Statistical model checking for parameterized models

Benoit Delahaye, Paulin Fournier, Didier Lime

Statistical model checking for parameterized models

Benoît Delahaye[1], Paulin Fournier[1], and Didier Lime[2]

[1] Université de Nantes - LS2N UMR CNRS 6004, Nantes, France
[2] École Centrale Nantes - LS2N UMR CNRS 6004, Nantes, France

Abstract. We propose a simulation-based technique, in the spirit of Statistical Model Checking, for approximate verification of probabilistic models with parametric transitions, and we focus in particular on parametric Markov chains. Our technique is based on an extension of Monte Carlo algorithms that allows to approximate the probability of satisfying a given finite trace property as a (polynomial) function of the parameters of the model. The confidence intervals associated with this approximation can also be expressed as a function of the parameters. In the paper, we present both the theoretical foundations of this technique and a prototype implementation in Python which we evaluate on a set of benchmarks.

1 Introduction

Nowadays, modelling and abstracting are widely accepted as crucial steps in the understanding and study of real-life systems. In many cases, it is necessary to incorporate probabilities in the models to cope with uncertainty, to abstract complex behaviour, or to introduce randomness. Markov chains and Markov decision processes, in particular, have been widely studied.

Statistical model checking. Though exact methods are known for such models they usually require solving huge equation systems, and therefore have scalability issues with the biggest models. A way to avoid this complexity is to consider approximation techniques through simulation. In particular, Monte Carlo simulation techniques allow to infer the real behaviour of the system via independent simulations up to a computable precision.

Parametric Markov Chains. The values given to probabilistic transitions can have a huge impact on the behaviour of the system. In the early stages of development, it may therefore be useful to have an insight on how the values of transition probabilities affect the system in order to be able to set the best value in terms of convergence speed for example. To this purpose, parametric Markov chains have been introduced in [1]. They allow to replace the probability values given to transitions by parameter variables, and therefore to be able to give guarantees on the system for all possible values of the parameters.

Results. The aim of this paper is to apply Monte Carlo simulation to parametric Markov chains in order to approximate the probability of the considered property as a polynomial function of the parameters. In addition, we also derive a confidence interval on the obtained probabilities as a polynomial function of the parameters. Aside from using parameterized models, which comes in particular with better flexibility in the modelling, robustness of the results, and usability at the earliest stages of conception, the expected benefits of our new approach are largely those of such simulation techniques for non-parameterized Markov chains: better scalability through a reduced memory footprint, a complexity that is largely independent of the model complexity (be it in terms of size of the state-space, or of features used as long as they are executable). More specific to our approach, since we derive polynomial function approximations, where exact methods lead to rational functions, these results should be easier to post-process. Finally the complexity of our approach is largely independent of the number of parameters.

In order to experimentally confirm the interest of our approach we have implemented it in a (fairly crude) prototype in Python, and we report very encouraging results on case-studies from the literature.

Related work. Model checking and parameter synthesis for parametric Markov chains have been widely studied in the last decade [2–5]. To the best of our knowledge, all existing works on this topic focus on exact techniques which either produce constraints on the parameter values [4, 5] or compute the probability of satisfying a given property as a rational function of the parameters [2, 3]. While these techniques have the advantage of precision, they only scale to models having few parameters. We conjecture that the approximation technique we propose in this paper will be advantageous in the context on models with a larger number of parameters because it allows to produce polynomial instead of rational functions.

On the other hand, Statistical Model Checking has, to the best of our knowledge, never been applied to parametric models as such. The closer existing techniques are reinforcement learning algorithms combined with Statistical Model Checking, which have been applied in the context of non-deterministic and probabilistic models such as Markov decision processes [6] (where non-determinism could be replaced with parameters). While these techniques allow to compute the best (or worst) probability of satisfying a given property, they do not provide error precision or confidence intervals. Moreover, since they only compute an approximation of the best (or worst) probability, they do not provide a complete analysis of the effect of non-deterministic choice (or parametric transitions) on the satisfaction of the given property. This is something which is easily provided using our technique.

Organization of the paper. In Section 2, we introduce the basic definitions related to the problem we study. Section 3 then introduces the parametric setting, and our main theoretical contribution. We then report on our prototype implementation and its use on several examples in Section 4. We give some leads for

the improvement of our approach and prototype in Section 5 and, finally, we conclude in Section 6.

2 Definitions

As usual, the set of real numbers and the set of natural numbers are respectively written $\mathbb{R}$ and $\mathbb{N}$. Given two real numbers $a < b$, the closed, semi-open and open intervals representing all real values between a and b are respectively written $[a, b]$, $(a, b]$, $[a, b)$ and (a, b).

Definition 1 (Markov chain). *A* Markov chain *(MC, for short) is a tuple* $\mathcal{M} = (S, s_0, P)$ *where* S *is a denumerable set of states,* $s_0 \in S$ *is the initial state and* $P : S \times S \to [0, 1]$ *is the transition probability function such that for all state* $s \in S$, $\sum_{s' \in S} P(s, s') = 1$.

A run of a Markov chain is a sequence of states $s_0 s_1 \dots$ such that for all i, $P(s_i, s_{i+1}) > 0$. Given a finite run $\rho = s_0 s_1 \dots s_n$, its length, written $|\rho|$ represents the number of transitions it goes through (including repetitions). Here $|\rho| = n$. We write $\Gamma_{\mathcal{M}}(l)$ (or simply $\Gamma(l)$ when $\mathcal{M}$ is clear from the context) for the set of all finite runs of length l, and $\Gamma_{\mathcal{M}}$ for the set of all finite runs *i.e.* $\Gamma_{\mathcal{M}} = \cup_{l \in \mathbb{N}} \Gamma_{\mathcal{M}}(l)$. As usual we define the probability measure, written $\mathbb{P}_{\mathcal{M}}$, on runs based on the sigma-algebra of cylinders (see *e.g.* [7]). This gives us that for any finite run $\rho = s_0 s_1 \dots s_n$, $\mathbb{P}_{\mathcal{M}}(\rho) = \Pi_{i=1}^{n} P(s_{i-1}, s_i)$. In the rest of the paper, we only consider *finite* runs.

Example 1 (Examples of properties). In this paper, we consider properties on bounded runs and we aim at computing approximations for the following values:

Reachability $\mathbb{P}_{\mathcal{M}}(\Diamond^{\leq l} s)$**.** A run $\rho = s_0 s_1 \dots$ is said to reach a state s in less than l steps, written $\rho \models \Diamond^{\leq l} s$, if there exists $i \leq l$ such that $s_i = s$.

Safety $\mathbb{P}_{\mathcal{M}}(\Box^{=l} E)$**.** A run $\rho = s_0 s_1 \dots$ is said to be safe for a set of states $E \subseteq S$ during l steps, written $\rho \models \Box^{=l} E$, if for all $i \leq l$, $s_i \in E$.

Expected reward $\mathbb{E}^{l}_{\mathcal{M}}(r)$**.** Given a reward function $r : \Gamma(l) \to \mathbb{R}$ we write $\mathbb{E}^{l}_{\mathcal{M}}(r) = \sum_{\rho \in \Gamma(l)} \mathbb{P}_{\mathcal{M}}(\rho) r(\rho)$ for the expected value of r on the runs of length l.

Notice that for any property $\varphi \subseteq \Gamma(l)$, $\mathbb{P}_{\mathcal{M}}(\varphi) = \mathbb{E}^{l}_{\mathcal{M}}(\mathbb{1}_{\varphi})$ where $\mathbb{1}_{\varphi}$ is the reward function such that $\mathbb{1}_{\varphi}(\rho) = 1$ if $\rho \in \varphi$ and 0 otherwise. In the following of the paper we will thus only consider properties of the form $\mathbb{E}^{l}_{\mathcal{M}}(r)$.

Given two Markov chains $\mathcal{M}^1 = (S^1, s^1_0, P^1)$ and $\mathcal{M}^2 = (S^2, s^2_0, P^2)$ we say that $\mathcal{M}^1$ and $\mathcal{M}^2$ have the same structure if $(S^1, s^1_0) = (S^2, s^2_0)$ and for all state $s, s' \in S^1$, $P^1(s, s') > 0$ if and only if $P^2(s, s') > 0$.

3 Approximation in parametric Markov chains

We now move to our main contribution: a simulation-based method for approximate verification of parametric Markov chains based on Monte Carlo. We start by recalling the vocabulary and main definitions in the context of parametric Markov chains.

3.1 Parametric Markov chains

Given a finite set of parameters $\mathbb{X}$ we write $Poly(\mathbb{X})$ for the set of all real (multivariate) polynomials on $\mathbb{X}$. Given a parameter valuation $v \in \mathbb{R}^{\mathbb{X}}$ and a polynomial $f \in Poly(\mathbb{X})$, the evaluation of f under valuation v is written $f(v)$.

Definition 2 (Parametric Markov chain). *A* Parametric Markov chain *(written pMC for short) is a tuple $\mathcal{M} = (S, s_0, P, \mathbb{X})$ such that S is a finite set of states, $s_0 \in S$ is the initial state, $\mathbb{X}$ is a finite set of parameters, and $P : S \times S \to Poly(\mathbb{X})$ is a parametric transition probability function.*

Remark 1 (Rational function). Notice that in the definition of pMC given above, we restrict ourselves to real (multivariate) polynomials on $\mathbb{X}$. However, all our results could naturally be extended to rational functions instead, *i.e.* functions of the form d/q where $d, q \in Poly(\mathbb{X})$.

Let $\mathcal{M}$ be a pMC and $v \in \mathbb{R}^{\mathbb{X}}$ be a valuation of the parameters of $\mathcal{M}$. Let P_v be the transition probability function obtained under valuation v, *i.e.* $P_v(s, s') = P(s, s')(v)$ for all $s, s' \in S$. We say that v is a *valid parameter valuation* with respect to $\mathcal{M}$ if the tuple (S, s_0, P_v) is a Markov chain, *i.e.* v defines valid probability distributions for the transitions of $\mathcal{M}$. If v is a valid parameter valuation w.r.t $\mathcal{M}$, the resulting Markov chain is written $\mathcal{M}^v$.

Remark 2 (Consistency). Notice that one can obtain the set of all valid parameter valuations as the result of a set of constraints stating that each transition has a probability between 0 and 1 and that the sum of outgoing transition probabilities is 1 for all states. The problem asking whether a pMC admits valid parameter valuations, and computing them is called the *consistency problem*. In this paper, we do not address the consistency problem further and refer instead the interested reader to [4] where this problem is addressed in the more general context of parametric interval Markov chains.

Given a pMC $\mathcal{M}$, a run ρ of $\mathcal{M}$ is a sequence of states $s_0 s_1 \dots$ such that for all $i \geq 0$, $P(s_i, s_{i+1}) \neq 0$ (*i.e.* the probability is either a strictly positive real constant or a function of the parameters). As for Markov chains we write $\Gamma_{\mathcal{M}}(l)$ the set of all finite runs of length l and $\Gamma_{\mathcal{M}}$ the set of all finite runs.

Observe that for any valid parameter valuation v, $\Gamma_{\mathcal{M}^v}(l) \subseteq \Gamma_{\mathcal{M}}(l)$ since v may assign 0 to some transition probabilities.

Example 2. A pMC $\mathcal{M}_1 = (S, s_0, P, \mathbb{X})$ is given as an example in Figure 1. In this figure, we have $S = \{0 \dots 4\}$, $s_0 = 0$, and $\mathbb{X} = \{p, q, r\}$. As depicted, some of the transitions are parametric. For instance, $P(1, 0) = p$. Let v be the parameter valuation such that $v(p) = v(q) = 0.5$ and $v(r) = 0$. According to the definition above, v is a valid parameter valuation for $\mathcal{M}_1$. Indeed, under this parameter valuation, all the transitions have values between 0 and 1 and the probabilities of the outgoing transitions of all states sum up to 1. The resulting Markov chain $\mathcal{M}_1^v$ is given in Figure 2.

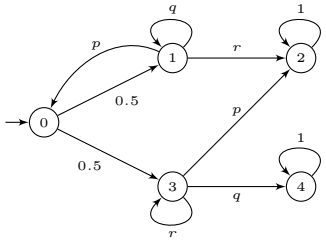


Fig. 1: pMC $\mathcal{M}_1$

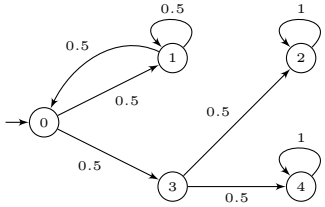


Fig. 2: MC $\mathcal{M}_1^v$ for parameter valuation v such that $v(p) = v(q) = 0.5$ and $v(r) = 0$

An example of run in $\mathcal{M}_1$ is $\rho = 0, 1, 2, 2$. The length of ρ is $|\rho| = 3$. As explained above, remark that ρ is not a run of $\mathcal{M}_1^v$ because the probability of the transition going from 1 to 2 has been set to 0 due to the parameter valuation v. On the other hand, all runs of $\mathcal{M}_1^v$ are also runs of $\mathcal{M}$.

3.2 Monte Carlo for parametric Markov chains

We start by recalling the central limit theorem, which is at the heart of our approach.

Theorem 1 (The central limit theorem (see *e.g.* [8])). *Let $X_1, X_2, \ldots$ be a sequence of independent and identically distributed random variables, each having mean γ and variance σ^2. Then the distribution of*

$$\frac{X_1 + \cdots + X_n - n\gamma}{\sigma\sqrt{n}}$$

tends to the standard normal distribution as $n \to \infty$. That is, for $-\infty < a < \infty$,

$$\lim_{n\to\infty} \mathbb{P}\left(\frac{X_1 + \cdots + X_n - n\gamma}{\sigma\sqrt{n}} \leq a\right) = \frac{1}{\sqrt{2\pi}} \int_{-\infty}^{a} e^{-x^2/2} dx.$$

The aim of this paper is to propose a statistical verification method, based on Monte Carlo, for approximating the expected value of a given reward function r on the runs $\Gamma(l)$ of a given pMC $\mathcal{M}$. In order to provide some intuition, we briefly recall how standard Monte Carlo analysis works in the context of statistical model checking of Markov chains. In this context, a set of n samples of the runs of the MC is produced. Each of these samples is evaluated, yielding a value 1 if it satisfies the desired property and 0 otherwise. According to the central limit theorem, the mean value of the samples provides a good estimator of the probability that a random run satisfies the desired property. Moreover, the central limit theorem provides a confidence interval that only depends on the number of samples (provided this number is large enough).

Unfortunately, as the transition probabilities are not known a priori in the context of pMCs, this technique cannot be applied directly (since we cannot produce samples according to the parametric transition probabilities). The intuition of the method we propose here is to fix the transition probabilities to an arbitrary function f, which we call *normalization function*, and to use these transition probabilities in order to produce samples of the pMC $\mathcal{M}$. However, instead of evaluating the obtained runs by directly using the desired reward function r, we define a new (parametric) reward function r' that takes into account the parametric transition probabilities. We show that, under any parameter valuation v, the evaluation of the mean value of r' on the set of samples is a good estimator for the expected value of the reward r on $\mathcal{M}^v$. The central limit theorem also allows to produce parametric confidence intervals. Contrary to standard Monte Carlo applied to Markov chains, the precision of the obtained approximation not only depends on the size of the sample, but also on the choice of parameter valuation as well as on the chosen normalization function. These issues are further discussed in Section 5.

In the following, we introduce some notations, then move to the random variable corresponding to our new parametric reward function, and finally show the correctness of our approach. Along the rest of the section, we consider a pMC $\mathcal{M} = (S, s_0, P, \mathbb{X})$ and a reward function $r : \Gamma_{\mathcal{M}} \to \mathbb{R}$.

Given a function $f : S \times S \to [0, 1]$ we say that f is valid w.r.t $\mathcal{M}$ if for all states s, $\sum_{s' \in S} f(s, s') = 1$. Given a valid function w.r.t $\mathcal{M}$, let $\mathcal{M}^f$ be the MC obtained from $\mathcal{M}$ by replacing P by f. Examples of such valid functions include the evaluation of the parametric transition probability by a valid valuation, or the function $u_{\mathcal{M}}$ such that, for each state, the distribution on its successors is uniform for all successors allowed in $\mathcal{M}$ (it respects the structure of $\mathcal{M}$). In the following, f is called a *normalization function* and, in particular, $u_{\mathcal{M}}$ is called the *uniform* normalization function. The choice of a good normalization function is discussed in Section 5.

Let $Pa : \Gamma_{\mathcal{M}} \to Poly(\mathbb{X})$ be a parametric reward function defined inductively as follows: $Pa(s_0) = 1$ and for all run $\rho ss' \in \Gamma_{\mathcal{M}}$, $Pa(\rho ss') = Pa(\rho s)P(s, s')$. Note that Pa can be seen as the counterpart of $\mathbb{P}$ for parametric Markov chains. Indeed for any valid valuation v and any run $\rho \in \Gamma_{\mathcal{M}^v}$ we have

$$\mathbb{P}_{\mathcal{M}^v}(\rho) = Pa(\rho)(v). \tag{1}$$

We now define the parametric reward function r' that will allow us to estimate the expectation of r. Given any valid normalization function f and any run $\rho \in \Gamma_{\mathcal{M}}$, let r' be such that

$$r'(\rho) = \frac{Pa(\rho)}{\mathbb{P}_{\mathcal{M}^f}(\rho)} r(\rho).$$

We now prove our main result. Let $\rho \in \Gamma_{\mathcal{M}^f}(l)$ be a random sample of $\mathcal{M}^f$ and let Y be the random variable defined as follows $Y = r'(\rho) = \frac{Pa(\rho)}{\mathbb{P}_{\mathcal{M}^f}(\rho)} r(\rho)$.

The following computation shows that, under any valid parameter valuation v such that $\mathcal{M}^f$ and $\mathcal{M}^v$ have the same structure, we have $\mathbb{E}(Y)(v) = \mathbb{E}^l_{\mathcal{M}^v}(r)$.

$$\mathbb{E}(Y)(v) = \left(\sum_{\rho \in \Gamma_{\mathcal{M}^f}(l)} \mathbb{P}_{\mathcal{M}^f}(\rho) y(\rho) \right)(v) \tag{2}$$

$$= \left(\sum_{\rho \in \Gamma_{\mathcal{M}^f}(l)} \mathbb{P}_{\mathcal{M}^f}(\rho) \frac{Pa(\rho)}{\mathbb{P}_{\mathcal{M}^f}(\rho)} r(\rho) \right)(v) \tag{3}$$

$$= \sum_{\rho \in \Gamma_{\mathcal{M}^f}(l)} Pa(\rho)(v) r(\rho) \tag{4}$$

$$= \sum_{\rho \in \Gamma_{\mathcal{M}^f}(l)} \mathbb{P}_{\mathcal{M}^v}(\rho) r(\rho) \tag{5}$$

$$= \sum_{\rho \in \Gamma_{\mathcal{M}^v}(l)} \mathbb{P}_{\mathcal{M}^v}(\rho) r(\rho) \tag{6}$$

$$= \mathbb{E}^l_{\mathcal{M}^v}(r) \tag{7}$$

(2) is obtained by definition of the expected value and of the distribution of Y; (3) is obtained by definition of the random variable Y; (4) is direct because we only consider runs ρ such that $\mathbb{P}_{\mathcal{M}^f}(\rho) \neq 0$; (5) is a consequence of (1); and finally, since $\Gamma_{\mathcal{M}^v}(l) = \Gamma_{\mathcal{M}^f}(l)$ because $\mathcal{M}^v$ has the same structure as $\mathcal{M}^f$ we obtain (6).

Our adaptation of the Monte Carlo technique for pMC is thus to estimate the expected value of Y in order to obtain a good estimator for the expectation of r.

Let $\rho_1, \ldots, \rho_n$ be a set of n runs of length l of $\mathcal{M}^f$. Let Y_i be the random variable with values in $Poly(\mathbb{X})$ such that $Y_i = r'(\rho_i)$. Notice that the Y_i are independent copies of the random variable Y. The random variables Y_i are therefore independent and identically distributed. Let γ be the parametric function giving their mean value and σ^2 be the parametric function giving their variance.

By the results above, for all valid parameter valuation v such that $\mathcal{M}^v$ and $\mathcal{M}^f$ have the same structure, $\mathbb{E}^l_{\mathcal{M}^v}(r) = \mathbb{E}(Y)(v) = \mathbb{E}(\sum_{i=1}^n Y_i/n)(v) = \gamma(v)$. Our parametric approximation of the expected value is therefore

$$\widehat{\gamma} = \sum_{i=1}^{n} Y_i/n.$$

We now use the central limit theorem in order to compute the *confidence intervals* associated to this estimation. As for the estimation $\widehat{\gamma}$ itself, the obtained confidence intervals will be given by parametric functions.

Since the random variables (Y_i) are independent and identically distributed, each having mean γ and variance σ^2, the expected value and variance of their sum $Y_1 + \cdots + Y_n$ is as follows: $\mathbb{E}(Y_1 + \cdots + Y_n) = n\gamma$ and $Var(Y_1 + \cdots + Y_n) = n\sigma^2$. Recall that both γ and σ are parametric functions. Let Z be the (parametric)

random variable such that

$$Z = \frac{(Y_1 + \cdots + Y_n - n\gamma)}{\sqrt{n\sigma^2}}.$$

By the central limit theorem, $Z(v)$ tends toward the standard normal distribution $\mathcal{N}(0,1)$, for all valid parameter valuation v such that $\mathcal{M}^v$ and $\mathcal{M}^f$ have the same structure, as $n \to \infty$. Therefore, for all valid parameter valuation v such that $\mathcal{M}^v$ and $\mathcal{M}^f$ have the same structure, assuming that n is large enough, we obtain that

$$\mathbb{P}(-z \leq Z(v) \leq z) \approx \boldsymbol{\varphi}(z) - \boldsymbol{\varphi}(-z) = 2\boldsymbol{\varphi}(z) - 1,$$

where $\boldsymbol{\varphi}(z) = \int_{-\infty}^{z} exp(-x^2/2)dx/(\sqrt{2\pi})$ is the cumulative distribution function of the standard normal distribution $\mathcal{N}(0,1)$. As a consequence, by definition of Z, we obtain that for all valid parameter valuation v such that $\mathcal{M}^v$ and $\mathcal{M}^f$ have the same structure, and for n large enough,

$$2\boldsymbol{\varphi}(z) - 1 \approx \mathbb{P}\left(\gamma(v) - z\frac{\sigma(v)}{\sqrt{n}} \leq \widehat{\gamma}(v) \leq \gamma(v) + z\frac{\sigma(v)}{\sqrt{n}}\right).$$

The (parametric) random interval $I = (\widehat{\gamma} - z\sigma/\sqrt{n}, \widehat{\gamma} + z\sigma/\sqrt{n})$ is called a *confidence interval* for γ with level $2\boldsymbol{\varphi}(z) - 1$. Typically we use $z = 1.96$ since $2\boldsymbol{\varphi}(1.96) - 1 = 0.95$ (or $z = 2.56$ for which the level is 0.99). According to our hypothesis, this (parametric) confidence interval is only valid for valid parameter valuations v such that $\mathcal{M}^v$ and $\mathcal{M}^f$ have the same structure.

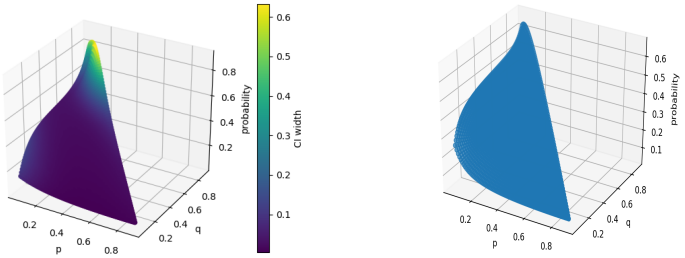
At this point, it is important to recall that the size of the confidence interval for $\widehat{\gamma}$ is also parametric: it is equal to $2 \cdot z\frac{\sigma}{\sqrt{n}}$. Indeed, the value of σ depends both on the valuation of the parameters and on the choice of the normalization function. We now explain how to compute a parametric estimation of σ. From classical probability theory, we know that an unbiased estimator for the variance σ^2 is:

$$\widehat{\sigma^2} = \frac{1}{(n-1)}\sum_{i=1}^{n} Y_i^2 - \frac{n}{(n-1)}\widehat{\gamma}^2.$$

For n large enough, the (parametric) confidence interval associated to our estimation $\widehat{\gamma}$ of γ can therefore be estimated by the (parametric) interval $I = (\widehat{\gamma} - z\widehat{\sigma}/\sqrt{n}, \widehat{\gamma} + z\widehat{\sigma}/\sqrt{n})$. The (parametric) size of this interval is therefore given by $2z\widehat{\sigma}/\sqrt{n}$. Thus, for an estimation of γ with a confidence interval of level 0.95 (for a large enough value of n), the (parametric) size of the confidence interval for the estimation of the variance is $3.92\widehat{\sigma}/\sqrt{n}$. Recall that $\widehat{\sigma}$ is a parametric function that also depends on the choice of the normalization function.

4 Implementation

We implemented our technique in python to validate the approach. All the experiments have been realized on a 2,5 GHz Intel Core i7 processor. The



(a) With 10 000 simulations and uniform normalization

(b) Theoretical probability

Fig. 3: Results on pMC $\mathcal{M}_1$.

prototype is still in the early development stages, thus we only experimented with the uniform normalization function as well as with valid valuation normalization functions. Moreover, the size of the samples is set manually. This implementation aims at validating our approach and not at competing with tools such as PARAM [9] or PROPhESY [2]. No optimizations have been implemented and we believe that this prototype could obtain much better result with simple optimizations. The code (still in development) is available at https://github.com/paulinfournier/MCpMC.

4.1 Toy example

We first tested our program on the pMC $\mathcal{M}_1$ given in example 2. The considered property is the probability of reaching state 4. The results of our program on this example are presented in Figure 3a. The number of simulations was set to 10 000 and the length of the run bounded to 100. The normalization function used is the uniform normalization. We also give in Figure 3b the graphical representation of the theoretical probability of reaching state 4. The parsing of the model took around 3ms and the simulations took around 17 seconds. The memory consumption of the program is depicted in Figure 4. Note that even if the program uses around 80MiB of memory, most of it is due to the loading of python libraries. A careful analysis of the memory consumption shows that the actual model only uses 0.7MiB, the simulations use 6.5MiB and the figure uses around 4MiB. The memory consumption could thus be easily reduced by using a lighter programming language with a better handling of memory such as C.

To show that our method is transparent with regard to the number of parameters we extended this model with 100 parameters $\{p_0, ..., p_{99}\}$. Since q remains the same and r is always $1 - (p + q)$, this new model is equipped with a total of 101 parameters. We therefore consider an unfolding of the pMC given in Figure 1

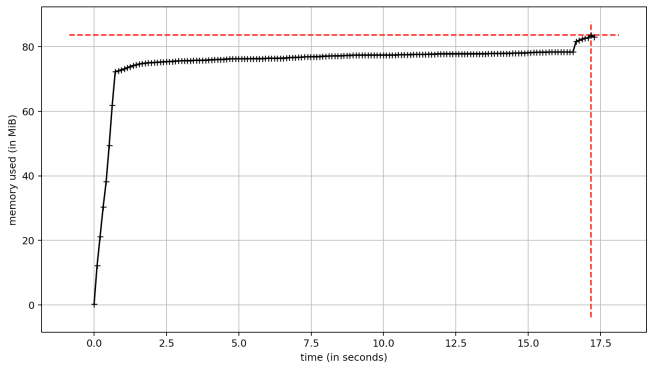


Fig. 4: Graphical representation of the memory consumption

of depth 101. Each time we enter a copy of state 1 or 3, new parameters are used. For example, the nth copy of states 1 and 3 use parameter $q_{n \mod 100}$ instead of q. The result of the experiment is shown in Figure 5. Since there are too many dimensions to plot in 2D, we only plot the result with respect to parameter q for random valuations of the parameters in (p_i). The number of simulations and simulation length are set as before. Note that the size and shade of the confidence intervals are not what one could expect but this is due to the fact that each point is evaluated for a different random valuation of the parameters in (p_i). The important point of this experiment is that the results were obtained in 21 second, which is approximately the same as with only 2 parameters. Note that this increase in time is due to a slight increase in the complexity of the model (the states are not really duplicated but we have to keep track of the current depth in order to consider the right parameters in each simulation).

4.2 Zeroconf

The Zeroconf model, taken from the PARAM website[3], models the management of a network. When a new host joins the network, it randomly selects an address among K potential candidates. If there are already m hosts in the network the probability of collision is $q = m/K$, which is a parameter of the model. In order to detect collisions, the new host asks the others whether the address is free. If he receives a positive answer, then the new hosts considers that his address is valid. The second parameter of the model is the probability p that the new hosts does not receive an answer. In this case, the new host retries at most n times (here $n = 140$). In case the new host does not receive an answer after n attempts, he decides that his address is valid. We consider the expected number of attempts until the address is considered valid (either because of positive answer or because of the absence of answers after n attempts). In Figure 6 we present the results

[3] See https://depend.cs.uni-saarland.de/tools/param/casestudies/

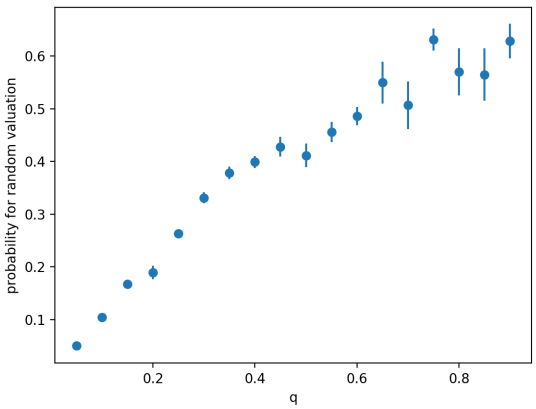


Fig. 5: Extension of $\mathcal{M}_1$ with 101 parameters

of our approach for 10 000 simulations, the uniform normalization function and for a simulation length of 500. Using our prototype, the experiment took around 60s. In Figure 6, we present as well the result obtained with PARAM (taken from their website). Remark that the shape of the distribution we obtain is similar to the one obtained with PARAM. The size of the confidence intervals are given as the color of the points.

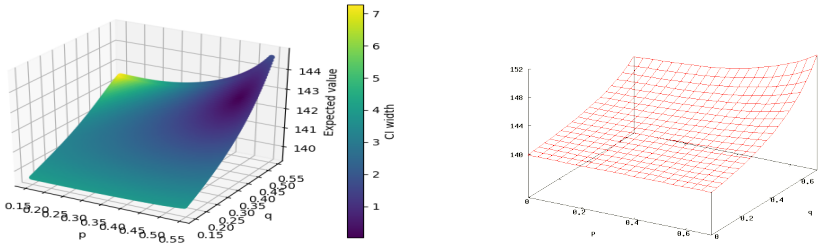


Fig. 6: Results for the zeroconf model obtained with our implementation (left) and taken from the PARAM website (right).

4.3 The crowds protocol

The Crowds protocol [10] aims at preserving anonymity of Internet users. To do so, each message is sent via random routes, with the assumption that a corrupted

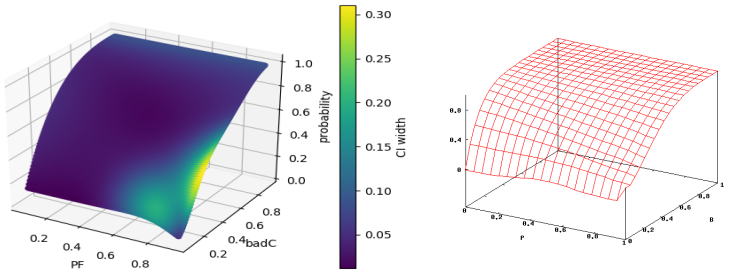


Fig. 7: Results on the Crowds protocol model for our implementation (left) and from the PARAM website (right).

router can only see the local sender of a message. This protocol guarantees that the probability of a corrupted router observing the real sender (and not just routing another user's message) is small. A model of this protocol as been proposed in Prism [11] and later extended with two parameters in [9]. One parameter PF represents the probability of forwarding a message to a random selected member (and therefore the probability of delivering the message directly to the receiver is $1 - PF$). The other parameter *badC* represents the probability of a router being corrupted. The reward function we are interested in corresponds to the probability of dishonest router observing the real sender more than other crowd members.

Figure 7 represents the output of our approach on this model for 10 000 simulations, using the uniform normalization function. This experiment was done with the same values as used on the PARAM website, that is 5 honest crowd members and 7 different path reformulates[4]. The simulation time took around 6 minute and used around 80MiB (again most of it due to loading of python libraries). Again, one can observe that the shape of the obtained distribution is similar to the one obtained with PARAM. The only zones where the two distributions are slightly different are those where the confidence intervals we produced are the largest.

5 Potential improvements

In this section we explore some variants of our technique in order to tackle some of its inherent problems.

5.1 Choice of normalization function

The choice of the normalization function f has a huge impact on the convergence speed of our method. Figure 9 illustrates this on a simple example where the

[4] See https://depend.cs.uni-saarland.de/tools/param/casestudies/ for details

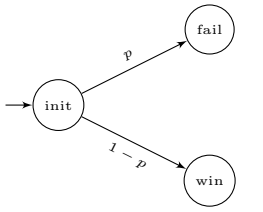


Fig. 8: A simple pMC.

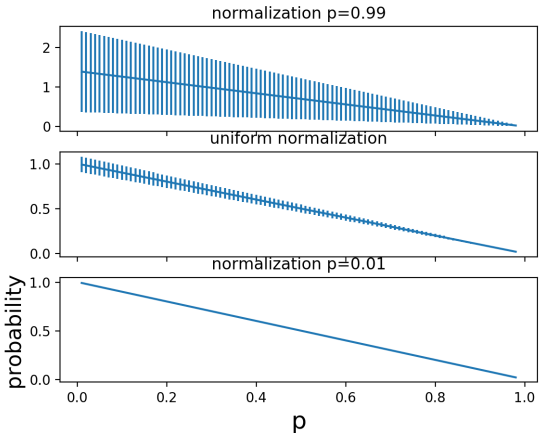


Fig. 9: Impact of the choice of the normalization function f on the size of confidence intervals.

property tested is the reachability of state *win* in the three state pMC given Figure 8. In Figure 9, where the number of simulations is set to 500, we can see that with a normalization function assigning 0.99 to p the confidence intervals are much larger (in particular for small values of p) than with a normalization function assigning 0.01 to p.

This can be explained by the following computation that bounds the variance of Y.

$$
\begin{aligned}
Var(Y)(v) &= \left(\mathbb{E}^l_{\mathcal{M}^f}(Y^2) - \mathbb{E}^l_{\mathcal{M}^f}(Y)^2\right)(v) && (8)\\
&= \left(\sum_{\rho \in \Gamma_{\mathcal{M}^f}(l)} \mathbb{P}_{\mathcal{M}^f}(\rho) y(\rho)^2\right)(v) - \mathbb{E}^l_{\mathcal{M}^v}(r)^2 && (9)\\
&= \sum_{\rho \in \Gamma_{\mathcal{M}^f}(l)} Pa(\rho)(v)^2 r(\rho)^2 / \mathbb{P}_{\mathcal{M}^f}(\rho) - \mathbb{E}^l_{\mathcal{M}^v}(r)^2 && (10)\\
&= \sum_{\rho \in \Gamma_{\mathcal{M}^f}(l)} \mathbb{P}_{\mathcal{M}^v}(\rho) r(\rho)^2 \mathbb{P}_{\mathcal{M}^v}(\rho) / \mathbb{P}_{\mathcal{M}^f}(\rho) - \mathbb{E}^l_{\mathcal{M}^v}(r)^2 && (11)\\
&\leq m \sum_{\rho \in \Gamma_{\mathcal{M}^f}(l)} \mathbb{P}_{\mathcal{M}^v}(\rho) r(\rho)^2 - \mathbb{E}^l_{\mathcal{M}^v}(r)^2 && (12)\\
&= m\mathbb{E}^l_{\mathcal{M}^v}(r^2) - \mathbb{E}^l_{\mathcal{M}^v}(r)^2 && (13)\\
&= (m-1)\mathbb{E}^l_{\mathcal{M}^v}(r^2) + Var^l_{\mathcal{M}^v}(r) && (14)
\end{aligned}
$$

Where $m = \max_{\rho \in \Gamma_{\mathcal{M}^f}(l), r(\rho) \neq 0} \mathbb{P}_{\mathcal{M}^v}(\rho) / \mathbb{P}_{\mathcal{M}^f}(\rho)$.

In the example, with the normalization function assigning 0.99 to p we have $m = (1-p)/0.01 = 100 - 100p$, thus for small values of p the obtained variance

is great. On the contrary, considering the normalization function assigning 0.01 to p, we have $m = (1-p)/0.99$, which is always quite small.

Following this remark, a good direction to improve the implemented prototype would be to automatically choose the normalization function minimizing m. Note that such a normalization function can be hard to compute, but one could first use the uniform normalization function then find which valid valuation maximizes the size of the confidence interval and restart an evaluation for this valid valuation. This technique may produce good results since the size of the confidence intervals decreases only with the square root of the number of runs. Thus decreasing m may be faster than increasing the number of simulations.

5.2 Modification of the structure

Notice that our technique is valid only for valuations such that $\mathcal{M}^f$ and $\mathcal{M}^v$ have the same structure. Indeed, to obtain the approximation of the parametric expected value we divide by the number of runs. However, when evaluating this parametric expected value on valuations modifying the structure of the MC $\mathcal{M}^f$, it may be the case that some of the runs are impossible for this valuation. For example, any run that takes a transition with a parametric probability p is impossible to obtain for a valuation v such that $v(p) = 0$, although it may appear in the simulations of $\mathcal{M}^f$ if $f(p) \neq 0$.

To address this problem, instead of dividing by the total number of runs obtained when simulating under the normalization function f, we can divide by the number of runs that are possible for the considered valuation v. Formally, instead of the estimator $(\sum_{i=1}^{n} Y_i)/n$, we can use $(\sum_{i=1}^{n} Y_i)/N$ where N is the function of v defined by:

$$N(v) = |\{i | \mathbb{P}_{\mathcal{M}^v}(\rho_i) > 0\}|$$

This can be computed on-the-fly. Defining the estimated variance and the random confidence interval with the same technique would give us a parametric approximation technique which is valid for any valid valuation (regardless of structure preservation). Note however that this is valid only if the structure of $\mathcal{M}^f$ allows more runs than the the structure of $\mathcal{M}^v$ *i.e.* if $\Gamma_{\mathcal{M}^v}(l) \subseteq \Gamma_{\mathcal{M}^f}(l)$. Otherwise, this technique only gives an approximation of $\mathbb{E}^l_{\mathcal{M}^v}(r(\rho)|\rho \in \Gamma_{\mathcal{M}^f}(l))$.

Notice also that this is useful only if $N(v)$ is large enough. Indeed if $N(v) << n$ it may be more relevant to estimate Y for a normalization function giving the same structure as $\mathcal{M}^v$.

5.3 Complement of the property

In classic Monte Carlo, there is no need to consider the complement of the property. Indeed, when one approaches both the probability of φ and that of $\neg\varphi$ by $\hat{\gamma}$ and $\hat{\gamma_\neg}$ respectively, it is always the case that $\hat{\gamma} = 1 - \hat{\gamma_\neg}$. In our approach however, since the probabilities of the runs are normalized, this does not always hold. Consider for example the pMC given figure 8 with a normalization function

such that $p = 1 - \epsilon$ with ϵ close to 0. With high probability, our approach would give that the probability of reaching *win* is $\hat{\gamma} = 0$ since it would miss the run reaching *win* which has a really low probability for this normalization function (note that this problem only appears if the number of simulations is not big enough). However, if we estimate the negation as well, we have $\hat{\gamma_\neg} = (n * p/(1-\epsilon))/n = p/(1-\epsilon)$, which is much closer to the truth for any valuation with $p \not\approx 1$.

It is thus relevant to consider both the property and its negation when considering a parametric approximation of a probability. Note that this holds only for the approximation of a probability and not for the general expected value of a reward. Note also that this is a good example of the importance of a good normalization function.

To implement this in practice, one could approach both the probabilities of φ and $\neg\varphi$ and whenever $\hat{\gamma_\neg}(v) \not\approx 1 - \hat{\gamma}(v)$ one could either increase the number of simulations or restart the approximation for the normalization function v.

6 Conclusion

In this paper, we have presented a new technique for statistical model checking of parametric Markov chains. This technique is based on a parametric adaptation of the standard Monte Carlo analysis. We show that our technique allows to approximate the expected value associated to any reward function by a polynomial function of the parameters, and propose as well a parametric confidence interval for the estimation. Compared to exact model-checking techniques, the technique we propose here offers the same benefits as standard simulation techniques for non-parametric models: better scalability and a complexity which is largely independent of the model complexity (be it in the size of the state space, in the type of features used, or in the number of parameters). Contrary to existing statistical model checking techniques for non-deterministic systems (such as the one presented in [6]), the one we propose in this paper allows to compute parametric confidence intervals, which offers a guarantee of the precision of our estimations. Finally, our technique has been implemented in a prototype tool using the Python language and has been tested on a set of benchmarks with encouraging results.

In the future, we plan on continuing the development of our prototype in a more efficient way in order to produce an extensive study both of the efficiency and of the accuracy of our technique compared to classical exact model checking techniques for pMCs. In particular, we plan on implementing the improvements presented in Section 5. We also plan on applying our technique to Markov Decision Processes, where nondeterminism could be replaced with parameters in order to study optimal schedulers. In this context, the parameters could be placed directly inside the model, or used inside a controler modeling the environment.

References

1. Alur, R., Henzinger, T.A., Vardi, M.Y.: Parametric real-time reasoning. In: Proceedings of the twenty-fifth annual ACM symposium on Theory of computing, ACM (1993) 592–601
2. Dehnert, C., Junges, S., Jansen, N., Corzilius, F., Volk, M., Bruintjes, H., Katoen, J.P., Ábrahám, E.: Prophesy: A probabilistic parameter synthesis tool. In Kroening, D., Păsăreanu, C.S., eds.: International Conference on Computer Aided Verification. Volume 9207 of LNCS., Cham, Springer (2015) 214–231
3. Hahn, E.M., Hermanns, H., Wachter, B., Zhang, L.: PARAM: A model checker for parametric Markov models. In: International Conference on Computer Aided Verification. Volume 6174 of LNCS., Springer (2010) 660–664
4. Bart, A., Delahaye, B., Lime, D., Monfroy, E., Truchet, C.: Reachability in parametric interval Markov chains using constraints. In: International Conference on Quantitative Evaluation of Systems. Volume 10503 of LNCS., Springer (2017) 173–189
5. Delahaye, B., Lime, D., Petrucci, L.: Parameter synthesis for parametric interval Markov chains. In: International Conference on Verification, Model Checking, and Abstract Interpretation. Volume 9583 of LNCS., Springer (2016) 372–390
6. Henriques, D., Martins, J., Zuliani, P., Platzer, A., Clarke, E.M.: Statistical model checking for markov decision processes. In: International Conference on Quantitative Evaluation of Systems, IEEE Computer Society (2012) 84–93
7. Baier, C., Katoen, J.P., Larsen, K.G.: Principles of model checking. MIT press (2008)
8. Ross, S.: A first course in probability. (2009)
9. Hahn, E.M., Hermanns, H., Wachter, B., Zhang, L.: Param: A model checker for parametric markov models. In: International Conference on Computer Aided Verification, Springer (2010) 660–664
10. Reiter, M.K., Rubin, A.D.: Crowds: Anonymity for web transactions. ACM transactions on information and system security (TISSEC) **1**(1) (1998) 66–92
11. Shmatikov, V.: Probabilistic model checking of an anonymity system. Journal of Computer Security **12**(3/4) (2004) 355–377