Efficient and Scalable Steady-state Dependability Verification

Diana El Rabih and Nihal Pekergin
LACL, University of Paris-Est Créteil,
61 avenue Général de Gaulle 94010, Créteil, France
Email : delrabih@u-pec.fr, nihal.pekergin@u-pec.fr

***Abstract*—We have proposed to perform statistical model checking by combining perfect sampling and statistical hypothesis testing based on single sampling plan method in order to verify steady-state formulas. This approach allows us to consider very large monotone models and to verify rare event properties efficiently. In this paper, we extend our proposed approach by implementing different statistical methods in our verification engine and by comparing their efficiency when we verify steady-state dependability properties for large non monotone models. We show that SPRT statistical method is generally more efficient than the other statistical methods. Moreover, we show that our statistical verification approach is efficient and scalable when we consider large non monotone models.**

***Index Terms*—Statistical model checking, Perfect simulation, Dependability verification, Continuous Stochastic Logic (CSL)**

I. Introduction

Probabilistic model checking is an extension of the formal verification methods for systems exhibiting stochastic behavior. The system model is usually specified as a state transition system, with probabilities attached to transitions, for example Markov chains. A wide range of quantitative performance, reliability, and dependability measures can be specified using temporal logics such as Continuous Stochastic Logic (CSL) defined over Continuous Time Markov Chains (CTMC) [2] and Probabilistic Computational Tree Logic (PCTL) defined over Discrete Time Markov Chains (DTMC) [2]. There are two distinct approaches to perform probabilistic model checking: the numerical model checking based on the computation of transient-state or steady-state distributions of the underlying Markov chain and the statistical model checking based on statistical methods and on sampling by means of discrete event simulation or by measurement. Statistical model checking techniques constitute an interesting alternative to numerical model checking techniques for large scale systems. In the last years, different statistical model checkers have been proposed [6][15][20] especially for properties specified by time-bounded until formulas. In the statistical model checker MRMC [8] statistical model checking of CSL steady-state property has been also considered.

We have proposed in [13][14] to perform statistical probabilistic model checking by combining perfect simulation and statistical hypothesis testing based on the single sampling plan method in order to check steady-state properties of large Markovian models. Perfect simulation is an extension of Monte Carlo Markov Chains (MCMC) methods allowing to obtain exact steady-state samples of the underlying Markov chain thus it avoids the burn-in time problem to detect the steady-state. Propp and Wilson have designed the algorithm of coupling from the past to perform perfect simulation [9]. A web page dedicated to this approach is maintained by them (http://research.microsoft.com/en-us/um/people/dbwilson/exact/). As a perfect sampler, we use ψ^2 proposed in [18], designed for the steady-state evaluation of various monotone queueing networks [19]. This tool [18] permits to simulate the stationary distribution or directly a cost function or a reward of large Markov chains. The significant advantage of perfect sampling is that it provides an *unbiased* sampling of the steady-state distribution, hence the accuracy of the verification result only belongs to the statistical testing. In other words, we ensure the correctness of our results considering a specified precision level. We have compared in [10][11][12], the numerical model checker PRISM [7], the statistical module of MRMC [8] and our statistical verification engine when they are applied to the verification of steady-state properties for very large models. We have shown the efficiency and the scalability of our approach to consider very large monotone models and to verify rare event properties efficiently.

In this paper, we extend our proposed approach by implementing in our verification engine other statistical methods existing in the litterature and by comparing their efficiency when we verify steady-state dependability properties. In fact, we consider two non-monotones queueing networks, such as network of queues with negative clients, and with coxian phase-type servers to show the efficiency and the scalability of our proposed approach also in the case of non monotone models. This paper is organized as follows: Section 2 briefly presents the temporal logic CSL, the perfect sampling and our proposed approach for statistical verification based on perfect sampling. We give a brief introduction of the implemented statistical methods in Section 3. Section 4 is devoted to the case studies. First we present the models. Next, we compare and analyze the results of our experiments. Finally, in Section 5 we summarize the conclusions and provide the future works.

II. Statistical Model Checking by Perfect Sampling

A. Continuous stochastic logic (CSL)

CSL is a branching-time temporal logic with state and path formulas and it is a powerful mean to state properties over

CTMCs. Thus it is useful to specify and to verify performance and dependability measures as logical formulas over CTMCs [1]. The steady-state operator (formula) $\psi = \mathcal{S}_{\bowtie\theta}(\varphi)$ lets us to analyze the long-run behaviour of the system. The steady state formula $\mathcal{S}_{\bowtie\theta}(\varphi)$ asserts that the steady-state probability for the set of the states satisfying φ meets the bound $\bowtie \theta$, where θ is a probability threshold, $\bowtie$ a comparison operator, for example $\bowtie \in \{<, >, \leq, \geq\}$, φ is a state formula (a boolean expression of state properties).

B. Perfect sampling and statistical verification

Propp and Wilson [9] have introduced the perfect/exact sampling method, which is based on the backward coupling, also called the coupling from the past: by coming from a distant time $-\tau$ sufficiently far in the past, if all trajectories (trajectories that come from all possible initial states in $\mathcal{X}$ at time $-\tau$) are coupled in one state at time 0, then the sampled state is exactly distributed according to the stationary distribution. The backward coupling provides steady-state sample in a controlled finite number of steps, that could not be obtained by a forward coupling scheme unless the model have a strong stationary time, which is rare in our examples [17]. Let $\{X_n\}_{n\in\mathbb{N}}$ be an irreducible and aperiodic discrete time Markov chain with a finite space $\mathcal{X}$ and a transition matrix $P = p_{i,j}$. Let π denote the steady-state distribution of the chain: $\pi = \pi P$. The evolution of the system can be given by a stochastic recurrence:

$$X_{n+1} = \eta(X_n, e_{n+1}) \tag{1}$$

with $\{e_n\}$ an independent and identically distributed sequence of events ($e_n \in \epsilon$). The transition function $\eta : \mathcal{X} \times \epsilon \to \mathcal{X}$ verifies the property that $Pr(\eta(i,e) = j) = p_{i,j}$ for every pair of states (i, j) and each random event e. An execution of the Markov chain is defined by an initial state x_0 and an sequence of events. The sequence of states given by Eq. 1 is called a trajectory. Trajectories are generated with the same sequence of events and if at time $t = 0$, two trajectories are in the same state, we say that they couple. The backward coupling is especially efficient when the underlying system is monotone. When the system is not monotone it is shown in [3] that the backward coupling can also be efficient. Given a partial order $\preceq$ on $\mathcal{X}$, an event e is said to be **monotone** if it preserves the partial ordering $\preceq$ on $\mathcal{X}$:

$$\forall (x, y) \in \mathcal{X} \quad x \preceq y \;\Rightarrow\; \eta(x, e) \preceq \eta(y, e) \tag{2}$$

If all events are monotone, the global system is said to be monotone. According to an order $\preceq$ on $\mathcal{X}$, there exists a set $\mathcal{M}_{\preceq} \subset \mathcal{X}$ of extremal states (maximal and minimal states). When a Markov chain is monotone, all trajectories issued from $\mathcal{X}$ are always bounded by trajectories issued from $\mathcal{M}_{\preceq}$. Thus, it is sufficient to compute trajectories issued from $\mathcal{M}_{\preceq}$ since when they couple, global coupling also occurs. As the size of $\mathcal{M}_{\preceq}$ is usually drastically smaller than the size of $\mathcal{X}$, monotone perfect sampling significantly improves the sampling time [9]. Efficiency of simulations is also improved by functional perfect sampling [19]. The algorithm samples a reward value, according to a user defined reward function $r : \mathcal{X} \to \mathcal{R}$; The algorithm stops when all trajectories are in a set of states at time 0 that belongs to the same reward value (going further in the past will inevitably couple in a state that belong to this reward value). To combine monotone and functional perfect sampling, the reward function r must be monotone, that is $x \preceq y \Rightarrow r(x) \preceq r(y)$. As $|\mathcal{R}|$ is smaller than $|\mathcal{X}|$, this technique may lead to an important reduction of the coupling time. In a property verification context, since we focus on reward functions that correspond to properties we want to check, $\mathcal{R} = \{0, 1\}$. In our statistical verification method we propose to apply functional perfect sampling, so at time 0, we test if the rewards are coupled at reward 0 or 1. In other words, we test if it is a positive or negative sample. Thus we associate the reward $r_\varphi(x)$ to each state $x \in \mathcal{X}$ for a given property φ: $r_\varphi(x) = 1$ if x satisfies φ, otherwise $r_\varphi(x) = 0$. Note that, as the reward function is monotone, values 0 and 1 cover contiguous zones of the state-space. Then, an interesting phenomenon happens when the property to be checked has a small set of positive states $\{x \in \mathcal{X} | r_\varphi(x) = 1\}$ (φ corresponds to a rare property / event): coupling frequently occurs in reward value 0 and the coupling time is very short. Moreover, if $|\{x \in \mathcal{X} | r_\varphi(x) = 1\}|$ does not depend on $\mathcal{X}$ (case of saturation properties for example), then the performance of perfect sampling algorithm will be as good for very large state-spaces as for small ones. This intuition is validated by results of Section 4.

The decision method tests if φ is satisfied (positive sample) or not (negative sample) on each generated sample path by counting the number of positive samples. Then it provides decision either *Yes* if the number of positive samples is greater or equal to m (ψ is satisfied) or *No* otherwise (ψ is not satisfied). The input parameters of the algorithm are: the model defined by a labelled CTMC, M, the property φ (to be verified on each sample), the threshold parameter θ, the indifference region parameter δ, and α, β for the strength of statistical hypothesis testing. In our work, we consider ergodic Markov chains $\mathcal{M}$, hence there is a unique steady-state distribution independent of the initial state. The satisfaction property is assigned to the model but not to an inital state. (we check whether the underlying model M satisfies the steady-state formula or not). $\mathcal{M} \models \mathcal{S}_{\bowtie\theta}(\varphi)$, if the property specified by the steady-state operator $\mathcal{S}$ is satisfied by the model $\mathcal{M}$. Note that the verification of $\mathcal{S}_{\geq\theta}(\varphi)$ is the same as $\mathcal{S}_{<1-\theta}(\neg\varphi)$ and also is the same as $\neg\mathcal{S}_{<\theta}(\varphi)$.

III. Statistical methods

The statistical decision method we have used in [11][12] when performing our statistical hypothesis testing is inspired from the Single Sampling Plan (SSP) method. In this section we present the different statistical methods we implement in our statistical verification engine.

A. Current methods

a) Statistical hypothesis testing

Suppose that we have generated n samples (simulations), and a sample X_i is a positive sample $(X_i = 1)$ if it satisfies φ and negative $(X_i = 0)$ otherwise. X_i is a random variable with

Bernoulli distribution with parameter p. Thus the probability to obtain a positive sample is p. In practice, two thresholds, p_0 and p_1 are defined in terms of the probability threshold θ, and the half-width δ of the indifference region: p_0 = θ + δ and p_1 = θ - δ. Then instead of testing $H : p \geq \theta$ against $K : p < \theta$, we test $H_0 : p \geq p_0$ against $H_1 : p \leq p_1$. In fact, the strength of the statistical test was determined by two error bounds, α and β, where α is a bound on the probability of accepting H_1 when H_0 holds (known as a type I error, or false negative) and β is a bound on the probability of accepting H_0 when H_1 holds (a type II error, or false positive). There are several methods for statistical hypothesis testing decision with constraints on error bounds (α, β) [22][21][16]:

a.1) Single Sampling Plan (SSP): It is based on the acceptance sampling with fixed sample size (n): if $\sum_{i=1}^{n} X_i \geq m$, then H_0 is accepted otherwise H_1 is accepted, where m is the acceptance threshold. The hypothesis H_1 will be accepted with probability $F(m, n, p)$ and the null hypothesis H_0 will be accepted with the probability $1 - F(m, n, p)$, where $F(m, n, p)$ is a binomial distribution: $F(m, n, p) = \sum_{i=1}^{m} C(n, i)p^i(1-p)^{n-i}$ with $C(n, i)$ is the combination of i from n. It is required that the probability of accepting H_1 when H_0 holds is at most α, and the probability of accepting H_0 when H_1 holds is at most β. These constraints can be illustrated as below:

- Pr[H_1 is accepted $|$ H_0 is true] $\leq \alpha$, which implies $F(m, n, p_0) \leq \alpha \qquad (C1)$
- Pr[H_0 is accepted $|$ H_1 is true] $\leq \beta$, which implies $1 - F(m, n, p_1) \leq \beta \qquad (C2)$

The sample size n and the acceptance threshold m must be chosen under these constraints and their formulas for optimal performance are given in [22].

a.2) Sequential Single Sampling Plan (SSSP): If we use a single sampling plan (n, m) and the sum of the first i observations, d_i=$\sum_{j=1}^{i} X_j$, $i < n$, is already greater than m, then we can accept H_0 without making further observations. Conversely, if $d_i + n - i \leq m$, regardless of the outcome of the remaining $n - i$ observations we already know that the sum of n observations will not exceed m, then we can safely accept H_1 after making only i observations. In the modified test procedure, after each observation, we decide whether sufficient information is available to accept either of the two hypotheses or additional observations are required.

a.3) Sequential Probability Ratio Test (SPRT): This method is based on the sequential probability ratio test [21][22]: after making the i^{th} simulation (generating the i^{th} sample), one computes the following quotient:

$$q_i = \prod_{j=1}^{i} \frac{Pr[X_j = x_j \mid p = p_1]}{Pr[X_j = x_j \mid p = p_0]} = \frac{p_1^{d_i}(1-p_1)^{i-d_i}}{p_0^{d_i}(1-p_0)^{i-d_i}}$$

where d_i denoting the number of positive samples. H_0 is accepted if $q_i \leq B$, and H_1 is accepted if $q_i \geq A$. Finding A and B with a given strength α, β is non trivial, in practice A is chosen as (1-β)/α and B as β/(1-α). Then a new test whose strength is (α^*, β^*) is obtained, but such that $\alpha^* + \beta^* \leq \alpha + \beta$, meaning that either $\alpha^* \leq \alpha$ or $\beta^* \leq \beta$. In practice, it is often found that both inequalities hold. When implementing the sequential probability ratio test, it is computationally more practical to work with the logarithm of q_i. Then we accept H_0 if $f_m \leq log\frac{\beta}{1-\alpha}$, we accept H_1 if $f_m \geq log\frac{1-\beta}{\alpha}$.

Note that, the sample size for a sequential test is a random variable, meaning that the required number of observations can vary from one use of such a test to another. Furthermore, the expected sample size typically depends on the unknown parameter p, so we cannot report a single value as was the case for acceptance sampling with fixed-size samples. The expected sample size varies with the distance of p from the indifference region (p_1, p_0). It tends to be largest when p is close to the center of the indifference region, and decreases the further away p is from the indifference region.

b) Statistical estimation

An alternative statistical solution method, based on estimation instead of hypothesis testing [6]. This approach uses n observations $x_1, ..., x_n$ to compute an estimate of p: p'=$\frac{\sum_{i=1}^{n} X_i}{n}$. The estimate is such that $Pr[|p' - p| < \delta] \geq 1 - \alpha$ $(E4)$. Using a result derived by Hoeffding [[21], Theorem 1], it can be shown that n = $\lceil \frac{1}{2\delta^2} log \frac{2}{\alpha} \rceil$ $(E5)$
is sufficient to satisfy $(E4)$. If we accept ψ as true when $p' \geq \theta$ and reject ψ as false otherwise, then it follows from $(E4)$ that the answer is correct with probability at least 1 - α if either s $\models \psi$ or s $\not\models \psi$ holds. Consequently, the verification procedure satisfies $(C1)$ and $(C2)$ with β = α. As with the solution method based on hypothesis testing, a definite answer is always generated (there is no undecided results).

c) Confidence interval

Another alternative statistical solution method based on confidence intervals has been proposed in [8]. To check whether $s \models P_{>\theta}(\varphi)$, an estimate $\tilde{p}$ of the probability p starting in s is determined using standard discrete event simulation techniques. Let ξ be the user-specified confidence of the result and $\delta^{'}$ the maximum width of the confidence interval. The probability of obtaining a correct answer to the model checking problem $s \models P_{>\theta}(\varphi)$ is now guaranteed to be at least ξ provided $\delta^{'} \leq |\theta - \tilde{p}|$. In this solution method, a slight adaptation of standard sequential confidence intervals is exploited in which the sample size and simulation depth can be adapted on demand. Although $\delta^{'} > |\theta - \tilde{p}|$, this solution method provides more accurate answers as its algorithm first simulates until the confidence interval is tighter than $\delta^{'}$ and then continues simulation until it reaches the definite answer to the model checking problem. This strategy increases the accuracy because the width of the resulting confidence interval can be much smaller than $\delta^{'}$. The penalty for this increased accuracy is an increase in the simulation times thus larger model-checking times.

B. Performance comparison of statistical methods

The estimation-based approach had been compared with the approach based on hypothesis testing in [21], by considering m = $\lfloor n\theta + 1 \rfloor$ and d=np'=$\sum_{i=1}^{n} x_i$. It had been demonstrated that $p' \geq \theta \iff d > m$. This means that the estimation-based approach can be interpreted as a single sampling plan (n, m). Therefore the approach proposed in [22], when using a single sampling plan, will always be at least as efficient as the estimation-based approach. In fact, it will be more efficient because: (i) the sample size is derived using the true underlying distribution, (ii) m is not restricted to be $\lfloor n\theta + 1 \rfloor$, and (iii) $\beta = \alpha$ can be accommodated. The last property, in particular, is important when dealing with conjunctive and nested probabilistic statements. The advantage of hypothesis testing is demonstrated numerically in [21]. Note, also, that the SPRT method often can be used to improve efficiency for the approach based on hypothesis testing. In fact, if a single sampling plan is used with strength (α, β) and indifference region of half-width δ, then the sample size n is roughly proportional to log α and log β and inversely proportional to δ^2 [22]. Using the SPRT method instead of a single sampling plan can reduce the expected sample size by orders of magnitude in most cases, although the SPRT method is not guaranteed always to be more efficient.

On the other hand, the method proposed by Sen et al. in [16] is not more efficient than the methods proposed by Younes et al. in [22]. In fact, Sen et al. manually selected the sample sizes for their single sampling plans. The selected sample sizes are not sufficient to achieve the same strength as used to produce the results for the SPRT method reported by Younes et al. in [22]. Finally, the confidence intervals statistical technique requires to use confidence interval of the width $< \delta$, whereas under the same conditions in hypothesis testing we would have to use the indifference region of the width less than only $2.\delta$. This can cause confidence intervals algorithms to require more samples than needed for the ones based on the hypothesis testing.

C. Statistical model checking complexity

The time complexity of any statistical solution method for probabilistic model checking can be understood in terms of two main factors: the number of observations (sample size) required to reach a decision, as well as the time required to generate each observation that depends of perfect simulation effort (coupling time). If an observation involves the verification of a path formula over a sample trajectory then the time complexity depends also of the length of trajectory prefixes (in terms of state transitions) required to determine if a path formula holds. The sample size depends on the method used for verifying probabilistic statements, as well as the desired strength (α, β) and of θ and δ. In fact, the sample size for a single sampling plan SSP is approximately proportional to the logarithm of α and β, and inversely proportional to δ^2 . If we use a sequential test SPRT, then the expected sample size also depends on the unknown probability measure p of the set of trajectories that satisfy φ. Moreover, the perfect simulation effort (coupling time) can be both model and implementation dependent, then it can be state space dependent, but models often have structure (monotone structure) [4] that can be exploited by the simulator to avoid such dependence. The length of trajectories depends on model characteristics and the property that is being verified but may be independent of the size of the state space. The space complexity of statistical probabilistic model checking is generally modest. It is needed to store the current state of a sample trajectory when generating an observation for the verification of a probabilistic statement, and this typically requires O(log $|\mathcal{X}|$) space where $|\mathcal{X}|$ is the size of state space. For systems that do not satisfy the Markov property, it may also be needed to store additional information to capture the execution history during simulation.

IV. Experimental Study

We now evaluate two non monotone models, taken from Ψ^2 and PRISM benchmarks, on which we will base our efficiency and scalability comparison. In fact, we verify the steady-state formula for these two case studies using the numerical verification approach implemented in PRISM tool and our statistical verification approach implemented in Ψ^2 tool using different statistical solution methods (Section 3), by varying the problem size (state space size related to the maximal queue capacity). We illustrate the statistical verification time ($\approx N_{samp}$*coupling time) in seconds, where N_{samp} is the sample size, for these case studies as a function of the maximal queue capacity (state space size) and we determine the memory limit for each case when using the verification tools. Since the considered Markovian models are ergodic (by construction), thus the steady-state probabilities are independent of the initial state. Thus, the considered steady-state formula is satisfied or not whatever the initial states.

a) Negative clients queueing network

We consider the following queueing model with both positive and negative clients (Figure 1). The non-monotonicity of this model (negative clients) is shown and its perfect sampling by envelope functions is given in [3]. We have implemented this non monotone model as a Ψ^2 model as explained in [3] and we have validated the correctness of our implementation. In fact, queueing models with negative clients, have found applications in computer communications and manufacturing settings. When a negative client arrives at the queue, it has the effect of a signal, which kills ordinary (positive) clients in the node. An example of a queueing system with both positive and negative clients (jobs) is computer networks with virus infection, which deletes jobs or failures, which causes other failures and removes jobs. Let N_{max} to be the maximal

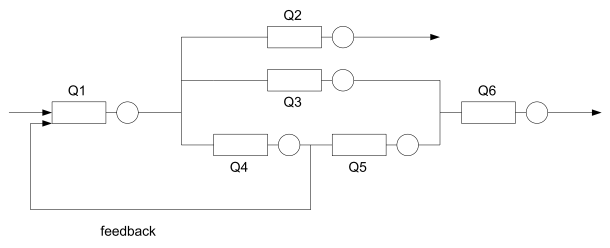

Fig. 1. Negative clients queueing network

capacity of each queue then the state space is $O((N_{max}+1)^6)$. Jobs arrive from exterior at the first queue with rates λ_1^+ (positive clients) and λ_1^- (negative clients), and exit the system from the second queue with rate μ_1 and from the sixth queue with rate μ_2. Jobs arrive also to the first queue from feedback link with rate λ_{feed}^+ and λ_{feed}^-. Jobs arrive to the i^{th} queue where $2 \leq i \leq 6$ with rates λ_i^+ (positive clients) and λ_i^- (negative clients). Also negative clients can arrive from exterior to the i^{th} queue where $2 \leq i \leq 6$ with rates κ_i^-. Let x_i denote the number of jobs currently in queue i. We define the atomic proposition that one queue of the system is full with the formula *negsysfull* = $(x_1 = N_{max})\ \vee (x_2 = N_{max})\ \vee (x_3 = N_{max})\ \vee (x_4 = N_{max})\ \vee (x_5 = N_{max})\ \vee (x_6 = N_{max})$. Based on this atomic proposition, we check the following *Steady-state formula:* $S_{\leq\theta}$ (*negsysfull*) to check whether the probability that the system is full in steady-state is less than θ or not.

b) Tandem Queueing Network with coxian phase (TQN) The TQN model (Figure 2) is taken from PRISM benchmark that consists of an $M/Cox_2/1$ queue sequentially composed with an M/M/1 queue . In [11], we have implemented this non monotone model as a Ψ^2 model by using non monotone techniques (envelope function) such as defined in [3] and we have validated the correctness of our implementation. The non-monotonicity of this model is shown in [5][11]. We consider 4 TQN connected in series then our considered system consists of 4 $M/Cox_2/1$ queues. Let N_{max} be the maximal capacity of each queue then the state space is $O((N_{max}+1)^2)$ for each TQN. In each TQN, jobs arrive at the first queue with rate λ, and exit the system from the second queue with rate κ. If the first queue is not empty and the second queue is not full, then jobs are routed from the first to the second queue. In each TQN, the routing time is governed by a two-phase Coxian distribution with parameters μ_1, μ_2, and a. Here, μ_i is the exit rate for the i^{th} phase of the distribution, and 1 - a is the probability of skipping the second phase. Let x_j denote the number of jobs currently in queue j, and $x_{ph_k} \in \{1, 2\}$, for $1 \leq k \leq 4$, denote the current phase of the Coxian distribution. We define the atomic proposition that one TQN component of the overall system is full with the formula *sys-full* = $[(x_1 = N_{max})\ \wedge (x_2 = N_{max})\ \wedge\ (x_{ph_1} = 2)] \vee [(x_3 = N_{max})\ \wedge (x_4 = N_{max})\ \wedge\ (x_{ph_2} = 2)] \vee [(x_5 = N_{max})\ \wedge\ (x_6 = N_{max})\ \wedge\ (x_{ph_3} = 2)] \vee [(x_7 = N_{max})\ \wedge (x_8 = N_{max})\ \wedge\ (x_{ph_4} = 2)]$. Based on this atomic proposition,

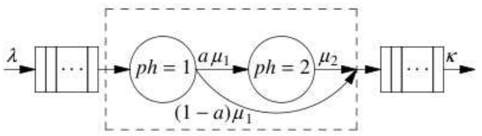

Fig. 2. Tandem queueing network with Coxian phase

we check the following *Steady-state formula:* $S_{\leq\theta}$ (*sys-full*) to check whether the probability that the system is full in steady-state is less than θ or not.

A. Experimental results

a) Negative clients network verification results: We consider λ_1^+=0.8, λ_1^-=0.2, λ_{feed}^+=0.7, λ_{feed}^-=0.3, all service rates will be state-independent with rate $\mu_1 = \mu_2 = 1$; λ_i^+=0.6, λ_i^-=0.4 and κ_i^-=0.1 for $2 \leq i \leq 6$. We give in Table I for $\theta = 0.001$ and $\epsilon = 10^{-4}$, the verification time for the considered steady-state formula $S_{<\theta}$ (*negsysfull*) by using PRISM Hybrid engine and Jacobi iterative method. Also we give in the same table for $\theta = 0.001$, $\delta = 10^{-4}/2$ respectively, and $\alpha = \beta = 10^{-2}$ the verification time for the same steady-state formula $S_{<\theta}$ (*negsysfull*) by using statistical verification methods implemented in Ψ^2 (Section 3). In fact, for $N_{max} = 21$ we obtain an out of memory message with PRISM. In all of the tables we denote by:

$PRISM$: numerical verification time in seconds for the steady-state formula by using PRISM hybrid engine.

$outm$: an out of memory message in PRISM tool.

The statistical verification time in seconds is given by combining with statistical techniques given in Section 3: $\Psi^2(SSP)$ for SSP, $\Psi^2(SPRT)$ for SPRT, $\Psi^2(SEst)$ for statistical estimation, $\Psi^2(CI)$ for confidence interval.

N_{max}	$\lvert\mathcal{X}\rvert$	$PRISM$	$\Psi^2(SSP)$	$\Psi^2(SPRT)$	$\Psi^2(SEst)$	$\Psi^2(CI)$
2	$7.29 * 10^2$	0.04	4.1	1.3	5.5	6.8
3	$4.09 * 10^3$	0.05	5.4	2.1	8.6	9.1
5	$4.66 * 10^4$	0.10	9.4	4.4	15.6	21.5
7	$2.62 * 10^5$	1.32	14.4	9.6	19.7	25.8
9	$1.00 * 10^6$	98.67	24.2	15.3	29.3	34.9
12	$4.82 * 10^6$	276.6	33.2	20.1	39.5	43.4
14	$1.13 * 10^7$	9213	42.6	29.7	54.7	67.1
21	$1.13 * 10^8$	$outm$	65.9	42.2	73.1	81.7
99	$1.00 * 10^{12}$	$outm$	98.1	53.1	126.1	155.4
999	$1.00 * 10^{18}$	$outm$	365.3	173.3	422.4	485.2
9999	$1.00 * 10^{24}$	$outm$	1315	633	1713	1929

TABLE I
NEGATIVE CLIENTS NETWORK: VERIFICATION TIME AS A FUNCTION OF STATE SPACE SIZE $\lvert\mathcal{X}\rvert$ FOR $S_{<0.001}$ (*negsysfull*)

b) Tandem network with coxian phase (4 TQN) verification results: For numerical application, for each TQN in the overall system (4 TQN in series) we consider $\lambda = 4 \times N_{max}$, $\mu_1 = 2$, $\mu_2 = 2$, $a = 0.1$ and κ =4. We give in Table II for $\theta = 0.001$ and for $\epsilon = 10^{-4}$, the verification time for the considered steady-state formula $S_{<\theta}$ (*sys-full*) by using PRISM Hybrid engine and Jacobi iterative method. Also we give in the same table for $\theta = 0.001$, $\delta = 10^{-4}/2$ respectively, $\alpha = \beta = 10^{-2}$, the verification time for the same steady-state formula $S_{<\theta}$ (*sys-full*) by using statistical verification methods implemented in Ψ^2 (Section 3). In fact, for $N_{max} = 10$ we obtain an out of memory message with PRISM.

N_{max}	$\lvert\mathcal{X}\rvert$	$PRISM$	$\Psi^2(SSP)$	$\Psi^2(SPRT)$	$\Psi^2(SEst)$	$\Psi^2(CI)$
2	$6.5 * 10^3$	0.4	7.1	4.22	8.5	9.8
3	$6.5 * 10^4$	0.5	9.4	5.12	11.4	17.1
4	$3.9 * 10^5$	1.93	17.9	8.14	20.3	22.8
5	$1.6 * 10^6$	33.2	21.8	12.3	23.6	26.4
6	$5.7 * 10^6$	150.6	34.3	21.3	39.6	44.2
7	$1.7 * 10^7$	290.6	53.2	34.1	60.9	71.5
8	$4.3 * 10^7$	476.6	78.6	57.3	98.7	117.1
9	$1.0 * 10^8$	8615	265.9	153.3	329.1	371.3
10	$2.1 * 10^8$	$outm$	386.6	233.1	422.6	492.6
99	$1.0 * 10^{16}$	$outm$	498.1	263.3	547.1	605.4
999	$1.0 * 10^{24}$	$outm$	565.3	302.1	626.3	715.2
9999	$1.0 * 10^{32}$	$outm$	1415	565.3	1826	2153

TABLE II
TQN: VERIFICATION TIME AS FUNCTION OF STATE SPACE SIZE $\lvert\mathcal{X}\rvert$ FOR $S_{<0.001}$ (*sys-full*)

B. Discussions

In Tables I and II, we have illustrated the statistical verification time ($\approx N_{samp}$*coupling time) in seconds for two non monotone models as a function of the maximal queue capacity (state space size), where N_{samp} is the sample size. In fact, the sample size, N_{samp} is the only factor that varies between the different statistical solution methods, regardless of implementation details. The sample size depends on the method used for verifying probabilistic statements, as well as the desired strength (α, β) and θ and δ. Note that, the coupling time of the perfect simulation varies with the state space size, with the implementation and with the verified property.

In Tables I and II, we show that the Single Sampling Plan (SSP) method is at least as efficient as the statistical estimation method and it will be more efficient since the sample size of the SSP method is derived using the true underlying distribution [21] (Section 3). We show also in these tables that the Single Sampling Plan (SSP) method is more efficient than the confidence intervals method, since the last method requires a smaller width of the confidence interval (Section 3). This can cause confidence intervals method to require more samples than needed for hypothesis testing based method (SSP method).

Moreover, we show in these tables that the Sequential Probability Ratio Test (SPRT) method is more efficient than the Single Sampling Plan (SSP) method. In fact, the sample size for a single sampling plan SSP is approximately proportional to the logarithm of α and β, and inversely proportional to δ^2 [21]. In our work, for SSP method we have determined the sample size using the approximation formulas given in [21]. For sequential test SPRT the expected sample size also depends on the unknown probability measure p of the set of trajectories that satisfy the property φ. In fact, for SPRT method the sample size is computed during the verification process. We show in tables I and II that using SPRT method instead of SSP method can reduce the verification time (depending on sample size) by an order of magnitude in most cases. Thus we show that SPRT statistical method is generally more efficient than the other statistical methods when performing steady state dependability verification for very large models and we show that the hypothesis testing based methods are generally more efficient than the estimation and the confidence intervals based methods. We also see in these tables that our statistical verification approach is efficient and scalable when we consider large non monotone models and it allows us to verify rare event properties efficiently on these models.

Finally, in Tables I and II we have determined the memory limit for each case when using the verification tools. There is no memory limit when using our statistical verification approach implemented in Ψ^2 tool, since the space complexity of statistical model checking is generally modest. In fact, in statistical model checking it is needed to store the current state of a sample trajectory when generating an observation for the verification of a probabilistic statement, and this typically requires O(log $|S|$) space where $|S|$ is the size of state space.

V. Conclusion and future works

In this paper, we extend our proposed approach [12][13][14] by implementing different statistical methods in our verification engine and by comparing their efficiency when we verify steady-state dependability properties for large non monotone models. We show that SPRT statistical method is generally more efficient than the other statistical methods when performing steady state dependability verification for very large models. Moreover, we show that our statistical verification approach is efficient and scalable when we consider large non monotone models and lets us to verify rare event properties efficiently on these models. Also we have found that our statistical verification approach scales better with the state space size and it is faster than PRISM tool especially for large models. In the future, we plan to complete our verification results for the CSL unbounded until formulas [14].

References

[1] A. Aziz, K. Sanwal, V. Singhal, and R. K. Brayton. Model-checking continous-time markov chains. *ACM Trans. Comput. Log.*, 1(1):162–170, 2000.

[2] C. Baier, B. Haverkort, H. Hermanns, and J.P. Katoen. Model-checking algorithms for continuous-time markov chains. *IEEE Trans. Software Eng.*, 29(6):524–541, 2003.

[3] A. Busic, J. M. Vincent, and B. Gaujal. Perfect simulation and non-monotone (markovian) systems. In *VALUETOOLS08*. ACM, 2008.

[4] P. Glasserman and D. Yao. *Monotone Structure in Discrete-Event Systems*. 1994.

[5] G. Gorgo. Envelope perfect sampling of 2-phases coxian service in queueing networks. INRIA. 2010.

[6] T. Hérault, R. Lassaigne, and S. Peyronnet. Apmc 3.0: Approximate verification of discrete and continuous time markov chains. In *QEST06*, pages 129–130. IEEE Computer Society, 2006.

[7] A. Hinton, M. Kwiatkowska, G. Norman, and D. Parker. Prism: A tool for automatic verification of probabilistic systems. In *TACAS06*, volume 3922 of *LNCS*, pages 441–444, 2006.

[8] J. P. Katoen, I. S. Zapreev, E. M. Hahn, H. Hermanns, and D. N. Jansen. The ins and outs of the probabilistic model checker mrmc. In *QEST09*, pages 167–176. IEEE Computer Society, 2009.

[9] D. Propp and J. Wilson. Exact sampling with coupled markov chains and applications to statistical mechanics. *Random Structures and Algorithms*, 9(1 and 2):223–252, 1996.

[10] D. El Rabih, G. Gorgo, N. Pekergin, and J.M. Vincent. Steady state dependability verification for very large systems. lacl. 2010.

[11] D. El Rabih, G. Gorgo, N. Pekergin, and J.M. Vincent. Steady state property verification: a comparison study. In *VECOS10*. eWic, British Computer Society, 2010.

[12] D. El Rabih, G. Gorgo, N. Pekergin, and J.M. Vincent. Steady state property verification for very large systems. In *International Journal of Critical Computer-Based Systems, IJCCBS11 (to appear)*, 2011.

[13] D. El Rabih and N. Pekergin. Statistical model checking for steady state dependability verification. In *DEPEND09*, pages 166–169. IEEE Computer Society, 2009.

[14] D. El Rabih and N. Pekergin. Statistical model checking using perfect simulation. In *ATVA09*, volume 5799 of *LNCS*, pages 120–134, 2009.

[15] K. Sen, M. Viswanathan, and G. Agha. Vesta: A statistical model-checker and analyzer for probabilistic systems. In *QEST05*, pages 251–252. IEEE Computer Society, 2005.

[16] Koushik Sen, Mahesh Viswanathan, and Gul Agha. On statistical model checking of stochastic systems. 3576:266–280, 2005.

[17] J. M. Vincent. Perfect simulation of monotone systems for rare event probability estimation. In *Winter Simulation Conference*, pages 528–537. ACM, 2005.

[18] J. M. Vincent and J. Vienne. ψ^2 a software tool for the perfect simulation of finite queueing networks. In *QEST07*, pages 113–114. IEEE Computer Society, 2007.

[19] J.M. Vincent and C. Marchand. On the exact simulation of functionals of stationary markov chains. *Linear Algebra and its Applications*, 386:285–310, 2004.

[20] H. L. S. Younes. Ymer: A statistical model checker. In *CAV05*, volume 3576 of *LNCS*, pages 429–433, 2005.

[21] H. L. S. Younes. Error control for probabilistic model checking. In *VMCAI06*, volume 3855 of *LNCS*, pages 142–156, 2006.

[22] H. L. S. Younes and R. G. Simmons. Statistical probabilistic model checking with a focus on time-bounded properties. *Inf. Comput.*, 204(9):1368–1409, 2006.