CvxPy [15] and the solver SCS [30], and the QCQP package [31], which implements several heuristics, including a naive CCP approach, for nonconvex QCQPs. Due to numerical instabilities, we could not automatically apply these two approaches to a wide range of benchmarks.

*Benchmarks.* We include the standard pMC benchmarks from the PARAM website, which contain two parameters. We furthermore have a rich selection of strategy synthesis problems obtained from partially observable MDPs (POMDPs), cf. [25]: GridX are gridworld problems with trap states (A), finite horizons (B), or movement costs (C). Maze is a navigation problem. Network and Repudiation originate from distributed protocols. We obtain the pMDP benchmarks either from the PARAM website, or as parametric variants to existing PRISM case studies, and describe randomized distributed protocols.

*Results.* Table 1 contains an overview of the results. The first two columns refer to the benchmark instance, the next column to the specification evaluated. We give the states (States), transitions (Trans.) and parameters (Par.) *in the bisimulation quotient*, which is then used for further evaluation. We then give the *minimum* (tmin), the *maximum* (tmax) and *average* (**tavg**) runtime (in seconds) for PSO with different seeds, the best runtime obtained using SMT (**t**), and the runtime for CCP (**t**). For CCP, we additionally give the fraction (in percent) of time spent in Gurobi (solv), and the number of CCP iterations (iter). Table 2 additionally contains the number of actions (Act) for pMDPs. The boldfaced measures **tavg**, and **t** for both SMT and CCP are the important measures to compare. Boldface values are the ones with the best performance for a specific benchmark.

There is a constant overhead for model building, which is in particular large if the bisimulation quotient computation is expensive, see the small fraction of time spent solving CCPs for Crowds. For the more challenging models, this overhead is negligible. Roughly 80–90% of the time is spent within Gurobi in these models, the remainder is used to feed the CCPs into Gurobi. A specification threshold closer to the (global) optimum typically induces a higher number of iterations (see Maze or Netw with different threshold). For the pMDP Coin, optimal parameter

**Table 1.** pMC benchmark results

| Problem | | | Info | | | PSO | | | SMT | CCP | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|
| Set | Inst | Spec | States | Trans. | Par. | tmin | tmax | **tavg** | **t** | **t** | solv | iter |
| Brp | 16,2 | $\mathbb{P}_{\leq 0.1}$ | 98 | 194 | **2** | 0 | 0 | **0** | 40 | 0 | 30% | 3 |
| Brp | 512,5 | $\mathbb{P}_{\leq 0.1}$ | 6146 | 12290 | **2** | 24 | 36 | **28** | TO | 33 | 24% | 3 |
| Crowds | 10,5 | $\mathbb{P}_{\leq 0.1}$ | 42 | 82 | **2** | 4 | 5 | 5 | 8 | **4** | 2% | 4 |
| Nand | 5,10 | $\mathbb{P}_{\leq 0.05}$ | 10492 | 20982 | **2** | 21 | 51 | 28 | TO | **22** | 21% | 2 |
| Zeroconf | 10000 | $\mathbb{E}_{\leq 10010}$ | 10003 | 20004 | **2** | 2 | 4 | **3** | TO | 57 | 81% | 3 |
| GridA | 4 | $\mathbb{P}_{\geq 0.84}$ | 1026 | 2098 | **72** | 11 | 11 | **11** | TO | 22 | 81% | 11 |
| GridB | 8,5 | $\mathbb{P}_{\geq 0.84}$ | 8653 | 17369 | **700** | 409 | 440 | 427 | TO | **213** | 84% | 8 |
| GridB | 10,6 | $\mathbb{P}_{\geq 0.84}$ | 16941 | 33958 | **1290** | 533 | 567 | 553 | TO | **426** | 84% | 7 |
| GridC | 6 | $\mathbb{E}_{\leq 4.8}$ | 1665 | 305 | **168** | 261 | 274 | 267 | TO | **169** | 90% | 23 |
| Maze | 5 | $\mathbb{E}_{\leq 14}$ | 1303 | 2658 | **590** | 213 | 230 | 219 | TO | **67** | 89% | 8 |
| Maze | 5 | $\mathbb{E}_{\leq 6}$ | 1303 | 2658 | **590** | – | – | TO | TO | **422** | 85% | 97 |
| Maze | 7 | $\mathbb{E}_{\leq 6}$ | 2580 | 5233 | **1176** | – | – | TO | TO | **740** | 90% | 60 |
| Netw | 5,2 | $\mathbb{E}_{\leq 11.5}$ | 21746 | 63158 | **2420** | 312 | 523 | 359 | TO | **207** | 39% | 3 |
| Netw | 5,2 | $\mathbb{E}_{\leq 10.5}$ | 21746 | 63158 | **2420** | – | – | TO | TO | **210** | 38% | 4 |
| Netw | 4,3 | $\mathbb{E}_{\leq 11.5}$ | 38055 | 97335 | **4545** | – | – | TO | TO | MO | - | - |
| Repud | 8,5 | $\mathbb{P}_{\geq 0.1}$ | 1487 | 3002 | **360** | 16 | 22 | 18 | TO | **4** | 36% | 2 |
| Repud | 8,5 | $\mathbb{P}_{\leq 0.05}$ | 1487 | 3002 | **360** | 273 | 324 | 293 | TO | **14** | 72% | 4 |
| Repud | 16,2 | $\mathbb{P}_{\leq 0.01}$ | 790 | 1606 | **96** | – | – | TO | TO | **15** | 78% | 9 |
| Repud | 16,2 | $\mathbb{P}_{\geq 0.062}$ | 790 | 1606 | **96** | – | – | TO | TO | TO | - | - |

**Table 2.** pMDP benchmark results

| Problem | | | Info | | | | PSO | | | SMT | CCP | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| Set | Inst | Spec | States | Act | Trans. | Par. | tmin | tmax | **tavg** | **t** | **t** | solv | iter |
| BRP | 4,128 | $\mathbb{P}_{\leq 0.1}$ | 17131 | 17396 | 23094 | **2** | 45 | 47 | 46 | TO | **39** | 33% | 4 |
| Coin | 32 | $\mathbb{E}_{\leq 500}$ | 4112 | 6160 | 7692 | **2** | 117 | 119 | **118** | TO | TO | - | - |
| CoinX | 32 | $\mathbb{E}_{\leq 210}$ | 16448 | 24640 | 30768 | **2** | 1196 | 1222 | 1208 | TO | **32** | 78% | 3 |
| Zeroconf | 1 | $\mathbb{P}_{\geq 0.99}$ | 31402 | 55678 | 70643 | **3** | 18 | 19 | **19** | TO | 79 | 82% | 2 |
| CSMA | 2,4 | $\mathbb{E}_{\leq 69.3}$ | 7958 | 7988 | 10594 | **26** | n.s. | n.s. | n.s. | TO | **79** | 86% | 10 |
| Virus | - | $\mathbb{E}_{\leq 10}$ | 809 | 3371 | 6741 | **18** | 113 | 113 | 113 | TO | **13** | 76% | 4 |
| Wlan | 0 | $\mathbb{E}_{\leq 580}$ | 2954 | 3972 | 5202 | **15** | n.s. | n.s. | n.s. | TO | **7** | 72% | 2 |

values are on the boundary of the parameter space and quickly reached by PSO. The small parameter values together with the rather large expected costs are numerically challenging for CCP. For CoinX, the parameter values are in the interior of the parameter space and harder to hit via sampling. For CCP, the difference between small and large coefficients is smaller than in Coin, which yields better convergence behavior. The benchmarks CSMA and WLAN are currently not supported by PSO due to the non-rectangular well-defined parameter space.

CCP does not solve all instances: In Netw (4,3), CCP exceeds the memory limit. In Repud, finding values close the global optimum requires too much time. While the thresholds used here are close to the global optima, actually finding the global optimum itself is always challenging.

*Effect of integrating model checking for CCP.* The benchmark-set Maze profits most: Discarding the model checking results in our CCP implementation always yields time-outs, even for the rather simple Maze, with threshold 14, which is

solved with usage of model checking results within 30 seconds. Here, using model checking results thus yields a speed-up by a factor of at least 60. More typical examples are Netw, where discarding the model checking results yields a factor 5 performance penalty. The Repud examples do not significantly profit from using intermediate model checking results.

*Evaluation of the QCQP package, GP and SMT.* We evaluate the GP on pMCs with two parameters: For the smaller BRP instance, the procedure takes 90 seconds, for Crowds 14 seconds. Other instances yield timeouts. We also evaluate the QCQP package on some pMCs. For the smaller BRP instance, the package finds a feasible solution after 113 seconds. For the Crowds instance, it takes 13 seconds. For a Repud instance with 44 states, and 26 parameters, the package takes 54 seconds and returns a solution that violates the specification. CCP with integrated model checking takes less than a second.

The results in Tables 1 and 2 make obvious that *SMT is not competitive*, irrespectively whether the NLP is preprocessed via state elimination. Moreover, state elimination (for pMCs) within the given time limit *is only possible for those (considered) models with 2 parameters*, using either PRISM, PARAM, or Storm.

### 6.3 Discussion

*A tuned variant of CCP improves the state-of-the-art.* Just applying out-of-the-box heuristics for QCQPs—like realized in the QCQP package or using our CCP implementation without integrated model checking—does not yield a scalable method. To solve the nonconvex QCQP, we require a CCP with a clever encoding, cf. Sect. 5, and several algorithmic improvements. State space reductions shrink the encoding, and model checking after each CCP iteration to terminate earlier typically saves 20% of iterations. Especially when convergence is slow, model checking saves significantly more iterations. Moreover, feeding model checking results into the CCP improves runtime by up to an additional order of magnitude, at negligible costs. These combined improvements to the CCP method outnumbers any solver-based approach by orders of magnitude, and is often superior to sampling based approaches, especially in the presence of many parameters. Benchmarks with many parameters pose two challenges for sampling based approaches: Sampling is necessarily sparse due to the high dimension, and optimal parameter valuations for one parameter often depend significantly on other parameter values.

*CCP performance can be boosted with particular benchmarks in mind.* For most benchmarks, choosing larger values for $\tau$ and $\mu$ improves the performance. Furthermore, for particular benchmarks, we can find a better initial value for $\hat{p}_s$ and $\hat{v}$. These adaptions, however, are not suitable for a general framework. Values used here reflect a more balanced performance over several types of benchmarks. On the downside, the dependency on the constants means that minor changes in the encoding may have significant, but hard to predict, effect. For SMT-solvers, additional and superfluous constraints often help steering the solver, but in the context of CCP, it diminishes the performance.

*Some benchmarks constitute numerically challenging problems.* For specification thresholds close to global optima and for some expected cost specifications in general, feasible parameter values may be very small. Such extremal parameter values induce CCPs with large differences between the smallest and largest coefficient in the encoding, which are numerically challenging. The pMDP benchmarks are more susceptible to such numerical issues.

## 7 Conclusion and Future Work

We presented a new approach to parameter synthesis for pMDPs. To solve the underlying nonconvex optimization problem efficiently, we devised a method to efficiently employ a heuristic procedure with integrated model checking. The experiments showed that our method significantly improves the state-of-the-art.

In the future, we will investigate how to automatically handle nonaffine transition functions. To further improve the performance, we will implement a hybrid approach between PSO and the CCP-based method.

## References


1. Aflaki, S., Volk, M., Bonakdarpour, B., Katoen, J.P., Storjohann, A.: Automated fine tuning of probabilistic self-stabilizing algorithms. In: SRDS. pp. 94–103. IEEE CS (2017)
2. Alizadeh, F., Goldfarb, D.: Second-order cone programming. Math Program. **95**(1), 3–51 (2003)
3. Amato, C., Bernstein, D.S., Zilberstein, S.: Solving POMDPs using quadratically constrained linear programs. In: AAMAS. pp. 341–343. ACM (2006)
4. Baier, C., Katoen, J.P.: Principles of Model Checking. MIT Press (2008)
5. Bartocci, E., Grosu, R., Katsaros, P., Ramakrishnan, C., Smolka, S.: Model repair for probabilistic systems. In: TACAS, LNCS, vol. 6605, pp. 326–340. Springer (2011)
6. Boyd, S., Kim, S.J., Vandenberghe, L., Hassibi, A.: A tutorial on geometric programming. Optimization and Engineering **8**(1) (2007)
7. Boyd, S., Vandenberghe, L.: Convex Optimization. Cambridge University Press, New York, NY, USA (2004)
8. Burer, S., Saxena, A.: The MILP road to MIQCP. Mixed Integer Nonlinear Programming pp. 373–405 (2012)
9. Calinescu, R., Ghezzi, C., Kwiatkowska, M., Mirandola, R.: Self-adaptive software needs quantitative verification at runtime. Commun. ACM **55**(9), 69–77 (2012)
10. Chen, T., Hahn, E.M., Han, T., Kwiatkowska, M., Qu, H., Zhang, L.: Model repair for Markov decision processes. In: TASE. pp. 85–92. IEEE CS (2013)
11. Cubuktepe, M., Jansen, N., Junges, S., Katoen, J.P., Papusha, I., Poonawala, H.A., Topcu, U.: Sequential convex programming for the efficient verification of parametric mdps. In: TACAS (2). LNCS, vol. 10206, pp. 133–150 (2017)
12. Daws, C.: Symbolic and parametric model checking of discrete-time Markov chains. In: ICTAC. LNCS, vol. 3407, pp. 280–294. Springer (2004)
13. Dehnert, C., Junges, S., Jansen, N., Corzilius, F., Volk, M., Bruintjes, H., Katoen, J.P., Ábrahám, E.: PROPhESY: A probabilistic parameter synthesis tool. In: CAV (1). LNCS, vol. 9206, pp. 214–231. Springer (2015)

14. Dehnert, C., Junges, S., Katoen, J.P., Volk, M.: A storm is coming: A modern probabilistic model checker. In: CAV (2). LNCS, vol. 10427, pp. 592–600. Springer (2017)
15. Diamond, S., Boyd, S.: CVXPY: A Python-embedded modeling language for convex optimization. Journal of Machine Learning Research **17**(83), 1–5 (2016)
16. Duflot, M., Fribourg, L., Hérault, T., Lassaigne, R., Magniette, F., Messika, S., Peyronnet, S., Picaronny, C.: Probabilistic model checking of the CSMA/CD protocol using PRISM and APMC. Electr. Notes TCS **128**(6), 195–214 (2005)
17. Filieri, A., Tamburrelli, G., Ghezzi, C.: Supporting self-adaptation via quantitative verification and sensitivity analysis at run time. IEEE Trans. Software Eng. **42**(1), 75–99 (2016)
18. Gainer, P., Hahn, E.M., Schewe, S.: Incremental verification of parametric and reconfigurable Markov chains. CoRR **abs/1804.01872** (2018)
19. Gurobi Optimization, Inc.: Gurobi optimizer reference manual. http://www.gurobi.com (2013)
20. Hahn, E.M., Hermanns, H., Wachter, B., Zhang, L.: PARAM: A model checker for parametric Markov models. In: CAV. LNCS, vol. 6174, pp. 660–664. Springer (2010)
21. Hahn, E.M., Hermanns, H., Zhang, L.: Probabilistic reachability for parametric Markov models. STTT **13**(1), 3–19 (2010)
22. Hahn, E.M., Li, Y., Schewe, S., Turrini, A., Zhang, L.: iscasMc: A web-based probabilistic model checker. In: FM. LNCS, vol. 8442, pp. 312–317. Springer (2014)
23. Hutschenreiter, L., Baier, C., Klein, J.: Parametric Markov chains: PCTL complexity and fraction-free Gaussian elimination. In: GandALF. EPTCS, vol. 256, pp. 16–30 (2017)
24. Jovanovic, D., de Moura, L.M.: Solving non-linear arithmetic. In: IJCAR. LNCS, vol. 7364, pp. 339–354. Springer (2012)
25. Junges, S., Jansen, N., Wimmer, R., Quatmann, T., Winterer, L., Katoen, J.P., Becker, B.: Finite-state controllers of POMDPs using parameter synthesis. In: UAI. AUAI Press (2018), to appear
26. Kwiatkowska, M., Norman, G., Parker, D.: PRISM 4.0: Verification of probabilistic real-time systems. In: CAV. LNCS, vol. 6806, pp. 585–591. Springer (2011)
27. Lanotte, R., Maggiolo-Schettini, A., Troina, A.: Parametric probabilistic transition systems for system design and analysis. Formal Aspects Comput. **19**(1), 93–109 (2007)
28. Linderoth, J.: A simplicial branch-and-bound algorithm for solving quadratically constrained quadratic programs. Math. Program. **103**(2), 251–282 (2005)
29. Lipp, T., Boyd, S.: Variations and extension of the convex–concave procedure. Optimization and Engineering **17**(2), 263–287 (2016)
30. O'Donoghue, B., Chu, E., Parikh, N., Boyd, S.: Conic optimization via operator splitting and homogeneous self-dual embedding. Journal of Optimization Theory and Applications **169**(3), 1042–1068 (2016)
31. Park, J., Boyd, S.: General heuristics for nonconvex quadratically constrained quadratic programming. arXiv preprint arXiv:1703.07870 (2017)
32. Shen, X., Diamond, S., Gu, Y., Boyd, S.: Disciplined convex-concave programming. In: CDC. pp. 1009–1014. IEEE (2016)
33. Su, G., Rosenblum, D.S., Tamburrelli, G.: Reliability of run-time quality-of-service evaluation using parametric model checking. In: ICSE. pp. 73–84. ACM (2016)

## A Proof of Theorem 2

*Proof.* Since the assignment of the convexified DC problem in (27) – (33) is feasible with

$$\tau \sum_{\forall s \in S \setminus T} \sum_{\forall \alpha \in Act} k_{s,\alpha} = 0,$$

we know that the assignment is feasible for the QCQP in (19) – (23) and, by definition, for the NLP in (1) – (7). We will show that for every $s \in S$ we have $\Pr(\mathcal{M}[u], \Diamond T, s) \leq p_s$, for any feasible assignment for the NLP in (1) – (7).

For $s \in S$, define $q_s = \Pr_s(\mathcal{M}[u], \Diamond T)$ (the probability to reach $T$ from $s$ in $\mathcal{M}[u]$) and $x_s = q_s - p_s$. Let $S_< = \{s \in S \mid p_s < q_s\}$.

For states $s \in T$ we have, by (2) that $p_s = 1 = q_s = 1$, meaning that $s \notin S_<$. For states $s$ with $q_s = 0$, i.e., states from which $T$ is almost surely not reachable, we have trivially $p_s \geq q_s$, also implying $s \notin S_<$. Therefore, for every $s \in S_<$, $p_s$ satisfies (6) and $T$ is reachable with positive probability.

Assume, for the sake of contradiction, that $S_< \neq \emptyset$, and let $x_{max} = \max\{x_s \mid s \in S\}$, and $S_{max} = \{s \in S \mid x_s = x_{max}\}$.

The assumption that $S_< \neq \emptyset$ implies $x_{max} > 0$. Let $s \in S$ be such that $x_s = x_{max}$. Therefore, $s \in S_<$, and thus for all $\alpha \in Act$, we have that

$$x_s \geq \sum_{s' \in S} \mathcal{P}(s, \alpha, s') \cdot p_{s'}. \tag{34}$$

On the other hand, there exists an $\alpha \in Act$ such that we have

$$q_s = \sum_{s' \in S} \mathcal{P}(s, \alpha, s') \cdot q_{s'}. \tag{35}$$

Thus,

$$q_s - p_s \leq \sum_{s' \in S} \mathcal{P}(s, \alpha, s') \cdot (q_{s'} - p_{s'}), \tag{36}$$

which is equivalent to

$$x_s \leq \sum_{s' \in S} \mathcal{P}(s, \alpha, s') \cdot x_{s'}. \tag{37}$$

Since for all $\alpha \in Act$, and $s' \in S$, we have that $\mathcal{P}(s, \alpha, s') \geq 0$ and $\sum_{s' \in S} \cdot \mathcal{P}(s, \alpha, s') = 1$, using the inequality in (36) we establish

$$x_{max} = x_s \leq \sum_{s' \in S} \mathcal{P}(s, \alpha, s') \cdot x_{s'} \tag{38}$$

$$\leq \sum_{s' \in S} \mathcal{P}(s, \alpha, s') \cdot x_{max} \tag{39}$$

$$\leq x_{max} \sum_{s' \in S} \mathcal{P}(s, \alpha, s') = x_{max}. \tag{40}$$

Which implies that all the inequalities are equalities, meaning that

$$x_{max} = x_s = \sum_{s' \in S} \mathcal{P}(s, \alpha, s') \cdot x_{s'} \tag{41}$$

$$= x_{max} \cdot \sum_{s' \in S} \mathcal{P}(s, \alpha, s'). \tag{42}$$

The equation in (42) gives us that $x_{s'} = x_{max} > 0$ for every successor $s'$ of $s$ in $\mathcal{M}[u]$. Since $s \in S_{max}$ was chosen arbitrarily, for every state $s \in S_{max}$, all successors of $s$ are also in $S_{max}$. As we established that $S_< \cap T = \emptyset$, it is necessary that $T$ is not reachable with positive probability from any $s \in S_{max}$, which is a contradiction with the fact that from every state in $S_<$, the set $T$ is reachable. □