Abstract—Probabilistic model checking aims to prove whether a Markov decision process (MDP) satisfies a temporal logic specification. The underlying methods rely on an often unrealistic assumption that the MDP is precisely known. Consequently, parametric MDPs (pMDPs) extend MDPs with transition probabilities that are functions over unspecified parameters. The parameter synthesis problem is to compute an instantiation of these unspecified parameters such that the resulting MDP satisfies the temporal logic specification. We formulate the parameter synthesis problem as a quadratically constrained quadratic program (QCQP), which is nonconvex and is NP-hard to solve in general. We develop two approaches that iteratively obtain locally optimal solutions. The first approach exploits the so-called convex-concave procedure (CCP), and the second approach utilizes a sequential convex programming (SCP) method. The techniques improve the runtime and scalability by multiple orders of magnitude compared to black-box CCP and SCP by merging ideas from convex optimization and probabilistic model checking. We demonstrate the approaches on a satellite collision avoidance problem with hundreds of thousands of states and tens of thousands of parameters and their scalability on a wide range of commonly used benchmarks.

I. INTRODUCTION

Markov decision processes (MDPs) are widely studied models for sequential decision-making [1]. MDPs have been used, for example, in robotic applications [2], aircraft collision avoidance systems [3], and Mars robot missions [4]. The formal verification of temporal logic specifications has been extensively studied for MDPs [5]. Such specifications are able to express properties such as “the maximum probability of reaching a set of goal states without colliding with an obstacle is more than 99%” or “the minimum expected time of reaching the target location is less than 10 seconds”. A crucial, yet possibly unrealistic, assumption in probabilistic model checking is that the transition and cost functions of the MDP are exactly known. However, these values are often estimated from data and may not be obtained exactly.

More general models express cost and probabilities as functions over parameters whose values are left unspecified [6]–[8]. Such parametric MDPs (pMDPs) describe uncountable sets of MDPs. A well-defined instantiation of the parameters yields an instantiated, parameter-free MDP. Applications of pMDPs include adaptive software systems [9], sensitivity analysis [10], optimizing randomized distributed algorithms [11], and synthesis of finite-memory strategies for partially observable MDPs (POMDPs) [12].

For a given finite-state pMDP, the parameter synthesis problem is to compute a parameter instantiation such that the instantiated MDP satisfies a given temporal logic specification. Solving this problem is ETR-complete, i.e., as hard as finding a root to a multivariate polynomial [13]. Consequently, the problem is NP-hard and in PSPACE. A straightforward approach to the parameter synthesis problem relies on an exact encoding into a nonlinear programming problem (NLP) [14] or a satisfiability modulo-theories formula [13]. These approaches are, in general, limited to a few states and parameters.

To solve the parameter synthesis problem, it suffices to guess a correct parameter instantiation. This insight has led to an adaptation of sampling-based techniques to the parameter synthesis problem [15], most prominently particle swarm optimization (PSO) [16]. After guessing an instantiation, one can efficiently verify the resulting associated parameter-free MDP to determine whether the specification is satisfied. These techniques can handle millions of states, but are restricted to a few parameters. Their performance degrades significantly with increasing parameters, e.g., with more than ten [15], [16].

How do these methods construct a good guess? One natural method is to solve the original problem exactly. More precisely, exact methods translate the parameter synthesis problem into an equivalent nonlinear program. The advantage of these methods is that they only need one iteration. However, this iteration is, in general, very costly. On the other hand, sampling-based methods completely disregard the model structure. Technically, they sample parameter instantiations from some prior, e.g., a uniform distribution from all samples. The advantage is that every iteration is fast, but one may need a tremendous amount of iterations or samples. These two approaches for the parameter synthesis problem are the extremal instances on the level of approximation, see Fig. 1.

Fig. 1: The spectrum of the solution approaches for the parameter synthesis problem.
To obtain a locally optimal solution to this QCQP, we use values for each parameter after solving the linearized problem of benchmarks shows that the CCP and SCP method can solve particular, an extensive empirical evaluation on a broad range convergence properties of the proposed CCP and SCP method. We discuss these values as an input for the next iteration if the probability is specification compared to the previous instantiation. We use the instantiation improves the probability of satisfying the and model check the instantiated MDP. We check whether step into SCP, similar to our first approach. Instead, we use the QCQP. Second, we integrate a probabilistic model checking constraints and re-solve the linearized problem. We discuss the issue while solving the linearized problem. First, we use so-called trust region constraints [23]–[25] to ensure that the existing SCP methods. Specifically, the solution obtained from solving the parameter synthesis problem.

In our first approach, we transform the NLP into a quadratically-constrained quadratic program (QCQP). However, the resulting QCQP is nonconvex and is NP-hard to solve [17], [18]. To obtain a locally optimal solution to this QCQP, we use the so-called convex-concave procedure (CCP) [19]. To that end, we reformulate the nonconvex QCQP as a difference-of-convex (DC) problem. All constraints and the objective of a DC problem are a difference of two convex functions. CCP computes a locally optimal solution to the resulting DC problem by convexifying it as a convex quadratic problem. The resulting convex quadratic problem can be solved by state-of-the-art solvers such as Gurobi [20]. We also integrate the CCP procedure with probabilistic model checking, which yields a speedup of multiple orders of magnitude compared to the existing CCP solvers. This approach was published as a preliminary conference paper [21], and our presentation is partially based on [22].

In our second approach, we exploit a sequential convex programming (SCP) method [23]–[25] to solve the parameter synthesis problem. We convexify the nonconvex QCQP into a linear program (LP) by linearizing the nonconvex constraints around a previous solution. Similar to CCP, SCP iteratively computes a locally optimal solution to the nonconvex QCQP. However, unlike CCP, the linearization in SCP does not over approximate the functions in the constraints Therefore, a feasible solution to the linearized problem in SCP may be infeasible to the parameter synthesis problem, unlike in CCP.

Existing SCP methods can ensure the correctness of the solution only when the starting point is feasible, which amounts to solving the parameter synthesis problem.

In this paper, we address the key critical shortcomings of the existing SCP methods. Specifically, the solution obtained from SCP may not be feasible to the parameter synthesis problem due to approximation errors in linearization and potential numerical issues while solving the linearized problem. First, we use so-called trust region constraints [23]–[25] to ensure that the linearized problem accurately approximates the nonconvex QCQP. Second, we integrate a probabilistic model checking step into SCP, similar to our first approach. Instead, we use the values for each parameter after solving the linearized problem and model check the instantiated MDP. We check whether the instantiation improves the probability of satisfying the specification compared to the previous instantiation. We use these values as an input for the next iteration if the probability is improved. Otherwise, we contract the radius of the trust region constraints and re-solve the linearized problem. We discuss the convergence properties of the proposed CCP and SCP method.

We integrate the proposed CCP and SCP methods with a probabilistic model checker in the tool PROPhESY [26]. In particular, an extensive empirical evaluation on a broad range of benchmarks shows that the CCP and SCP method can solve the parameter synthesis problem for models with hundreds of thousands of states and tens of thousands of parameters as opposed to few parameters for the existing parameter synthesis tools. Thus, the resulting methods (1) solve multiple orders of magnitude larger problems compared to other parameter synthesis tools, (2) yield an improvement of multiple orders of magnitude in runtime compared to just using CCP and SCP as a black box, and (3) ensure the correctness of the solution and have favourable convergence properties.

B. Related work

Traditionally, approaches for solving the parameter synthesis problems have been built around the notion of abstracting the parametric model into a solution function, similar to our approaches. The solution function is the probability of satisfying the temporal logic specification as a function of the model parameters [6], [8], [25], [27]. The solution function can be exploited the probabilistic model checking tools PARAM [8], PRISM [28] and Storm [29] to solve the parameter synthesis problem. This function is exponentially large in the parameters, and solving the problem is again exponential in the number of parameters, making the whole approach doubly exponential [30]. Consequently, these approaches typically can handle millions of states but only a handful of parameters.

Moreover, these approaches require a fixed policy or has to introduce a parameter for every state/action-pair in the MDP.

Orthogonally, Quatmann et al. [31] address an alternative parameter synthesis problem which focuses on proving the absence of parameter instantiations. The method iteratively solves simple stochastic games. Spel et al. [32] consider proving that the parameters behave monotonically, allowing for faster sampling-based approaches. However, this method is limited to a few parameters. A recent survey on parameter synthesis in Markov models can be found in [33].

Further variations of parameter synthesis, e.g., consider statistical guarantees for parameter synthesis, often with some prior on the parameter values [34]–[36]. These approaches cannot provide the absolute guarantees on an answer that the methods in this paper provide.

Parametric MDPs generalize interval models [37], [38]. Such interval models have also been considered with convex uncertainties [39]–[42]. However, the resulting problems with interval models are easier to solve due to the lack of dependencies (or couplings) between parameters in different states.

Finally, similar convex-optimization-based methods to our approaches have been used to synthesize finite-memory strategies for POMDPs [43], [44]. However, our techniques can solve POMDPs that are significantly larger than these methods.

C. Organization

We first provide the necessary preliminaries in Section [IV]. Then we provide the problem statement and formulate the problem as a nonconvex QCQP in Section [III]. We develop the CCP method in Section [IV] and the SCP method in Section [V]. Section [VI] discusses the convergence rate of the methods. Section [VII] shows numerical examples and demonstrates the scalability of the approaches. We conclude our paper and discuss possible future directions in Section [VIII].
II. Preliminaries

A probability distribution over a finite or countably infinite set $X$ is a function $\mu: X \to [0, 1] \subseteq \mathbb{R}$ with $\sum_{x \in X} \mu(x) = 1$. The set of all distributions on $X$ is denoted by $\text{Distr}(X)$. Let $V = \{v_1, \ldots, v_n\}$ be a finite set of variables over the real numbers $\mathbb{R}$. The set of multivariate polynomials in $[0, 1]$ over $V$ is $\mathbb{Q}[V]$. An instantiation for $V$ is a function $v: V \to \mathbb{R}$.

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

For $s \in S$, $A(s) = \{\alpha \in Act \mid \exists s' \in S. \mathcal{P}(s, \alpha, s') \neq 0\}$ 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 $s \in S$, $M$ is a parametric discrete-time Markov chain (pMC). pMDPs can be equipped with a state-action cost function $c: S \times Act \to \mathbb{R}_{\geq 0}$.

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

To define measures on MDPs, nondeterministic choices are resolved by a so-called strategy $\sigma: S \to Act$ with $\sigma(s) \in A(s)$. The set of all strategies over $M$ is $\text{Str}^{M}$. For the measures in this paper, memoryless deterministic strategies suffice [5].

Applying a strategy to an MDP yields an induced Markov chain where all nondeterminism is resolved.

For an MC $D$, the reachability specification $\varphi_r = P_{\leq \lambda}(\diamond T)$ 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 $D$, we write $D \models \varphi_r$. Accordingly, for an expected cost specification $\varphi_c = E_{\leq \kappa}(\diamond G)$, $D \models \varphi_c$ holds 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 [5] Ch. 10. We note that linear temporal logic specifications can be reduced to reachability specifications, and we refer the reader to [5] for a detailed introduction. An MDP $M$ satisfies a specification $\varphi$, written $M \models \varphi$, if and only if for all strategies $\sigma \in \text{Str}^{M}$ it holds that $M^{\sigma} \models \varphi$.

III. Formal Problem Statement

In this section, we state the parameter synthesis problem, which is to compute a parameter instantiation such that the instantiated MDP satisfies the given temporal logic specification. We then discuss the nonlinear program formulation of the parameter synthesis problem, which forms the basis of our solution methods.

Problem 1 (Parameter synthesis problem). Given a pMDP $M = (S, s_i, Act, V, \mathcal{P})$, and a reachability specification $\varphi_r = P_{\leq \lambda}(\diamond T)$, compute a well-defined instantiation $v: V \to \mathbb{R}$ for $M$ such that $M[v] \models \varphi_r$.

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

For a given well-defined instantiation $v$, Problem 1 can be solved by verifying whether $M[v] \models \varphi_r$. The standard formulation uses a linear program (LP) to minimize the probability $p_{s_1}$ of reaching the target set $T$ from the initial state $s_1$ while ensuring that this probability is realizable under any strategy [5] Ch. 10. The straightforward extension of this approach to pMDPs to compute a satisfiable instantiation $v$ yields the following nonlinear program (NLP) [21], [43] with the variables $p_s$ for $s \in S$, and the parameter variables in $V$ in the transition function $\mathcal{P}(s, \alpha, s')$ for $s, s' \in S$ and $\alpha \in Act$:

\[
\begin{align*}
\text{minimize} & \quad p_{s_1} \\
\text{subject to} & \quad p_s = 1, \quad \forall s \in T, \\
& \quad \mathcal{P}(s, \alpha, s') \geq 0, \quad \forall s, s' \in S \setminus T, \forall \alpha \in A(s), \\
& \quad \sum_{s' \in S} \mathcal{P}(s, \alpha, s') = 1, \quad \forall s \in S \setminus T, \forall \alpha \in A(s), \\
& \quad \lambda \geq p_{s_1}, \\
& \quad p_s \geq \sum_{s' \in S} \mathcal{P}(s, \alpha, s') \cdot p_{s'} \quad \forall s \in S \setminus T, \forall \alpha \in A(s).
\end{align*}
\]

For $s \in S$, the probability variable $p_s \in [0, 1]$ represents an upper bound of the probability of reaching target set $T \subseteq S$. The parameters in the set $V$ enter the NLP as part of the functions from $\mathbb{Q}[V]$ in the transition function $\mathcal{P}$. The constraint (5) ensures that the probability of reaching $T$ is below the threshold $\lambda$. This constraint is optional for stating the problem, but we use the constraint in our methods for finding a parameter instantiation that satisfies the specification $\varphi$. We minimize $p_{s_1}$ to assign probability variables their minimal values with respect to the parameters $V$.

The probability of reaching a state in $T$ from $T$ is set to one (6). The constraints (3) and (4) ensure well-defined transition probabilities. Recall that $\mathcal{P}(s, \alpha, s')$ is an affine function in $V$. Therefore, the constraints (3) and (4) only depend on the parameters in $V$, and they are affine in the parameters. Constraint (5) is optional but necessary later, and ensures that the probability of reaching $T$ is below the threshold $\lambda$. For each state $s \in S \setminus T$ and action $\alpha \in A(s)$, the probability induced by the maximizing scheduler is a lower bound to the probability variables $p_s$ (6). To assign probability variables to their minimal values with respect to the parameters in $V$, $p_{s_1}$ is minimized in the objective (1). We state the correctness of the NLP in Proposition 1.

Proposition 1. The NLP in (1) – (6) computes the minimal probability of reaching $T$ under a maximizing strategy, and an instantiation $v$ is feasible to the NLP if and only if $M[v] \models \varphi_r$.

Proof. The NLP in (1) – (6) is an extension of the LP in [5] Theorem 10.105. We refer to [22] Theorem 4.20 for a formal proof.

Remark 1 (Graph-preserving instantiations). In the LP for-
ulation 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 ensure that the variables encode the actual reachability probabilities. We do the same. This preprocessing requires the underlying graph of the pMDP to be preserved under any valuation of the parameters. Thus, as in [8], [26], we consider only graph-preserving valuations. Concretely, we exclude valuations \( v \) with \( f[v] = 0 \) for \( f \in \mathcal{P}(s, \alpha, s') \) for all \( s, s' \in S \) and \( \alpha \in \text{Act} \). We replace the set of constraints (3) by

\[
\forall s, s' \in S. \forall \alpha \in \mathcal{A}(s). \quad \mathcal{P}(s, \alpha, s') \geq \varepsilon_{\text{graph}}, \quad (7)
\]

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

We demonstrate the constraints for the NLP in (1) – (7) for a pMC by Example 1

**Example 1.** Consider the pMC in Fig. 2 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:

\[
\text{minimize} \quad p_{s_0} \\
\text{subject to} \quad p_{s_3} = 1, \quad (9) \\
\lambda \geq p_{s_0} \geq v \cdot p_{s_1}, \quad (10) \\
p_{s_1} \geq (1 - v) \cdot p_{s_2}, \quad (11) \\
p_{s_2} \geq v \cdot p_{s_3}, \quad (12) \\
1 - \varepsilon_{\text{graph}} \geq v \geq \varepsilon_{\text{graph}}. \quad (13)
\]

- **a) Expected cost specifications:** The NLP in (1) – (7) considers reachability probabilities. If we have instead an expected cost specification \( \phi_c = E_{s \in S}(\mathcal{G}) \), we replace (2), (5), and (6) in the NLP by the following constraints:

\[
p_s = 0, \forall s \in G, \quad (14) \\
p_s \geq c(s, \alpha) + \sum_{s' \in S} \mathcal{P}(s, \alpha, s') \cdot p_{s'}, \forall s \in S \setminus G, \forall \alpha \in \mathcal{A}(s), \quad (15) \\
k \geq p_{s_1}. \quad (16)
\]

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, \( p_{s_1} \) is bounded by the threshold \( k \).

For the remainder of this paper, we restrict pMDPs to be affine, see Definition 1. For an affine pMDP \( \mathcal{M} \), the functions in the resulting NLP (1) – (5) for pMDP synthesis are affine in \( V \). However, the functions in the constraints (6) are quadratic, as a result of multiplying affine functions occurring in \( \mathcal{P} \) with the probability variables \( p_{s'} \). Therefore, the problem in (1) – (6) is a quadratically constrained quadratic program (QCQP) [46] and is generally nonconvex [21].

**Fig. 2:** A pMC with a single parameter \( v \).

**Fig. 3:** Penalty CCP loop. It starts with some assignment for parameters \( \hat{v} \) and probability variables \( \hat{p} \) and iteratively solves convex problems until the penalty variables converge to zero.

**Remark 2.** In the literature, pMDPs and pMCs appearing in benchmarks and case-studies are almost exclusively affine. Furthermore, from a complexity-theoretic point of view, solving for these pMDPs is as hard as \( \varepsilon \) when considering the general pMDP definition [13]. We refer to [22] Sec. 5.1.1 for a discussion of subclasses of pMDPs.

**IV. CONVEX-CONCAVE PROCEDURE**

In this section, we present our solution based on the penalty convex-concave procedure (CCP) [19], which iteratively over-approximates a nonconvex optimization problem. Specifically, we rewrite the quadratic functions in (6) as a sum of convex and concave functions and linearize the concave functions. The resulting convex problem can then be solved efficiently, and the process is iterated until a suitable solution is found. However, the convergence conditions of CCP might be too conservative if the initial solution is infeasible. Specifically, the obtained parameter instantiation and the instantiated MDP might solve the parameter synthesis problem, even though it cannot be certified by the convergence conditions of CCP. Motivated by this fact, we integrate a model checking procedure into the CCP, which ensures the numerical stability of the solution, and certifies whether a computed instantiation solves the parameter synthesis problem. We remark that the model checking procedure is also a critical part of our second approach, which is based on sequential convex programming.

We depict the CCP approach for solving the nonconvex QCQP in (1) – (6) in Fig. 3. The approach searches for a solution of the nonconvex QCQP by solving approximations of the QCQP in the form of convexified problems around some initial assignment for the parameters \( \hat{v} \) and probability variables \( \hat{p} \). After solving the convexified problem, we obtain the values of the penalty variables of this solution. If this penalty is zero, we have found a solution to the original nonconvex QCQP. Otherwise, we update (or guess) a new \( \hat{v} \) and \( \hat{p} \) which we use to convexify the QCQP. We denote the obtained solution from the convexified problem for the parameter variables as \( \hat{v} \) and for the probability variables as \( \hat{p} \). This loop may converge to a solution with positive values of the penalty variables, requiring restarting from another initial parameter instantiation.

**A. Constructing a convex approximation**

We start with the construction of the penalty DC problem, then discuss updating the variables. For compact notation, let
be the quadratic function in \( V \) and \( p_s \), i.e.,
\[
h(s, \alpha, s') = \mathcal{P}(s, \alpha, s') \cdot p_s
\]
for any \( s, s' \in S, \alpha \in A(s) \) whose right-hand side is part of the constraint \( (2) \) in the nonconvex QCQP. Note that \( h(s, \alpha, s) \) is an affine function in \( V \) for affine MDPs. We first write this quadratic function as a difference of two convex functions. For simplicity, let
\[
\mathcal{P}(s, \alpha, s') = 2d \cdot y + c, \text{ and } p_s = z,
\]
where \( y \) is the parameter variable, \( z \) is the probability variable, and \( c \in \mathbb{R}, d \in \mathbb{R}_+ \) are constants. We equivalently rewrite each bilinear function \( h(s, \alpha, s') \) as
\[
2d \cdot y + d(y^2 + z^2) - d(y^2 + z^2) + c \cdot z = d(y + z)^2 - d(y^2 + z^2) + c \cdot z.
\]
The function \( d(y + z)^2 + c \cdot z \) is a quadratic convex function in \( y \) and \( z \). In the remainder, let \( h_{cvx}(s, \alpha, s') := d(y + z)^2 \). We show an example of the resulting DC problem in Example \( 2 \)

**Example 2.** Recall the pMC in Fig. \( 2 \) and the QCQP from Example \( 1 \). All quadratic constraints in the example are nonconvex. We construct a DC problem with \( d = 0.5 \) as

**Proposition 2.** A feasible solution to the nonconvex QCQP in \( (1) - (6) \) is also feasible to the convexified QCQP in \( (1) - (5) \) together with \( (18) \).

**Proof.** For any concave function \( f \), its first-order approximation is a global over-estimator, i.e., for \( v, v' \in V \):
\[
f(v) \leq f(v') + \nabla f(v')^\top (v' - v),
\]
leading to \( h_{cvx}(s, \alpha, s') \leq h_{aff}(s, \alpha, s') \). Therefore, by construction, any feasible solution that satisfies \( (6) \) also satisfies \( (18) \). The claim follows, as all other constraints coincide.

The stricter set of constraints is often too strict and may not have any feasible solution. We add a penalty variable \( k_s \) for all \( s \in S \setminus T \) to all convexified constraints, which guarantees that the DC problem is feasible. These variables allow us to measure the “amount” of infeasibility. The larger the assigned value for a penalty variable, the larger the violation. We then seek to minimize the violation of the original DC constraints by minimizing the sum of the penalty variables. The resulting convexified problem with the penalty variables is given by

**Theorem 1.** A feasible solution of the convex DC problem in \( (20) - (26) \)

\[
\begin{aligned}
\text{minimize} & \quad p_s + \tau \sum_{s \in S \setminus T} k_s \\
\text{subject to} & \quad p_s = 1, \quad \forall s \in T,
\end{aligned}
\]

\[
\begin{aligned}
\mathcal{P}(s, \alpha, s') & \geq \varepsilon_{\text{graph}}, \quad \forall s, s' \in S \setminus T, \forall \alpha \in A(s), \\
\sum_{s' \in S} \mathcal{P}(s, \alpha, s') & = 1, \quad \forall s \in S \setminus T, \forall \alpha \in A(s), \\
\lambda & \geq k_s,
\end{aligned}
\]

\[
\begin{aligned}
k_s + p_s & \geq \sum_{s' \in T} (h_{cvx}(s, \alpha, s') + h_{aff}(s, \alpha, s')) , \\
\forall s \in S \setminus T, \forall \alpha \in A(s), \\
k_s & \geq 0, \quad \forall s \in S \setminus T,
\end{aligned}
\]

where \( \tau > 0 \) is a fixed penalty parameter. This convexified DC problem is, in fact, a convex QCQP. We use the constraint \( (24) \), which is the same as constraint \( (5) \), to ensure that we compute a solution at each iteration to minimize the violation of the convexified constraints, and guide the methods to a feasible solution with respect to the specification \( \varphi \). We show an example of the convexified problem in Example \( 3 \).

**Example 3.** Recall the pMC in Fig. \( 2 \) and the DC problem from Example \( 2 \). We introduce the penalty variables \( k_s \), and assume a fixed \( \tau \). We convexify around \( \hat{v} \), and the probability values of each state, \( \hat{p}_{s_0}, \hat{p}_{s_1}, \hat{p}_{s_2} \). The resulting convex problem with \( d = 0.5 \) is given by

**Proposition 2.** A feasible solution to the nonconvex QCQP in \( (1) - (6) \) is also feasible to the convexified QCQP in \( (1) - (5) \) together with \( (18) \).

**Proof.** For any concave function \( f \), its first-order approximation is a global over-estimator, i.e., for \( v, v' \in V \):
\[
f(v) \leq f(v') + \nabla f(v')^\top (v' - v),
\]
leading to \( h_{cvx}(s, \alpha, s') \leq h_{aff}(s, \alpha, s') \). Therefore, by construction, any feasible solution that satisfies \( (6) \) also satisfies \( (18) \). The claim follows, as all other constraints coincide. \( \square \)

The strictly set of constraints is often too strict and may not have any feasible solution. We add a penalty variable \( k_s \) for all \( s \in S \setminus T \) to all convexified constraints, which guarantees that the DC problem is feasible. These variables allow us to measure the “amount” of infeasibility. The larger the assigned value for a penalty variable, the larger the violation. We then seek to minimize the violation of the original DC constraints by minimizing the sum of the penalty variables. The resulting convexified problem with the penalty variables is given by

**Theorem 1.** A feasible solution of the convex DC problem in \( (20) - (26) \)

\[
\begin{aligned}
\text{minimize} & \quad p_s + \tau \sum_{s \in S \setminus T} k_s \\
\text{subject to} & \quad p_s = 1, \quad \forall s \in T,
\end{aligned}
\]

\[
\begin{aligned}
\mathcal{P}(s, \alpha, s') & \geq \varepsilon_{\text{graph}}, \quad \forall s, s' \in S \setminus T, \forall \alpha \in A(s), \\
\sum_{s' \in S} \mathcal{P}(s, \alpha, s') & = 1, \quad \forall s \in S \setminus T, \forall \alpha \in A(s), \\
\lambda & \geq k_s,
\end{aligned}
\]

\[
\begin{aligned}
k_s + p_s & \geq \sum_{s' \in T} (h_{cvx}(s, \alpha, s') + h_{aff}(s, \alpha, s')) , \\
\forall s \in S \setminus T, \forall \alpha \in A(s), \\
k_s & \geq 0, \quad \forall s \in S \setminus T,
\end{aligned}
\]

where \( \tau > 0 \) is a fixed penalty parameter. This convexified DC problem is, in fact, a convex QCQP. We use the constraint \( (24) \), which is the same as constraint \( (5) \), to ensure that we compute a solution at each iteration to minimize the violation of the convexified constraints, and guide the methods to a feasible solution with respect to the specification \( \varphi \). We show an example of the convexified problem in Example \( 3 \).
With the given condition, the constraints in (20) – (26) reduce we find a feasible solution, or it converges. The procedure we update CCP with the model checking results and convexify (2) Often, all instantiations with admissible parameter values yield well-defined MDPs. We verify this property via an easy time constructing the encoding, and the solver reuses previous functions around the current (not feasible) solution and solve the resulting convex problem. We repeat this procedure until the procedure converges or we find a feasible solution.

Proof. Let \( v \) be a feasible solution as in the theorem statement. With the given condition, the constraints in (20) – (26) reduce to the constraints in (11) – (15) and (18). From Proposition 2 we conclude that the feasible solution is also feasible to the QCQP in (1) – (6). Finally, using Proposition 1 we conclude that the feasible solution is also a solution to Problem 1. We provide another proof in the Appendix A.

We update the penalty parameter \( \tau \) by \( \mu + \tau \) for a \( \mu > 0 \), if any of the penalty variables are positive. The penalty parameter is updated until an upper limit for \( \tau_{\text{max}} \) is reached to avoid numerical problems. Then, we convexify the \( h_{\text{cev}} \) functions around the current (not feasible) solution and solve the resulting convex problem. We repeat this procedure until we find a feasible solution, or it converges. The procedure may be restarted with a different initial value if the procedure converges to an infeasible solution.

B. Efficiency Improvements in CCP

In this section, we discuss our second method, which is a sequential convex programming (SCP) approach with trust region constraints [23] – [25]. Similar to the penalty CCP method, the SCP method computes a locally optimal solution by iteratively approximating a nonconvex optimization problem. The approximate problem is a linear program (LP), instead of a QCQP. This comes at the cost of a generally coarser approximation.

More precisely, the main differences between SCP and CCP are, (1) the resulting convex problem is an LP, and solving is thus generally faster than solving a similar-sized QCQP, and (2) the convexified functions in the constraints are no longer upper bounds of the original functions. The approximation may generate optimal solutions in the convexified problem that are infeasible in the original problem. Therefore, we include trust regions and an additional model checking step similar to the CCP method to ensure that the new solution improves the objective. The trust regions ensure that the resulting LP accurately approximates the nonconvex QCQP. If the new solution indeed improves the objective, we accept and update the assignment of the variables and enlarge the trust region. Otherwise, we contract the trust region, and do not update the assignment of the variables.

A. Constructing the affine approximation

We now detail how we linearize the bilinear functions in the constraints in (6), similar to Section [1]. Recall that this constraint appears as

\[
p_s \geq \sum_{s' \in S} P(s, \alpha, s') \cdot p_{s'} \quad \forall s \in S \setminus T, \forall \alpha \in A(s).
\]
Similar to the previous section, consider the bilinear function in the above constraint
\[ h(s, \alpha, s') = \mathcal{P}(s, \alpha, s') \cdot p_{s'} \]  
and let
\[ \mathcal{P}(s, \alpha, s') = 2d \cdot y + c, \] 
where \( y \) is the parameter variable, \( z \) is the probability, and \( c,d \) are constants, similar to the previous section. We then convexify \( h(s, \alpha, s') \) as
\[ h_{\alpha}(s, \alpha, s') := 2d \cdot (\langle \hat{y} + \hat{z} \rangle + \hat{y} \cdot (z - \hat{z}) + \hat{z} \cdot (y - \hat{y})) + c \cdot z, \]
where \( \langle \hat{y}, \hat{z} \rangle \) are any assignments to \( y \) and \( z \). Note that the function \( h_{\alpha}(s, \alpha, s') \) is affine in the parameter variable \( y \) and the probability variable \( z \). After the linearization, the set of constraints (32) is replaced by the convex constraints
\[ \forall s \in S \setminus T, \forall \alpha \in A(s), p_s \geq \sum_{s' \in S} h_{\alpha}(s, \alpha, s'), \]

**Remark 3.** If the pMDP is not affine, i.e., \( \mathcal{P}(s, \alpha, s') \) is not affine in \( V \) for every \( s, s' \in S \) and \( \alpha \in A(s) \), then \( h(s, \alpha, s') \) will not be a quadratic function in \( V \) and probability variables \( p_{s'} \). In this case, we can compute \( h_{\alpha}(s, \alpha, s') \) by computing a first order approximation with respect to \( V \) and \( p_{s'} \) around the previous assignment.

Similar to the CCP method, we use penalty variables \( k_s \) for all \( s \in S \setminus T \) to all linearized constraints, ensuring that they are always feasible. However, the functions in these constraints do not over-approximate the functions in the original constraints. Therefore, a feasible solution to the linearized problem is potentially infeasible to the parameter synthesis problem. To make sure that the linearized problem accurately approximates the parameter synthesis problem, we use a trust region constraint around the previous parameter instantiations. The resulting LP is:

\[
\begin{align*}
\text{minimize} & \quad p_{s_i} + \tau \sum_{s \in S \setminus T} k_s \\
\text{subject to} & \quad p_s = 1, \quad \forall s \in T, \\
\mathcal{P}(s, \alpha, s') & \geq \varepsilon_{\text{graph}}, \quad \forall s, s' \in S \setminus T, \forall \alpha \in A(s), \\
\sum_{s' \in S} \mathcal{P}(s, \alpha, s') & = 1, \quad \forall s \in S, \forall \alpha \in A(s), \quad (33) \\
\lambda & \geq p_{s_i}, \quad (34) \\
k_s + p_s & \geq \sum_{s' \in T} h_{\alpha}(s, \alpha, s'), \quad \forall s \in S \setminus T, \forall \alpha \in A(s), \\
k_s & \geq 0, \quad \forall s \in S \setminus T. \quad (35) \\
\hat{p}_{s_i} / \delta' & \leq p_s \leq \hat{p}_{s_i} \cdot \delta', \quad \forall s \in S \setminus T, \quad (37) \\
\hat{p}_{s_i} / \delta' & \leq \hat{p}_{s_i} \cdot \delta', \quad (38) \\
\end{align*}
\]

where \( \tau > 0 \) is as defined before, and \( \mathcal{P}(s, \alpha, s') \) and \( \hat{p}_s \) denotes the previous assignment for the parameter and probability variables. The constraints (37–38) are the trust region constraints. \( \delta > 0 \) is the size of the trust region, and \( \delta' = \delta + 1 \).

Unlike the convexification in the CCP method, the linearization step in the SCP does not over approximate functions in the constraint (35). Therefore, we cannot provide the soundness guarantees for the solutions to the above LP as we did for the CCP method in Theorem 1. On the other hand, the trust region constraints ensure that the linearization is accurate by restricting the set of feasible solutions around the previous solution. Recall that we integrate the model-checking step into the SCP method to obtain soundness for the SCP method. We also show in the examples by a comparison with another SCP method that integrating the model-checking step significantly improves the performance. Similar to the CCP method, we use the constraint (34) to minimize the violation of the convexified constraints. We demonstrate the linearization in Example 4.

**Example 4.** Recall the pMC in Fig. 2 and the QCQP from Example 2. After linearizing around an assignment for \( \hat{v}, \hat{p}_{s_0}, \hat{p}_{s_1}, \) and \( \hat{p}_{s_2} \), the resulting LP with a trust region radius \( \delta > 0 \) is given by

\[
\begin{align*}
\text{minimize} & \quad p_{s_0} + \tau \sum_{i=0}^{2} k_{s_i} \\
\text{subject to} & \quad p_{s_0} = 1, \lambda \geq p_{s_0}, \\
p_{s_0} + p_{s_1} & \geq \hat{v} \cdot \hat{p}_{s_1} + \hat{p}_{s_1} \cdot (v - \hat{v}) + \hat{v} \cdot (p_{s_1} - \hat{p}_{s_1}), \\
k_{s_1} + p_{s_1} & \geq p_{s_2} - \hat{v} \cdot \hat{p}_{s_2} - p_{s_2} \cdot (v - \hat{v}) - \hat{v} \cdot (p_{s_2} - \hat{p}_{s_2}), \quad (39) \\
k_{s_2} + p_{s_2} & \geq \hat{v} \cdot \hat{p}_{s_2} + \hat{p}_{s_2} \cdot (v - \hat{v}) + \hat{v} \cdot (p_{s_2} - \hat{p}_{s_2}), \quad (40) \\
k_{s_0} & \geq 0, k_{s_1} \geq 0, k_{s_2} \geq 0, \\
\hat{p}_{s_0} / \delta' & \geq p_{s_0} \geq \hat{p}_{s_0} \cdot \delta', \quad \hat{p}_{s_1} / \delta' \geq p_{s_1} \geq \hat{p}_{s_1} \cdot \delta', \quad (42) \\
\hat{p}_{s_2} / \delta' & \geq p_{s_2} \geq \hat{p}_{s_2} \cdot \delta' \cdot \hat{v} / \delta' \geq v \geq \hat{v} \cdot \delta'. \quad (43)
\end{align*}
\]

We detail our SCP method in Fig. 5. We initialize the method with a guess for the parameters \( \hat{v} \), for the probability variables \( \hat{p} \), and the trust region \( \delta > 0 \). Then, we solve the LP (30)–(37) that is linearized around \( \hat{v} \) and probability variables \( \hat{p} \).

After obtaining an instantiation to the parameters \( v \), we model check the instantiated MDP \( \mathcal{M}[v] \) to obtain the values of probability variables res(\( p \)) for the instantiation \( v \). If the instantiated MDP indeed satisfies the specification, we return the instantiation \( v \). Otherwise, we check whether the probability of reaching the target set \( \beta \) is larger than the previous best value \( \beta_\hat{v} \). If \( \beta \) is larger than \( \beta_\hat{v} \), we update the values for the probability and the parameter variables, and enlarge the trust region. Else, we reduce the size of the trust region, and resolve the problem that is linearized around \( \hat{v} \) and \( p \). This procedure is repeated until a parameter instantiation that satisfies the specification is found, or the value of \( \delta \) is smaller than \( \omega > 0 \). The intuition behind enlarging the trust region is as follows: If the instantiation to the parameters \( v \) increases the probability of reaching the target set \( \beta \) over the previous solution, then we conclude that the linearization is accurate. Consequently, the SCP method may take a larger step in the next iteration for faster convergence in practice.

For expected cost specifications, the resulting algorithm is similar, except, we accept the parameter instantiation if the
After each iteration, we model check the instantiated MDP variables and parameter variables satisfy the constraints in (6). In this section, we discuss the implementation details SCP. It also provides an accurate point for the next iteration as a model checking to ensure that the obtained probability Fig. 5 and the existing SCP methods. First, we perform large, and contract its radius.

Fig. 5: SCP with model checking in the loop. The NLP (1)–(6) is linearized around \( \hat{v}, \hat{p} \). Then, we solve the LP (30)–(37) an optimal solution to the parameter values, denoted by \( P \). After each iteration, we model check the instantiated MDP \( M[v] \) to determine whether the specification is satisfied. If the instantiated MDP satisfies the specifications, we return the parameter instantiation. Otherwise, we check whether the reachability probability, denoted by \( \beta \), is improved compared to the previous iteration, denoted by \( \hat{\beta} \). If the probability is improved, we accept this step, update the assignment for the parameters and the probability variables, and increase the size of the trust region \( \delta \) by \( \gamma \). Otherwise, we do not update the assignment, and decrease the size of the trust region.

expected cost is reduced compared to the previous iteration, and initialize \( \hat{\beta} \) with a large constant.

B. Efficiency Improvements in SCP

Similar to the previous section, we consider several efficiency improvements for the SCP method. We also apply the algorithmic improvements in Section IV-B about building the encoding to the SCP method.

1) Integrating Trust Regions and Model Checking with SCP: In this section, we discuss the implementation details for the procedure in Fig. 5 and discuss how we update trust regions in our implementation. In nonlinear optimization, trust region algorithms obtain the new iterate point by searching in a trust region of the current iterate point \([23]–[25]\). Trust region algorithms check if the approximated problem accurately represents the nonlinear optimization problem, and adapt the size of the trust region in each iteration. In our algorithm, we check if the convexification is accurate by checking whether the obtained parameter instantiation improves the objective value. If we improve the objective value, i.e., the probability of satisfying the specification, we will enlarge the trust region. Otherwise, we conclude that the size of the trust region is too large, and contract its radius.

There are two critical differences between the method in Fig. 5 and the existing SCP methods. First, we perform a model checking to ensure that the obtained probability variables and parameter variables satisfy the constraints in (6). It also provides an accurate point for the next iteration as the probability and parameter variables are consistent on the underlying pMDP.

Second, we can compute the actual objective value by model checking and determine to accept the iterate instead of checking the feasibility of the solution for each constraint and the change of an approximate objective function [24], [47]. Computing the actual objective by model checking allows us to take larger steps in each iteration and significantly improve the algorithm’s performance in practice.

VI. CONVERGENCE PROPERTIES OF THE PROPOSED METHODS

In this section, we discuss the convergence properties of the proposed methods. We also discuss the conditions on the constrained and penalty problem having the same set of locally optimal solutions.

a) Convergence properties of CCP: The convergence of CCP is discussed in [19], [48]. If CCP is started with a feasible point, then all of the solutions in each iteration will be feasible. Additionally, the objective will decrease monotonically and will converge. Reference [49] showed that CCP converges to a solution that satisfies the full KKT conditions, which are necessary conditions for a solution to be locally optimal. The convergence rate of CCP is established in [50] if the feasible set is convex, which is not the case for the parameter synthesis problem. Therefore, we are not aware of any convergence rate results for the CCP method in the parameter synthesis problem.

b) Convergence properties of SCP: The convergence rate statements of the trust region methods [23]–[25] and other SCP methods [47], [51] rely on regularization assumptions such as Lipschitz continuity for the gradients of the functions in the objective and the constraints. The QCQP in (1)–(6) satisfies the regularization assumption, as all the functions in the objective and constraints are quadratic. The critical step in the convergence proofs is to show the existence of a small enough trust region to approximate the nonlinear optimization problem and sufficiently decrease the objective function in each iteration. Recent linear and superlinear convergence results [24], [47] rely on results from dynamical systems [52] to show linear convergence of iterative methods with a real-analytic objective and constraint functions. We state a convergence property of the SCP method described by Fig. 5.

Proposition 3. The sequence \((v, p)\) generated by SCP method in Fig. 5 converges to a limit point.

Proof. Let \(f^k(v, p)\) be the reachability probability of the instantiated MDP \(M[v]\) at the \(k\)–th iteration. It is clear that this sequence is bounded below by 0, and above by 1, and is monotonically increasing as we accept the parameter solutions only if the reachability probability is improved. Therefore, due to Lebesgue’s monotone convergence theorem [53, 11.28], this sequence is convergent for reachability probabilities. For expected cost specifications, the sequence is monotonically decreasing, and bounded above by the instantiated cost in the first iteration, and below by 0; therefore it is convergent.

The model checking step in Fig. 5 involves a nonsmooth mapping from the parameter instantiations to probability
variables due to maximizing the reachability probability at each action and makes the convergence rate analysis difficult. For completeness, we show the convergence rate of a similar iterative method without model checking in Appendix B. The local convergence rate of the SCP method is linear in Appendix B. However, in our numerical examples, we observed that the SCP method in Appendix B does not perform well in practice due to requiring a small enough step size to ensure convergence. On the other hand, we can check if we improved the objective directly in Step 2 of Algorithm 1, allowing us to take larger step sizes and reduce the required number of iterations compared to the existing methods.

c) Justification of penalty functions: In this section, we discuss the merit of using penalty functions for the CCP and SCP methods. Using penalty functions \[ 54-56 \] ensures that each convexified problem is feasible. A penalty problem \[ 24 \] puts a penalty for violating each constraint in a constrained optimization problem instead of enforcing them as hard constraints. A penalty function is exact if a constrained optimization problem and the penalty problem have the same optimality conditions.

For simplicity, we denote our variable pair \((v, p)\) as \(x\) in this section. Following the definitions in \[ 46 \] Section 5.1.1, we denote objective in \[ 1 \] as \(f_0(x)\), the inequality constraints in \[ 3, 5-6 \] as
\[
   f_i(x) \leq 0, \quad i = 1, \ldots, m,
\]
where \(m\) is the number of inequality constraints, and the equality constraints in \[ 2 \] and \[ 4 \] as
\[
   g_i(x) = 0, \quad i = 1, \ldots, p,
\]
where \(p\) is the number of inequality constraints. We now state the stationary part of the first order necessary conditions for a point \(\bar{x}\) to be a locally optimal solution.

**Theorem 2** (Karush-Kuhn-Tucker conditions, \[ 46 \]). If \(\bar{x}\) is a locally optimal solution to the NLP in \[ 1-6 \], then there exist optimal Lagrange multipliers \(\mu^*_i \geq 0, i = 1, \ldots, m\) and \(\nu^*_i, i = 1, \ldots, p\) that satisfies the following
\[
   \nabla f_o(\bar{x}) + \sum_{i=1}^{m} \mu^*_i \nabla f_i(\bar{x}) + \sum_{i=1}^{p} \nu^*_i \nabla g_i(\bar{x}) = 0,
\]
under an appropriate constraint qualifier condition.

**Proof.** We provide the proof for satisfying the linear independence constraint qualification in Appendix B. \(\square\)

For the penalty problem, let the objective be
\[
   f_0(x) + \sum_{i=1}^{m} \mu_i \max(0, f_i(x)) + \sum_{i=1}^{p} \nu_i |g_i(x)|,
\]
where \(\mu_i, i = 1, \ldots, m\) and \(\nu_i, i = 1, \ldots, p\) are the penalty weights and the penalty problem has no constraints. We now state the *exactness* condition, which states the necessary and sufficient condition on the original QCQP and the penalty problem having the same set of locally optimal solutions.

**Theorem 3** (Exactness conditions, \[ 56 \]). If \(\bar{x}\) satisfies the condition in Theorem 2 with Lagrange multipliers \(\mu^*_i \geq 0, i = 1, \ldots, m\) and \(\nu^*_i, i = 1, \ldots, p\), and let the penalty weights \(\bar{\mu}_i, i = 1, \ldots, m\) and \(\bar{\nu}_i, i = 1, \ldots, p\) satisfy
\[
   \bar{\mu}_i > |\mu^*_i|, \quad i = 1, \ldots, m,
\]
\[
   \bar{\nu}_i > |\nu^*_i|, \quad i = 1, \ldots, p,
\]
then \(\bar{x}\) is also a locally optimal solution for the penalty problem. The converse result also holds.

Theorem 3 guarantees the existence of penalty weights that preserve the set of locally optimal solutions. In practice, we select a large enough \(\tau\) in SCP and CCP to minimize the violations of the constraints.

**VII. Numerical Examples**

We evaluate our approaches on benchmark examples that are subject to either reachability or expected cost specifications. We implement the proposed CCP and SCP methods with the discussed efficiency improvements from Section IV-B and Section V-B in the parameter synthesis framework PROPhESY \[ 26 \]. We use the probabilistic model checker Storm \[ 29 \] to extract an explicit representation of a 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 and LPs, we use Gurobi 9.0 \[ 20 \], configured for numerical stability.

a) Environment: We evaluate the SCP and CCP methods with 16 3.1 GHz cores, a 32 GB memory limit, and an 1200 seconds time limit (TO). The objective is to find a feasible parameter valuation for pMCs and pMDPs with specified thresholds on probabilities or costs in the specifications, as in Problem 1. We ask for a well-defined valuation of the parameters, with \(\varepsilon_{\text{graph}} = 10^{-6}\). We run all the approaches with the same configuration of Storm. For pMCs, we enable weak bisimulation, which reduces the number of states of the pMC in all examples. We do not use bisimulation for pMDPs. We provide all the codes and log files for the numerical examples in https://github.com/mcubuktepe/pMDPsynthesis.

b) Baselines: We compare the runtimes with three baselines: a particle swarm optimization (PSO) implementation, solving the nonconvex QCQP directly with Gurobi \[ 20 \], indicated by “GRB” in the following results, and an SCP algorithm S\(^l\)QP, indicated by “S\(^l\)QP” in the following results, from \[ 47 \], with convergence guarantees to KKT points. PSO is a heuristic sampling approach that searches the parameter space, inspired by \[ 16 \]. For each valuation, PSO performs model checking without rebuilding the model, instead it adapts the matrix from previous valuations. As PSO is a randomized procedure, we run it five times 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 nontrivial task. For pMCs that originate from POMDPs, we also compare with a POMDP solver PRISM-POMDP \[ 57 \] by using the relation between pMCs and POMDPs \[ 12 \]. We note that PRISM-POMDP computes a lower and an upper bound on the probability of satisfying a temporal logic specification by discretizing the uncountable belief space. Our approach computes a finite-memory policy
that maximizes or minimizes the probability of satisfying a temporal logic specification.

c) Tuning constants: For the CCP method, 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, we observed that a larger \( \tau \) increases numerical stability. For CCP, we pick \( \mu = \max_{s \in S^T} \hat{p}_s \). We update \( \tau \) by adding \( \mu \) after each iteration. Empirically, increasing \( \tau \) with larger 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., \([19]\), \( \tau \) is multiplied by \( \mu \) after each iteration.

For the SCP method, we use a constant \( \tau = 10^4 \) for the reachability and expected cost specifications. The initial trust-region value is \( \delta = 2 \) and \( \gamma = 1.5 \), where we adjust the size of the trust region after each iteration. Finally, we use \( \omega = 10^{-4} \) and terminate the procedure if the trust region is smaller than \( \omega \).

Our initial guess \( \hat{v} \) is 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 \( \mathbb{P}_{\leq \lambda}(\hat{v}T) \) to initialize the probability variables, and analogously for expected cost specifications.

A. Case Study: Satellite Collision Avoidance

In this example, we consider a satellite collision avoidance problem \([58], [59]\). We formulate the spacecraft dynamics in Hill’s reference frame, with the origin at a specified location on a circular spacecraft orbit \([60], [61]\). We define the three axes of the motion as in \([60], [61]\). We consider circular orbits in this example, as it yields time-invariant satellite dynamics with natural motion trajectories (NMTs). The nominal orbital radius for the satellite dynamics is 7728 km.

The spacecraft dynamics include periodic behavior around the earth, and a set of natural motion trajectories (NMTs) \([58], [62]\). In this example, we maximize the probability of successfully finishing a cycle in orbit by avoiding a collision with other objects in space. Given a set of NMTs, we form an undirected graph with one node corresponding to each closed NMT and time index \( t \), and two nodes share an edge if the distance between them is 250km. If two nodes share an edge, then it is possible to perform a transfer between two nodes. The state variables of the POMDP are (1) \( n \in \{1, \ldots, 36\} \) depicting the current NMT of the satellite, and (2) \( t \in \{1, \ldots, I\} \), depicting the current time index for a fixed NMT. We use different values of \( I \) in the examples. The satellite can follow its current NMT with no fuel usage. The satellite can get an observation every 10 steps, resulting in 720 different observations. In each NMT and time index, the satellite can either stay in the current orbit, incrementing the time index by 1 or can transfer into a different nearby NMT if the underlying two nodes share an edge. In our model, the probability of successfully switching to another NMT is 0.9, and the satellite will transfer to a different nearby orbit with a probability of 0.1.

The specification is \( \mathbb{P}_{\geq \lambda}(\hat{v}T) \), which asserts that the probability of finishing a cycle in orbit without colliding another object is greater than a threshold \( \lambda \). The objective is to compute a switching strategy that satisfies the specification with a probability that is greater than \( \lambda \), ensuring that the satellite does not collide with another object with high probability. For example, in the satellite collision avoidance problem, the strategy can determine at which point in space to perform a transfer between two orbits. This strategy and the corresponding controller for the transfer can then be implemented in the underlying system to perform collision avoidance.

In our first model, denoted by “Satellite,” we discretize the trajectory into \( I = 200 \) time indices. To further show the scalability of our SCP method, we also compute a finite-memory policy with \( I = 200 \), denoted by “Satellite-fm.” We also discretize the trajectory into \( I = 8000 \) time indices with 36 NMTs, allowing a control input to be given to the satellite around every 0.2 second. We also consider the effect of having different levels of sensor quality, where the satellite can get an observation in every 40, 80, and 200 time steps. The resulting POMDPs have 288000 states and 7200, 3600, and 1440 observations, denoted by “Satellite-7200,” “Satellite-3600,” and “Satellite-1440,” respectively. The resulting pMCs have 10042, 5337, and 2448 parameters, respectively.

The detailed results for the models are shown in Table I. The first two columns refer to the benchmark instance, the next column to the specification. We give the number of states (States), transitions (Trans.), and parameters (Par.). We then give the minimum (tmin), the maximum (tmax) and average (tavg) runtime (in seconds) for PSO with different seeds, and the runtime obtained using CCP, SCP and \( \text{St}^1 \text{QP} \) (t). We also give the number of CCP, SCP and \( \text{St}^1 \text{QP} \) iterations (iter).

We show the trajectories from a memoryless policy in Fig. 6a and from a policy with finite memory in Fig. 6b. The average length of the trajectory with the memoryless policy is 402, which is twice the length of the trajectory from the finite-memory policy, given by 215. The expected cost is due to switching to a different NMT, for the finite-memory policy 619 Newton \cdot \text{seconds}. In contrast, it is 1249 Newton \cdot \text{seconds} for the memoryless policy. We give detailed results of the computation in Table II. We observe that after 6 minutes of computation, using larger pMCs with finite-memory policy yields superior policies in the probability of satisfying the specification with \( I = 200 \).

We obtained a solution with a collision probability of less than \( 10^{-3} \) on models with \( I = 8000 \) and using memoryless policies, showing the benefit of increasing the number of time indices compared to the previous examples. The computation took 370, 336, and 386 seconds for the models with 7200, 3600, and 1440 observations, respectively, showing that our approach does not scale exponentially with the number of observations and parameters.

B. Results for Further Models

a) Benchmarks: We include the standard pMC benchmarks from the PARAM website \([\text{http://www.avacs.org/tools/param/}]\) which contain two parameters. We furthermore have a rich selection of strategy synthesis problems obtained from partially observable MDPs
Fig. 6: The obtained trajectories are shown in red that successfully finish an orbit around the origin. We highlight the initial location by a big circle. We highlight the NMTs used during the orbit and depict the objects to avoid by black spheres.

TABLE I: Results for the satellite collision avoidance example.

| Set       | Problem | Info              | States | Info Trans. | Par. | PSO max | PSO tavg | SCP t iter | CCP t iter | SlQP t iter | GRB t |
|-----------|---------|-------------------|--------|-------------|------|---------|----------|------------|------------|-------------|-------|
| Satellite | P ≥ 0.5 |卫星 | 6265     | 17436 | 231 | –       | –        | 8          | 6          | 1142 421    | 977 1061 | TO    |
| Satellite | P ≥ 0.9 |卫星 | 6265     | 17436 | 231 | –       | –        | 14         | 12         | TO          | TO   | TO |
| Satellite | P ≥ 0.95| 卫星 | 31325    | 156924 | 2555| –       | –        | 146        | 10         | MO          | TO   | TO |
| Satellite-fm | P ≥ 0.95| 卫星 | 31325    | 156924 | 2555| –       | –        | 239        | 18         | MO          | TO   | TO |
| Satellite-1440 | P ≥ 0.995| 卫星 | 217561  | 615433 | 2248| –       | –        | 386        | 4          | MO          | TO   | TO |
| Satellite-3600 | P ≥ 0.995| 卫星 | 217561  | 615433 | 5337| –       | –        | 536        | 4          | MO          | TO   | TO |
| Satellite-7200 | P ≥ 0.995| 卫星 | 217561  | 615433 | 10042| –       | –        | 370        | 4          | MO          | TO   | TO |

(POMDPs) [12]. The first example is a maze setting (Maze), introduced in [63]. The objective is to reach a target location in the minimal expected time in a maze. The initial state is randomized, and the robot can only get an observation if it hits a wall. We consider an example with five memory nodes.

The second example from [12] schedules wireless traffic over a network channel (Netw), where at each time a scheduler generates a new packet for each user at each time [64]. The scheduler has partial observability of the quality of the channels. The main trade-off is to probabilistically schedule the traffic to the user with the best or worst perceived channel condition.

Finally, in the drone example (Drone), the objective is to compute a policy to arrive at a target location while avoiding an intruder. The intruder is only visible within a radius, and the controller has to take account of the partial information. We consider a memoryless and a finite-memory policy with five memory for observations (Drone-fm) to show the trade-offs between the computation time and the reachability probability. The pMDP benchmarks originate from the PARAM website, or as parametric variants to existing PRISM case studies.

b) Results: Similar to Table I, Table II contains an overview of the results for pMCs. Table III additionally contains the number of actions (Actions) for pMDPs.

For the Maze example, PRISM-POMDP found a solution that satisfies the 6 threshold in 1.1 seconds, performing better than all methods considered. However, PRISM-POMDP ran out of memory in the other POMDPs. We observe the effect of including memory clearly in the Drone benchmark. In the examples with memoryless strategies, both SCP and CCP methods can find solutions for lower thresholds much faster than with finite-memory strategies. With a threshold of 0.99, PSO, CCP and SlQP ran out of time before finding a solution. On the other hand, only SCP can find a solution for the 0.99 threshold with a finite-memory strategy before the time-out.

Specifically, in most of the considered examples, SCP outperforms other methods, especially on the models with a large number of parameters. The empirical results demonstrate that in the examples where SlQP successfully finds a solution, it requires significantly more steps than SCP. The reason is, the linearization in SlQP is only accurate on a very small region around the previous solution. The algorithm prevents it from taking a large step to ensure that the linearization is accurate in all iterations. On the other hand, by incorporating model checking results, SCP can take larger steps for faster convergence while ensuring the soundness of the solutions using the model checking results. On a few instances, such as Crowds and Network, CCP can outperform the other methods.

Similarly, on instances with 2 parameters, PSO is the best method in terms of the performance. Overall, we observe that the SCP method is significantly faster than PSO, SlQP and
TABLE II: pMC benchmark results

| Problem | Set | Spec  | States | Trans. | Par. | tmin | tmax | tavg | SCP  | CCP  | S1/4P | GRB |
|---------|-----|-------|--------|--------|------|------|------|------|------|------|-------|-----|
| Bsp     | F ≤ 0.1 | 324 | 452 | 2 | 0 | 0 | 0 | 0 | 1 | 0 | 4 | 0 | 19 | 0 |
| Bsp     | F ≤ 0.1 | 20999 | 29703 | 2 | 2 | 4 | 2 | 6 | 1 | 180 | 87 | TO | – | 5 |
| Crowds  | F ≤ 0.1 | 80 | 120 | 2 | 1 | 1 | 1 | 2 | 2 | 2 | 3 | 2 | 2 | 2 |
| Nand    | F ≤ 0.05 | 5447 | 7374 | 2 | 0 | 0 | 0 | 1 | 1 | 1 | 1 | 1 | 1 | 14 |
| Maze    | E ≤ 14 | 1303 | 2658 | 590 | 125 | 201 | 167 | 3 | 56 | 27 | 430 | TO |
| Maze    | E ≤ 10 | 1303 | 2658 | 590 | – | – | TO | 2 | 5 | 36 | 54 | 378 | 1047 | TO |
| Maze    | E ≤ 6  | 1303 | 2658 | 590 | – | – | TO | 3 | 9 | 43 | 77 | TO | – | TO |
| Maze    | E ≤ 5.3 | 1303 | 2658 | 590 | – | – | TO | 7 | 1 | 152 | 192 | TO | – | TO |
| Netw    | E ≤ 10 | 5040 | 13327 | 596 | 213 | 273 | 243 | 3 | 1 | 3 | 1 | TO | – | TO |
| Netw    | E ≤ 5  | 5040 | 13327 | 596 | – | – | TO | 3 | 2 | 6 | 3 | TO | – | TO |
| Netw    | E ≤ 3.3 | 5040 | 13327 | 596 | – | – | TO | 20 | 27 | 193 | 97 | TO | – | TO |
| Drone   | F ≥ 0.9 | 4179 | 9414 | 1053 | – | – | TO | 3 | 3 | 23 | 13 | 96 | 207 | TO |
| Drone   | F ≥ 0.95 | 4179 | 9414 | 1053 | – | – | TO | 3 | 4 | 177 | 99 | 995 | 2212 | TO |
| Drone-fm| F ≥ 0.9 | 32403 | 70099 | 12286 | – | – | TO | 93 | 3 | 1179 | 13 | TO | – | TO |
| Drone-fm| F ≥ 0.95 | 32403 | 70099 | 12286 | – | – | TO | 96 | 4 | 10 | – | TO | – | TO |
| Drone-fm| F ≥ 0.99 | 32403 | 70099 | 12286 | – | – | TO | 160 | 20 | 10 | – | TO | – | TO |

Fig. 7: The obtained expected cost of SCP and CCP versus the number of iterations for the Network example with a threshold of 3.3. SCP can find a feasible solution with fewer iterations than CCP on this example.

CPC on benchmarks with many parameters.

We show the convergence of the CCP and SCP in Figures 7-9 on three benchmarks with different thresholds. We show the expected cost in each iteration for both methods until they find a feasible solution. In all of these examples, SCP requires fewer iterations compared to CCP to compute a feasible parameter instantiation. The obtained expected cost for SCP is also less than CCP after each iteration. We observe that the objective does not decrease for SCP for several iterations in each figure, due to rejecting the parameter instantiation and contracting the trust region for these iterations. However, SCP can minimize the expected cost after sufficiently reducing the size of the trust region to approximate the nonconvex problem accurately.

Finally, solving the nonconvex QCQP using Gurobi yields a similar runtime with SCP and CCP in pMCs with two parameters. However, Gurobi failed to find a solution for all pMCs obtained from POMDPs before the time limit.

For the pMDP benchmarks, see Table III, we observe a similar pattern, where the SCP method is significantly faster than PSO and CCP in all but one benchmark. For the CSMA benchmark, the SCP method fails to find a feasible solution for the thresholds 68.7 and 68.4, whereas CCP can find a feasible solution in seconds. In this benchmark, it takes 916 iterations for SCP to find a solution that induces a cost less than 68.7, and SCP converges to a locally optimal solution that induces a cost higher than 68.4, therefore not satisfying the specification. The benchmarks CSMA and WLAN are currently not supported by PSO due to the nonrectangular well-defined parameter space. Similar to the pMC benchmarks, Gurobi was able to solve a pMDP with 2 parameters at a similar time than CCP and SCP. However, Gurobi failed to find a solution for other pMDPs within the time limit.

c) Effect of values of hyperparameters \( \delta \) and \( \gamma \): In Table IV, we report the running time and the number of iterations for the SCP method on finding a feasible solution with threshold 0.95 for different values of the hyperparameters \( \delta \) and \( \gamma \). In all of these examples, the running time and number of iterations for the SCP method vary for different values. For example, selecting \( \delta = 1.0 \) and \( \gamma = 1.5 \) gives the best performance, and the running time and the number of iterations are highest with \( \delta = 1.5 \) and \( \gamma = 2.0 \). However, for all values of hyperparameters, SCP was able to compute a solution within a few minutes.

d) Effect of integrating model checking for CCP and SCP: Discarding the model checking results in our CCP implementation always yields time-outs, even for the relatively simple benchmark Maze with threshold 10, which is solved with usage of model checking results within a minute. Here, using model checking results thus yields a speed-up by a factor of at least 60. For the Netw example, not using model checking increases the runtime by ten on average. For the Drone examples that CCP solves before the timeout, we observe that CCP always exceeds time if we do not include the model checking results.

Fig. 8: The obtained expected cost of SCP and CCP versus the number of iterations for the Maze example with a threshold of 5.3. SCP can find a feasible solution with fewer iterations than CCP on this example.
For SCP, we observed that model checking reduces the runtime of the procedure significantly in all examples and guarantees the solution’s correctness. For the Drone examples, the size of the trust region becomes too small before reaching the required threshold without model checking, and SCP returns an infeasible solution for all thresholds. On the other hand, we can observe that SCP can find a feasible solution within seconds or minutes if we include model checking.

C. Discussion

The main results from the experiments is, a tuned variant of SCP improves the state-of-the-art. We observed that directly applying SCP methods does not yield a scalable method and may cause incorrect terminations. To solve the nonconvex QCQP that originates from a pMDP efficiently, we needed to update parts of the model between each iteration instead of rebuilding. Additionally, Gurobi can use the results from the previous iterations, which improves the overall method’s runtime. Using state reduction techniques from model checking significantly reduces the number of variables in each convex problem, which improves the runtime and the numerical stability of the procedure. The critical ingredient is to incorporate model checking, which reduces the number of iterations significantly, as shown by the comparison with the $S^1QP$ algorithm. It also allows a sound termination criterion, which cannot be realized by directly applying SCP methods as the linearized problem is not an over-approximation of the original QCQP.

These improvements to SCP yield a procedure that can handle much bigger problems than existing POMDP solvers. Our SCP method is also superior to sampling-based approaches in problems with many parameters. It also significantly outperforms CCP \([21]\) in almost all examples considered. 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.

VIII. CONCLUSION AND FUTURE WORK

We studied the applicability of convex optimization for parameter synthesis of parametric Markov decision processes (pMDPs). To solve the underlying nonconvex optimization problem efficiently, we proposed two methods that combine techniques from convex optimization and formal methods. The experiments showed that our methods significantly improve the state-of-the-art and can solve large-scale satellite collision avoidance problems. In the future, we will investigate how to handle the case if some of the parameters cannot be controlled and may be adversarial. We will also apply our techniques to solving uncertain partially observable MDPs with different uncertainty structures.

REFERENCES

[1] M. L. Puterman, Markov Decision Processes: Discrete Stochastic Dynamic Programming. John Wiley and Sons, 1994.
[2] X. Ding, S. L. Smith, C. Belta, and D. Rus, “Optimal Control of Markov Decision Processes with Linear Temporal Logic Constraints,” IEEE Transactions on Automatic Control, vol. 59, no. 5, pp. 1244–1257, 2014.
[3] C. von Essen and D. Giannakopoulou, “Analyzing the Next Generation Airborne Collision Avoidance System,” in TACAS, ser. LNCS, vol. 8413. Springer, 2014, pp. 620–635.
[4] P. Nilsson, S. Haesaert, R. Thakker, K. Otsu, C.-I. Vasile, A.-A. Aghamohammadi, R. M. Murray, and A. D. Ames, “Toward Specification-Guided Active Mars Exploration for Cooperative Robot Teams,” in RSS. Robotics: Science and Systems Foundation, 2018.
[5] C. Baier and J.-P. Katoen, Principles of Model Checking. MIT Press, 2008.
[6] C. Daws, “Symbolic and Parametric Model Checking of Discrete-Time Markov Chains,” in ICTAC, ser. LNCS, vol. 3407. Springer, 2004, pp. 280–294.
[7] R. Lanotte, A. Maggiolo-Schettini, and A. Troina, “Parametric Probabilistic Transition Systems for System Design and Analysis,” Formal Aspects Comput., vol. 19, no. 1, pp. 93–109, 2007.
which is equivalent to

\[ \alpha \]

Thus, \( \alpha \) from which \( s \) reach \( T \) in (20) – (26) is feasible with

\[ \text{Proof.} \]

\[ \text{A. McCallum, “Overcoming Incomplete Perception with Utile Distinction Memory,” in ICML, P. E. Utgoff, Ed. Morgan Kaufmann, 1993, pp. 190–196.} \]

\[ \text{B. Wie, “Space Vehicle Dynamics and Control.” American Institute of Aeronautics and Astronautics, 2008.} \]

\[ \text{C. Petersen, M. Baldwin, R. S. Erwin, and I. Kolmanovsky, “Safe Positively Invariant Sets for Spacecraft Obstacle Avoidance,”} \]

\[ \text{Journal of Guidance, Control, and Dynamics, vol. 38, no. 4, pp. 720–732, 2015.} \]

\[ \text{D. B. Wie, A. Weiss, C. Petersen, M. Baldwin, R. S. Erwin, and I. Kolmanovsky, “Mission Design and Trajectory Analysis for Inspection of a Host Spacecraft by a Microsatellite,” in 2007 IEEE Aerospace Conference. IEEE, 2007, pp. 1–23.} \]

\[ \text{E. McCallum, “Overcoming Incomplete Perception with Utile Distinction Memory,” in ICML, P. E. Utgoff, Ed. Morgan Kaufmann, 1993, pp. 190–196.} \]

\[ \text{F. Yang, S. Murugesan, and J. Zhang, “Real-Time Scheduling over Markovian Channels: When Partial Observability Meets Hard Deadlines,” in 2011 IEEE Global Telecommunications Conference-GLOBECOM. IEEE, 2011, pp. 1–5.} \]

**APPENDIX**

**A. Proof of Theorem 7**

Proof. Since the assignment of the convexified DC problem in (20) – (26) is feasible with

\[ \tau \sum_{\forall s \in S \setminus T} k_s = 0, \]

we know that the assignment is feasible for the QCQP in (1) – (6). We will show that for every \( s \in S \) we have \( \Pr(\mathcal{M}[v] \cap T, s) \leq p_s \), for any feasible assignment for the QCQP in (1) – (6).

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

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

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

The assumption that \( S_\prec \neq \emptyset \) implies \( x_{\text{max}} > 0 \). Let \( s \in S \) be such that \( x_s = x_{\text{max}} \). Therefore, \( s \in S_\prec \), and for all \( \alpha \in A(s) \), we have that

\[ p_s \geq \sum_{s' \in S} \mathcal{P}(s, \alpha, s') \cdot p_{s'}. \]

(39)

On the other hand, there exists an \( \alpha \in A(s) \) such that

\[ q_s = \sum_{s' \in S} \mathcal{P}(s, \alpha, s') \cdot q_{s'}. \]

(40)

Thus,

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

(41)

which is equivalent to

\[ x_s \leq \sum_{s' \in S} \mathcal{P}(s, \alpha, s') \cdot x_{s'}. \]

(42)

Since for all \( \alpha \in A(s) \), and \( s' \in S \), we have that \( \mathcal{P}(s, \alpha, s') \geq 0 \) and \( \sum_{s' \in S} \mathcal{P}(s, \alpha, s') = 1 \), using (41) we establish

\[ x_{\text{max}} = x_s \leq \sum_{s' \in S} \mathcal{P}(s, \alpha, s') \cdot x_{s'} \]

\[ \leq \sum_{s' \in S} \mathcal{P}(s, \alpha, s') \cdot x_{\text{max}} \]

(43)

\[ \leq x_{\text{max}} \sum_{s' \in S} \mathcal{P}(s, \alpha, s') = x_{\text{max}}. \]

(44)

Which implies that all the inequalities are equalities, meaning

\[ x_{\text{max}} = x_s = \sum_{s' \in S} \mathcal{P}(s, \alpha, s') \cdot x_{s'}. \]

(45)

The equation in (47) gives us that \( x_{s'} = x_{\text{max}} > 0 \) for every successor \( s' \) in \( \mathcal{M}[v] \). Since \( s \in S_{\text{max}} \) was chosen arbitrarily, for every state \( s \in S_{\text{max}} \), all successors of \( s \) are also in \( S_{\text{max}} \). As we established that \( S_\prec \cap T = \emptyset \), it is necessary that \( T \) is not reachable with positive probability from any \( s \in S_{\text{max}} \), which is a contradiction with the fact that from every state in \( S_\prec \), the set \( T \) is reachable. \( \square \)

**B. Proof of Theorem 2**

Proof. We show that the NLP in (1) – (6) satisfies linear independence constraint qualification, which states that the gradients of the active inequality and equality constraints are linearly independent at a locally optimal solution. Without loss of generality, we assume that the pMDP is simple, i.e., \( \mathcal{P}(s, \alpha, s') \in \{ v, 1 - v \mid v \in \mathbb{V} \cup R_{[0,1]} \} \).

First, the constraints (3) – (4) reduce to the constraints

\[ \varepsilon_{\text{graph}} \leq v \leq 1 - \varepsilon_{\text{graph}} \quad \forall v \in \mathbb{V}, \]

and the gradients of the active constraints would therefore be independent at any solution \( \forall v^* \in \mathbb{V} \).

Second, we now focus on the constraints (6). This constraint is affine in the probability variables \( p_s \) for \( s \in S \setminus T \) for a fixed value of the parameters in \( V \). Further, it is known that for any value of the parameters, the solution for the probability variables \( p_s \) for \( s \in S \setminus T \) instantiated MDP is unique [5, Theorem 10.19, Theorem 10.100], after a preprocessing step removing all states \( s \in S \) that reach a target state \( t \in T \) with probability 0 and 1 using Algorithm 46 in [5]. Note that this preprocessing step also removes the constraint (2).

Furthermore, the solution \( p^* \) for the probability variables can be obtained by solving the equation system \( p = \mathbf{P}p + \mathbf{b} \), where \( p = (p_s)_{s \in S \setminus T} \), i.e., the probability variables, the matrix \( \mathbf{P} \) contains the transition probabilities for the states in \( S \setminus T \) for the optimal deterministic strategy \( \sigma(s) \), i.e., \( \mathbf{P} = (\mathcal{P}(s, \sigma(s), s'))_{s, s' \in S \setminus T} \), and the vector \( \mathbf{b} = (b_s)_{s \in S \setminus T} \) contains the probabilities of reaching a state in the target set \( T \) from any state \( s \in S \setminus T \) within one step under the optimal action, i.e., \( b_s = \sum_{t \in T} \mathcal{P}(s, \sigma(s), t) \). This equation system correspond to the active inequality constraints in (6), which can be written as

\[ p^* = \sum_{s' \in S} \mathcal{P}(s, \sigma(s), s') \cdot p^*_{s'} \quad \forall s \in S \setminus T, \]
where \( \hat{P} \) denote the transition function for the instantiated pMDP, and \( p_s^* \) is the unique solution for the probability variable at state \( s \). Note that the solution for the probability variables is unique, and the above equation system is affine in the probability variables. Therefore, the gradients of the above active inequality constraints are linearly independent at their unique solution \( p_s^* \) for the probability variables \( p_s \).

Combining the two facts, we conclude that the NLP in (1)–(5) satisfies the linear independence constraint qualification for any locally optimal solution.

C. Convergence of SCP Methods for Parameter Synthesis

In this section, we show the convergence of a SCP method that is a variant of Fig. 5. Similar to Section VI-c, we denote our variable pair \((v, p)\) by \( x \) in this section. Let the objective in (1) be \( h_0(x) \), the inequality constraints in (5) be

\[
h_i(x) \leq 0, \quad i = 1, \ldots, m
\]

where \( m \) is the number of inequality constraints. As the other constraints in the QCQP (1)–(4) are convex, we compactly represent the constraints in (2)–(4) as

\[
x \in Q.
\]

Note that the convex set \( Q \) is compact, which is a required assumption for the convergence analysis of SCP methods.

Remark 4. We remark that we do not include the threshold constraint in (5), as SCP methods require an initial feasible point for convergence analysis. Such an initial feasible point can be obtained by a model checking step with any well-defined parameter instantiations without the threshold constraints.

We also note that the functions \( h_i(x) \) are \( C^2 \) with Lipschitz continuous gradients as they are quadratic. For each \( i = 0, 1, \ldots, m \), we denote by \( L_i > 0 \) the Lipschitz constants of \( \nabla h_i \).

At each iteration \( l \), we solve the following convex problem with variables \( x \) and \( k \):

\[
\begin{align*}
\text{minimize} \quad & h_0(x) + \beta_l k + \frac{\mu + \beta_l \mu'}{2} ||x_l - x||^2 \\
\text{subject to} \quad & h_i(x_l) + \nabla h_i(x_l)^\top (x - x_l) \leq k, \quad i = 1, \ldots, m, \\
& k \geq 0, \\
& x \in Q.
\end{align*}
\]

where \( \mu > L_0 \) and \( \mu' \geq \max_{i=1,\ldots,m} L_i, \beta_l > 0 \). We show the SCP method in Algorithm 1.

We now state the convergence result of Algorithm 1

Theorem 4. [47, Theorem 2.2, Theorem 2.3] The sequence \( \{x_l\} \) generated by Algorithm 1 converges to a feasible point \( x_\infty \) that satisfies the KKT conditions in Theorem 2 for the QCQP in (1)–(4) and (6). Furthermore, the convergence rate is in the form

\[
\begin{align*}
||x_k - x_\infty||_2 & = O(q^k), \quad \text{with} \quad q \in (0, 1), \\
||x_k - x_\infty||_2 & = O(1/k^\xi), \quad \text{with} \quad \xi > 0.
\end{align*}
\]

Murat Cubuktepe joined the Department of Aerospace Engineering at the University of Texas at Austin as a Ph.D. student in Fall 2015. He received his B.S degree in Mechanical Engineering from Bogazici University in 2015. His research is on the theoretical and algorithmic aspects of the design and verification of autonomous systems in the intersection of formal methods, convex optimization, and artificial intelligence.

Nils Jansen is an assistant professor with the Institute for Computing and Information Science (iCIS) at the Radboud University, Nijmegen, the Netherlands. He received his Ph.D. in computer science with distinction from RWTH Aachen University, Germany, in 2015. Prior to Radboud University, he was a postdoctoral researcher and research associate with the Institute for Computational Engineering and Sciences at the University of Texas at Austin. His current research focuses on formal reasoning about safety aspects in machine learning and robotics. At the heart is the development of concepts inspired from formal methods to reason about uncertainty and partial observability.

Sebastian Junges is a postdoctoral researcher at the University of California, Berkeley. In 2020, he received his Ph.D degree with distinction from RWTH Aachen University, Germany. His research focuses on model-based analysis of controllers for uncertain environments, either applied for runtime assurance or as part of the design process. In his research, he applies and extends ideas from formal methods, in particular from satisfiability checking and probabilistic model checking.

Joost-Pieter Katoen is a Distinguished Professor with RWTH Aachen University, Germany, and holds a parttime professorship at the University of Twente, The Netherlands. He received a honorary doctorate degree from Aalborg University, Denmark. His research interests include formal methods, model checking, concurrency theory, and probabilistic computation. He coauthored more than 180 conference papers, 75 journal papers, and the book “Principles of Model Checking.” Prof. Katoen is the Chairman of the steering committee of ETAPS, and steering committee member of the conferences CONCUR, QEST, and FFORMATS. He is a member of Academia Europaea and holds an ERC Advanced Research Grant (2017).

Ufuk Topcu joined the Department of Aerospace Engineering at the University of Texas at Austin as an assistant professor in Fall 2015. He received his Ph.D. degree from the University of California at Berkeley in 2008. He held research positions at the University of Pennsylvania and California Institute of Technology. His research focuses on the theoretical, algorithmic, and computational aspects of design and verification of autonomous systems through novel connections between formal methods, learning theory and controls.