Sampling-Based Verification of CTMCs with Uncertain Rates

Thom S. Badings[1(✉)], Nils Jansen[1], Sebastian Junges[1], Marielle Stoelinga[1,2], and Matthias Volk[2]

[1] Radboud University, Nijmegen, The Netherlands
thom.badings@ru.nl
[2] University of Twente, Enschede, The Netherlands

Abstract. We employ uncertain parametric CTMCs with parametric transition rates and a prior on the parameter values. The prior encodes uncertainty about the actual transition rates, while the parameters allow dependencies between transition rates. Sampling the parameter values from the prior distribution then yields a standard CTMC, for which we may compute relevant reachability probabilities. We provide a principled solution, based on a technique called scenario-optimization, to the following problem: From a finite set of parameter samples and a user-specified confidence level, compute prediction regions on the reachability probabilities. The prediction regions should (with high probability) contain the reachability probabilities of a CTMC induced by any additional sample. To boost the scalability of the approach, we employ standard abstraction techniques and adapt our methodology to support approximate reachability probabilities. Experiments with various well-known benchmarks show the applicability of the approach.

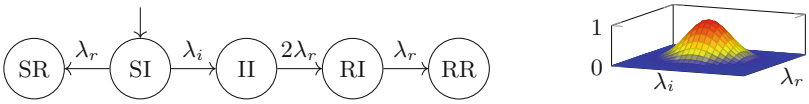
1 Introduction

Continuous-time Markov chains (CTMCs) are widely used to model complex probabilistic systems in reliability engineering [51], network processes [36,38], systems biology [11,23] and epidemic modeling [2]. A key verification task is to compute aspects of system behavior from these models, expressed as, e.g., continuous stochastic logic (CSL) formulae [4,7]. Typically, we compute reachability probabilities for a set of horizons, such as: *what is the probability that a target state is reached before time* $t_1, \ldots, t_n$*?* Standard algorithms [7] implemented in mature model checking tools such as Storm [37] or Prism [42] provide efficient means to compute these *reachability probabilities*. However, these methods typically require that transition rates and probabilities are precisely known. This assumption is often unrealistic [34] and led to some related work, which we discuss in Sect. 7.

Illustrative Example. An epidemic can abstractly be modeled as a finite-state CTMC, e.g., the SIR (susceptible-infected-recovered) model [3], which is shown

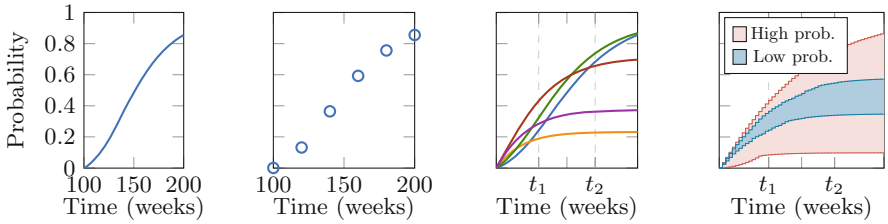
This work has been partially funded by NWO under the grant PrimaVera, number NWA.1160.18.238, and by EU Horizon 2020 project MISSION, number 101008233.

S. Shoham and Y. Vizel (Eds.): CAV 2022, LNCS 13372, pp. 26–47, 2022.
https://doi.org/10.1007/978-3-031-13188-2_2



(a) pCTMC $\mathcal{M}$ with parameters λ_i, λ_r. (b) Distribution $\mathbb{P}$ over values for (λ_i, λ_r).

Fig. 1. An upCTMC $(\mathcal{M}, \mathbb{P})$ for the SIR (pop=2) model.



(a) Curve for a single CTMC with precise transition rates.

(b) Point abstraction of a curve for a single CTMC.

(c) Curves for five CTMCs with different rates.

(d) Two prediction regions with different probabilities.

Fig. 2. The probability of extinction in the SIR (140) model for horizons $[100, t]$. (Color figure online)

in Fig. 1a for a population of two. Such a CTMC assumes a *fixed set of transition rates*, in this case an infection rate λ_i, and a recovery rate λ_r. The outcome of analyzing this CTMC for fixed values of λ_i and λ_r may yield a *probability curve* like in Fig. 2a[1], where we plot the probability (y-axis) of reaching a target state that corresponds to the epidemic becoming extinct against varying time horizons (x-axis). In fact, the plot is obtained via a smooth interpolation of the results at finitely many horizons, cf. Fig. 2b. To acknowledge that λ_i, λ_r are in fact unknown, we may analyze the model for different values of λ_i, λ_r, resulting in a set of curves as in Fig. 2c. These individual curves, however, provide no guarantees about the shape of the curve obtained from another infection and recovery rate. Instead, we assume a *probability distribution* over the transition rates and aim to compute *prediction regions* as those in shown Fig. 2d, in such a way that with a certain (high) probability, any rates λ_i and λ_r yield a curve within this region.

Overall Goal. From the illustrative example, we state the following goal. Each fixed set of transition rates induces a *probability curve*, i.e., a mapping from horizons to the corresponding reachability probabilities. We aim to construct *prediction regions* around a set of probability curves, such that with high probability and high confidence, sampling a set of transition rates induces a probability curve within this region. Our key contribution is an efficient *probably approximately correct*, or PAC-style method that computes these prediction regions. The remainder of the introduction explores the technical steps toward this goal.

[1] For visual clarity, we plot the reachability probability *between* time 100 and $t_1, \ldots, t_n$.

Uncertain CTMCs. The setting above is formally captured by parametric CTMCs (pCTMCs). Transition rates of pCTMCs are not given precisely but as (polynomials over) parameters [15,34], such as those shown in Fig. 1a. We assume a *prior* on each parameter valuation, i.e., assignment of values to parameters, similar to settings in [11,44] and in contrast to, e.g., [23,34]. These priors may result from asking different experts which value they would assume for, e.g., the infection rate. The prior may also be the result of Bayesian reasoning [56]. Formally, we capture the uncertainty in the rates by an arbitrary and potentially unknown *probability distribution* over the parameter space, see Fig. 1b. We call this model an *uncertain pCTMC (upCTMC)*. The distribution allows drawing independent and identically distributed (i.i.d.) *samples* that yield (parameter-free) CTMCs.

Problem Statement. We consider prediction regions on probability curves in the form of a pair of two curves that 'sandwich' the probability curves, as depicted in Fig. 2d. Intuitively, we then aim to find a prediction region R that is sufficiently large, such that sampling parameter valuations yields a probability curve in R with high probability p. We aim to compute a lower bound on this *containment probability* p. Naturally, we also aim to compute a meaningful, i.e. small (tight), prediction region R. As such, we aim to solve the following problem:

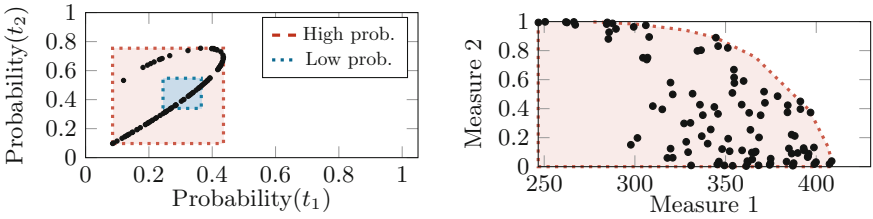
Problem Statement. Given a upCTMC with a target state, compute
1. a (tight) *prediction region* R on the probability curves, and
2. a (tight) *lower bound on the containment probability* that a sampled parameter valuation induces a probability curve that will lie in R.

We solve this problem with a user-specified confidence level β.

The Problem Solved. In this paper, we present a method that samples probability curves as in Fig. 2c, but now for, say 100 curves. From these curves, we compute prediction regions (e.g., both tubes in Fig. 2d) and compute a lower bound (one for both tubes) on the containment probability that the curve associated with any sampled parameter value will lie in the specific prediction region (tube). Specifically, for a confidence level of 99% and considering 100 curves, we conclude that this lower bound is 79.4% for the red region and 7.5% for the blue region. For a higher confidence level of 99.9%, the lower bounds are slightly more conservative.

A Change in Perspective. Toward the algorithm, we make a change in perspective. For two horizons t_1 and t_2, reachability probabilities for fixed CTMCs are two-dimensional points in $[0,1]^2$ that we call *solution vectors*, as shown in Fig. 3a. Here, these solution vectors represent pairs of the probabilities that the disease becomes extinct before time t_1 and before t_2. The prediction regions as in Fig. 2d are shown as the shaded boxes in Fig. 3a.

Solving the problem algorithmically. We solve the problem using a sampling-based approach. Starting with a set of solution vectors, we use techniques from *scenario optimization*, a data-driven methodology for solving stochastic optimization problems [18,21]. As such, we construct the prediction region from the solution to an optimization problem. Our method can balance the size of the prediction region with the containment probability, as illustrated by the two boxes in Fig. 3a.



(a) Reachability at time points t_1 and t_2. (b) Pareto front for two measures.

Fig. 3. Prediction regions on the solutions vectors for two different upCTMCs.

Extensions. Our approach offers more than prediction regions on probability curves from precise samples. The change in perspective mentioned above allows for solution vectors that represent *multiple objectives*, such as the reachability with respect to different goal states, expected rewards or even the probability mass of paths satisfying more complex temporal properties. In our experiments, we show that this multi-objective approach —also on probability curves— yields much tighter bounds on the containment probability than an approach that analyzes each objective independently. We can also produce prediction regions as other shapes than boxes, as, for example, shown in Fig. 3b. To accelerate our approach, we significantly extend the methodology for dealing with *imprecise verification results*, given as an interval on each entry of the solution vector.

Contributions. Our key contribution is the approach that provides prediction regions and lower bounds on probability curves for upCTMCs. The approach requires only about 100 samples and scales to upCTMCs with tens of parameters. Furthermore: (1) We extend our approach such that we can also handle the case where only imprecise intervals on the verification results are available. (2) We develop a tailored batch verification method in the model checker Storm [37] to accelerate the required batches of verification tasks. We accompany our contributions by a thorough empirical evaluation and remark that our batch verification method can be used beyond scenario optimization. Our scenario optimization results are independent of the model checking and are, thus, applicable to any model where solution vectors are obtained in the same way as for upCTMCs.

Data Availability. All source code, benchmarks, and logfiles used to produce the data are archived: https://doi.org/10.5281/zenodo.6523863.

2 Problem Statement

In this section, we introduce pCTMCs and upCTMCs, and we define the formal problem statement. We use probability distributions over finite and infinite sets; see [9] for details. The set of all distributions over a set X is denoted by $Dist(X)$. The set of polynomials over parameters V, with rational coefficients, is denoted by $\mathbb{Q}[V]$. An *instantiation* $u\colon V \to \mathbb{Q}$ maps parameters to concrete values. We often fix a parameter ordering and denote instantiations as vectors, $u \in \mathbb{Q}^{|V|}$.

Definition 1 (pCTMC). *A* pCTMC *is a tuple* $\mathcal{M} = (S, s_I, V, \mathbf{R})$*, where* S *is a finite set of states,* $s_I \in Dist(S)$ *is the initial distribution,* V *are the (ordered) parameters, and* $\mathbf{R}\colon S \times S \to \mathbb{Q}[V]$ *is a parametric transition rate function. If* $\mathbf{R}(s,s) \in \mathbb{Q}_{\geq 0}$ *for all* $s, s' \in S$*, then* $\mathcal{M}$ *is a (parameter-free)* CTMC*.*

For any pair of states $s, s' \in S$ with a non-zero rate $\mathbf{R}(s,s') > 0$, the probability of triggering a transition from s to s' within t time units is $1 - e^{-\mathbf{R}(s,s')\cdot t}$ [41].

Applying an *instantiation* u to a pCTMC $\mathcal{M}$ yields an *instantiated* CTMC $\mathcal{M}[u] = (S, s_I, V, \mathbf{R}[u])$ where $\mathbf{R}[u](s,s') = \mathbf{R}(s,s')[u]$ for all $s, s' \in S$. In the remainder, we only consider instantiations u for a pCTMC $\mathcal{M}$ which are *well-defined*. The set of such instantiations is the parameter space $\mathcal{V}_\mathcal{M}$.

A central *measure* on CTMCs is the *(time-bounded) reachability* $\Pr(\Diamond^{\leq \tau} E)$, which describes the probability that one of the error states E[2] is reached within the horizon $\tau \in \mathbb{Q}$. Other measures include the expected time to reach a particular state, or the average time spent in particular states. We refer to [41] for details.

Given a concrete (instantiated) CTMC $\mathcal{M}[u]$, the *solution* for measure φ is denoted by $\mathsf{sol}^{\varphi}_{\mathcal{M}[u]} \in \mathbb{R}$; the *solution vector* $\mathsf{sol}^{\Phi}_{\mathcal{M}[u]} \in \mathbb{R}^m$ generalizes this concept to an (ordered) set of m measures $\Phi = \varphi_1, \ldots, \varphi_m$. We abuse notation and introduce the *solution function* to express solution vectors on a pCTMC:

Definition 2 (Solution function). *A* solution function $\mathsf{sol}^{\Phi}_{\mathcal{M}}\colon \mathcal{V}_\mathcal{M} \to \mathbb{R}^{|\Phi|}$ *is a mapping from a parameter instantiation* $u \in \mathcal{V}_\mathcal{M}$ *to the solution vector* $\mathsf{sol}^{\Phi}_{\mathcal{M}[u]}$*.*

We often omit the scripts in $\mathsf{sol}^{\Phi}_{\mathcal{M}}(u)$ and write $\mathsf{sol}(u)$ instead. We also refer to $\mathsf{sol}(u)$ as the solution vector of u. For n parameter samples $\mathcal{U}_n = \{u_1, \ldots, u_n\}$ with $u_i \in \mathcal{V}_\mathcal{M}$, we denote the solution vectors by $\mathsf{sol}(\mathcal{U}_n) \in \mathbb{R}^{m \times n}$.

Using solution vectors, we can define the probability curves shown in Fig. 2c.

Definition 3 (Probability curve). *The* probability curve *for reachability probability* $\phi_\tau = \Pr(\Diamond^{\leq \tau} E)$ *and CTMC* $\mathcal{M}[u]$ *is given by* $\mathsf{probC} : \tau \mapsto \mathsf{sol}^{\varphi_\tau}_{\mathcal{M}[u]}$*.*

We can approximate the function probC for a concrete CTMC by computing $\mathsf{probC}(t_1), \ldots, \mathsf{probC}(t_m)$ for a finite set of time horizons. As such, we compute the solution vector w.r.t. m different reachability measures $\Phi = \{\varphi_{t_1}, \ldots, \varphi_{t_m}\}$. By exploiting the monotonicity[3] of the reachability over time, we obtain an upper and lower bound on $\mathsf{probC}(\tau)$ as two step functions, see Fig. 2d. We can smoothen the approximation, by taking an upper and lower bound on these step functions.

We study pCTMCs where the parameters follow a probability distribution. This probability distribution can be highly complex or even unknown; we merely assume that we can sample from this distribution.

Definition 4 (upCTMC). *A* upCTMC *is a tuple* $(\mathcal{M}, \mathbb{P})$ *with* $\mathcal{M}$ *a pCTMC and* $\mathbb{P}$ *a probability distribution over the parameter space* $\mathcal{V}_\mathcal{M}$ *of* $\mathcal{M}$*.*

[2] Formally, states are labeled and E describes the label, see [8].

[3] In Definition 3, only the upper limit on the timebound is varied, so measures are monotonic.

A upCTMC defines a probability space $(\mathcal{V}_\mathcal{M}, \mathbb{P})$ over the parameter values, whose domain is defined by the parameter space $\mathcal{V}_\mathcal{M}$. In the remainder, we denote a *sample* from $\mathcal{V}_\mathcal{M}$ drawn according to $\mathbb{P}$ by $u \in \mathcal{V}_\mathcal{M}$.

To quantify the performance of a upCTMC, we may construct a *prediction region* on the solution vector space, such as those shown in Fig. 3a. In this paper, we consider only prediction regions which are compact subsets $R \subseteq \mathbb{R}^{|\Phi|}$. We define the so-called *containment probability* of a prediction region, which is the probability that the solution vector $\mathsf{sol}(u)$ for a randomly sampled parameter $u \in \mathcal{V}_\mathcal{M}$ is contained in R, as follows:

Definition 5 (Containment probability). *For a prediction region R, the* containment probability $\mathsf{contain}_\mathcal{V}(R)$ *is the probability that the solution vector* $\mathsf{sol}(u)$ *for any parameter sample $u \in \mathcal{V}_\mathcal{M}$ is contained in R:*

$$\mathsf{contain}_\mathcal{V}(R) = \Pr\{u \in \mathcal{V}_\mathcal{M} : \mathsf{sol}(u) \in R\}. \tag{1}$$

Recall that we solve the problem in Sect. 1 with a user-specified confidence level, denoted by $\beta \in (0, 1)$. Formally, we solve the following problem:

Formal Problem. Given a upCTMC $(\mathcal{M}, \mathbb{P})$, a set Φ of measures, and a confidence level $\beta \in (0, 1)$, compute a (tight) prediction region R and a (tight) lower bound $\mu \in (0, 1)$ on the containment probability, such that $\mathsf{contain}(R) \geq \mu$ holds with a confidence level of at least β.

The problem in Sect. 1 is a special case of the formal problem, with Φ the reachability probability over a set of horizons. In that case, we can overapproximate a prediction region as a rectangle, yielding an interval $[\underline{c}, \bar{c}]$ for every horizon t that defines where the two step functions (see below Definition 3) change. We smoothen these step functions (similar to probability curves) to obtain the following definition:

Definition 6 (Prediction region for a probability curve). *A* prediction region R *over a probability curve* probC *is given by two curves $\underline{c}, \bar{c} : \mathbb{Q}_{\geq 0} \to \mathbb{R}$ as the area in-between: $R = \{(t, y) \in \mathbb{Q} \times \mathbb{R} \mid \underline{c}(t) \leq y \leq \bar{c}(t)\}$.*

We solve the problem by sampling a finite set $\mathcal{U}_n$ of parameter values of the upCTMC and computing the corresponding solution vectors $\mathsf{sol}(\mathcal{U}_n)$. In Sect. 3, we solve the problem assuming that we can compute solution vectors exactly. In Sect. 4, we consider a less restricted setting in which every solution is imprecise, i.e. only known to lie in a certain interval.

3 Precise Sampling-Based Prediction Regions

In this section, we use scenario optimization [16,18] to compute a high-confidence lower bound on the containment probability. First, in Sect. 3.1, we describe how to compute a prediction region using the solution vectors $\mathsf{sol}(\mathcal{U}_n)$ for the parameter samples $\mathcal{U}_n$. In Sect. 3.2, we clarify how to compute a lower bound on the containment probability with respect to this prediction region. In Sect. 3.3, we construct an algorithm based on those results that solves the formal problem.

3.1 Constructing Prediction Regions

We assume that we are given a set of solution vectors $\mathsf{sol}(\mathcal{U}_n)$ obtained from n parameter samples. We construct a prediction region R based on these vectors such that we can annotate these regions with a lower bound on the containment probability, as in the problem statement. For conciseness, we restrict ourselves to the setting where R is a hyperrectangle in $\mathbb{R}^m$, with $m = |\Phi|$ the number of measures, cf. Remark 1 below. In the following, we represent R using two vectors (points) $\underline{x}, \bar{x} \in \mathbb{R}^m$ such that, using pointwise inequalities, $R = \{x \mid \underline{x} \leq x \leq \bar{x}\}$. For an example of such a rectangular prediction region, see Fig. 3a.

As also shown in Fig. 3a, we do *not* require R to contain all solutions in $\mathsf{sol}(\mathcal{U}_n)$. Instead, we have two orthogonal goals: we aim to minimize the size of R, while also minimizing the (Manhattan) distance of samples to R, measured in their 1-norm. Solutions contained in R are assumed to have a distance of zero, while solutions not contained in R are called *relaxed*. These goals define a *multi-objective problem*, which we solve by weighting the two objectives using a fixed parameter $\rho > 0$, called the *cost of relaxation*, that is used to scale the distance to R. Then, $\rho \to \infty$ enforces $\mathsf{sol}(\mathcal{U}_n) \subseteq R$, as in the outer box in Fig. 3a, while for $\rho \to 0$, R is reduced to a point. Thus, the cost of relaxation ρ is a tuning parameter that determines the size of the prediction region R and hence the fraction of the solution vectors that is contained in R (see [19,21] for details).

We capture the problem described above in the following convex optimization problem $\mathfrak{L}^\rho_{\mathcal{U}}$. We define the decision variables $\underline{x}, \bar{x} \in \mathbb{R}^m$ to represent the prediction region. In addition, we define a decision variable $\xi_i \in \mathbb{R}^m_{\geq 0}$ for every sample $i = 1, \ldots, n$ that acts as a slack variable representing the distance to R.

$$\mathfrak{L}^\rho_{\mathcal{U}}: \text{ minimize} \quad \|\bar{x} - \underline{x}\|_1 + \rho \sum_{i=1}^{n} \|\xi_i\|_1 \tag{2a}$$

$$\text{subject to} \quad \underline{x} - \xi_i \leq \mathsf{sol}(u_i) \leq \bar{x} + \xi_i \quad \forall i = 1, \ldots, n. \tag{2b}$$

The objective function in Eq. (2a) minimizes the size of R —by minimizing the sum of the width of the prediction region in all dimensions— plus ρ times the distances of the samples to R. We denote the optimal solution to problem $\mathfrak{L}^\rho_{\mathcal{U}}$ for a given ρ by R^*_ρ, ξ^*_ρ, where $R^*_\rho = [\underline{x}^*_\rho, \bar{x}^*_\rho]$ for the rectangular case.

Assumption 1. *The optimal solution R^*_ρ, ξ^*_ρ to $\mathfrak{L}^\rho_{\mathcal{U}}$ exists and is unique.*

Note that Definition 2 ensures finite-valued solution vectors, thus guaranteeing the existence of a solution to Eq. (2). If the solution is not unique, we apply a suitable tie-break rule that selects one solution of the optimal set (e.g., the solution with a minimum Euclidean norm, see [16]). The following example shows that values of ρ exist for which such a tie-break rule is necessary to obtain a unique solution.

Example 1. Figure 4 shows a set of solution vectors in one dimension, labeled A–F. Consider prediction region $R_1 = [A, F]$. The corresponding objective value Eq. (2a) is $\|\bar{x} - \underline{x}\| + \rho \cdot \sum \xi_i = \|\bar{x} - \underline{x}\| = \delta_1 + \cdots + \delta_5$, as all $\xi_i = 0$. For prediction region $R_2 = [B, E]$, the objective value is $\delta_2 + \delta_3 + \delta_4 + \rho \cdot \delta_1 + \rho \cdot \delta_5$. Thus, for $\rho > 1$,

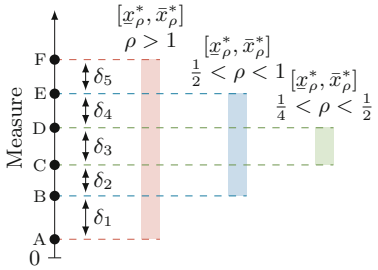


Fig. 4. The prediction region changes with the cost of relaxation ρ.

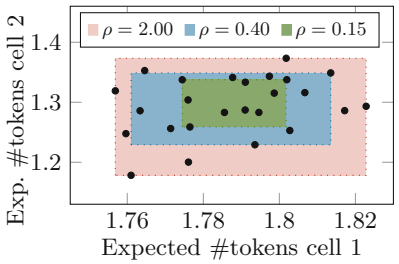


Fig. 5. Prediction regions as boxes, for different costs of relaxations ρ.

solving $\mathfrak{L}^\rho_{\mathcal{U}}$ yields R_1 whereas for $\rho < 1$, relaxing solutions A and F is cheaper than not doing so, so R_2 is optimal. When $\rho = 1$, however, relaxing solutions A and F yields the same cost as not relaxing these samples, so a tie-break rule is needed (see above). For $\rho < \frac{1}{2}$, relaxing samples A, B, E, and F is cost-optimal, resulting in the prediction region containing exactly $\{C, D\}$. □

Similarly, we can consider cases with more samples and multiple measures, as shown in Fig. 5 (see [6, Appendix A] for more details). The three prediction regions in Fig. 5 are obtained for different costs of relaxation ρ. For $\rho = 2$, the region contains all vectors, while for a lower ρ, more vectors are left outside.

Remark 1. While problem $\mathfrak{L}^\rho_{\mathcal{U}}$ in Eq. (2) yields a rectangular prediction region, we can also produce other shapes. We may, e.g., construct a Pareto front as in Fig. 3b, by adding additional affine constraints [12]. In fact, our only requirement is that the objective function is convex, and the constraints are convex in the decision variables (the dependence of the constraints on u may be arbitrary) [21]. □

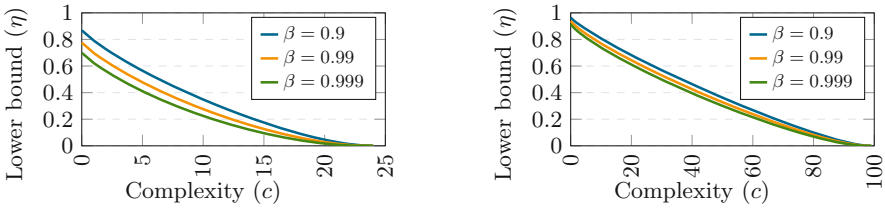
3.2 Bounding the Containment Probability

The previous section shows how we compute a prediction region based on convex optimization. We now characterize a valid high-confidence lower bound on the containment probability w.r.t. the prediction region given by the optimal solution to this optimization problem. Toward that result, we introduce the so-called *complexity* of a solution to problem $\mathfrak{L}^\rho_{\mathcal{U}}$ in Eq. (2), a concept used in [21] that is related to the compressibility of the solution vectors $\mathsf{sol}(\mathcal{U}_n)$:

Definition 7 (Complexity). *For $\mathfrak{L}^\rho_{\mathcal{U}}$ with optimal solution R^*_ρ, ξ^*_ρ, consider a set $\mathcal{W} \subseteq \mathcal{U}_n$ and the associated problem $\mathfrak{L}^\rho_{\mathcal{W}}$ with optimal solution $\tilde{R}_\rho, \tilde{\xi}_\rho$. The set $\mathcal{W}$ is* critical, *if*

$$\tilde{R}_\rho = R^*_\rho \quad \text{and} \quad \{u_i \mid \xi^*_{\rho,i} > 0\} \subseteq \mathcal{W}.$$

The complexity *c^*_ρ of R^*_ρ, ξ^*_ρ is the cardinality of the smallest critical set. We also call c^*_ρ the complexity of $\mathfrak{L}^\rho_{\mathcal{U}}$.*



(a) Number of samples $n = 25$.

(b) Number of samples $n = 100$.

Fig. 6. Lower bounds η on the containment probability as a function of the complexity c, obtained from Theorem 1 for different confidence levels β.

If a sample u_i has a value $\xi^*_{\rho,i} > 0$, its solution vector has a positive distance to the prediction region, R^*_ρ. (i.e., $[\underline{x}^*_\rho, \bar{x}^*_\rho]$ for the rectangular case). Thus, the complexity c^*_ρ is the number of samples for which $\mathsf{sol}(u_i) \notin R^*_\rho$, plus the minimum number of samples needed on the boundary of the region to keep the solution unchanged. We describe in Sect. 3.3 how we algorithmically determine the complexity.

Example 2. In Fig. 5, the prediction region for $\rho = 2$ contains all solution vectors, so $\xi^*_{2,i} = 0 \, \forall i$. Moreover, if we remove *all but four* solutions (the ones on the boundary of the region), the optimal solution to problem $\mathfrak{L}^\rho_{\mathcal{U}}$ remains unchanged, so the complexity is $c^*_{1.12} = 0 + 4$. Similarly, the complexity for $\rho = 0.4$ is $c^*_{0.4} = 8 + 2 = 10$ (8 solutions outside the region, and 2 on the boundary). □

Recall that Definition 5 defines the containment probability of a generic prediction region R, so $\mathsf{contain}(R^*_\rho)$ is the containment probability w.r.t. the optimal solution to $\mathfrak{L}^\rho_{\mathcal{U}}$. We adapt the following theorem from [21], which gives a lower bound on the containment probability $\mathsf{contain}(R^*_\rho)$ of an optimal solution to $\mathfrak{L}^\rho_{\mathcal{U}}$ for a predefined value of ρ. This lower bound is correct with a user-defined confidence level of $\beta \in (0, 1)$, which we typically choose close to one (e.g., $\beta = 0.99$).

Theorem 1. *Let $\mathcal{U}_n$ be a set of n samples, and let c^* be the complexity of problem $\mathfrak{L}^\rho_{\mathcal{U}}$. For any confidence level $\beta \in (0, 1)$ and any upper bound $d^* \geq c^*$, it holds that*

$$\mathbb{P}^n \Big\{ \mathsf{contain}\big(R^*_\rho\big) \geq \eta(d^*) \Big\} \geq \beta, \tag{3}$$

where R^*_ρ is the prediction region for $\mathfrak{L}^\rho_{\mathcal{U}}$. Moreover, η is a function defined as $\eta(n) = 0$, and otherwise, $\eta(c)$ is the smallest positive real-valued solution to the following polynomial equality in the t variable for a complexity of c:

$$\binom{n}{c} t^{n-c} - \frac{1-\beta}{2n} \sum_{i=c}^{n-1} \binom{i}{c} t^{i-c} - \frac{1-\beta}{6n} \sum_{i=n+1}^{4n} \binom{i}{c} t^{i-c} = 0. \tag{4}$$

We provide the proof of Theorem 1 in [6, Appendix B.1]. With a probability of at least β, Theorem 1 yields a correct lower bound. That is, if we solve $\mathfrak{L}^\rho_{\mathcal{U}}$

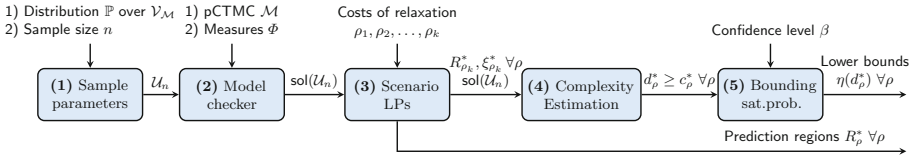


Fig. 7. Overview of our approach for solving the problem statement.

for many more sets of n parameter samples (note that, as the samples are i.i.d., these sets are drawn according to the product probability $\mathbb{P}^n$), the inequality in Eq. (3) is incorrect for *at most* a $1-\beta$ fraction of the cases. We plot the lower bound $\eta(c)$ as a function of the complexity $c = 0, \ldots, n$ in Fig. 6, for different samples sizes n and confidence levels β. These figures show that an increased complexity leads to a lower η, while increasing the sample size leads to a tighter bound.

Example 3. We continue Example 2. Recall that the complexity for the outer region in Fig. 5 is $c^*_{1.12} = 4$. With Theorem 1, we compute that, for a confidence level of $\beta = 0.9$, the containment probability for this prediction region is at least $\eta = 0.615$ (cf. Figure 6a). For a stronger confidence level of $\beta = 0.999$, we obtain a more conservative lower bound of $\eta = 0.455$. □

3.3 An Algorithm for Computing Prediction Regions

We combine the previous results in our algorithm, which is outlined in Fig. 7. The goal is to obtain a set of prediction regions as in Fig. 5 and their associated lower bounds. To strictly solve the problem statement, assume $k = 1$ in the exposition below. We first outline the complete procedure before detailing Steps 4 and 5.

As preprocessing steps, given a upCTMC $(\mathcal{M}, \mathbb{P})$, we first (1) sample a set $\mathcal{U}_n$ of n parameter values. Using $\mathcal{M}$ and Φ, a (2) model checking algorithm then computes the solution vector $\mathsf{sol}^{\Phi}_{\mathcal{M}}(u)$ for each $u \in \mathcal{U}_n$, yielding the set of solutions $\mathsf{sol}(\mathcal{U}_n)$. We then use $\mathsf{sol}(\mathcal{U}_n)$ as basis for (3) the scenario problem $\mathfrak{L}^{\rho}_{\mathcal{U}}$ in Eq. (2), which we solve for k predefined values $\rho_1, \ldots, \rho_k$, yielding k prediction regions $R^*_{\rho_1}, \ldots R^*_{\rho_k}$. We (4) compute an upper bound d^*_ρ on the complexity $c^*_\rho\ \forall\rho$. Finally, we (5) use the result in Theorem 1, for a given confidence β, to compute the lower bound on the containment probability $\eta(d^*_\rho)$ of R^*_ρ. Using Definition 6, we can postprocess this region to a prediction region over the probability curves.

Step (3): Choosing values for ρ. Example 1 shows that relaxation of additional solution vectors (and thus a change in the prediction region) only occurs at *critical* values of $\rho = \frac{1}{n}$, for $n \in \mathbb{N}$. In practice, we will use $\rho = \frac{1}{n+0.5}$ for ± 10 values of $n \in \mathbb{N}$ to obtain gradients of prediction regions as in Sect. 6.

Step (4): Computing complexity. Computing the complexity c^*_ρ is a combinatorial problem in general [30], because we must consider the removal of all combinations

of the solutions on the boundary of the prediction region R^*_ρ. In practice, we compute an upper bound $d^*_\rho \geq c^*_\rho$ on the complexity via a greedy algorithm. Specifically, we iteratively solve $\mathfrak{L}^\rho_{\mathcal{U}}$ in Eq. (2) with *one more sample on the boundary removed*. If the optimal solution is unchanged, we conclude that this sample does not contribute to the complexity. If the optimal solution is changed, we put the sample back and proceed by removing a different sample. This greedy algorithm terminates when we have tried removing all solutions on the boundary.

Step (5): Computing lower bounds. Theorem 1 characterizes a computable function $B(d^*, n, \beta)$ that returns zero for $d^* = n$ (i.e., all samples are critical), and otherwise uses the polynomial Eq. (4) to obtain η, which we solve with an approximate root finding method in practice (see [31] for details on how to ensure that we find the smallest root). For every upper bound on the complexity d^* and any requested confidence, we obtain the lower bound $\eta = B(d^*, n, \beta)$ for the containment probability w.r.t. the prediction region R^*_ρ.

4 Imprecise Sampling-Based Prediction Regions

Thus far, we have solved our problem statement under the assumption that we compute the solution vectors precisely (up to numerics). For some models, however, computing precise solutions is expensive. In such a case, we may choose to compute an approximation, given as an *interval* on each entry of the solution function. In this section, we deal with such *imprecise solutions*.

Setting. Formally, imprecise solutions are described by the bounds $\mathsf{sol}^-(u), \mathsf{sol}^+(u) \in \mathbb{R}^m$ such that $\mathsf{sol}^-(u) \leq \mathsf{sol}(u) \leq \mathsf{sol}^+(u)$ holds with pointwise inequalities. Our goal is to compute a prediction region R and a (high-confidence) lower bound μ such that $\mathsf{contain}(R) \geq \mu$, i.e., a lower bound on the probability that any *precise solution* $\mathsf{sol}(u)$ is contained in R. However, we must now compute R and $\mathsf{contain}(R)$ from the imprecise solutions $\mathsf{sol}^-, \mathsf{sol}^+$. Thus, we aim to provide a guarantee with respect to the *precise* solution $\mathsf{sol}(u)$, based on *imprecise* solutions.

Challenge. Intuitively, if we increase the (unknown) prediction region R^* from problem $\mathfrak{L}^\rho_{\mathcal{U}}$ (for the unknown precise solutions) while also overapproximating the complexity of $\mathfrak{L}^\rho_{\mathcal{U}}$, we obtain sound bounds. We formalize this idea as follows.

Lemma 1. *Let R^*_ρ be the prediction region and c^*_ρ the complexity that result from solving $\mathfrak{L}^\rho_{\mathcal{U}}$ for the precise (unknown) solutions $\mathsf{sol}(\mathcal{U}_n)$. Given a set $R \in \mathbb{R}^n$ and $d \in \mathbb{N}$, for any confidence level $\beta \in (0, 1)$, the following implication holds:*

$$R^*_\rho \subseteq R \text{ and } c^*_\rho \leq d \implies \mathbb{P}^n\Big\{\mathsf{contain}\big(R\big) \geq \eta(d)\Big\} \geq \beta, \tag{5}$$

where $\eta(n) = 0$, and otherwise, $\eta(d)$ is the smallest positive real-valued solution to the polynomial equality in Eq. (4).

The proof is in [6, Appendix B.2]. In what follows, we clarify how we compute the appropriate R and d in Lemma 1. As we will see, in contrast to Sect. 3, these results do *not* carry over to other definitions $\mathfrak{L}^\rho_{\mathcal{U}}$ (for non-rectangular regions R).

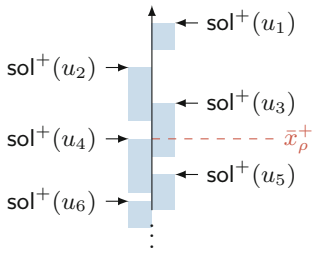


Fig. 8. Imprecise solutions and the upper bound $\bar{x}'_\rho$ of the prediction region.

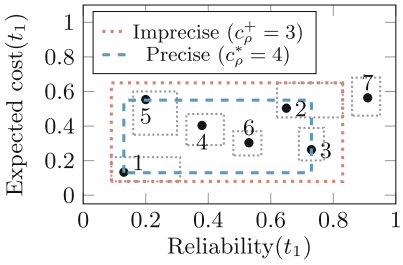


Fig. 9. Complexity of the imprecise solution vs. that of the precise solution.

4.1 Prediction Regions on Imprecise Solutions

In this section, we show how to compute $R \supseteq R^*_\rho$, satisfying the first term in the premise of Lemma 1. We construct a *conservative box* around the imprecise solutions as in Fig. 9, containing both $\mathsf{sol}^-(u)$ and $\mathsf{sol}^+(u)$. We compute this box by solving the following problem $\mathfrak{G}^\rho_{\mathcal{U}}$ as a modified version of $\mathfrak{L}^\rho_{\mathcal{U}}$ in Eq. (2):

$$\mathfrak{G}^\rho_{\mathcal{U}}: \text{ minimize } \|\bar{x} - \underline{x}\|_1 + \rho \sum_{i=1}^{n} \|\xi_i\|_1 \tag{6a}$$

$$\text{subject to } \quad \underline{x} - \xi_i \leq \mathsf{sol}^-(u_i), \;\; \mathsf{sol}^+(u_i) \leq \bar{x} + \xi_i \quad \forall i = 1, \ldots, n. \tag{6b}$$

We denote the optimal solution of $\mathfrak{G}^\rho_{\mathcal{U}}$ by $[\underline{x}'_\rho, \bar{x}'_\rho], \xi'_\rho$ (recall that the optimum to $\mathfrak{L}^\rho_{\mathcal{U}}$ is written as $[\underline{x}^*_\rho, \bar{x}^*_\rho], \xi^*_\rho$).[4] If a sample $u_i \in \mathcal{V}_{\mathcal{M}}$ in problem $\mathfrak{G}^\rho_{\mathcal{U}}$ is relaxed (i.e., has a non-zero ξ_i), part of the interval $[\mathsf{sol}^-(u_i), \mathsf{sol}^+(u_i)]$ is not contained in the prediction region. The following result (for which the proof is in [6, Appendix B.3]. relates $\mathfrak{L}^\rho_{\mathcal{U}}$ and $\mathfrak{G}^\rho_{\mathcal{U}}$, showing that we can use $[\underline{x}'_\rho, \bar{x}'_\rho]$ as R in Lemma 1.

Theorem 2. *Given ρ, sample set $\mathcal{U}_n$, and prediction region $[\underline{x}'_\rho, \bar{x}'_\rho]$ to problem $\mathfrak{G}^\rho_{\mathcal{U}}$, it holds that $[\underline{x}^*_\rho, \bar{x}^*_\rho] \subseteq [\underline{x}'_\rho, \bar{x}'_\rho]$, with $[\underline{x}^*_\rho, \bar{x}^*_\rho]$ the optimal solution to $\mathfrak{L}^\rho_{\mathcal{U}}$.*

We note that this result is not trivial. In particular, the entries ξ_i from both LPs are incomparable, as are their objective functions. Instead, Theorem 2 relies on two observations. First, due to the use of the 1-norm, the LP $\mathfrak{G}^\rho_{\mathcal{U}}$ can be decomposed into n individual LPs, whose results combine into a solution to the original LP. This allows us to consider individual dimensions. Second, the solution vectors that are relaxed depend on the value of ρ and on their *relative order*, but not on the *precise position* within that order, which is also illustrated by Example 1. In combination with the observation from Example 1 that the *outermost* samples are relaxed at the (relatively) highest ρ, we can provide conservative guarantees on which samples are (or are surely not) relaxed. We formalize these observations and provide a proof of Theorem 2 in [6, Appendix B.3].

[4] We write $[\underline{x}^*_\rho, \bar{x}^*_\rho]$ and $[\underline{x}'_\rho, \bar{x}'_\rho]$, as results in Sect. 4 apply only to rectangular regions.

4.2 Computing the Complexity

To satisfy the second term of the premise in Lemma 1, we compute an upper bound on the complexity. We first present a negative result. Let the complexity c'_ρ of problem $\mathfrak{G}^\rho_{\mathcal{U}}$ be defined analogous to Definition 7, but with $[\underline{x}'_\rho, \bar{x}'_\rho]$ as the region.

Lemma 2. *In general, $c^*_\rho \leq c'_\rho$ does not hold.*

Proof. In Fig. 9, the smallest critical set for the imprecise solutions are those labeled $\{1, 2, 7\}$, while this set is $\{1, 3, 5, 7\}$ under precise solutions, so $c^*_\rho > c'_\rho$. □

Thus, we cannot upper bound the complexity directly from the result to $\mathfrak{G}^\rho_{\mathcal{U}}$. We can, however, determine the samples that are certainly *not* in any critical set (recall Definition 7). Intuitively, a sample is *surely noncritical* if its (imprecise) solution is strictly within the prediction region and does not overlap with any solution on the region's boundary. In Fig. 8, sample u_6 is surely noncritical, but sample u_5 is not (whether u_5 is critical depends on its precise solution). Formally, let δR be the boundary[5] of region $[\underline{x}'_\rho, \bar{x}'_\rho]$, and let $\mathcal{B}$ be the set of samples whose solutions overlap with δR, which is $\mathcal{B} = \{u \in \mathcal{U}_n : [\mathsf{sol}^-(u), \mathsf{sol}^+(u)] \cap \delta R \neq \varnothing\}$.

Definition 8. *For a region $[\underline{x}'_\rho, \bar{x}'_\rho]$, let $\mathcal{I} \subset [\underline{x}'_\rho, \bar{x}'_\rho]$ be the rectangle of largest volume, such that $\mathcal{I} \cap [\mathsf{sol}^-(u), \mathsf{sol}^+(u)] = \varnothing$ for any $u \in \mathcal{B}$. A sample $u_i \in \mathcal{V}_\mathcal{M}$ is* surely noncritical *if $[\mathsf{sol}^-(u_i), \mathsf{sol}^+(u_i)] \subseteq \mathcal{I}$. The set of all surely noncritical samples w.r.t. the (unknown) prediction region $[\underline{x}^*_\rho, \bar{x}^*_\rho]$ is denoted by $\mathcal{X} \subset \mathcal{U}_n$.*

As a worst case, any sample not surely noncritical can be in the smallest critical set, leading to the following bound on the complexity as required by Lemma 1.

Theorem 3. *Let $\mathcal{X}$ be the set of surely noncritical samples. Then $c^*_\rho \leq |\mathcal{U}_n \setminus \mathcal{X}|$.*

The proof is in [6, Appendix B.4]. For imprecise solutions, the bound in Theorem 3 is conservative but can potentially be improved, as discussed in the following.

4.3 Solution Refinement Scheme

Often, we can *refine* imprecise solutions arbitrarily (at the cost of an increased computation time). Doing so, we can improve the prediction regions and upper bound on the complexity, which in turn improves the computed bound on the containment probability. Specifically, we propose the following rule for refining solutions. After solving $\mathfrak{G}^\rho_{\mathcal{U}}$ for a given set of imprecise solutions, we refine the solutions on the boundary of the obtained prediction region. We then resolve problem $\mathfrak{G}^\rho_{\mathcal{U}}$, thus adding a loop back from (4) to (2) in our algorithm shown in Fig. 7. In our experiments, we demonstrate that with this refinement scheme, we iteratively improve our upper bound $d \geq c^*_\rho$ and the smallest superset $R \supseteq R^*_\rho$.

[5] The boundary of a compact set is defined as its closure minus its interior [45].

5 Batch Verification for CTMCs

One bottleneck in our method is to obtain the necessary number of solution vectors $\mathsf{sol}(\mathcal{U}_n)$ by model checking. The following improvements, while mild, are essential in our implementation and therefore deserve a brief discussion.

In general, computing $\mathsf{sol}(u)$ via model checking consists of two parts. First, the high-level representation of the upCTMC —given in Prism [42], JANI [13], or a dynamic fault tree[6]— is translated into a concrete CTMC $\mathcal{M}[u]$. Then, from $\mathcal{M}[u]$ we construct $\mathsf{sol}(u)$ using off-the-shelf algorithms [7]. We adapt the pipeline by tailoring the translation and the approximate analysis as outlined below.

Our implementation supports two methods for building the concrete CTMC for a parameter sample: (1) by first instantiating the valuation in the specification and then building the resulting concrete CTMC, or (2) by first building the pCTMC $\mathcal{M}$ (only once) and then instantiating it for each parameter sample to obtain the concrete CTMC $\mathcal{M}[u]$. Which method is faster depends on the specific model (we only report results for the fastest method in Sect. 6 for brevity).

Partial models. To accelerate the time-consuming computation of solution vectors by model-checking on large models, it is natural to abstract the models into smaller models amenable to faster computations. Similar to ideas used for dynamic fault trees [55] and infinite CTMCs [48], we employ an abstraction which only keeps the most relevant parts of a model, i.e., states with a sufficiently large probability to be reached from the initial state(s). Analysis on this partial model then yields best- and worst-case results for each measure by assuming that all removed states are either target states (best case) or are not (worst case), respectively. This method returns imprecise solution vectors as used in Sect. 4, which can be refined up to an arbitrary precision by retaining more states of the original model.

Similar to building the complete models, two approaches are possible to create the partial models: (1) fixing the valuation and directly abstracting the concrete CTMC, or (2) first building the complete pCTMC and then abstracting the concrete CTMC. We reuse partial models for similar valuations to avoid costly computations. We cluster parameter valuations which are close to each other (in Euclidean distance). For parameter valuations within one cluster, we reuse the same partial model (in terms of the states), albeit instantiating it according to the precise valuation.

6 Experiments

We answer three questions about (a prototype implementation of) our approach:

Q1. Can we verify CTMCs taking into account the uncertainty about the rates?
Q2. How well does our approach scale w.r.t. the number of measures and samples?
Q3. How does our approach compare to naïve baselines (to be defined below)?

Setup. We implement our approach using the explicit engine of Storm [37] and the improvements of Sect. 5 to sample from upCTMCs in Python. Our current

[6] Fault trees are a common formalism in reliability engineering [51].

Table 1. Excerpt of the benchmark statistics (sampling time is per 100 CTMCs).

Benchmark	$\lvert\Phi\rvert$	Model size			Storm run time [s]		Scen.opt. time [s]	
		#pars	#states	#trans	Init.	Sample (×100)	$N = 100$	$N = 200$
SIR (140)	26	2	9 996	19 716	0.29	2947.29	18.26	63.27
SIR (140)[a]	26	2	9 996	19 716	0.29	544.27	25.11	129.66
Kanban (3)	4	13	58 400	446 400	4.42	46.95	2.28	6.69
Kanban (5)	4	13	2 546 432	24 460 016	253.39	4363.63	2.03	5.94
Polling (9)	2	2	6 912	36 864	0.64	22.92	2.13	6.66
buffer	2	6	5 632	21 968	0.48	20.70	1.21	4.15
Tandem (31)	2	5	2 016	6 819	0.11	862.41	5.19	24.30
rbc	40	6	2 269	12 930	0.01	1.40	5.27	16.88
rc (1,1)	25	21	8 401	49 446	27.20	74.90	5.75	20.34
rc (1,1)[a]	25	21	n/a[b]	n/a[b]	0.02	2.35	29.23	150.61
rc (2,2)[a]	25	29	n/a[b]	n/a[b]	0.03	27.77	24.86	132.63
hecs (2,1)[a]	25	5	n/a[b]	n/a[b]	0.02	9.83	26.78	145.77
hecs (2,2)[a]	25	24	n/a[b]	n/a[b]	0.02	194.25	33.06	184.32

[a] Computed using approximate model checking up to a relative gap between upper bound $\mathsf{sol}^+(u)$ and lower bound $\mathsf{sol}^-(u)$ below 1% for every sample $u \in \mathcal{V}_{\mathcal{M}}$.
[b] Model size is unknown, as the approximation does not build the full state-space.

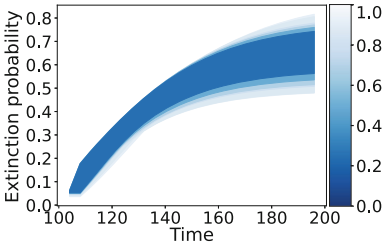


Fig. 10. Prediction regions for the SIR (60) benchmark with $n = 400$.

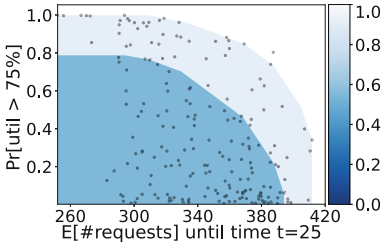


Fig. 11. Pareto front for the buffer benchmark with $n = 200$ samples.

implementation is limited to pCTMC instantiations that are *graph-preserving*, i.e. for any pair $s, s' \in S$ either $\mathbf{R}(s, s')[u] = 0$ or $\mathbf{R}(s, s')[u] > 0$ for all u. We solve optimization problems using the ECOS solver [29]. All experiments ran single-threaded on a computer with 32 3.7 GHz cores and 64 GB RAM. We show the effectiveness of our method on a large number of publicly available pCTMC [35] and fault tree benchmarks [50] across domains (details in [6, Appendix C]).

Q1. Applicability

An excerpt of the benchmark statistics is shown in Table 1 (see [6, Appendix C] for the full table). For all but the smallest benchmarks, sampling and computing the solution vectors by model checking is more expensive than solving the scenario problems. In the following, we illustrate that 100 samples are sufficient to provide qualitatively good prediction regions and associated lower bounds.

Table 2. Lower bounds $\bar{\mu}$ and standard deviation (SD), vs. the observed number of 1 000 additional solutions that indeed lie within the obtained regions.

(a) Kanban (3).

	$\beta = 0.9$		$\beta = 0.999$		Frequentist
n	$\bar{\mu}$	SD	$\bar{\mu}$	SD	Observed
100	0.862	0.000	0.798	0.000	959 ± 22.7
200	0.930	0.000	0.895	0.000	967 ± 17.4
400	0.965	0.001	0.947	0.001	984 ± 8.6
800	0.982	0.000	0.973	0.000	994 ± 3.2

(b) Railway crossing (1,1,hc).

	$\beta = 0.9$		$\beta = 0.999$		Frequentist
n	$\bar{\mu}$	SD	$\bar{\mu}$	SD	Observed
100	0.895	0.018	0.835	0.020	954 ± 26.8
200	0.945	0.007	0.912	0.008	980 ± 12.8
400	0.975	0.004	0.958	0.005	990 ± 8.3
800	0.986	0.002	0.977	0.003	995 ± 4.3

Plotting prediction regions. Figure 10 presents prediction regions on the extinction probability of the disease in the SIR model and is analogous to the tubes in Fig. 2d (see [6, Appendix C.1] for plots for various other benchmarks). These regions are obtained by applying our algorithm with varying values for the cost of relaxation ρ. For a confidence level of $\beta = 99\%$, the widest (smallest) tube in Fig. 10 corresponds to a lower bound probability of $\mu = 91.1\%$ ($\mu = 23.9\%$). Thus, we conclude that, with a confidence of at least 99%, the curve created by the CTMC for any sampled parameter value will lie within the outermost region in Fig. 10 with a probability of at least 91.1%. We highlight that our approach supports more general prediction regions. We show $n = 200$ solution vectors for the buffer benchmark with two measures in Fig. 11 and produce regions that approach the Pareto front. For a confidence level of $\beta = 99\%$, the outer prediction region is associated with a lower bound probability of $\mu = 91.1\%$, while the inner region has a lower value of $\mu = 66.2\%$. We present more plots in [6, Appendix C.1].

Tightness of the solution. In Table 2 we investigate the tightness of our results. For the experiment, we set $\rho = 1.1$ and solve $\mathfrak{L}^{\rho}_{\mathcal{U}}$ for different values of n, repeating every experiment 10 times, resulting in the average bounds $\bar{\mu}$. Then, we sample 1 000 solutions and count the *observed* number of solutions contained in every prediction regions, resulting in an empirical approximation of the containment probability. Recall that for $\rho > 1$, we obtain a prediction region that contains all solutions, so this observed count grows toward n. The lower bounds grow toward the empirical count for an increased n, with the smallest difference (RC, $n = 800$, $\beta = 0.9$) being as small as 0.9%. Similar observations hold for other values of ρ.

Handling imprecise solutions. The approximate model checker is significantly faster (see Table 1 for SIR (140) and RC), at the cost of obtaining imprecise solution vectors.[7] For SIR (140), the sampling time is reduced from 49 to 9 min, while the scenario optimization time is slightly higher at 129 s. This difference only grows larger with the size of the CTMC. For the larger instances of RC and HECS, computing exact solutions is infeasible at all (one HECS (2,2) sample alone takes 15 min). While the bounds on the containment probability under imprecise solu-

[7] We terminate at a relative gap between upper/lower bound of the solution below 1%.

Table 3. Run times in [s] for solving the scenario problems for SIR and RC with $\rho = 0.1$ (timeout (TO) of 1 hour) for different sample sizes n and measures m.

(a) SIR (population 20).

n / m	50	100	200	400	800
100	0.97	1.59	3.36	9.17	25.41
200	3.69	7.30	22.91	59.45	131.78
400	29.43	76.13	153.03	310.67	640.70
800	261.97	491.73	955.77	1924.15	TO

(b) Railway crossing (1,1,hc).

n / m	50	100	200	400
100	1.84	3.40	8.18	24.14
200	6.35	14.56	45.09	113.09
400	34.74	96.68	203.77	427.80
800	292.32	579.09	1215.67	2553.98

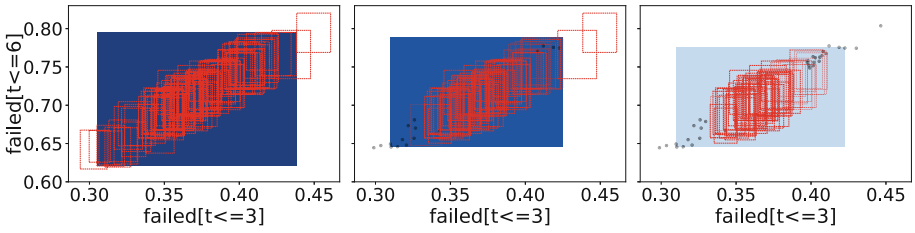
tions may initially be poor (see Fig. 12a, which results in $\mu = 2.1\%$), we can improve the results significantly using the refinement scheme proposed in Sect. 4.3. For example, Fig. 12c shows the prediction region after refining 31 of the 100 solutions, which yields $\mu = 74.7\%$. Thus, *by iteratively refining only the imprecise solutions on the boundary of the resulting prediction regions, we significantly tighten the obtained bounds on the containment probability.*

Q2. Scalability

In Table 3, we report the run times for steps (3)–(5) of our algorithm shown in Fig. 7 (i.e., for solving the scenario problems, but not for computing the solution vectors in Storm). Here, we solve problem $\mathfrak{L}^{\rho}_{\mathcal{U}}$ for $\rho = 0.1$, with different numbers of samples and measures. Our approach scales well to realistic numbers of samples (up to 800) and measures (up to 400). The computational complexity of the scenario problems is largely *independent of the size of the CTMC*, and hence, similar run times are observed across the benchmarks (cf. Table 1).

Q3. Comparison to baselines

We compare against two baselines: (1) Scenario optimization to analyze each measure independently, yielding a separate probabilistic guarantee on each measure. (2) A frequentist (Monte Carlo) baseline, which samples a large number of parameter values and counts the number of associated solutions within a region.



(a) No solutions refined. (b) Intermediate step. (c) 31 refined solutions.

Fig. 12. Refining imprecise solution vectors (red boxes) for RC (2,2), $n = 100$. (Color figure online)

Analyzing measures independently. To show that analyzing a full *set of measures* at once, e.g., the complete probability curve, is essential, we compare our method to the baseline that analyzes *each measure independently* and combines the obtained bounds on each measure afterward. We consider the PCS benchmark with precise samples and solve $\mathfrak{L}^{\rho}_{\mathcal{U}}$ for $\rho = 2$ (see [6, Table 5] for details). For $n = 100$ samples and $\beta = 99\%$, our approach returns a lower bound probability of $\mu = 84.8\%$. By contrast, the naïve baseline yields a lower bound of only 4.5%, and similar results are observed for different values of n (cf. [6, Table 5 in Appendix C]). There are two reasons for this large difference. First, the baseline applies Theorem 3 once for each of the 25 measures, so it must use a more conservative confidence level of $\tilde{\beta} = 1 - \frac{1-\beta}{25} = 0.9996$. Second, the baseline takes the conjunction over the 25 independent lower bounds, which drastically reduces the obtained bound.

Frequentist baseline. The comparison to the frequentist baseline on the Kanban and RC benchmarks yields the previously discussed results in Table 2. The results in Tables 1 and 3 show that *the time spent for sampling is (for most benchmarks) significantly higher than for scenario optimization.* Thus, our scenario-based approach has a relatively low cost, while resulting in valuable guarantees which the baseline does not give. To still obtain a high confidence in the result, a much larger sample size is needed for the frequentist baseline than for our approach.

7 Related Work

Several verification approaches exist to handle uncertain Markov models.

For (discrete-time) *interval* Markov chains (DTMCs) or Markov decision processes (MDPs), a number of approaches verify against all probabilities within the intervals [32,39,46,53,54]. Lumpability of interval CTMCs is considered in [22]. In contrast to upCTMCs, interval Markov chains have no dependencies between transition uncertainties and no distributions are attached to the intervals.

Parametric Markov models generally define probabilities or rates via functions over the parameters. The standard parameter synthesis problem for discrete-time models is to find all valuations of parameters that satisfies a specification. Techniques range from computing a solution function over the parameters, to directly solving the underlying optimization problems [24,28,33,40]. Parametric CTMCs are investigated in [23,34], but are generally restricted to a few parameters. The work [15] aims to find a robust parameter valuation in pCTMCs.

For all approaches listed so far, the results may be rather conservative, as no prior information on the uncertainties (the intervals) is used. That is, the uncertainty is not quantified and all probabilities or rates are treated equally as likely. In our approach, we do not compute solution functions, as the underlying methods are computationally expensive and usually restricted to a few parameters.

Quantified uncertainty is studied in [44]. Similarly to our work, the approach draws parameter values from a probability distribution over the model parameters and analyzes the instantiated model via model checking. However, [44] studies DTMCs and performs a frequentist (Monte Carlo) approach, cf. Sect. 6,

to compute estimates for a single measure, without prediction regions. Moreover, our approach requires significantly fewer samples, cf. the comparison in Sect. 6.

The work in [10,11] takes a sampling-driven Bayesian approach for pCTMCs. In particular, they take a prior on the solution function over a single measure and update it based on samples (potentially obtained via statistical model checking). We assume no prior on the solution function, and, as mentioned before, do not compute the solution function due to the expensive underlying computations.

Statistical model checking (SMC) [1,43] samples path in stochastic models to perform model checking. This technique has been applied to numerous models [25–27,47], including CTMCs [52,57]. SMC analyzes a *concrete* CTMC by sampling from the known transition rates, whereas for upCTMC these rates are parametric.

Finally, scenario optimization [16,21] is widely used in control theory [14] and recently in machine learning [20] and reliability engineering [49]. Within a verification context, closest to our work is [5], which considers the verification of single measures for uncertain MDPs. [5] relies on the so-called sampling-and-discarding approach [17], while we use the risk-and-complexity perspective [31], yielding better results for problems with many decision variables like we have.

8 Conclusion

This paper presents a novel approach to the analysis of parametric Markov models with respect to a set of performance characteristics. In particular, we provide a method that yields statistical guarantees on the typical performance characteristics from a finite set of samples of those parameters. Our experiments show that high-confidence results can be given based on a few hundred of samples. Future work includes supporting models with nondeterminism, exploiting aspects of parametric models such as monotonicity, and integrating methods to infer the distributions on the parameter space from observations.

References

1. Agha, G., Palmskog, K.: A survey of statistical model checking. ACM Trans. Model. Comput. Simul. **28**(1), 6:1–6:39 (2018)
2. Allen, L.J.: A primer on stochastic epidemic models: Formulation, numerical simulation, and analysis. Infect. Dis. Model. **2**(2), 128–142 (2017)
3. Andersson, H., Britton, T.: Stochastic Epidemic Models and Their Statistical Analysis, vol. 151. Springer Science & Business Media, New York (2012). https://doi.org/10.1007/978-1-4612-1158-7
4. Aziz, A., Sanwal, K., Singhal, V., Brayton, R.: Model-checking continuous-time Markov chains. ACM Trans. Comput. Logic **1**(1), 162–170 (2000)
5. Badings, T.S., Cubuktepe, M., Jansen, N., Junges, S., Katoen, J.P., Topcu, U.: Scenario-based verification of uncertain parametric MDPs. CoRR **abs/2112.13020** (2021)
6. Badings, T.S., Jansen, N., Junges, S., Stoelinga, M., Volk, M.: Sampling-based verification of CTMCs with uncertain rates. Technical report, CoRR, abs/2205.08300 (2022)

7. Baier, C., Haverkort, B.R., Hermanns, H., Katoen, J.P.: Model-checking algorithms for continuous-time Markov chains. IEEE Trans. Softw. Eng. **29**(6), 524–541 (2003)
8. Baier, C., Katoen, J.P.: Principles of Model Checking. MIT Press, Cambridge (2008)
9. Bertsekas, D.P., Tsitsiklis, J.N.: Introduction to Probability. Athena Scientinis (2000)
10. Bortolussi, L., Milios, D., Sanguinetti, G.: Smoothed model checking for uncertain continuous-time Markov chains. Inf. Comput. **247**, 235–253 (2016)
11. Bortolussi, L., Silvetti, S.: Bayesian statistical parameter synthesis for linear temporal properties of stochastic models. In: Beyer, D., Huisman, M. (eds.) TACAS 2018. LNCS, vol. 10806, pp. 396–413. Springer, Cham (2018). https://doi.org/10.1007/978-3-319-89963-3_23
12. Boyd, S., Vandenberghe, L.: Convex Optimization. Cambridge University Press, New York (2004)
13. Budde, C.E., Dehnert, C., Hahn, E.M., Hartmanns, A., Junges, S., Turrini, A.: JANI: quantitative model and tool interaction. In: Legay, A., Margaria, T. (eds.) TACAS 2017. LNCS, vol. 10206, pp. 151–168. Springer, Heidelberg (2017). https://doi.org/10.1007/978-3-662-54580-5_9
14. Calafiore, G.C., Campi, M.C.: The scenario approach to robust control design. IEEE Trans. Autom. Control. **51**(5), 742–753 (2006)
15. Calinescu, R., Ceska, M., Gerasimou, S., Kwiatkowska, M., Paoletti, N.: Efficient synthesis of robust models for stochastic systems. J. Syst. Softw. **143**, 140–158 (2018)
16. Campi, M.C., Garatti, S.: The exact feasibility of randomized solutions of uncertain convex programs. SIAM J. Optim. **19**(3), 1211–1230 (2008)
17. Campi, M.C., Garatti, S.: A sampling-and-discarding approach to chance-constrained optimization: feasibility and optimality. J. Optim. Theory App. **148**(2), 257–280 (2011)
18. Campi, M.C., Garatti, S.: Introduction to the scenario approach. SIAM (2018)
19. Campi, M.C., Garatti, S.: Wait-and-judge scenario optimization. Math. Program. **167**(1), 155–189 (2018)
20. Campi, M.C., Garatti, S.: Scenario optimization with relaxation: a new tool for design and application to machine learning problems. In: CDC, pp. 2463–2468. IEEE (2020)
21. Campi, M., Carè, A., Garatti, S.: The scenario approach: a tool at the service of data-driven decision making. Ann. Rev. Control **52**, 1–17 (2021)
22. Cardelli, L., Grosu, R., Larsen, K.G., Tribastone, M., Tschaikowski, M., Vandin, A.: Lumpability for uncertain continuous-time Markov chains. In: Abate, A., Marin, A. (eds.) QEST 2021. LNCS, vol. 12846, pp. 391–409. Springer, Cham (2021). https://doi.org/10.1007/978-3-030-85172-9_21
23. Ceska, M., Dannenberg, F., Paoletti, N., Kwiatkowska, M., Brim, L.: Precise parameter synthesis for stochastic biochemical systems. Acta Inform. **54**(6), 589–623 (2017)
24. Cubuktepe, M., Jansen, N., Junges, S., Katoen, J.P., Topcu, U.: Convex optimization for parameter synthesis in MDPs. IEEE Trans Autom Control pp. 1–1 (2022)
25. D'Argenio, P.R., Hartmanns, A., Sedwards, S.: Lightweight statistical model checking in nondeterministic continuous time. In: Margaria, T., Steffen, B. (eds.) ISoLA 2018. LNCS, vol. 11245, pp. 336–353. Springer, Cham (2018). https://doi.org/10.1007/978-3-030-03421-4_22
26. David, A., Larsen, K.G., Legay, A., Mikucionis, M., Poulsen, D.B.: UPPAAL SMC tutorial. Int. J. Softw. Tools Technol. Transf. **17**(4), 397–415 (2015)

27. David, A., Larsen, K.G., Legay, A., Mikučionis, M., Wang, Z.: Time for statistical model checking of real-time systems. In: Gopalakrishnan, G., Qadeer, S. (eds.) CAV 2011. LNCS, vol. 6806, pp. 349–355. Springer, Heidelberg (2011). https://doi.org/10.1007/978-3-642-22110-1_27
28. Daws, C.: Symbolic and parametric model checking of discrete-time Markov chains. In: Liu, Z., Araki, K. (eds.) ICTAC 2004. LNCS, vol. 3407, pp. 280–294. Springer, Heidelberg (2005). https://doi.org/10.1007/978-3-540-31862-0_21
29. Domahidi, A., Chu, E., Boyd, S.P.: ECOS: an SOCP solver for embedded systems. In: ECC, pp. 3071–3076. IEEE (2013)
30. Garatti, S., Campi, M.C.: The risk of making decisions from data through the lens of the scenario approach. IFAC-PapersOnLine **54**(7), 607–612 (2021)
31. Garatti, S., Campi, M.: Risk and complexity in scenario optimization. Math. Program. **191**, 1–37 (2019)
32. Givan, R., Leach, S.M., Dean, T.L.: Bounded-parameter Markov decision processes. Artif. Intell. **122**(1–2), 71–109 (2000)
33. Hahn, E.M., Hermanns, H., Zhang, L.: Probabilistic reachability for parametric Markov models. Int. J. Softw. Tools Technol. Transf. **13**(1), 3–19 (2011)
34. Han, T., Katoen, J.P., Mereacre, A.: Approximate parameter synthesis for probabilistic time-bounded reachability. In: RTSS, pp. 173–182. IEEE CS (2008)
35. Hartmanns, A., Klauck, M., Parker, D., Quatmann, T., Ruijters, E.: The quantitative verification benchmark set. In: Vojnar, T., Zhang, L. (eds.) TACAS 2019. LNCS, vol. 11427, pp. 344–350. Springer, Cham (2019). https://doi.org/10.1007/978-3-030-17462-0_20
36. Haverkort, B.R., Hermanns, H., Katoen, J.P.: On the use of model checking techniques for dependability evaluation. In: SRDS, pp. 228–237. IEEE CS (2000)
37. Hensel, C., Junges, S., Katoen, J.P., Quatmann, T., Volk, M.: The probabilistic model checker Storm. Softw. Tools Technol. Transf. (2021)
38. Hermanns, H., Meyer-Kayser, J., Siegle, M.: Multi terminal binary decision diagrams to represent and analyse continuous time Markov chains. In: 3rd International Workshop on the Numerical Solution of Markov Chains, pp. 188–207. Citeseer (1999)
39. Jonsson, B., Larsen, K.G.: Specification and refinement of probabilistic processes. In: LICS, pp. 266–277. IEEE CS (1991)
40. Junges, S., et al.: Parameter synthesis for Markov models. CoRR **abs/1903.07993** (2019)
41. Katoen, J.P.: The probabilistic model checking landscape. In: LICS, pp. 31–45. ACM (2016)
42. Kwiatkowska, M., Norman, G., Parker, D.: PRISM 4.0: verification of probabilistic real-time systems. In: Gopalakrishnan, G., Qadeer, S. (eds.) CAV 2011. LNCS, vol. 6806, pp. 585–591. Springer, Heidelberg (2011). https://doi.org/10.1007/978-3-642-22110-1_47
43. Legay, A., Lukina, A., Traonouez, L.M., Yang, J., Smolka, S.A., Grosu, R.: Statistical Model Checking. In: Steffen, B., Woeginger, G. (eds.) Computing and Software Science. LNCS, vol. 10000, pp. 478–504. Springer, Cham (2019). https://doi.org/10.1007/978-3-319-91908-9_23
44. Meedeniya, I., Moser, I., Aleti, A., Grunske, L.: Evaluating probabilistic models with uncertain model parameters. Softw. Syst. Model. **13**(4), 1395–1415 (2014)
45. Mendelson, B.: Introduction to topology. Courier Corporation (1990)
46. Puggelli, A., Li, W., Sangiovanni-Vincentelli, A.L., Seshia, S.A.: Polynomial-time verification of PCTL properties of MDPs with convex uncertainties. In: CAV. LNCS, vol. 8044, pp. 527–542. Springer (2013)

47. Rao, K.D., Gopika, V., Rao, V.V.S.S., Kushwaha, H.S., Verma, A.K., Srividya, A.: Dynamic fault tree analysis using Monte Carlo simulation in probabilistic safety assessment. Reliab. Eng. Syst. Saf. **94**(4), 872–883 (2009)
48. Roberts, R., Neupane, T., Buecherl, L., Myers, C.J., Zhang, Z.: STAMINA 2.0: improving scalability of infinite-state stochastic model checking. In: Finkbeiner, B., Wies, T. (eds.) VMCAI 2022. LNCS, vol. 13182, pp. 319–331. Springer, Cham (2022). https://doi.org/10.1007/978-3-030-94583-1_16
49. Rocchetta, R., Crespo, L.G.: A scenario optimization approach to reliability-based and risk-based design: soft-constrained modulation of failure probability bounds. Reliab. Eng. Syst. Saf. **216**, 107900 (2021)
50. Ruijters, E., et al.: FFORT: a benchmark suite for fault tree analysis. In: ESREL (2019)
51. Ruijters, E., Stoelinga, M.I.A.: Fault tree analysis: a survey of the state-of-the-art in modeling, analysis and tools. Comput. Sci. Rev. **15**, 29–62 (2015)
52. Sen, K., Viswanathan, M., Agha, G.: On statistical model checking of stochastic systems. In: Etessami, K., Rajamani, S.K. (eds.) CAV 2005. LNCS, vol. 3576, pp. 266–280. Springer, Heidelberg (2005). https://doi.org/10.1007/11513988_26
53. Sen, K., Viswanathan, M., Agha, G.: Model-checking Markov chains in the presence of uncertainties. In: Hermanns, H., Palsberg, J. (eds.) TACAS 2006. LNCS, vol. 3920, pp. 394–410. Springer, Heidelberg (2006). https://doi.org/10.1007/11691372_26
54. Skulj, D.: Discrete time Markov chains with interval probabilities. Int. J. Approx. Reason. **50**(8), 1314–1329 (2009)
55. Volk, M., Junges, S., Katoen, J.P.: Fast dynamic fault tree analysis by model checking techniques. IEEE Trans. Ind. Inform. **14**(1), 370–379 (2018)
56. Wijesuriya, V.B., Abate, A.: Bayes-adaptive planning for data-efficient verification of uncertain Markov decision processes. In: Parker, D., Wolf, V. (eds.) QEST 2019. LNCS, vol. 11785, pp. 91–108. Springer, Cham (2019). https://doi.org/10.1007/978-3-030-30281-8_6
57. Younes, H.L.S., Simmons, R.G.: Statistical probabilistic model checking with a focus on time-bounded properties. Inf. Comput. **204**(9), 1368–1409 (2006)