Simulation-Based CTMC Model Checking: An Empirical Evaluation

Joost-Pieter Katoen[a] and Ivan S. Zapreev[b]
[a]RWTH Aachen University, D-52056 Aachen, Germany
[b]CWI, 1098 XG Amsterdam, The Netherlands

Abstract

This paper provides an experimental study of the efficiency of simulation-based model-checking algorithms for continuous-time Markov chains by comparing: MRMC – the only tool that implements (new) confidence-interval-based algorithms for verification of all main CSL formulae; Ymer – that allows for verification of time-bounded and time-interval until using sequential acceptance sampling; and VESTA – that can verify time-bounded and unbounded until by means of simple hypothesis testing. The study shows that MRMC provides the most accurate verification results. Ymer and VESTA, unlike MRMC, have almost constant memory consumption. Ymer requires the least number of observations to assess the model-checking problem, but MRMC is mostly the fastest. This implies that the tools' efficiency does not, so much, depend on sampling but is rather determined by supplementary computations.

I. Introduction

The applicability of probabilistic model checking ranges from areas such as randomised distributed algorithms to planning and AI, security [21], and even biological process modelling [18]. Probabilistic model-checking engines have been integrated in existing tool chains for widely used formalisms such as stochastic Petri nets [6], Statemate [4], the stochastic process algebra PEPA [11], and a probabilistic variant of Promela [2]. Popular logics are Probabilistic CTL (PCTL) [7] and Continuous Stochastic Logic (CSL) [3]. At present, there are several model checkers, such as PRISM [22], MRMC [19], VESTA [23], Ymer [27], and APMC [17], that support verification of finite-state continuous-time Markov chains. The typical kind of properties that they can verify are time-bounded until properties–"Does the probability to reach a certain set of goal states (by avoiding bad states) within a maximal time span exceed 0.5?", unbounded until which is similar to the previous one, but with no time bound, and steady-state–"In equilibrium, does the likelihood to leak confidential information remain below 10^{-4}?"

Probabilistic model checking can be done employing numerical or statistical approaches. The former one, is carried out by symbolic and numerical methods. It typically guarantees a high degree of accuracy, but often requires a lot of intricate computations. The latter approach, is based on sampling and Monte Carlo simulation and allows for a much simpler algorithms. Being statistical in nature, simulations can not guarantee that the verification result is 100% correct, but the approach allows to bound the probability of generating an incorrect answer to the verification problem.

Like in the traditional setting, probabilistic model checking suffers from the state-space explosion: the number of states grows exponentially in the number of system components and cardinality of data domains. This brings a great deal of inefficiency when using numerical model-checking algorithms. Fortunately, the simulation-based approach often overcomes this problem due to simpler algorithms and "on-the-fly" state-space generation.

In this paper we provide a comparative experimental study of the simulation-based model-checking techniques for CSL. This study (empirically) evaluates the three distinct approaches: one, implemented in MRMC, and based on confidence intervals (*c. i.*), see the PhD thesis of Ivan S. Zapreev [31]; another, realised in Ymer, and based on sequential acceptance sampling [30]; and the third one, supported by VESTA, and based on simple hypothesis testing [24].

Let us note that both PRISM and APMC allow for statistical model checking of DTMCs and CTMCs, based on the theoretical results of [8]. The algorithms allow for

This research was performed as part of the MC=MC project financed by the Netherlands Organization for Scientific Research (NWO). We thank Håkan L. S. Younes (Google) for valuable discussions and advising about experimental setup.

model-checking until formulae, by considering the *finite* path prefixes and computing the probability estimates by means of the Chernoff-Hoeffding bounds. The reasons why we did not consider PRISM and APMC in our experiments are as follows: *(i)* when using its simulation engine, PRISM allows to compute the probability estimates, but does not support probability bounds in the formulae; *(ii)* algorithms implemented in MRMC, except for not using Chernoff-Hoeffding bounds, seem to be a generalisation that of [8] in case of knowing structural properties of the Markov chain.

Our experiments are aimed at the following main points: *(i)* the verification time – the required time to verify a formula; *(ii)* the confidence levels – the match between the theoretically guaranteed confidence and the one obtained in practice; *(iii)* the peak memory usage (VSZ) – the maximal amount of virtual memory (RAM + swap) needed by the tools during the verification; *(iv)* the required number of observations – an indicator of the simulation effort.

For our experiments, we have chosen two case studies (CTMCs): Cyclic Server Polling System (CPS) and Tandem Queueing Network (TQN), also used in [14] for performance evaluation of probabilistic model checkers.

The rest of the paper is organised as follows: Section II contains a detailed description of the considered case studies and Section III provides further details about the employed model checkers and outlines their differences. Further, in Section IV we discuss and match the tool's parameters, justifying their values selected in Section V. The latter goes over the experimental setup. Section VI discusses the experimental results and Section VII concludes.

II. Case Studies

Cyclic Server Polling System (CPS). A cyclic polling system [13] consists of N stations and a server. Each station has a buffer with capacity 1 and the stations are attended by a single server in cyclic order. The server starts by polling the first station. If this station has a message in its buffer, the server serves it. Once the station has been served, or if its buffer was empty, the server moves to the next station cyclically. The polling and service times are exponentially distributed with rates $\gamma = 200$ and $\mu = 1$, respectively. The arrival rate of messages at each station is exponentially distributed with rate $\lambda = \frac{\mu}{N}$. Applications of this case study can be found in e. g. [27], [9], [23], [28].

Tandem Queueing Network (TQN). The Tandem Queueing Network [10] (see also [9], [28], [23]) consists of two queues of capacity N in sequence. Messages arrive at the first queue; when they get served, they are routed to the second queue, from where they leave the system. The message arrivals are exponentially distributed with rate $\lambda = 4 \cdot N$. The server handles messages from the first queue according to a two-phase Coxian [5] distribution. The time between departures from the second queue is exponentially distributed with rate $\kappa = 4$.

III. Tools

MRMC [15], [14] (version $1.4.1$, April 2009) is a command-line tool, written in C. The tool implements numerical model-checking techniques for DTMC and CTMC models, and reward extensions thereof. Since $v1.4.1$, it has a full support for the statistical model checking of CSL properties on CTMC models. For time-interval until formulae the tool employs simple terminating simulation. For unbounded-until formulae, the Markov chain model is divided into transient and absorbing states and then the long-run reachability probability is bounded by transient probabilities. For the steady-state formulae the probability is estimated based on steady-state simulation of bottom strongly connected components (BSCCS) and estimates for the probabilities to reach those BSCCs. The tool's distinguishing features are that: *(i)* to verify a formula it estimates its probability using a *c. i.* of desired width and then compares it against the formula's probability bound; *(ii)* MRMC does not employ standard sequential *c. i.* but rather emulates it by gradually increasing the sample size; *(iii)* the tool requires two independent samples when model-checking unbounded-until formulae.

Ymer [27] (version 3.0, February 2005) is a command-line tool, written in C and C++, for verifying transient properties of CTMCs and generalisations. Ymer implements statistical CSL model checking techniques based on discrete event simulation [26] and sequential acceptance sampling [29]. It also incorporates simple acceptance sampling and a numerical engine adopted from PRISM [16]. For time-interval formulae Ymer uses terminating simulations but instead of *c. i.* employs sequential acceptance sampling. The latter minimises the number of required observations by rejecting/accepting the verified property at early stages, when the simulations show that the formula is clearly satisfied/violated. The procedure has the advantage of requiring fewer observations, on average, than fixed sample size tests, e. g. *c. i.*, for similar levels of accuracy.

VESTA [23] (version 2.0, 2005) is a Java-based tool for statistical analysis of probabilistic systems. In particular, VESTA allows to verify CSL (PCTL) properties on CTMC (DTMC) models. The tool implements model-checking techniques, based on simple hypothesis testing [12], discussed in [29] and [24]. For until formulae the tool uses terminating simulations. For unbounded until, a *terminal* state $\perp$ is added to the model. Every state of the original model is then extended with a transition to this state (taken with some fixed probability $p_{\perp}$), and the existing transition probabilities are renormalized to form proper

probability distributions. Such modification allows to avoid infinite simulation paths, but at the same time requires an extra conditions for guarantying confidence levels, see Section IV. Note that, simple hypothesis testing – a simplified version of a sequential acceptance sampling that uses fixed sample sizes.

Tool differences: Before we proceed let us overview the differences between the considered tools, and the techniques they implement, and try to estimate their possible influence on the experimental results. *(i)* VESTA is implemented in Java and thus, can be slower than the other tools. Also, its VSZ values should mostly reflect the total memory allocated by JVM. *(ii)* VESTA uses simple hypothesis testing whereas Ymer uses sequential. Therefore, we expect VESTA to be slower than Ymer, since to achieve the same level of confidence, sequential hypothesis testing requires fewer observations than simple hypothesis testing [1]. *(iii)* Ymer and VESTA, unlike MRMC: *(iii.a) have on-the-fly model generation:* MRMC accepts pre-generated CTMCs, and thus the tool's VSZ values should depend on the model size; *(iii.b) can only verify properties in the initial state of the model:* Thus, our results correspond to model checking formulae in the initial state; *(iii.c) do not provide the probability estimates:* Ymer has a special option that allows to request such estimates (Ymer P). In this case, results are computed using sequential confidence-interval based approach [20].

IV. Tool Parameters

For a fair experimental comparison of model-checking algorithms it is vital to have their input parameters matching each other in the best possible way. Further, we consider the main simulation parameters of MRMC, Ymer, and VESTA. We will assume that $\widetilde{p} := \mathit{Prob}\,(s_0,\ \Phi\ \mathcal{U}\ \Psi)$, $\mathit{Prob}\left(s_0,\ \Phi\ \mathcal{U}^{[t_1,t_2]}\ \Psi\right)$, or $\mathit{Prob}^{\infty}\,(s,\ \Psi)$, and b is the probability bound of the formulae, e. g. when we want to verify $\mathcal{P}_{\geq b}\,(\Phi\ \mathcal{U}\ \Psi)$. Note that, since we consider formulae without nested probabilistic operators, the correctness conditions of the algorithms of Ymer and VESTA are the relaxed versions thereof given in [25]. An extended discussion about the tool's simulation parameters can be found in Section 7.1 of [31].

MRMC has two parameters: ξ – the desired confidence of the result; δ' – the upper bound on the width of the considered *c. i. Confidence-level guarantees*: if δ' is such that $\delta' \leq |b - \widetilde{p}|$, then the probability of getting the correct answer to the model-checking problem is guaranteed to be at least ξ.

Ymer has three parameters: α – the desired probability of the false-positive answer; β – the desired probability of the false-negative answer; δ – the half width of the indifference region. *Confidence-level guarantees*: if δ is such that $\widetilde{p} \notin (b-\delta,\ b+\delta)$, then the probability of getting the correct answer to the model-checking problem is guaranteed to be $1-\alpha$. Here we assume that $\alpha = \beta$ since we do not want do distinguish between false-positive and false-negative error probabilities.

VESTA inherits the parameters and the error-level guarantees of Ymer. In addition it has two parameters and one condition specific for unbounded-until formulae: $p_\perp > 0$ – the stopping probability; δ_1 – the width of the indifference region for the problem $\mathcal{P}_{=0}\,(\mathcal{A}\,\mathcal{U}\,\mathcal{G})$. *Confidence-level guarantees*: if $p_\perp$ and δ_1 are such that: $\widetilde{p} \notin \left(0,\ \frac{\delta_1}{p_m^{(|S|-1)}\cdot(1-p_\perp)^{(|S|-1)}}\right]$, where p_m is the smallest non-zero transition probability in the model, then the probability of getting the correct answer for the unbounded-until formulae is guaranteed to be $1-\alpha$ (here we take $\alpha = \beta$).

A. Relating parameters

To match parameters of Ymer, VESTA and MRMC, we take $1 - \xi = \alpha = \beta$, because we want to have the equal bounds on probabilities of having incorrect answers. In addition, we take $\delta' = \delta$ since then fulfilling $\delta' \leq |b - \widetilde{p}|$ is equivalent to choosing δ such that $\widetilde{p} \notin (b-\delta,\ b+\delta)$.

The extra condition of VESTA, required for the unbounded-until operator, does not have analogs in MRMC and Ymer. Therefore, in our experiments we use the default tool values for $p_\perp$ and δ_1. Note that, trying to satisfy this condition can cause serious problems when model checking large models due to the exponentials in the divider of the interval's right border. Moreover, according to [24], the decrease of $p_\perp$ dramatically increases the model-checking times. The same increase of verification time is likely to happen when δ_1 is decreased.

V. Experimental setup

Every experiment, unless stated otherwise, was repeated 100 times. Average verification times (milliseconds) and number of used observations, have logarithmic scale and are based on tool's statistics [1]. Peak memory usage of the tools was collected by sampling process-memory consumption (approximately) every 100 msec. The (actual) confidence levels are computed as the average number of successful model-checking runs on each experiment. The experiments were performed on a cluster-computer node with two 2.33 GHz Intel Dual-Core Xeon processors (64-bit) and 16 GB of RAM (time bounded- and unbounded-until formulae) and an Intel® Core™2 Quad 2.40 GHz processor (64-bit), 8 GB of RAM (steady-state formulae). The operating system was Linux, because it is supported

[1] A minor output modification was introduced into Ymer, see [32].

by all the tools. Considering the discussion in Section IV, the main tool parameters were set as follows: $1-\xi = \alpha = \beta = 0.05$, $\delta' = \delta = 0.01$, $p_\perp = 0.01$ and $\delta_1 = 0.1$.

These tool settings are expected to guarantee the 95% accuracy of the verification results. The accuracy can be lower if the conditions specified in Section IV are violated. Also, when verifying the unbounded-until formulae with VESTA, we use the default tool's settings. For the steady-sate formulae we have chosen the minimal sample size and the sample-size step to be $1,000$. The latter was done because for smaller model sizes we had premature *c. i.* convergence, that was resulting in low confidence.

VI. Experimental Results

Note that, Ymer does not support unbounded-until and VESTA cannot verify interval-until properties. Thus, our experimental results do not always include all the tools. MRMC is the only tool that supports verification of the steady-state operator, which can be verified in a pure simulation (*P*) or hybrid (*H*) mode. In the latter case the probabilities of reaching bottom strongly connected components are computed by means of numerical computations. Also, the regeneration method, used in steady-state simulations, can be run in the original setting (*O*), when the regeneration point is chosen arbitrarily, or using the heuristic (*H*), when it is chosen to be the most recurring state in a test run preceding the verification. Moreover, the sample-size step for sequential *c. i.* computations can be chosen to be fixed (*C*) or dynamic (*A*). In the latter case, the tool exponentially increases the sample-size step during the simulations. Therefore, for each steady-state formula, we have MRMC curves with names formed as $MRMC_{TMS}$ where $T \in \{P, H\}$, $M \in \{O, H\}$, and $S \in \{C, A\}$.

For both case studies, each tool and each model size (CPS: $N \geq 15$, TQN $N \geq 511$), the VSZ did not show any significant correlation (not more than a 2% difference) with the verified until formulae. This means, that in case of MRMC, the tool's memory consumption was mostly caused by storing large state spaces in RAM. Also, the memory consumption of Ymer and VESTA were practically constant. Due to these facts, we do not provide the VSZ plots of the until formulae, except for the first one verified on the CPS case study.

A. Cyclic Server Polling System (CPS)

For this case study we verified a bounded-until, an interval-until, two unbounded-until, and a steady-state formulae on the models with number of stations N ranging from 3 to 18 and the corresponding state-space sizes ranging from 36 to $7,077,888$. With the increase of N, the numerically-computed probabilities for the considered properties change as follows: for *Prob* $\left(true\ \mathcal{U}^{[0,80]}\ busy_1\right)$: from 1.0000 to 0.9882; for *Prob* $\left(true\ \mathcal{U}^{[40,80]}\ serve_1\right)$: from 0.9999 to 0.8944; for *Prob* $\left(poll_1\ \mathcal{U}\ serve_1\right)$: from 0.0016 to 0.0002; for *Prob* $\left(\neg serve_2\ \mathcal{U}\ serve_1\right)$: from 0.5213 to 0.5386; for $\mathcal{S}\left(busy_1\right)$: from 0.3481 to 0.1717.

$\mathcal{P}_{\geq 0.95}\left(true\ \mathcal{U}^{[0,80]}\ busy_1\right)$ – the probability that station 1 becomes busy within 80 time units is at least 0.95. With increase of N, all the tools show increase of the model-checking times (cf. Fig. 1) and the number of observations (cf. Fig. 2). This is because: *(a)* *Prob* $\left(true\ \mathcal{U}^{[0,80]}\ busy_1\right)$ decreases and approaches the probability constraint (0.95); *(b)* the model state space grows, requiring for more and longer simulation paths. For the largest model size ($N = 18$), MRMC uses (respectively) 1.2, 3.6 and 10.2 times more observations than VESTA, Ymer P and Ymer. Yet, MRMC is 1.5, 3.2, and 4.4 times faster than (respectively) Ymer, VESTA, and Ymer P. For Ymer it means that either the tool does not have a sufficiently efficient implementation or that sampling does not have a significant impact on verification times, when compared to the effort needed for, e. g., performing hypothesis testing. Still, the verification times of MRMC are growing faster than that of the other tools.

According to Fig. 3, Ymer and VESTA use constant

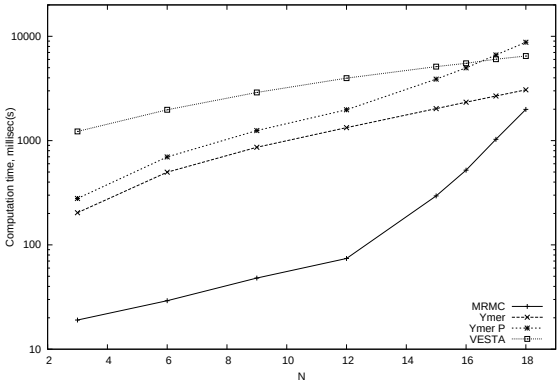

Fig. 1: CPS : $\mathcal{P}_{\geq 0.95}\left(true\ \mathcal{U}^{[0,80]}\ busy_1\right)$ (time)

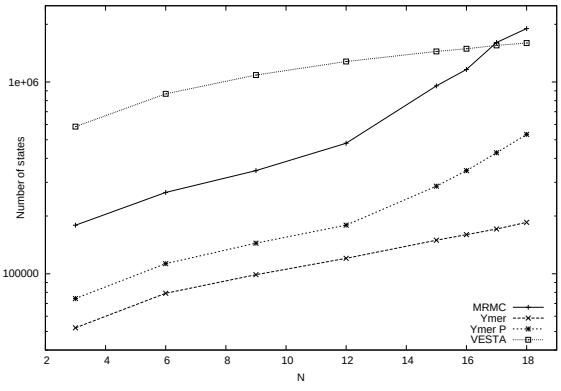

Fig. 2: CPS : $\mathcal{P}_{\geq 0.95}\left(true\ \mathcal{U}^{[0,80]}\ busy_1\right)$ (# observations)

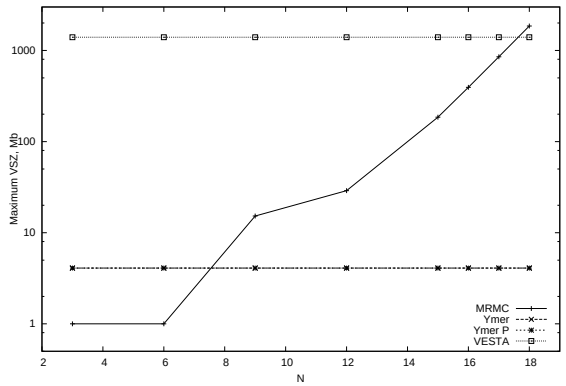

Fig. 3: CPS: $\mathcal{P}_{\geq 0.95}\left(true\ \mathcal{U}^{[0,80]}\ busy_1\right)$ (VSZ)

memory and the VSZ of MRMC, as predicted, grows with the model size. This implies that, since both Ymer and VESTA do not generate the model's state space, the memory consumption for sampling is insignificant. The large memory usage of VESTA is dominated by the amount of memory acquired by the JVM.

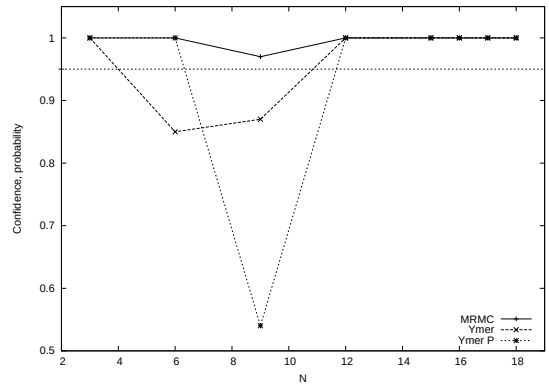

Fig. 4: CPS: $\mathcal{P}_{\geq 0.99}\left(true\,\mathcal{U}^{[40,80]}\ serve_1\right)$ (confidence)

$\mathcal{P}_{\geq 0.99}\left(true\,\mathcal{U}^{[40,80]}\ serve_1\right)$ – the probability that station 1 is served within the time interval $[40,\ 80]$ is at least 0.99. The confidence levels for $N \in \{6,\ 9\}$ (cf.

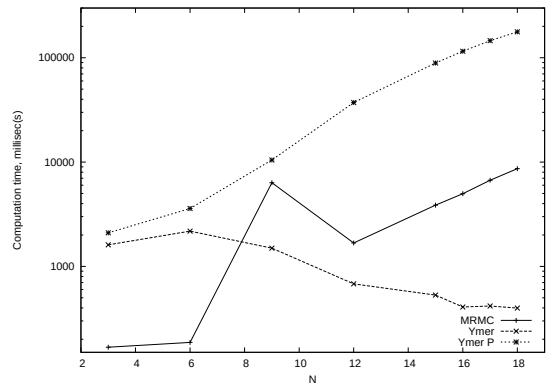

Fig. 5: CPS : $\mathcal{P}_{\geq 0.99}\left(true\,\mathcal{U}^{[40,80]}\ serve_1\right)$ (time)

Fig. 4) are compromised, especially in case of Ymer and Ymer P. This happens because the corresponding probabilities *Prob* $\left(true\,\mathcal{U}^{[40,80]}\ serve_1\right)$ are 0.9988 and

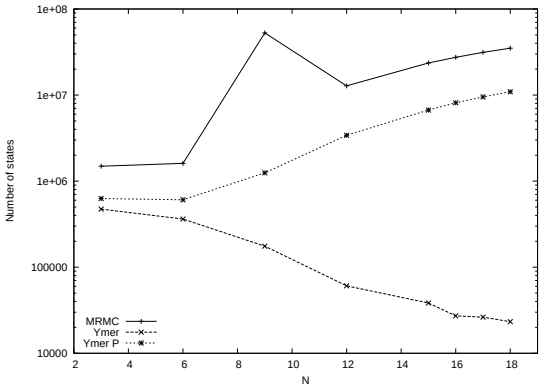

Fig. 6: CPS : $\mathcal{P}_{\geq 0.99}\left(true\,\mathcal{U}^{[40,80]}\ serve_1\right)$ (# observations)

0.9888, i. e., they fall in the indifference region. Moreover, the condition $\delta' \leq |b - \widetilde{p}|$, required by MRMC for ensuring the confidence $\xi = 0.95$, is also violated. MRMC provides more accurate answers as its algorithm first simulates until the *c. i.* is tighter than δ' and then continues simulation until it reaches the definite answer to the model-checking problem. This increases the accuracy because the width of the resulting *c. i.* can be much smaller than δ'. Also, MRMC uses the Agresti-Coull *c. i.* that is known to have a coverage probability that exceeds the specified confidence.

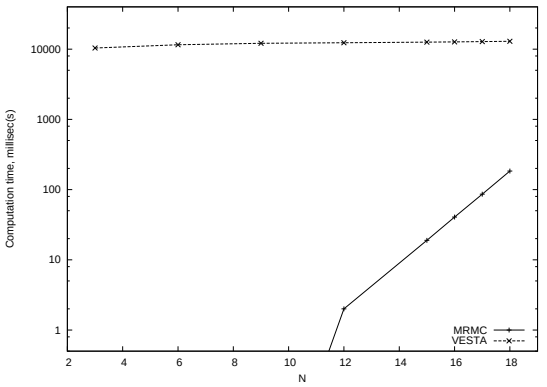

Fig. 7: CPS: $\mathcal{P}_{\geq 0.2}\left(poll_1\ \mathcal{U}\ serve_1\right)$ (time)

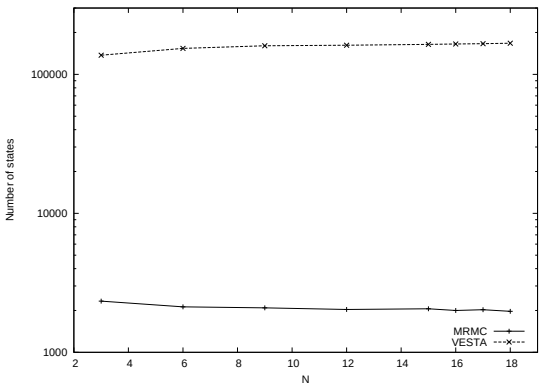

Fig. 8: CPS: $\mathcal{P}_{\geq 0.2}\left(poll_1\ \mathcal{U}\ serve_1\right)$ (# observations)

The model-checking times (cf. Fig. 5) and the number of observations (cf. Fig. 6) indicate that the accuracy of

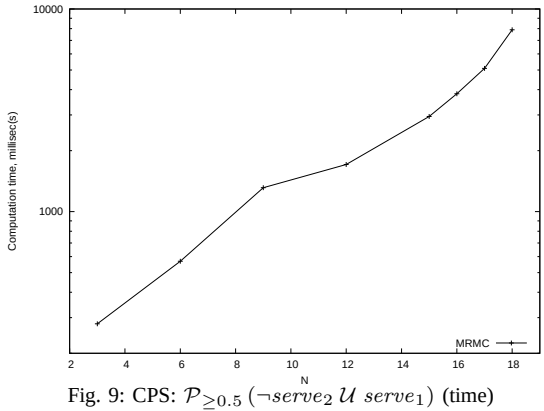

Fig. 9: CPS: $\mathcal{P}_{\geq 0.5}\left(\neg serve_2\ \mathcal{U}\ serve_1\right)$ (time)

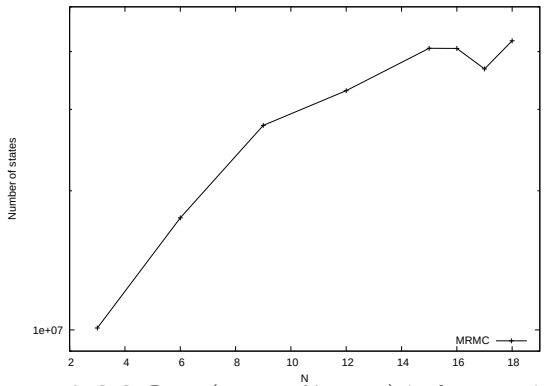

Fig. 10: CPS: $\mathcal{P}_{\geq 0.5}\left(\neg serve_2\ \mathcal{U}\ serve_1\right)$ (# observations)

MRMC comes at a price, as witnessed by the peaks for $N = 9$. In general ($N = 18$), MRMC is up to 8 times faster than Ymer P, but is up to 21.7 times slower than Ymer. The performance of the latter one is improving with the growth of N. The reason for that is likely to be the rapid increase of distance between the values of *Prob* $\left(true\ \mathcal{U}^{[40,80]}\ serve_1\right)$ and the probability bound of the formula. In this case, Ymer P and MRMC continue simulations until they reach the *c. i.* of the desired width but Ymer, that uses sequential hypothesis testing, does not need that, so it stops much earlier.

$\mathcal{P}_{\geq 0.2}\left(poll_1\ \mathcal{U}\ serve_1\right)$ – the probability that station 1 is served after being polled is at least 0.2. Both MRMC and VESTA showed 100% accuracy when model checking this property. The performance results given in Fig. 7 indicate that the time required by VESTA is almost constant for all model sizes and for MRMC it is insignificantly small in the beginning (up to $N = 12$) and then starts growing. This contradictions to Fig. 8. One can notice that the number of observations required by VESTA is growing whereas for MRMC it is decreasing. Putting these facts together, we conclude that the increase of verification time for MRMC might be caused by: *(i)* the effort required for traversing the large (up to about $7 \cdot 10^6$ states) Markov chain stored in RAM; *(ii)* the need to search for BSCCs. Still, MRMC is at least 10 times faster than VESTA.

$\mathcal{P}_{\geq 0.5}\left(\neg serve_2\ \mathcal{U}\ serve_1\right)$ – the probability that station 1 is served before station 2 is at least 0.5. Fig. 9 provides the model-checking times for MRMC which again showed $> 95\%$ accuracy. The plots for VESTA are not present because it did not terminate within the 15 minutes time-out (compared to seconds required by MRMC). Fig. 10 shows the number of required observations. Notice that, there is a significant drop for $N = 17$ and also the values for $N = 15$ and 16 are almost equal. At the same time, the model-checking times for these values of N show a stable and continuous increase. This strengthen our belief in that supplementary computations, such as traversal through a large pre-generated Markov chain, stored in RAM, give a much stronger influence on the model-checking time than the increase in the number of required observations.

$\mathcal{S}_{>0.19}\left(busy_1\right)$ – the steady-state probability of station 1 being busy is greater than 0.19. Fig. 11 provides the

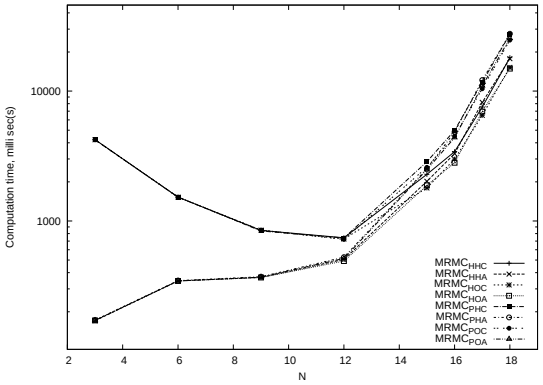

Fig. 11: CPS : $\mathcal{S}_{>0.19}\left(busy_1\right)$ (time)

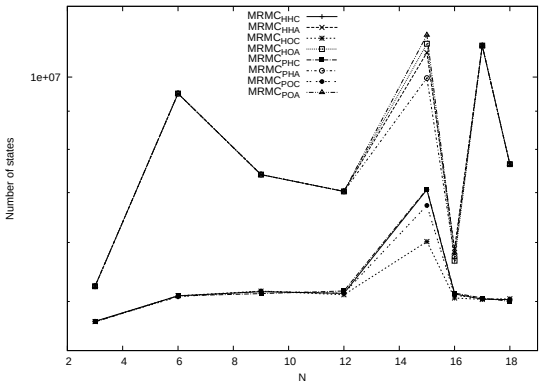

Fig. 12: CPS : $\mathcal{S}_{>0.19}\left(busy_1\right)$ (# observations)

model-checking times for MRMC which showed 100% accuracy. Notice that up to $N = 12$ there is a significant difference between MRMC runs with the constant sample-size step, the upper bunch of curves, and the dynamic step, the lower bunch. Also, the latter ones are using significantly more observations (cf. Fig. 12), from which we conclude that re-computation of *c. i.*, on small models, can significantly slow down model checking. Besides, for $N = 15, 16$ the estimated probabilities fall into the indifference interval. The corresponding picks are especially

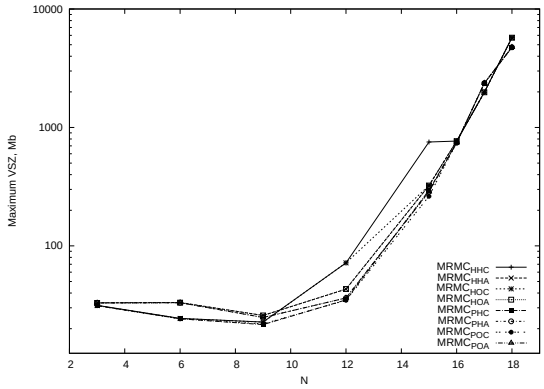

Fig. 13: CPS: $\mathcal{S}_{>0.19}\left(busy_1\right)$ (VSZ)

distinctive for the $MRMC_{**C}$ curves of Fig. 12. Yet, this does not have any drastic affect on the corresponding model-checking times. This is because for $N >= 12$ the effort needed for additional computations still exceeds the effort required for simulating a large model. The memory consumption curves, cf. Fig. 13, all showing similar behaviour. Yet, memory required to store samples, starts playing an important role. Notice that, the VSZ values of MRMC are significantly higher (up to 3.1 times for $N = 18$) than in Fig. 3.

B. Tandem Queueing Network (TQN)

Here we verified two bounded-until, one interval-until, one unbounded-until, and one steady-state formulae on the models with the queue capacities N ranging from 2 to 1023 and the corresponding state-space sizes are ranging from 15 to $2,096,128$. With the increase of N, the numerically-computed probabilities for the considered properties change as follows: for *Prob* $\left(true\ \mathcal{U}^{[0,2]}\ full\right)$: from 0.0262 to 0.0000; for *Prob* $\left(true\ \mathcal{U}^{[0.5,2]}\ full\right)$: from 0.0225 to 0.0000; for *Prob* $\left(true\ \mathcal{U}^{[0,10]}\ full_1\right)$: it is constantly 1.0000; for *Prob* $\left(\neg full_1\ \mathcal{U}\ full_2\right)$: from 0.0177 to 0.0000; for $\mathcal{S}\left(full_1\right)$: from 0.8032 to 0.9995. Since the value of N is changed in a non-linear manner, the horizontal axis of the plots given in this section is *logarithmic*.

$\mathcal{P}_{\leq 0.01}\left(true\ \mathcal{U}^{[0,2]}\ full\right)$ – the probability that both queues become full within 2 time units is at most 0.01. There are no results for Ymer P because it was not terminating within the 15 minutes time-out. The confidence estimates in Fig. 14 exhibit a slight decrease of confidence for Ymer and VESTA at $N = 2$. This is due to the fact that in this case *Prob* $\left(true\ \mathcal{U}^{[0,2]}\ full\right) = 0.0262$ is relatively close to the probability bound. Still, the confidence levels stay above the theoretically predicted one. As before, MRMC is generally faster than the other tools (cf. Fig. 15) but levels out with Ymer at $N = 1023$. Also, its model checking times grow faster that that of the other tools. The peaks in MRMC plots for $N = 2$ (see also Fig. 16) is the price it pays for being 100% accurate. Notice that, for

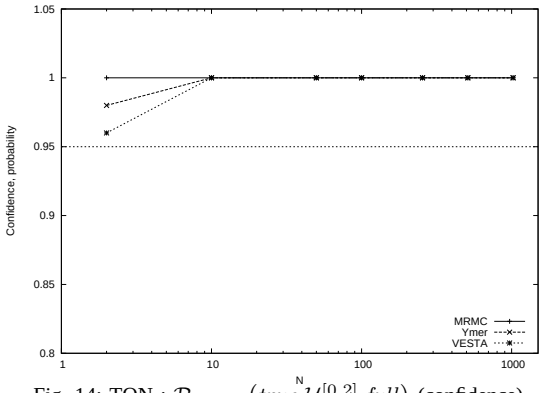

Fig. 14: TQN : $\mathcal{P}_{\leq 0.01}\left(true\ \mathcal{U}^{[0,2]}\ full\right)$ (confidence)

$N \geq 10$ the number of observations grows uniformly for all tools. E. g., MRMC requires from 3 to 3.2 times more observations than Ymer, and VESTA needs about 12% more samples than MRMC. The verification times show a different behaviour. VESTA has the slowest increase of time, Ymer's times grow a bit faster, and MRMC has the fastest time increase. In the worst case ($N = 1023$), Ymer is only 2.3 times faster than VESTA, and about 7% faster than MRMC. Considering the corresponding increase in the number of observations, this might mean that the Ymer's implementation is either not very efficient or that sampling does not have a sufficient effect on model checking times, compared to supplementary computations. MRMC most likely suffers from the need to store and traverse the complete CTMC.

$\mathcal{P}_{\leq 0.1}\left(true\ \mathcal{U}^{[0.5,2]}\ full\right)$ – the probability that both queues become full within time interval $[0.5,\ 2]$ is at most 0.1. For this property all the tools showed 100% accuracy. Once again, Ymer P was not able to finish verification within 15 minutes. The performance results given in Fig. 17 and 16 show the behaviour similar to the one for $\mathcal{P}_{\leq 0.01}\left(true\ \mathcal{U}^{[0,2]}\ full\right)$.

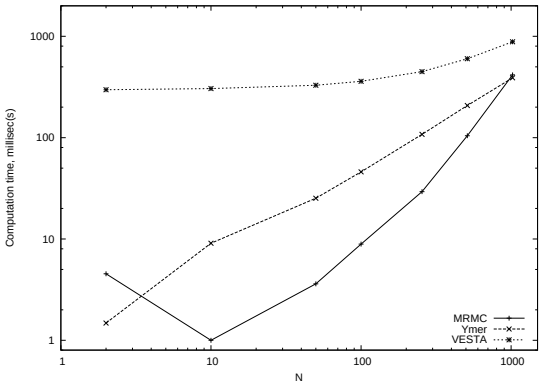

Fig. 15: TQN: $\mathcal{P}_{\leq 0.01}\left(true\ \mathcal{U}^{[0,2]}\ full\right)$ (time)

$\mathcal{P}_{\leq 0.98}\left(true\ \mathcal{U}^{[0,10]}\ full_1\right)$ – the probability that the first queue becomes full within 10 time units is at most 0.98. In this case Ymer P successfully verified the formula, and all the tools were 100% accurate. The performance

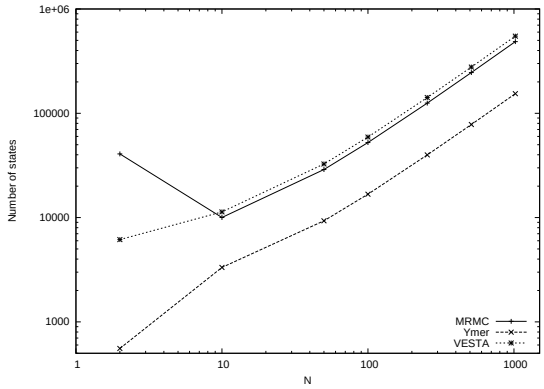

Fig. 16: TQN: $\mathcal{P}_{\leq 0.01}\left(true\ \mathcal{U}^{[0,2]}\ full\right)$ (# observations)

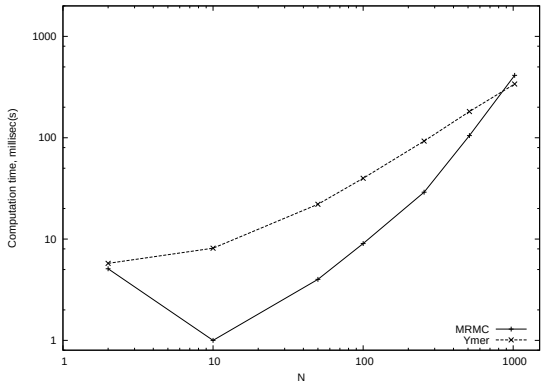

Fig. 17: TQN: $\mathcal{P}_{\leq 0.1}\left(true\ \mathcal{U}^{[0.5,2]}\ full\right)$ (time)

displayed in Fig. 19 and Fig. 20, are similar to the ones for the previous two properties.

$\mathcal{P}_{\leq 0.03}\left(\neg full_1\ \mathcal{U}\ full_2\right)$ – the probability that the second queue becomes full before the first queue is at most 0.03. Both, VESTA and MRMC were completely accurate in their model-checking results. The verification times and the number of observations in Fig. 21 and 22 reflect that, since $Prob\left(\neg full_1\ \mathcal{U}\ full_2\right) = 0.000$ for all $N \geq 10$, VESTA needs an almost constant amount of observations to decide on the property. This can be because it uses hypothesis testing and that the distance

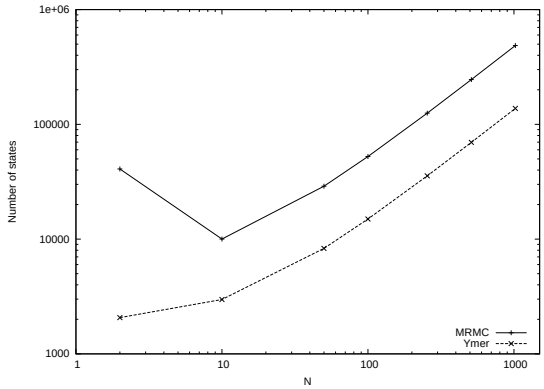

Fig. 18: TQN: $\mathcal{P}_{\leq 0.1}\left(true\ \mathcal{U}^{[0.5,2]}\ full\right)$ (# observations)

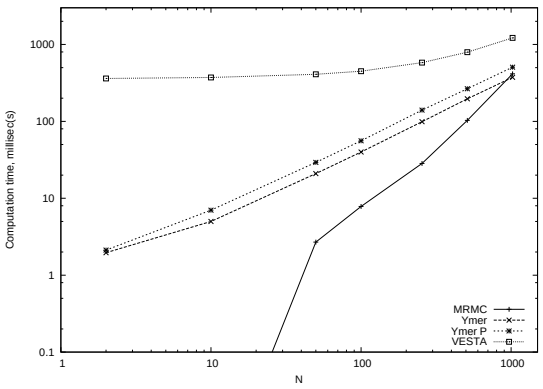

Fig. 19: TQN: $\mathcal{P}_{\leq 0.98}\left(true\ \mathcal{U}^{[0,10]}\ full_1\right)$ (time)

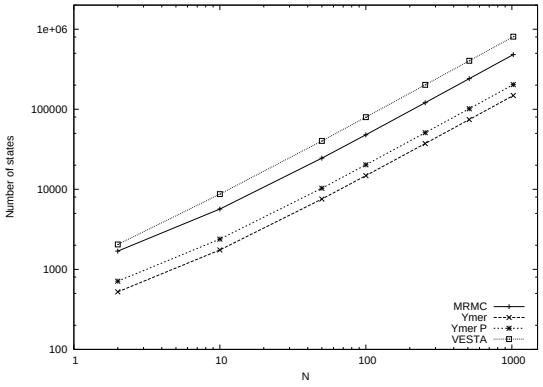

Fig. 20: TQN: $\mathcal{P}_{\leq 0.98}\left(true\ \mathcal{U}^{[0,10]}\ full_1\right)$ (# observations)

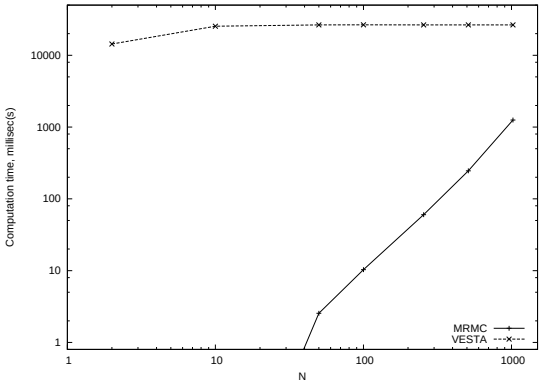

Fig. 21: TQN: $\mathcal{P}_{\leq 0.03}\left(\neg full_1\ \mathcal{U}\ full_2\right)$ (time)

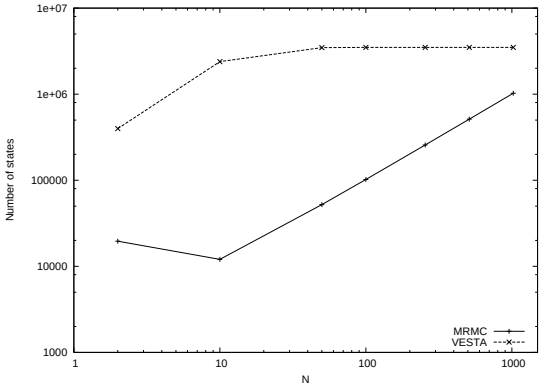

Fig. 22: TQN: $\mathcal{P}_{\leq 0.03}\left(\neg full_1\ \mathcal{U}\ full_2\right)$ (# observations)

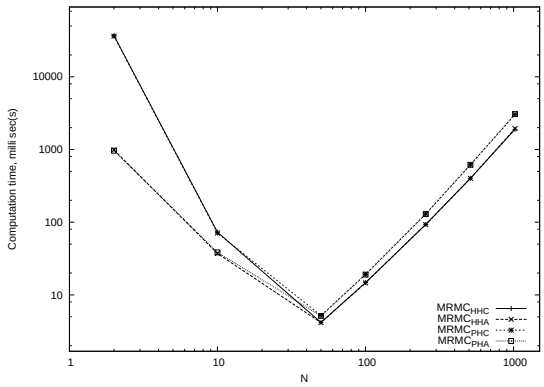

Fig. 23: TQN: $\mathcal{S}_{>0.999}\,(full_1)$ (time)

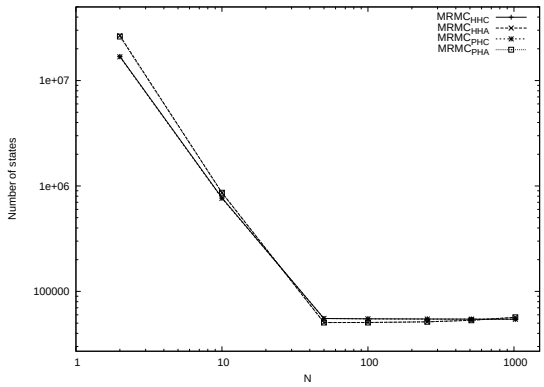

Fig. 24: TQN: $\mathcal{S}_{>0.999}\,(full_1)$ (# observations)

between the probability bound 0.03 and the true value of $\mathit{Prob}\,(\neg full_1\ \mathcal{U}\ full_2)$ stays constant. Still, MRMC is at least 6 times faster than VESTA.

$\mathcal{S}_{>0.999}\,(full_1)$ – the steady-state probability of the first queue being full is greater than 0.999 For this property we set the width of the indifference region to be 0.0003. Although MRMC was 100% accurate, we should note that in case of $N = 511$ the estimated probability falls in the indifference region. Also, using pure regeneration method, without the heuristic for choosing the regeneration point, failed. MRMC was unable to finish simulations withing 15 minutes timeout. The reason for that is that the TQN's model is an ergodic Markov chain. The latter, especially for larger models, causes most of the regeneration cycles to be enormously long. Once again, cf. Fig. 23, we see that for smaller model sizes ($N < 50$) having a dynamic sample-size increase saves a lot of effort needed for re-computation of the *c. i.* For $N \geq 50$ the pure simulation method requires more time. This is because, although for an ergodic CTMC there is no need to compute reachability probabilities, the pure and hybrid simulation methods have two different implementations and the former has a higher complexity. The required observation in Fig. 24 show that MRMC with the dynamic sample-size increase needs more observations for $N < 50$, and for $N \geq 50$ all curves exhibit comparable behaviour. At the moment, we do not

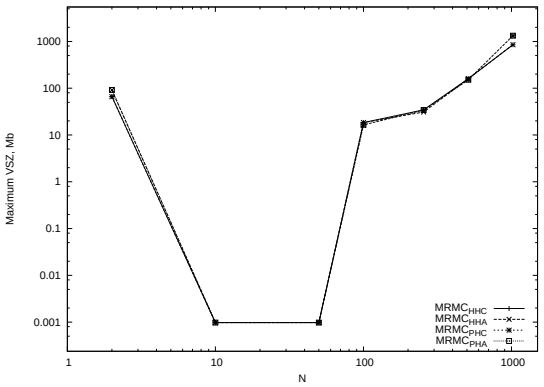

Fig. 25: TQN: $\mathcal{S}_{>0.999}\,(full_1)$ (VSZ)

have any good explanation for the decrease in the number of needed observations and the almost constant values for $N \geq 50$. Note that, verifying the given steady-state property in the worst case ($N = 1023$, $MRMC_{PHA}$) requires about 4.2 times more memory (cf. Fig. 25[2]) than verifying, e. g., $\mathcal{P}_{\leq 0.01}\left(true\ \mathcal{U}^{[0,2]}\ full\right)$ for $N = 1023$.

VII. Conclusions

Our analysis showed that for until formulae the peak-memory consumption (VSZ) of MRMC grows in accordance with the growth of the model sizes. This is due to using the pre-generated Markov chain, as opposed to the on-demand state-space generation implemented in Ymer and VESTA. The later tools show (almost) constant memory consumption. For the steady-state operator the situation is different. When model checked with MRMC, for the same model size, VSZ values can be up to 4.5 (TQN, $N = 1023$) times larger than the ones for the until formula. This means that memory needed for storing sampled data is almost negligible when verifying until, and is significant when verifying steady-state formulae.

The actual confidence levels of all tools were within theoretically predicted bounds. At the same time MRMC showed high accuracy even in cases when the sufficient conditions for providing these bounds were violated.

Ymer P and VESTA were not always able to provide model checking times within the 15 minutes time out. In all other cases, the model-checking times for all the tools were within seconds. The exception is Ymer P, cf. Fig. 5. On the considered models, verification times of MRMC were mostly several times (up to 10) smaller than that of Ymer and VESTA, but the performance of MRMC was rapidly decreasing with growth of the model sizes. This might be because, e. g., generating random paths through a large Markov chain requires addressing far distant blocks of RAM. Another observation is that, for steady-state simulations on smaller models ($N \leq 12$ for CPS, and $N \leq 511$

[2] $N = 10, 50$: An inadequate statistics due to small verification times.

for TQN) computation of confidence intervals requires much more effort than doing sampling. To conclude, we must admit that Ymer showed an excellent performance on larger models, where in one case it was 21.7 times faster than MRMC (cf. Fig. 5). Also, Ymer always needed fewer observations to provide correct model-checking results than other tools. This means that its algorithms are more efficient from the simulation point of view. Considering its performance on smaller models, we must conclude that either its implementation is not very efficient or that the sampling effort does not play a significant role when compared to supplementary computations. Last, but not least is VESTA which, considering that it is implemented in Java, showed a reasonably good performance. The tool typically required more observations, but, with the growth of the model sizes, the increase in their numbers was not as significant as in case of MRMC.

References

[1] A.Wald and J. Wolfowitz. Optimum character of the sequential probability ratio test. *Annals of Mathematical Statistics*, 19:326–339, 1948.

[2] C. Baier, F. Ciesinski, and M. Größer. ProbMela and verification of Markov decision processes. *ACM SIGMETRICS Performance Evaluation Review*, 32(4):22–27, 2005.

[3] C. Baier, B. Haverkort, H. Hermanns, and J.-P. Katoen. Model-Checking Algorithms for Continuous-Time Markov Chains. *IEEE Transactions on Software Engineering*, 29(6):524–541, 2003.

[4] E. Bode, M. Herbstritt, H. Hermanns, S. Johr, T. Peikenkamp, R. Pulungan, R. Wimmer, and B. Becker. Compositional Performability Evaluation for STATEMATE. In *Quantitative Evaluation of Systems (QEST)*, pages 167–178. IEEE Computer Society, 2006.

[5] D. R. Cox. A use of complex probabilities in the theory of stochastic processes. In *Cambridge Philosophical Society*, volume 51, pages 313–319, 1955.

[6] D. D'Aprile, S. Donatelli, and J. Sproston. CSL Model Checking for the GreatSPN Tool. In C. Aykanat, T. Dayar, and I. Korpeoglu, editors, *Computer and Information Sciences*, volume 3280 of *LNCS*, pages 543–553. Springer, 2004.

[7] H. Hannsson and B. Jonsson. A logic for reasoning about time and reliability. *Formal Aspects of Computing*, 6(5):512–535, 1994.

[8] T. Hérault, R. Lassaigne, F. Magniette, and S. Peyronnet. Approximate Probabilistic Model Checking. In B. Steffen and G. Levi, editors, *Verification, Model Checking, and Abstract Interpretation (VMCAI'04)*, volume 2937 of *Lecture Notes in Computer Science*, pages 73–84. Springer-Verlag, 2004.

[9] H. Hermanns, J.-P. Katoen, J. Meyer-Kayser, and M. Siegle. A Markov Chain Model Checker. In S. Graf and M. Schwartzbach, editors, *Tools and Algorithms for the Construction and Analysis of Systems (TACAS)*, volume 1785 of *LNCS*, pages 347–362. Springer, 2000.

[10] H. Hermanns, J. Meyer-Kayser, and M. Siegle. Multi Terminal Binary Decision Diagrams to Represent and Analyse Continuous Time Markov Chains. In B. Plateau, W. J. Stewart, and M. Silva, editors, *Numerical Solutions of Markov Chains*, pages 188–207. Prensas Universitarias, 1999.

[11] J. Hillston. *A Compositional Approach to Performance Modelling*. Distinguished Dissertations Series. Cambridge University Press, New York, NY, USA, 1996.

[12] R. V. Hogg and A. T. Craig. *Introduction to Mathematical Statistics*. MacMillan, New York, NY, USA, 4th edition, 1978.

[13] O. C. Ibe and K. S. Trivedi. Stochastic Petri Net Models of Polling Systems. *Selected Areas in Communications*, 8(9):1649–1657, 1990.

[14] D. N. Jansen, J.-P. Katoen, M. Oldenkamp, M. Stoelinga, and I. S. Zapreev. How Fast and Fat Is Your Probabilistic Model Checker? In *Haifa Verification Conference (HVC)*, volume 4899 of *LNCS*, pages 65 – 79. Springer, 2008.

[15] J.-P. Katoen, M. Khattri, and I. S. Zapreev. A Markov Reward Model Checker. In *Quantitative Evaluation of Systems (QEST)*, pages 243–244. IEEE Computer Society, 2005. www.mrmc-tool.org.

[16] M. Kwiatkowska, G. Norman, and D. Parker. PRISM: Probabilistic Symbolic Model Checker. In T. Field, P. Harrison, J. Bradley, and U. Harder, editors, *Modelling Techniques and Tools for Computer Performance Evaluation (TOOLS)*, volume 2324 of *LNCS*, pages 200–204. Springer, 2002.

[17] R. Lassaigne and S. Peyronnet. Approximate Verification of Probabilistic Systems. In H. Hermanns and R. Segala, editors, *Process Algebra and Probabilistic Methods, Performance Modeling and Verification (PAPM/PROBMIV)*, pages 213–214. Springer, 2002.

[18] P. Lecca and C. Priami. Cell cycle control in eukaryotes: A BioSpi model. Technical Report DIT-03-045, Informatica e Telecomunicazioni: University of Trento, 2003.

[19] MRMC: Markov Reward Model Checker. http://www.mrmc-tool.org/.

[20] A. Nadas. An extension of a theorem of Chow and Robbins on sequential confidence intervals for the mean. *Annals of Mathematical Statistics*, 40(2):667–671, 1969.

[21] G. Norman and V. Shmatikov. Analysis of probabilistic contract signing. *Journal of Computer Security*, 14(6):561–589, 2006.

[22] PRISM: Probabilistic Symbolic Model Cchecker. http://www.prismmodelchecker.org.

[23] K. Sen, M. Viswanathan, and G. Agha. Statistical Model Checking of Black-Box Probabilistic Systems. In R. Alur and D. A. Peled, editors, *Computer Aided Verification (CAV)*, volume 3114 of *LNCS*, pages 202–215. Springer, 2004.

[24] K. Sen, M. Viswanathan, and G. Agha. On Statistical Model Checking of Stochastic Systems. In K. Etessami and S. K. Rajamani, editors, *Computer Aided Verification (CAV)*, volume 3576 of *LNCS*, pages 266–280. Springer, 2005.

[25] K. Sen, M. Viswanathan, and G. Agha. VESTA: A Statistical Model-checker and Analyzer for Probabilistic Systems. In *Quantitative Evaluation of Systems (QEST)*, pages 251–252. IEEE Computer Society, 2005.

[26] G. S. Shedler. *Regenerative Stochastic Simulation*. Academic Press, London, UK, 1993.

[27] H. Younes. Ymer: A Statistical Model Checker. In K. Etessami and S. K. Rajamani, editors, *Computer Aided Verification (CAV)*, volume 3576 of *LNCS*, pages 429–433. Springer, 2005.

[28] H. Younes, M. Kwiatkowska, G. Norman, and D. Parker. Numerical vs. Statistical Probabilistic Model Checking. *Software Tools for Technology Transfer (STTT)*, 8(3):216–228, 2006.

[29] H. Younes and R. Simmons. Probabilistic Verification of Discrete Event Systems using Acceptance Sampling. In E. Brinksma and K. G. Larsen, editors, *Computer Aided Verification (CAV)*, volume 2404 of *LNCS*, pages 223–235. Springer, 2002.

[30] H. Younes and R. Simmons. Statistical Probabilistic Model Checking with a Focus on Time-Bounded Properties. *Information and Computation*, 204(9):1368–1409, 2006.

[31] I. S. Zapreev. *Model Checking Markov Chains: Techniques and Tools*. PhD thesis, University of Twente, Enschede, The Netherlands, 2008. http://doc.utwente.nl/58974/1/thesis_Zapreev.pdf.

[32] I. S. Zapreev and C. Jansen. MRMC: Test-suite manual. http://www.mrmc-tool.org/trac/wiki/Specifications, 2008.