Numerical vs. Statistical Probabilistic Model Checking: An Empirical Study*

Håkan L. S. Younes[1], Marta Kwiatkowska[2], Gethin Norman[2], and David Parker[2]

[1] Computer Science Department, Carnegie Mellon University
Pittsburgh, PA 15213, USA
[2] School of Computer Science, University of Birmingham
Birmingham B15 2TT, United Kingdom

Abstract. Numerical analysis based on uniformisation and statistical techniques based on sampling and simulation are two distinct approaches for transient analysis of stochastic systems. We compare the two solution techniques when applied to the verification of time-bounded until formulae in the temporal stochastic logic CSL. This study differs from most previous comparisons of numerical and statistical approaches in that CSL model checking is a hypothesis testing problem rather than a parameter estimation problem. We can therefore rely on highly efficient sequential acceptance sampling tests, which enables statistical solution techniques to quickly return a result with some uncertainty. This suggests that statistical techniques can be useful as a first resort during system prototyping, rather than as a last resort as often suggested. We also propose a novel combination of the two solution techniques for verifying CSL queries with nested probabilistic operators.

1 Introduction

Continuous-time Markov chains (CTMCs) are an important class of stochastic models, widely used in performance and dependability evaluation. The temporal logic CSL (Continuous Stochastic Logic) introduced by Aziz et al. [2, 3] and since extended by Baier et al. [5] provides a powerful means to specify both path-based and traditional state-based performance measures on CTMCs in a concise and flexible manner. CSL contains a time-bounded until operator, the focus of this study, that allows one to express properties such as "the probability of n servers becoming faulty within 15.07 seconds is at most 0.01".

The two dominating techniques used for analysis of stochastic systems are *numerical* methods and *statistical* methods. Numerical solution techniques can often provide a higher accuracy than statistical methods, whose results are probabilistic in nature. However, numerical methods are far more memory intensive, which often leaves statistical solution techniques as a last resort [23, 7].

* Supported in part by DARPA and ARO under contract no. DAAD19–01–1–0485, a grant from the Royal Swedish Academy of Engineering Sciences (IVA), FORWARD and EPSRC grants GR/N22960, GR/S11107 and GR/S46727.

A. Podelski and K. Jensen (Eds.), 10th International Conference on Tools and Algorithms for the Construction and Analysis of Systems (TACAS'04), volume 2988 of LNCS, pages 46–60, 2004.

The verification of time-bounded CSL formulae can be reduced to transient analysis [4, 5]. Efficient numerical solution techniques, such as uniformisation [14, 20, 18, 7], for transient analysis of CTMCs have existed for decades and are well-understood. Younes and Simmons [26] have proposed a statistical approach for verifying time-bounded CSL formulae, based on acceptance sampling and discrete event simulation. The use of acceptance sampling is possible because CSL formulae only ask if a probability is above or below some threshold. Previous comparisons of numerical and statistical solution techniques have typically been based on estimation problems. This study is concerned with hypothesis testing problems, for which there exists highly efficient *sequential* acceptance sampling tests that make statistical solution techniques look more favourable than in a comparison with numerical techniques on estimation problems.

We have implemented the statistical model checking algorithm in the PRISM tool[3][16], a probabilistic model checker developed at the University of Birmingham. PRISM already implements numerical solution techniques and makes use of symbolic data representation in order to reduce memory requirements for those techniques.

Probabilistic model checking in general, and the two approaches implemented in PRISM, are described in Sect. 2. In this section we also propose a combination of numerical and statistical solution techniques to handle CSL formulae with nested probabilistic operators. The idea of combining the two techniques has been explored before [21, 7], but not in the context of nested CSL queries. The mixed solution technique has also been implemented in PRISM.

In Sect. 4, we present empirical results obtained with PRISM on a number of case studies, described in Sect. 3, which serve as the comparison of the two approaches. The results demonstrate that the complexity of both the numerical and the statistical approach is typically linear in the time-bound of the property, but that the statistical approach scales better with the size of the state space. Furthermore, the statistical approach requires considerably less memory than the numerical approach, allowing us to verify models far beyond the scope of numerical solution methods. The principal advantage of numerical techniques based on uniformisation is that increased accuracy in the result comes at almost no price. The statistical solution method can very rapidly provide solutions with some uncertainty, however reducing the uncertainty is costly. This suggest that statistical techniques can be quite useful as a *first* resort during system prototyping, while numerical techniques may be more appropriate when very high accuracy in the result is required.

2 CTMCs and Probabilistic Model Checking

Probabilistic model checking refers to a range of techniques for the formal analysis of systems that exhibit stochastic behaviour. The system is usually specified as a state transition system, with probability values attached to the transitions.

[3] PRISM web site: www.cs.bham.ac.uk/~dxp/prism

In this paper, we consider the case where this model is a continuous-time Markov chain (CTMC).

A CTMC $\mathcal{C}$ is a tuple $(S, \mathbf{R}, L)$ where S is a finite set of *states*, $\mathbf{R} : S \times S \to \mathbb{R}_{\geq 0}$ is the *rate matrix* and $L : S \to 2^{AP}$ is a *labelling function*, mapping each state to a subset of the set of atomic propositions AP. For any state $s \in S$, the probability of leaving state s within t time units is given by $1 - e^{-E(s) \cdot t}$ where $E(s) = \sum_{s' \in S} \mathbf{R}(s, s')$. $E(s)$ is known as the *exit rate*. If $\mathbf{R}(s, s') > 0$ for more than one $s' \in S$, then there is a *race* between the transitions leaving s, where the probability of moving to s' in a single step equals the probability that the delay corresponding to moving from s to s' "finishes before" the delays of any other transition leaving s. A *path* of the CTMC is a sequence of states, between each of which there is a non-zero probability of making a transition. A path of the CTMC can be seen as a single execution of the system being modelled.

In probabilistic model checking, properties of the system to be verified are specified in a temporal logic. For CTMCs, we use the temporal logic CSL [2, 3, 5], an extension of CTL. The syntax of CSL is defined as

$$\Phi \ ::= \ true \ \big| \ a \ \big| \ \Phi \wedge \Phi \ \big| \ \neg \Phi \ \big| \ \mathcal{P}_{\bowtie p}\left(\Phi \ \mathcal{U}^{\leq t} \ \Phi\right) \ \big| \ \mathcal{P}_{\bowtie p}\left(\Phi \ \mathcal{U} \ \Phi\right) \ \big| \ \mathcal{S}_{\bowtie p}\left(\Phi\right) \ ,$$

where $p \in [0, 1]$, $t \in \mathbb{R}_{\geq 0}$, $\bowtie \in \{<, \leq, \geq, >\}$ and a is an atomic proposition from the set AP used to label states of the CTMC.

A state s of a CTMC satisfies the formula $\mathcal{P}_{\bowtie p}(\phi)$, denoted $s \models \mathcal{P}_{\bowtie p}(\phi)$, if $P(s, \phi) \bowtie p$, where $P(s, \phi)$ is the probability that a path starting in state s satisfies the path formula ϕ. Here, a path formula ϕ is either $\Phi \ \mathcal{U}^{\leq t} \ \Psi$, meaning that formula Ψ is satisfied within t time units and formula Φ is satisfied up until that point, or $\Phi \ \mathcal{U} \ \Psi$, meaning that $\Phi \ \mathcal{U}^{\leq t} \ \Psi$ holds for some $t \in \mathbb{R}_{\geq 0}$. The value $P(s, \phi)$ is defined in terms of the probability measure over paths starting in state s, as defined by Baier et al. [5]. The $\mathcal{S}_{\bowtie p}(\Phi)$ operator describes the behaviour of the CTMC in the *steady-state* or long-run. The precise semantics of this and the other CSL operators are given by Baier et al. [5]. In this paper, we focus on the *time-bounded until* operator $\mathcal{P}_{\bowtie p}\left(\Phi \ \mathcal{U}^{\leq t} \ \Psi\right)$.

2.1 Numerical Probabilistic Model Checking

The numerical model checking approach for verifying a time-bounded until formula $\mathcal{P}_{\bowtie p}\left(\Phi \ \mathcal{U}^{\leq t} \ \Psi\right)$ in a state $s \in S$ is based on first computing the probability $P(s, \Phi \ \mathcal{U}^{\leq t} \ \Psi)$, and then comparing this probability with the threshold $\bowtie p$.

First, as initially proposed by Baier et al. [4], the problem is reduced to the computation of transient probabilities on a modified CTMC. For a CTMC $\mathcal{C} = (S, \mathbf{R}, L)$, we construct the CTMC $\mathcal{C}' = (S, \mathbf{R}', L)$ by making all states satisfying $\neg\Phi \vee \Psi$ absorbing, i.e. removing all of their outgoing transitions. Hence, $\mathbf{R}'$ is obtained from $\mathbf{R}$ by removing all entries from the appropriate rows. The probability $P(s, \Phi \ \mathcal{U}^{\leq t} \ \Psi)$ in the CTMC $\mathcal{C}$ is now equal to the probability of, in the CTMC $\mathcal{C}'$, being in a state satisfying Ψ at time t having started in state s.

The computation of this probability is carried out via a process know as *uniformisation* (also known as *randomisation*), originally proposed by Jensen

[14]. We construct the *uniformised* discrete-time Markov chain (DTMC) of $\mathcal{C}'$, whose probability transition matrix $\mathbf{P}$ equals $\mathbf{I} + (\mathbf{R}' - \mathbf{E}')/q$, where $\mathbf{I}$ is the identity matrix, $\mathbf{E}'$ is a diagonal matrix containing exit rates of $\mathcal{C}'$, i.e. $\mathbf{E}'(s, s')$ equals $E'(s)$ if $s = s'$ and 0 otherwise, and $q \geq \max\{E'(s) \mid s \in S\}$ is the *uniformisation constant* of the CTMC $\mathcal{C}'$.

It then follows that $P(s, \Phi\ \mathcal{U}^{\leq t}\ \Psi)$ can be computed simultaneously for all states $s \in S$ by computing the vector of probabilities

$$\underline{P}(\Phi\ \mathcal{U}^{\leq t}\ \Psi) = \sum_{k=0}^{\infty} \gamma(k, q{\cdot}t) \cdot \left(\mathbf{P}^k \cdot \underline{\Psi}\right) \quad , \tag{1}$$

where $\gamma(k, q{\cdot}t)$ is the kth Poisson probability with parameter $q{\cdot}t$ (i.e. $\gamma(k, q{\cdot}t) = e^{-q \cdot t} \cdot (q{\cdot}t)^k / k!$), and $\underline{\Psi}$ characterises the set of states satisfying Ψ (i.e. $\underline{\Psi}(s) = 1$ if $s \models \Psi$, and 0 otherwise). If we are only interested in verifying $\mathcal{P}_{\bowtie p}\left(\Phi\ \mathcal{U}^{\leq t}\ \Psi\right)$ in a single state s, then we only need to carry out the summation in (1) for $P(s, \Phi\ \mathcal{U}^{\leq t}\ \Psi)$, which in practice can save us both memory and time. However, as pointed out by Katoen et al. [15], the asymptotic time complexity is the same when computing the entire vector $\underline{P}(\Phi\ \mathcal{U}^{\leq t}\ \Psi)$. In this paper, we only compute the entire vector for nested probabilistic formulae.

In practice, the infinite summation in (1) is truncated from the left by L_ϵ and from the right by R_ϵ by using the techniques of Fox and Glynn [8] so that the truncation error is bounded by an a priori error tolerance ϵ. This means that if $\underline{\hat{P}}(\Phi\ \mathcal{U}^{\leq t}\ \Psi)$ is the solution vector obtained with truncation, then

$$0 \leq P(s, \Phi\ \mathcal{U}^{\leq t}\ \Psi) - \hat{P}(s, \Phi\ \mathcal{U}^{\leq t}\ \Psi) \leq \epsilon \quad \forall s \in S \ . \tag{2}$$

Note that, since iterative squaring is not attractive for sparse matrices due to fill-in [22, 20], the matrix products $\mathbf{P}^k$ are typically computed in an iterative fashion: $\mathbf{P}^k \cdot \underline{\Psi} = \mathbf{P} \cdot \left(\mathbf{P}^{k-1} \cdot \underline{\Psi}\right)$. Also, although the left truncation point L_ϵ allows us to skip the first L_ϵ terms of (1), we still need to compute $\mathbf{P}^k \cdot \underline{\Psi}$ for $k < L_\epsilon$, so the total number of iterations required by the algorithm is R_ϵ.

Steady-State Detection. To potentially reduce the number of iterations required by the numerical model checking algorithm, we can use on-the-fly steady-state detection in conjunction with uniformisation [20, 18]. If the uniformised DTMC reaches steady-state after $k_s < R_\epsilon$ iterations, then $\mathbf{P}^k \cdot \underline{\Psi} = \mathbf{P}^{k_s} \cdot \underline{\Psi}$ for all $k \geq k_s$, which means that we can compute $\underline{\hat{P}}(\Phi\ \mathcal{U}^{\leq t}\ \Psi)$ as follows using only k_s iterations:

$$\underline{\hat{P}}(\Phi\ \mathcal{U}^{\leq t}\ \Psi) = \sum_{k=L_\epsilon}^{k_s} \gamma(k, q{\cdot}t) \cdot \left(\mathbf{P}^k \cdot \underline{\Psi}\right) + \left(\mathbf{P}^{k_s} \cdot \underline{\Psi}\right) \cdot \left(1 - \sum_{k=L_\epsilon}^{k_s} \gamma(k, q{\cdot}t)\right) \ . \tag{3}$$

We can ensure that a steady-state vector actually exists by choosing q strictly greater than $\max\{E(s) \mid s \in S\}$ [20, 18].

Malhotra et al. [18] derive an error bound for (3) under the assumption that the steady-state point can be detected exactly within a given error tolerance.

Let $\underline{\Pi}^*$ denote the true steady-state vector. Malhotra et al. claim that if $\|\mathbf{P}^{k_s} \cdot \underline{\Psi} - \underline{\Pi}^*\| \leq \epsilon/4$ for $L_\epsilon < k_s < R_\epsilon$, then the same error bound as in (2) is guaranteed. The error analysis is flawed, however, in that it results in an error region twice as wide as the original error region. This is a result of the error due to steady-state detection being two-sided, while the truncation error is one-sided. To guarantee an error region of width ϵ instead of 2ϵ, it is necessary to bound $\|\mathbf{P}^{k_s} \cdot \underline{\Psi} - \underline{\Pi}^*\|$ by $\epsilon/8$ instead of $\epsilon/4$. This correction yields the error bounds $-\epsilon/4 \leq P(s, \Phi\, \mathcal{U}^{\leq t}\, \Psi) - \hat{P}(s, \Phi\, \mathcal{U}^{\leq t}\, \Psi) \leq 3\epsilon/4$ for all $s \in S$.

In practice, the true steady-state vector $\underline{\Pi}^*$ is not known a priori, so instead we stop when the norm of the difference between successive iterates is sufficiently small (at most $\epsilon/8$ by the above analysis), as suggested by Malhotra et al. [18].

Sequential Stopping Rule. To potentially reduce the number of iterations even further, we note that the CSL query $\mathcal{P}_{\bowtie p}\left(\Phi\, \mathcal{U}^{\leq t}\, \Psi\right)$ does not require that we compute $P(s, \Phi\, \mathcal{U}^{\leq t}\, \Psi)$ with higher accuracy than is needed to determine whether $P(s, \Phi\, \mathcal{U}^{\leq t}\, \Psi) \bowtie p$ holds. In the following analysis we restrict $\bowtie$ to $\geq$ as the other three cases are essentially the same.

Let $\hat{P}^k(s, \Phi\, \mathcal{U}^{\leq t}\, \Psi)$ denote the accumulated probability up until and including iteration k. Because each term in (1) is non-negative, we know that $\hat{P}^i(s, \Phi\, \mathcal{U}^{\leq t}\, \Psi) \geq \hat{P}^k(s, \Phi\, \mathcal{U}^{\leq t}\, \Psi)$ for all $i > k$. Therefore if $\hat{P}^k(s, \Phi\, \mathcal{U}^{\leq t}\, \Psi) \geq p$ holds for some $k < R_\epsilon$, then we can answer the query $\mathcal{P}_{\geq p}\left(\Phi\, \mathcal{U}^{\leq t}\, \Psi\right)$ affirmatively after only k iterations instead of R_ϵ (or k_s) iterations.

For early termination with a negative result, we can use the upper bound on the right Poisson tail provided by Fox and Glynn [8] for $k \geq 2 + \lfloor q \cdot t \rfloor$ to determine if $\mathcal{P}_{\geq p}\left(\Phi\, \mathcal{U}^{\leq t}\, \Psi\right)$ is false before completing R_ϵ iterations. Let $\hat{T}$ be the upper bound on the right Poisson tail. If $\hat{P}^k(s, \Phi\, \mathcal{U}^{\leq t}\, \Psi) + \hat{T} < p$, then we know already after k iterations that $\mathcal{P}_{\geq p}\left(\Phi\, \mathcal{U}^{\leq t}\, \Psi\right)$ is false.

We now have a sequential stopping rule for our algorithm, but note that the potential savings are limited by the fact that the positive part of the rule applies first after L_ϵ iterations and the negative part first after $2 + \lfloor q{\cdot}t \rfloor$ iterations, and both L_ϵ and R_ϵ are of the same order of magnitude as $q{\cdot}t$. We will see later that the sequential component of the statistical approach is much more significant.

Symbolic Representation. The PRISM tool uses binary decision diagrams (BDDs) [6] and multi-terminal BDDs (MTBDDs) [9] to construct a CTMC from a model description in the PRISM language, a variant of Alur and Henzinger's Reactive Modules formalism [1]. For numerical computation though, PRISM includes three separate *engines* making varying use of symbolic methods.

The first engine uses MTBDDs to store the model and iteration vector, while the second uses conventional data structures for numerical analysis: sparse matrices and arrays. The latter nearly always provides faster numerical computation than its MTBDD counterpart, but sacrifices the ability to conserve memory by exploiting structure. The third, *hybrid*, engine provides a compromise by storing the models in an MTBDD-like structure, which is adapted so that numerical computation can be carried out in combination with array-based storage for vectors. This hybrid approach is generally faster than MTBDDs, while handling larger systems than sparse matrices, and hence is the one used in this paper. For further details and comparisons between the engines see [17, 19].

2.2 Statistical Probabilistic Model Checking

Statistical techniques, involving simulation and sampling, have been in use for decades to analyse stochastic systems. Younes and Simmons [26] show how discrete event simulation and acceptance sampling can be used to verify properties of general discrete event systems expressed as CSL formulae not including $\mathcal{P}_{\bowtie p}(\Phi\,\mathcal{U}\,\Phi)$ and $\mathcal{S}_{\bowtie p}(\Phi)$. We focus here on $\mathcal{P}_{\geq p}\left(\Phi\,\mathcal{U}^{\leq t}\,\Psi\right)$, noting that $\mathcal{P}_{\leq p}\left(\Phi\,\mathcal{U}^{\leq t}\,\Psi\right) \equiv \neg\mathcal{P}_{>1-p}\left(\Phi\,\mathcal{U}^{\leq t}\,\Psi\right)$, and that $>$ ($<$) is practically indistinguishable from $\geq$ ($\leq$) to any acceptance sampling test (this can be said of numerical approaches as well due to the use of finite precision floating-point arithmetic).

Given a state $s \in S$ and a CSL formula $\mathcal{P}_{\geq p}\left(\Phi\,\mathcal{U}^{\leq t}\,\Psi\right)$, we wish to test whether $P(s, \Phi\,\mathcal{U}^{\leq t}\,\Psi)$ is above or below the threshold p. More specifically, we want to test the hypothesis $P(s, \Phi\,\mathcal{U}^{\leq t}\,\Psi) \geq p$ against the alternative hypothesis $P(s, \Phi\,\mathcal{U}^{\leq t}\,\Psi) < p$.

In order for acceptance sampling to be applicable, however, we first need to relax the hypotheses. For some $\delta > 0$, let H_0 be the hypothesis $P(s, \Phi\,\mathcal{U}^{\leq t}\,\Psi) \geq p + \delta$ and let H_1 be the hypothesis $P(s, \Phi\,\mathcal{U}^{\leq t}\,\Psi) \leq p - \delta$. Clearly, the formula $\mathcal{P}_{\geq p}\left(\Phi\,\mathcal{U}^{\leq t}\,\Psi\right)$ is true if H_0 holds and false if H_1 holds. An acceptance sampling test, such as Wald's *sequential probability ratio test* [24], will limit the probability of accepting H_1 when H_0 holds (false negative) to α and the probability of accepting H_0 when H_1 holds (false positive) to β. We refer to the region $I = (p-\delta, p+\delta)$ as the *indifference region*, because no guarantees regarding the error probability is given if $P(s, \Phi\,\mathcal{U}^{\leq t}\,\Psi) \in I$, and we assume that the user selects δ with this in mind. The parameters α and β determine the *strength* of the acceptance sampling test.

Wald's sequential probability ratio test is carried out as follows. Let $p_0 = p+\delta$ and $p_1 = p - \delta$, and let x_i be a sample of a Bernoulli variate X representing the result of verifying $\Phi\,\mathcal{U}^{\leq t}\,\Psi$ over a sample path starting in state s. Sample paths are generated through discrete event simulation, and we only generate as much of a sample path as is needed to determine the truth value of $\Phi\,\mathcal{U}^{\leq t}\,\Psi$. Note that we can perform simulation at the level of the PRISM language and never need to generate the underlying CTMC. At the mth stage of the test, we calculate the quantity

$$\frac{p_{1m}}{p_{0m}} = \prod_{i=1}^{m} \frac{\Pr[X = x_i \,|\, P(s, \Phi\,\mathcal{U}^{\leq t}\,\Psi) = p_1]}{\Pr[X = x_i \,|\, P(s, \Phi\,\mathcal{U}^{\leq t}\,\Psi) = p_0]} \ ,$$

where x_i is the ith sample of the Bernoulli variate X (1 if path formula holds and 0 otherwise), and $\Pr[X = x_i \,|\, P(s, \Phi\,\mathcal{U}^{\leq t}\,\Psi) = p_j] = p_j^{x_i}(1-p_j)^{1-x_i}$. Hypothesis H_0 is accepted if $p_{1m}/p_{0m} \leq \beta/(1-\alpha)$, and hypothesis H_1 is accepted if $p_{1m}/p_{0m} \geq (1-\beta)/\alpha$. Otherwise, an additional sample is required. This gives an acceptance sampling test that, for all practical purposes, has strength α and β. For further details on the test see [24].

2.3 Mixing Numerical and Statistical Techniques

Although the algorithm of Younes and Simmons [26] can handle CSL formulae with nested probabilistic operators, the way in which it is done requires in the worst case that the nested formula be verified in each state along a sample path, each time with an error inversely proportional to the length of the sample path. The numerical approach, on the other hand, can verify the nested formula for all states simultaneously at the same (asymptotic) cost as verifying the formula for a single state. This is a clear advantage when dealing with nested probabilistic operators.

We therefore propose a mixed approach, implemented in our system, where statistical sampling is used to verify the outermost probabilistic operator, while numerical techniques are used to verify the nested probabilistic operators. We can mix the numerical and statistical techniques by assuming that the result of the numerical technique holds with certainty (i.e. $\alpha = \beta = 0$ in terms of a statistical test). The nested formulae are first verified for all states using numerical methods. When verifying a path formula over a sample path we only need to read the value for each state along the path without any additional verification effort for the nested formulae. The cost for verifying the nested components of a formula is exactly the same for the mixed approach as for the numerical approach, but the use of sampling for the outermost probabilistic operator can provide a faster solution.

3 Case Studies

We now introduce three case studies, taken from the literature on performance evaluation and probabilistic model checking, on which we will base our empirical evaluation. A fourth simple case study is also introduced to illustrate the use of nested probabilistic operators in CSL.

Tandem Queueing Network. The first case study is based on a CTMC model of a tandem queueing network presented by Hermanns et al. [11]. The network consists of an $M/Cox_2/1$ queue sequentially composed with an $M/M/1$ queue. The capacity of each queue is n, and the state space is $\mathcal{O}(n^2)$. The property of interest is given by the CSL formula $\mathcal{P}_{<0.5}\left(\mathit{true}\ \mathcal{U}^{\leq T}\ \mathit{full}\right)$ which is true in a state if there is less than a 50% chance of the queueing network becoming full within T time units, and, in the case of the sampling-based approach, we verify the correctness of this property in the state where the both queues are empty.

Symmetric Polling System. For this case study we consider an n-station symmetric polling system described by Ibe and Trivedi [12]. Each station has a single-message buffer and the stations are attended by a single server in cyclic order. The server begins polling station 1. If there is a message in the buffer of station 1, the server starts polling that station. Once station i has been polled, or if there is no message in the buffer of station i when it is being served, the server starts polling station $i+1$ (or 1 if $i = n$). The polling and service times are exponentially distributed, as is the time from when a station has been polled until a new message fills its buffer. The fact that all rates are equal for all stations makes the system symmetric. The size of the state space for a system with n stations is $\mathcal{O}(n \cdot 2^n)$.

We will verify the property that, if station 1 is full, then it is polled within T time units with probability at least 0.5. We do so in the state where station 1 has just been polled and the buffers of all stations are full. Let $s \in \{1, \dots, n\}$ be the station currently getting the server's attention, let $a \in \{0, 1\}$ represent the activity of the server (0 for polling and 1 for serving), and let $m_i \in \{0, 1\}$ be the number of messages in the buffer of station i. We can then represent the property of interest with the CSL formula $m_1{=}1 \implies \mathcal{P}_{\geq 0.5}\big(\mathit{true}\ \mathcal{U}^{\leq T}\ \mathit{poll1}\big)$ where $\mathit{poll1} \equiv s{=}1 \wedge a{=}0$, and the state in which we verify the formula is given by $s{=}1 \wedge a{=}1 \wedge m_1{=}1 \wedge \ldots \wedge m_n{=}1$.

Dependable Workstation Cluster. The third case study is a dependable cluster of workstations due to Haverkort et al. [10]. The system consists of two sub-clusters each containing n workstations. Communication between the two sub-clusters is performed over a backbone connection. The workstations of each sub-cluster are connected in a star topology, with a single switch providing connectivity to the backbone. Each of the components can fail at any time, and the time to failure is exponentially distributed with different rates for different components. There is a single repair unit that can restore failed units. The repair time is assumed to be exponentially distributed. The size of the state space is $\mathcal{O}(n^2)$ for a cluster with $2n$ workstations.

The minimum quality of service (QoS) for a cluster is defined as having at least k interconnected operational workstations. This can be achieved by having at least k operational workstations in one sub-cluster with a functional switch, or by having at least k operational workstations in total with the backbone and both switches functioning properly. Let w_l and w_r denote the number of operational workstations in the left and right sub-clusters respectively. Furthermore, let b represent the atomic proposition that the backbone is working, and s_l (s_r) that the left (right) switch is up. Minimum QoS can then be defined as $\mathit{minimum} \equiv (w_\mathrm{l}{\geq}k \wedge s_\mathrm{l}) \vee (w_\mathrm{r}{\geq}k \wedge s_\mathrm{r}) \vee (w_\mathrm{l}{+}w_\mathrm{r}{\geq}k \wedge b \wedge s_\mathrm{l} \wedge s_\mathrm{r})$. The property we will verify is $\mathcal{P}_{<0.1}\big(\mathit{true}\ \mathcal{U}^{\leq T}\ \neg\mathit{minimum}\big)$, corresponding to Φ_4 of [10], that there is a less than 10% chance of the QoS dropping below minimum within T time units, and this property will be verified in the state where all units are functional.

Grid World. For the last case study we consider an $n \times n$ grid world with a robot moving from the bottom left corner to the top right corner, first along the bottom edge and then along the right edge. There is a janitor moving randomly around the grid, and the robot cannot move into a square occupied by the janitor. The objective is for the robot to reach the goal square at top right corner within T_1 time units with probability at least 0.9, while maintaining at least a 0.5 probability of periodically communicating with the base station. The CSL formula $\mathcal{P}_{\geq 0.9}\left(\mathcal{P}_{\geq 0.5}\left(\mathit{true}\ \mathcal{U}^{\leq T_2}\ \mathit{communicate}\right)\ \mathcal{U}^{\leq T_1}\ \mathit{goal}\right)$ expresses the given objective. The size of the state space is $\mathcal{O}(n^3)$.

4 Empirical Evaluation

We base our empirical evaluation on the case studies presented in Sect. 3. We have verified the time-bounded until formulae for the first three case studies using both the numerical and the statistical approach, varying the problem size (and thereby the size of the state space) and the time bound. Fig. 1 shows the verification time in seconds for these case studies, both as a function of the state space size and as a function of the time bound. All results, for both the numerical and the statistical approach, are for the verification of a CSL property in a *single* state. The results were generated on a 500 MHz Pentium III PC running Linux, and with a 700 MB memory limit set per process.

Memory Requirements. In the case of the numerical solution method, all experiments were run using the hybrid engine (see Sect. 2.1) which, although not necessarily the fastest engine, in general allows the analysis of larger problems than the other engines. The limiting factor in the hybrid approach is the space required to store the iteration vector: however compact the matrix representation is, memory proportional to the number of states is required for numerical solution. More precisely, the hybrid engine with steady-state detection requires storage of three double precision floating point vectors of size $|S|$, which for the memory limit of 700 MB means that systems with at most 31 million states can be analysed.[4] In practice, for the first three case studies, we were able to handle systems with about 27 million states, showing that the symbolic representation of the probability matrix is fairly compact.

The memory requirements for the statistical approach are very conservative. In principle, all that we need to store during verification is the current state, which only requires memory logarithmic in the size of the state space. We never exhausted memory during verification when using the statistical solution method.

Performance of Numerical Solution Method. For model checking time-bounded until formulae using the numerical approach, PRISM computes the Poisson probabilities (see Sect. 2.1) using the Fox-Glynn algorithm [8], which, for the hybrid engine, yields an overall time complexity of $\mathcal{O}(q \cdot t \cdot M)$, where q is the uniformisation constant of the CTMC, t is the time bound of the until formula and M is the number of non-zero entries in $\mathbf{R}$.

In all the examples considered, the number of non-zeros in the rate matrix is linear in the size of the state space. Hence, the verification time for a given

[4] We need an additional floating point vector of size $|S|$ for verifying a formula in all states simultaneously, which would make the limit 23 million states.

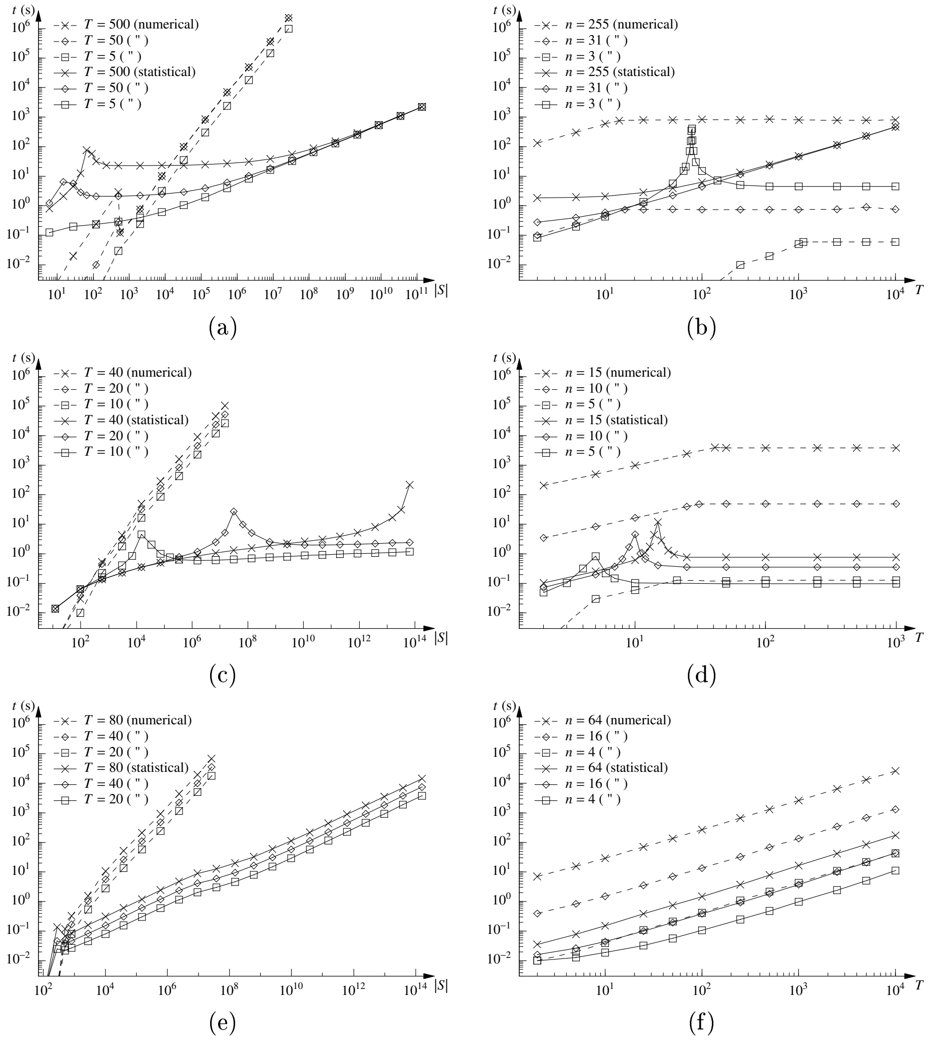


Fig. 1. The verification time, in seconds, for the tandem queueing network (top), the symmetric polling system (centre), and the dependable workstation cluster (bottom). To the left, the verification time is given as a function of the state space size, and to the right as a function of the time bound for the property that is being verified. Solid curves are for the statistical approach with $\alpha = \beta = 10^{-2}$ and $2\delta = 10^{-2}$, while dashed curves are for the numerical approach using the hybrid engine and $\epsilon = 10^{-6}$.

time-bounded until formula is linear in the size of the state space, as can be observed in Figs. 1(a), (c), and (e). For a single model, the complexity is linear in the time bound, as demonstrated by the results in Figs. 1(b), (d), and (f). Note that Figs. 1(b) and (d) show the verification time to be constant once the time bound has become sufficiently large. This is caused by steady-state detection as described in Sect. 2.1. We can also see the effect of steady-state detection in Fig. 1(a), with the curve for $T = 50$ and $T = 500$ coinciding for larger state spaces.

Performance of Statistical Solution Method. There are two main factors influencing the verification time for the statistical approach: the number of samples and the length of sample paths (in terms of state transitions). Consider the problem of verifying the formula $\mathcal{P}_{\geq p}(\phi)$ in a state s.

For fixed α and β (test strength) and δ (indifference region), the number of samples grows larger the closer p gets to the probability $P(s, \phi)$ that ϕ is satisfied by a path starting in state s. The results in Fig. 1 were generated using $\alpha = \beta = 10^{-2}$ and $2\delta = 10^{-2}$. The peaks in the curves for the statistical solution method all coincide with $P(s, \phi)$ being in the indifference region $[p - \delta, p + \delta]$.

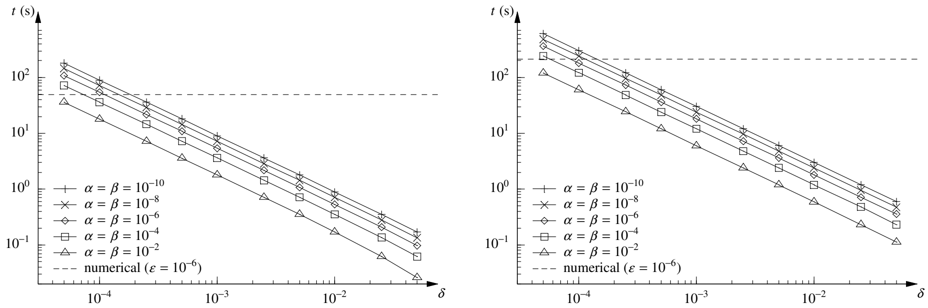
The number of samples required to verify a formula of the form $\mathcal{P}_{\geq p}(\phi)$ rapidly decreases as $P(s, \phi)$ gets further away from the threshold p. The key performance factor then becomes the length of sample paths, which depends on the exit rates of the CTMC and on the path formula ϕ. An upper bound on the expected length of sample paths is $\mathcal{O}(q{\cdot}t)$. We can see in Fig. 1(f), where $P(s, \phi)$ remains far from the threshold and the number of samples is close to constant (about 400 samples) for all values of the time bound, that the curves for both methods have roughly the same slope. The statistical approach scales better with the size of the state space giving it the edge over the numerical approach in the cluster example, but steady-state detection gives the numerical approach a clear advantage in the tandem case study (Fig. 1(b)).

For the tandem queueing network, the arrival rate for messages is $4n$, where n is the capacity of the queues. This has as a result that sample path lengths are proportional to n. As n increases, the sample path length becomes the dominant performance factor, meaning that verification time for the statistical approach becomes proportional to n. This is to be compared with the numerical approach, whose performance is linear in the size of the state space, which is quadratic in n. Similar results can be seen for the dependable workstation cluster. In the polling example, the arrival rate λ is inversely proportional to the number of polling stations n, while the other rates remain constant for all n. This explains the levelling-off of the curves for the statistical solution method in Fig. 1(c).

Recall that we only need to generate as much of a sample path as is needed to determine the truth value of ϕ. For $\phi = \Phi\, \mathcal{U}^{\leq t}\, \Psi$, we can stop if we reach a state satisfying $\neg\Phi \vee \Psi$ (cf. the CTMC $\mathcal{C}'$ constructed in the numerical approach in Sect. 2.1). The effect of this is seen most clearly for the polling case study as we increase the time bound. Once the path formula is satisfied the average length of the sample paths does not increase (Fig. 1(d)).

Trading Accuracy for Speed. With both solution methods, it is possible to adjust the accuracy of the result. For the statistical approach, we can control the parameters α, β, and δ so as to trade accuracy for efficiency. By setting these parameters high, we can get an answer quickly. We could then gradually tighten the error bounds and/or the indifference region to obtain higher accuracy. This approach is taken by Younes et al. [25], who modify the statistical solution method for verifying CSL formulae of the form $\mathcal{P}_{\geq p}(\phi)$ without nested probabilistic operators so that it can be stopped at any time to return a result.

Figure 2 shows how the verification time for a polling system problem and a workstation cluster problem depends on the strength of the test and the width of the indifference region. We can see that the verification time is inversely proportional both to the error bounds and the width of the indifference region, and that for some parameter values the numerical approach is faster while for others the statistical approach is the fastest. Using the statistical approach with error bounds $\alpha = \beta = 10^{-10}$ and half-width of the indifference region $\delta \approx 2 \cdot 10^{-4}$, for example, we could obtain a verification result for the polling system problem ($n = 10$) in roughly the same time as is required by the numerical approach. For larger models, we would of course be able to obtain even higher accuracy with the statistical approach if allowed as much time as needed by the numerical approach to solve the problem.



(a) Symmetric polling system ($n = 10$ and $T = 40$).

(b) Dependable workstation cluster ($n = 64$ and $T = 80$).

Fig. 2. Verification time as a function of the half-width of the indifference region for different error bounds. The dashed line in each graph represents the verification time for the numerical approach.

We can adjust the accuracy for the numerical solution method by varying the parameter ϵ, but increasing or decreasing ϵ has very little effect on the verification time as shown by Reibman and Trivedi [20] and Malhotra et al. [18]. This means that the numerical solution method can provide very high accuracy without much of a performance degradation, while the statistical solution method is well suited if a quick answer with some uncertainty is more useful. This suggest that statistical techniques can be quite useful as a *first* resort, instead of a last resort as often suggested.

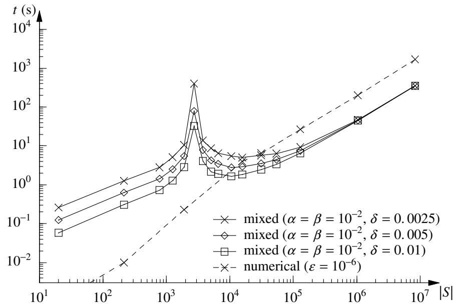


Fig. 3. Verification time as a function of state space size for the grid world example, with $T_1 = 100$ and $T_2 = 7$.

Mixing Solution Techniques. Finally, we present some results for the grid world case study, where the CSL property has nested probabilistic operators. We can see in Fig. 3 that the mixed approach shares performance characteristics with both approaches, outperforming the pure numerical solution method for larger state spaces.

5 Discussion

In this paper, we have empirically compared numerical and statistical solution techniques for probabilistic model checking on case studies taken from the literature on performance evaluation and probabilistic model checking. We focused our attention on time-bounded properties as these are the type of properties most suited for statistical methods (the time-bound provides a natural limit for simulations).

The nature of CSL formulae allows us to use statistical hypothesis testing instead of *estimation* (we only need to know if the probability of a path formula holding is above or below some threshold). The use of sequential acceptance sampling allows the statistical approach to adapt to the difficulty of the problem: verifying a property $\mathcal{P}_{\geq p}(\phi)$ in a state s takes more time if the true probability of ϕ holding in s is close to the threshold p. This can give a clear edge for statistical methods over numerical approaches for model checking CSL formulae. Most previous assessments of statistical techniques (see, e.g., [21]) are based on parameter estimation problems, which are clearly harder in that they typically require a large number of samples. Our results show that the intuition from earlier studies does not necessarily carry over to CSL model checking. Instead of a last resort, statistical solution methods can be seen as a first resort providing a result rapidly, for example during system prototyping when high accuracy may not be a priority, while numerical techniques often remain superior when very high accuracy is required.

Our results are otherwise in line with known complexity results for the two techniques. We show a linear complexity in the time-bound for both approaches. Our results also confirm that statistical methods scale better with the size of the state space, but that high accuracy comes at a greater price than for numerical methods.

The case studies we considered in this paper were all CTMCs. For more complex models with general distributions, such as semi-Markov processes, numerical methods rely on even more elaborate techniques for verifying time-bounded properties (see e.g. [13]). A statistical approach, on the other hand, would work just as well for semi-Markov processes (assuming, of course, that samples from the distributions used in the model can be generated in roughly the same amount of time as samples from the exponential distribution).[5]

References

[1] Alur, R. and Henzinger, T. A. Reactive modules. *Formal Methods in System Design*, 15(1):7–48, 1999.

[2] Aziz, A., Sanwal, K., Singhal, V., and Brayton, R. Verifying continuous time Markov chains. In *Proc. 8th International Conference on Computer Aided Verification*, volume 1102 of *LNCS*, pages 269–276. Springer, 1996.

[3] Aziz, A., Sanwal, K., Singhal, V., and Brayton, R. Model-checking continuous-time Markov chains. *ACM Transactions on Computational Logic*, 1(1):162–170, 2000.

[4] Baier, C., Haverkort, B. R., Hermanns, H., and Katoen, J.-P. Model checking continuous-time Markov chains by transient analysis. In *Proc. 12th International Conference on Computer Aided Verification*, volume 1855 of *LNCS*, pages 358–372. Springer, 2000.

[5] Baier, C., Haverkort, B. R., Hermanns, H., and Katoen, J.-P. Model-checking algorithms for continuous-time Markov chains. *IEEE Transactions on Software Engineering*, 29(6):524–541, 2003.

[6] Bryant, R. E. Graph-based algorithms for Boolean function manipulation. *IEEE Transactions on Computers*, C-35(8):677–691, 1986.

[7] Buchholz, P. A new approach combining simulation and randomization for the analysis of large continuous time Markov chains. *ACM Transactions on Modeling and Computer Simulation*, 8(2):194–222, 1998.

[8] Fox, B. L. and Glynn, P. W. Computing Poisson probabilities. *Communications of the ACM*, 31(4):440–445, 1988.

[9] Fujita, M., McGeer, P. C., and Yang, J. C.-Y. Multi-terminal binary decision diagrams: An efficient data structure for matrix representation. *Formal Methods in System Design*, 10(2/3):149–169, 1997.

[10] Haverkort, B. R., Hermanns, H., and Katoen, J.-P. On the use of model checking techniques for dependability evaluation. In *Proc. 19th IEEE Symposium on Reliable Distributed Systems*, pages 228–237. IEEE Computer Society, 2000.

[5] ProVer, an experimental tool for verifying time-bounded CSL properties of general discrete event systems, is available at www.cs.cmu.edu/~lorens/prover.html.

[11] Hermanns, H., Meyer-Kayser, J., and Siegle, M. Multi terminal binary decision diagrams to represent and analyse continuous time Markov chains. In *Proc. 3rd International Workshop on the Numerical Solution of Markov Chains*, pages 188–207. Prensas Universitarias de Zaragoza, 1999.

[12] Ibe, O. C. and Trivedi, K. S. Stochastic Petri net models of polling systems. *IEEE Journal on Selected Areas in Communications*, 8(9):1649–1657, 1990.

[13] Infante López, G. G., Hermanns, H., and Katoen, J.-P. Beyond memoryless distributions: Model checking semi-Markov chains. In *Proc. 1st Joint International PAPM-PROBMIV Workshop*, volume 2165 of *LNCS*, pages 57–70. Springer, 2001.

[14] Jensen, A. Markoff chains as an aid in the study of Markoff processes. *Skandinavisk Aktuarietidskrift*, 36:87–91, 1953.

[15] Katoen, J.-P., Kwiatkowska, M., Norman, G., and Parker, D. Faster and symbolic CTMC model checking. In *Proc. 1st Joint International PAPM-PROBMIV Workshop*, volume 2165 of *LNCS*, pages 23–38. Springer, 2001.

[16] Kwiatkowska, M., Norman, G., and Parker, D. PRISM: Probabilistic symbolic model checker. In *Proc. 12th International Conference on Modelling Techniques and Tools for Computer Performance Evaluation*, volume 2324 of *LNCS*, pages 200–204. Springer, 2002.

[17] Kwiatkowska, M., Norman, G., and Parker, D. Probabilistic symbolic model checking with PRISM: A hybrid approach. In *Proc. 8th International Conference on Tools and Algorithms for the Construction and Analysis of Systems*, volume 2280 of *LNCS*, pages 52–66. Springer, 2002.

[18] Malhotra, M., Muppala, J. K., and Trivedi, K. S. Stiffness-tolerant methods for transient analysis of stiff Markov chains. *Microelectronics and Reliability*, 34(11):1825–1841, 1994.

[19] Parker, D. *Implementation of Symbolic Model Checking for Probabilistic Systems.* PhD thesis, University of Birmingham, 2002.

[20] Reibman, A. and Trivedi, K. S. Numerical transient analysis of Markov models. *Computers & Operations Research*, 15(1):19–36, 1988.

[21] Shanthikumar, J. G. and Sargent, R. G. A unifying view of hybrid simulation/analytic models and modeling. *Operations Research*, 31(6):1030–1052, 1983.

[22] Stewart, W. J. A comparison of numerical techniques in Markov modeling. *Communications of the ACM*, 21(2):144–152, 1978.

[23] Tiechroew, D. and Lubin, J. F. Computer simulation—discussion of the techniques and comparison of languages. *Communications of the ACM*, 9(10):723–741, 1966.

[24] Wald, A. Sequential tests of statistical hypotheses. *Annals of Mathematical Statistics*, 16(2):117–186, 1945.

[25] Younes, H. L. S., Musliner, D. J., and Simmons, R. G. A framework for planning in continuous-time stochastic domains. In *Proc. Thirteenth International Conference on Automated Planning and Scheduling*, pages 195–204. AAAI Press, 2003.

[26] Younes, H. L. S. and Simmons, R. G. Probabilistic verification of discrete event systems using acceptance sampling. In *Proc. 14th International Conference on Computer Aided Verification*, volume 2404 of *LNCS*, pages 223–235. Springer, 2002.