Importance Sampling for Model Checking of Continuous Time Markov Chains

Benoît Barbot, Serge Haddad, Claudine Picaronny
LSV, ENS Cachan & CNRS & INRIA Cachan, France
{*barbot,haddad,picaronny*}*@lsv.ens-cachan.fr*

***Abstract*—Model checking real time properties on probabilistic systems requires computing transient probabilities on continuous time Markov chains. Beyond numerical analysis ability, a probabilistic framing can only be obtained using simulation. This statistical approach fails when directly applied to the estimation of very small probabilities. Here combining the uniformization technique and extending our previous results, we design a method which applies to continuous time Markov chains and formulas of a timed temporal logic. The corresponding algorithm has been implemented in our tool COSMOS. We present experimentations on a relevant system, with drastic time reductions w.r.t. standard statistical model checking.**

***Keywords*-statistical model checking, rare events, importance sampling, coupling, uniformization**

I. INTRODUCTION

Many complex systems exhibit probabilistic behaviour either in an inherent way or through interaction with unreliable environment (communication protocols, biological systems...). Quantitative model checking is an efficient technique to verify properties of these systems. It consists in estimating the probability of a real time property, expressed by some temporal logic formula like CSL [1], as "the probability that the airbag fails to deploy within 10ms is less than 10^{-3}" [2]. This requires to compute transient probabilities on a probabilistic model of the system. Whenever numerical methods cannot be used because of the inherent state explosion, statistical sampling techniques prove to be efficient as soon as it is possible to perform a Monte-Carlo simulation of the model. Simulation usually requires a very small amount of space comparatively, thus allows to deal with huge models [3]. In principle, it only requires to maintain a current state (and some numerical values in case of a non Markovian process). Furthermore no regenerative assumption is required and it is easier to parallelise the methods. Several tools include statistical model checking: COSMOS [4], GREATSPN [5], PRISM [6], UPPAAL [7], YMER [8].

The main drawback of statistical model checking is its inefficiency in dealing with very small probabilities. The size of the sample of simulations required to estimate these small probabilities exceeds achievable capacities. This difficulty is known as the *rare event* problem.

Several methods have been developed to cope with this problem whose main one is *importance sampling* [9]. Importance sampling method is based on a modification of the underlying probability distribution in such a way that a specific rare event occurs much more frequently. Theoretical results have been obtained for importance sampling but none of them includes any *true confidence interval*. Indeed all previous works propose *asympotic confidence intervals* based on the central limit theorem. For rare event simulation, such an interval is inappropriate since to be close to a true confidence interval, it requires to generate a number of trajectories far beyond the current computational capabilities.

In [10], we proposed an efficient method based on importance sampling to estimate in a reliable way (the first one with a true confidence interval) tiny steady-state probabilities, required for logical formula using a standard "Until" property (aUb) when the model operational semantic is a Discrete Time Markov Chain (DTMC).

Our contribution. We extend here our previous results in order to deal with simultaneous timed and probabilistic assessments: We improve our method to estimate transient probabilities of rare events on Continuous Time Markov Chains (CTMC). More precisely, given a bounded delay τ, we statistically estimate the (tiny) probability that a random path generated by the CTMC reaches a certain state before instant τ. In order to design and prove the correctness of the method we proceed in three stages:

- We show using uniformisation [11] that a confidence interval for the estimation can be computed from confidence intervals of several estimations in the embedded DTMC of the CTMC.
- Our importance sampling approach for time bounded reachability in DTMC is then developped by generalizing the one we have proposed in [10], based on the mapping of the original model to a reduced one using coupling [12].
- However contrary to the original approach, the memory requirements are no more negligible and depend on the considered (discrete) time interval. Thus we propose three algorithms with a different trade-off between time and space so that we can handle very large time intervals.

To the best of our knowledge, our method is the first one among importance sampling methods for CTMC that provides a true confidence interval. Furthermore we have implemented it in the statistical model checker COSMOS [4]. We tested our tool on a classical relevant model getting impressive time and/or memory reductions.

Organisation. In section II, we recall our previous results [10]. In section III, we extend this method to the

estimation of transient probabilities on continuous time Markov chains. In section IV, we develop algorithmic issues in order to overcome excessive memory consumption. Afterwards we present our implementation in the tool COSMOS and an experimentation on a classical example. Finally in section VI, we conclude and give some perspectives to this work.

II. IMPORTANCE SAMPLING METHOD WITH GUARANTEED VARIANCE FOR UNBOUNDED REACHABILITY

Let us summarize the method developed in [10]: first remark that the modeller does not usually specify its system with a Markov chain. He rather defines a higher level model $\mathcal{M}$ (a queueing network, a stochastic Petri net, etc.), whose operational semantic is a Markov chain $\mathcal{C}$. If $\mathcal{C}$ is a Discrete Time Markov Chain (DTMC) with state space S, transition probability matrix $\mathbf{P}$ and two absorbing states s_+ and s_- which are reached with probability 1, we define $\mu(s)$ $s \in S$ the probability to reach s_+ starting from s. We want to estimate the probability $\mu(s_0)$ with s_0 being the initial state of $\mathcal{C}$.

By generating a large sample of trajectories the Monte Carlo algorithm provides an estimation of $\mu(s_0)$ as the ratio of the trajectories reaching s_+ by the total number of generated trajectories. For a rare event this approach is not suitable as it is due to the size of the sample far too big when one wants a precise result. The variance of the underlying random variable is in fact too big [9].

The importance sampling method uses a modified transition matrix $\mathbf{P}'$ during the generation of paths. $\mathbf{P}'$ must satisfy:

$$\mathbf{P}(s,s') > 0 \Rightarrow \mathbf{P}'(s,s') > 0 \vee s' = s_- \tag{1}$$

which means that this modification cannot remove transitions that have not s_- as target, but can add new transitions. The method maintains a correction factor called L initialized to 1; this factor represents the *likelihood* of the path. When a path crosses a transition $s \to s'$ with $s' \neq s_-$, L is updated by $L \leftarrow L\frac{\mathbf{P}(s,s')}{\mathbf{P}'(s,s')}$. When a path reaches s_-, L is set to zero. If $\mathbf{P}' = \mathbf{P}$ (i.e. no modification of the chain), the value of L when the path reaches s^+ (resp. s^-) is 1 (resp. 0).

Let V_s (resp. W_s) be the random variable associated with the final value of L for a path starting in s in the original model (resp. in the modified one). By definition, $\mathbf{E}(V_{s_0}) = \mu(s_0)$. A classical result [9] p. 25, states that $\mathbf{E}(W_{s_0}) = \mathbf{E}(V_{s_0})$.

The difficulty of the importance sampling method is to find a suitable $\mathbf{P}'$. We proposed in [10] to perform numerical analysis of an approximation of the chain to produce a suitable matrix $\mathbf{P}'$ having in mind a variance reduction.

We associate with the model $\mathcal{M}$ a smaller one $\mathcal{M}^\bullet$ whose associated DTMC $\mathcal{C}^\bullet$ is a smaller Markov chain with similar attributes $(S^\bullet, \mathbf{P}^\bullet, \mu^\bullet, \dots)$. The Markov chain $\mathcal{C}^\bullet$ is *reduced* from $\mathcal{C}$ if their exists a *reduction* f, that is a mapping from S to $S^\bullet$ such that $s^\bullet_- = f(s_-)$ and $s^\bullet_+ = f(s_+)$. Note that this reduction is designed at the model level. Our method only uses a particular kind of reductions:

Definition 1: Let $\mathcal{C}$ be a DTMC and $\mathcal{C}^\bullet$ reduced from $\mathcal{C}$ by f. $\mathcal{C}^\bullet$ is a *reduction with guaranteed variance* if for all $s \in S$ such that $\mu^\bullet(f(s)) > 0$ we have :

$$\sum_{s' \in S} \mu^\bullet(f(s')) \cdot \mathbf{P}(s,s') \leq \mu^\bullet(f(s)) \tag{2}$$

Fortunately, we do not need to compute the function $\mu^\bullet$ in order to check that $\mathcal{C}^\bullet$ is a reduction with guaranteed variance. In [10], we developed a structural requirement using coupling theory to ensure these hypotheses are fulfilled. We can now construct an efficient important sampling based on a reduced chain with guaranteed variance.

Proposition 1: Let $\mathcal{C}$ be a DTMC and $\mathcal{C}^\bullet$ be a reduction with guaranteed variance by f. Let $\mathbf{P}'$ be defined by:

- if $\mu^\bullet(f(s)) = 0$ then for all $s' \in S$, $\mathbf{P}'(s,s') = \mathbf{P}(s,s')$
- if $\mu^\bullet(f(s)) > 0$ then for all $s' \in S \setminus \{s_-\}$,
  $\mathbf{P}'(s,s') = \frac{\mu^\bullet(f(s'))}{\mu^\bullet(f(s))}\mathbf{P}(s,s')$ and
  $\mathbf{P}'(s,s_-) = 1 - \sum_{s' \in S} \frac{\mu^\bullet(f(s'))}{\mu^\bullet(f(s))}\mathbf{P}(s,s')$.

The importance sampling based on matrix $\mathbf{P}'$ has the following properties:

- For all s such that $\mu(s) > 0$, W_s is a random variable which has value in $\{0, \mu^\bullet(f(s))\}$.
- $\mu(s) \leq \mu^\bullet(f(s))$ and $\mathbf{V}(W_s) = \mu(s)\mu^\bullet(f(s)) - \mu^2(s)$.
- One can compute a true confidence interval for this importance sampling.

We can now describe the full method:

1) Specify a model $\mathcal{M}^\bullet$ with associated DTMC $\mathcal{C}^\bullet$, and a reduction function f satisfying hypotheses of proposition 1.
2) Compute function $\mu^\bullet$ with a numerical model checker applied on $\mathcal{M}^\bullet$.
3) Compute $\mu(s_0)$ with a statistical model checker applied on $\mathcal{M}$ using the importance sampling of proposition 1.

We extend the requirement of definition 1 to the context of bounded reachability but due to lack of space this extension is kept in our research report [13].

III. EXTENSION TO BOUNDED REACHABILIY

We now want to apply the previously defined method to estimate bounded reachability probabilities. We extend it to bounded reachability in DTMC and then to *Continuous Time Markov Chain* CTMC.

A. Bounded Reachability in DTMC

Given a finite integer horizon u we denote by $\mu_u(s)$ the probability to reach s_+ from s in u steps. The goal now is to estimate $\mu_u(s_0)$.

Adding a countdown timer, we define a new Markov chain $\mathcal{C}_u$ whose state space is $(S \setminus \{s_-, s_+\}) \times [1,u] \cup \{s_-, s_+\}$.

The timer is initialized to u. Except from the two absorbing states s_+ and s_-, all transitions decrease this timer by one. All trajectories of length u not ending in s_+ are sent by the mean of their last transition into the sink state s_-. Therefore the probability to reach s_+ in $\mathcal{C}$ in at most u steps is equal to the probability to reach s_+ in C_u.

Theoretically this allows to use the method described in the previous section in the bounded reachability context. In practice the size of $\mathcal{C}_u$, which is u times the size of $\mathcal{C}$ often make the direct computation intractable. In the following we describe several algorithms bypassing this problem.

B. Bounded Reachability in CTMC

In a continuous time Markov chain, each state s is equipped with an exit rate λ_s. The waiting time in each state s is then distributed according to an exponential law of parameter λ_s.

To apply our method in the continuous setting we use the standard method of uniformization which reduces the problem of bounded reachability in a CTMC to some problems of bounded reachability in the embedded DTMC.

A chain is said to be uniform when the rate $\lambda = \lambda_s$ is independent from s. Given a uniform chain, the probability $\mu_\tau(s)$ to reach the state s_+ in τ time is equal to:

$$\mu_\tau(s) = \sum_{n \geq 0} \frac{e^{-\lambda\tau}(\lambda\tau)^n}{n!} \mu_n(s)$$

Indeed using the uniform hypothesis, $\frac{e^{-\lambda\tau}(\lambda\tau)^n}{n!}$ is the probability that n transitions take place in interval $[0, \tau]$.

Given a non uniform chain with bounded rates, it is routine to transform it in a uniform chain with the same distribution [11] . It consists to select some upper bound of the rates (say λ), consider λ as the uniform transition rate and set a transition matrix $\mathbf{P}_u$ defined by:

$$\begin{aligned} \forall s \neq s' \in S \ \mathbf{P}_u(s, s') &= \tfrac{\lambda_s}{\lambda} \mathbf{P}_u(s, s') \\ \mathbf{P}_u(s, s) &= 1 - \textstyle\sum_{s' \neq s} \mathbf{P}_u(s, s') \end{aligned}$$

We estimate this value by truncating this infinite sum. The Fox-Glynn algorithm [14] allows to compute left (n^-) and right (n^+) truncation points given an error threshold. The errors made by this truncation have to be added to the confidence interval. We obtain a precise formulation of a true confidence interval combining errors from the statistical simulation and from truncation in Fox-Glynn algorithm. Due to lack of space this formulation is kept in the research report [13]. Then terms $\mu_n(s)$ are estimated using the previously defined method.

IV. Algorithmic Considerations

Based on the previous developments, we describe a methodology to perform statistical model checking using importance sampling to estimate the tiny probability $\mu_\tau(s_0)$ to reach the state s_+ in time less than τ in several steps.

1) Specify a reduced a model $\mathcal{M}^\bullet$ whose embedded DTMC $\mathcal{C}^\bullet$ is a reduction with guaranteed variance.
2) Fix some uniform rate λ for the uniformization of $\mathcal{C}$. Compute left and right truncation points n^-, n^+ for the desired error threshold. Then compute for each n between n^- and n^+ the coefficient $\frac{e^{-\lambda\tau}(\lambda\tau)^n}{n!}$.
3) Compute the distributions $\{\mu_n^\bullet\}_{0<n\leq n^+}$ (numerical computations of the iterated power of the transition matrix on $\mathcal{C}^\bullet$).
4) Use these distributions to perform importance sampling on the simulation of the initial model in order to estimate $\mu_u(s)$ for $n^- \leq u \leq n^+$. We generate a large sample of trajectories using the transition system corresponding to matrix P'_u obtained by applying proposition 1 to the DTMC $\mathcal{C}_u$; compute along each path the likelihood L in order to obtain an estimation with accurate confidence interval.
5) Deduce from these confidence intervals the final confidence interval.

The first step requires some understanding of the system to design an appropriate reduced chain. Steps 2 and 3 only require standard computations on finite Markov chains. Step 5 is obtained by weighting with the Poisson probabilities confidence intervals obtained in step 4 and combining them with the numerical error produced by the Fox-Glynn algorithm. See [13] for a precise formulation. We now detail step 4 since it rises algorithmic problems.

We denote by m the number of states of the Markov chain $\mathcal{C}^\bullet$ and by d the maximum of outdegrees of vertices of $\mathcal{C}^\bullet$. Let us remark that in typical modellings, d is very small compared to m. A simulation takes at most u steps going through states $(s_u, u), \ldots, (s_1, 1), s_\pm$ where $s_u = s_0$ and $s_\pm \in \{s_+, s_-\}$. In state (s_v, v), we compute the distribution $P'_u((s_v, v), -)$ (cf. proposition 1), which requires the values of $\mu_v^\bullet(f(s))$ and $\mu_{v-1}^\bullet(f(s'))$, for each possible target state s' from s_v.

Vectors $\{\mu_v^\bullet\}_{0<v\leq u}$ may be computed iteratively one from the other with complexity $\Theta(mdu)$: Precisely, define $\widetilde{\mathbf{P}}^\bullet$ as the substochastic matrix obtained from $\mathbf{P}^\bullet$ by removing state s_- and $\mu_0^\bullet$ as the null vector except for $\mu_0^\bullet(s_+) = 1$; then $\mu_v^\bullet = \widetilde{\mathbf{P}}^\bullet \cdot \mu_{v-1}^\bullet$. But for large values of u, the space complexity to store them becomes intractable and the challenge is to obtain a space-time trade-off. So we propose three methods. The methods consist of a precomputation stage and a simulation stage. Their difference lies in the information stored during the first stage and the additional numerical computations during the second stage. In the precomputation, each method computes iteratively the u vectors $\mu_v^\bullet = (\widetilde{\mathbf{P}}^\bullet)^v(\mu_0^\bullet)$ for v from 1 to u.

1) First method is the "natural" implementation. It consists in storing all these vectors during the precomputation stage and then proceeding to the simulation without any additional numerical computations. The storage of

vectors $\{\mu_v^\bullet\}_{v\leq u}$ is the main memory requirement.

2) Let $l(< u)$ be an integer. In the precomputation stage, the second method only stores the $\lfloor \frac{u}{l} \rfloor + 1$ vectors $\mu_\tau^\bullet$ with τ multiple of l in list Ls and $\mu_{l\lfloor \frac{u}{l} \rfloor+1}^\bullet, \ldots, \mu_u^\bullet$ in list K (see the precomputation stage of algorithm 2). During the simulation stage, in a state (s, τ), with $\tau = ml$, the vector $\mu_{\tau-1}^\bullet$ is present neither in Ls nor in K. So the method uses the vector $\mu_{l(m-1)}^\bullet$ stored in Ls to compute iteratively all vectors $\mu_{l(m-1)+i}^\bullet = P^{\bullet i}(\mu_{l(m-1)}^\bullet)$ for i from 1 to $l-1$ and store them in K (see the step computation stage of algorithm 2). Then it proceeds to l consecutive steps of simulation without anymore computations. We choose l close to $\sqrt{u}$ in order to minimize the space complexity of such a factorization of steps.

3) Let $k = \lfloor \log_2(u) \rfloor + 1$. In the precomputation stage, the third method only stores $k+1$ vectors in Ls. More precisely, initially using the binary decomposition of u ($u = \sum_{i=0}^{k} a_{u,i} 2^i$), the list Ls of $k+1$ vectors consists of $w_{i,v} = \mu^\bullet_{\sum_{j=i}^{k} a_{v,j} 2^j}$, for all $1 \leq i \leq k+1$ (see the precomputation step of algorithm 3). During the simulation stage in a state (s, v), with the binary decomposition of v ($v = \sum_{i=0}^{k} a_{v,i} 2^i$), the list Ls consists of $w_{i,v} = \mu^\bullet_{\sum_{j=i}^{k} a_{v,j} 2^j}$, for all $1 \leq i \leq k+1$. Observe that the first vector $w_{1,v}$ is equal to $\mu_v^\bullet$. We obtain $\mu_{v-1}^\bullet$ by updating Ls according to $v-1$. Let us describe the updating of the list performed by the step-computation of algorithm 3. Let i_0 be the smallest index such that $a_{v,i_0} = 1$. Then for $i > i_0$, $a_{v-1,i} = a_{v,i}$, $a_{v-1,i_0} = 0$ and for $i < i_0$, $a_{v-1,i} = 1$. The new list Ls is then obtained as follows. For $i > i_0$ $w_{i,v-1} = w_{i,v}$, $w_{i_0,v-1} = w_{i_0-1,v}$. Then the vectors for $i_0 < i$, the vectors $w_{i,v-1}$ are stored along iterated $2^{i_0-1} - 1$ matrix-vector products starting from vector $w_{i_0,v-1}$: $w(j, v-1) = P_0^{\bullet 2^j} w(j+1, v-1)$. The computation associated with v requires $1 + 2 + \cdots + 2^{i_0-1}$ products matrix-vector , i.e. $\Theta(md2^{i_0})$. Noting that the bit i is reset at most $m2^{-i}$ times, the complexity of the whole computation is $\sum_{i=1}^{k} 2^{k-i}\Theta(md2^i) = \Theta(mdu\log(u))$.

The three methods are numbered according to their decreasing space complexity. The corresponding space-time trade-off is summarized by table I, where the space unit is the storage of a float.

V. Experimentation

A. Implementation

Tools. Our experiments have been performed on Cosmos, a statistical model checker whose input model is a stochastic Petri net with general distributions and formulas are expressed by the logic HASL [4]. We have also used the model checker Prism for comparisons with our method. All

Algorithm 2:

```
Precomputation(u, μ_0^•, P_0^•)
Result: Ls, K
// List Ls fulfills Ls(i) = μ_{i·l}^•
l ← ⌊√u⌋
w ← μ_0^•
for i from 1 to ⌊u/l⌋l do
    w ← P_0^• w
    if i mod l = 0 then
        Ls(i/l) ← w
// List K contains μ_{⌊u/l⌋l+1}^•, ..., μ_u^•
for i from ⌊u/l⌋l + 1 to u do
    w ← P_0^• w
    K(i mod l) ← w

Stepcomputation(v, l, P_0^•, K, Ls)
// Updates K when needed
if v mod l = 0 then
    w ← Ls(v/l − 1)
    for i from (v/l − 1)l + 1 to v − 1 do
        w ← P_0^• w
        K(i mod l) ← w
```

Algorithm 3:

```
Precomputation(u, μ_0^•, P_0^•)
Result: Ls
// Ls fulfills Ls(i) = μ^•_{Σ_{j=i}^k a_{u,j} 2^j}
k ← ⌊log_2(u)⌋ + 1
v ← μ_0^•
Ls(k + 1) ← v
for i from k downto 0 do
    if a_{u,i} = 1 then
        for j from 1 to 2^i do
            w ← P_0^• w
    Ls(i) ← w

Stepcomputation(v, l, P_0^•, Ls)
// Ls is updated accordingly to v − 1
i_0 ← min(i | a_{v,i} = 1)
w ← Ls(i_0 + 1)
Ls(i_0) ← v
for i from i_0 − 1 downto 0 do
    for j = 1 to 2^i do
        w ← P_0^• w
    Ls(i) ← w
```

Table I
COMPARED COMPLEXITIES

Complexity	Method 1	Method 2	Method 3
Space	mu	$2m\sqrt{u}$	$m \log u$
Time for the precomputation	$\Theta(mdu)$	$\Theta(mdu)$	$\Theta(mdu)$
Additional time for the simulation	0	$\Theta(mdu)$	$\Theta(mdu \log(u))$

the experiments have been performed on a computer with twelve 2.6Ghz processors and 48G of memory[1].

Adaptation of COSMOS. In addition to the implementation of our algorithms, we have done two main modifications on the tool in order to integrate our method. First, the probabilities of the Poisson distribution are computed by a freely available implementation of the Fox-Glynn algorithm [15]. Second, COSMOS sequentially generates a batch of trajectories. In our context this is highly inefficient since the numerical computations of $\mu_n^{\bullet}$ required by algorithms 1 and 2 should be repeated for every trajectory. So one generates a bunch of trajectories in parallel step by step. Different sizes of bunches are possible but they cannot exceed the size required for the numerical computations. Based on the asymptotic time and space cost of these computations, we handle m^2 trajectories.

B. Global Overflow in Tandem Queues

Let us present an experimentation on tandem queues. This example is a classical benchmark for importance sampling. It has also practical interest as a standard modeling of networks [16]. Such a modeling allows to accurately dimension a network for a given loadwork.

Specification. We consider a system of k queues in serie. A client arrives in the first queue with rate ρ_0. In queue i ($i < k$), a client is served with rate ρ_i and then go to the next queue. In the last queue, clients leave the system with rate ρ_k. For this model we can construct a reduced one by bounding the number of clients except in the first queue by a parameter R. A suitable coupling relation can be established in order to ensure the hypotheses of definition 1 as described in [13]. We are interested in estimating the probability for the system to overflow i.e. there is more than $H = 50$ clients in the whole system before being empty in less than $\tau = 100$ time units.

Choice of parameters. We choose the parameters of the system as follows. $\rho_0 = 0.25$ and for all $1 \leq i$ $\rho_i = 0.375$. We study the behaviour of the methods for different values of k. We have chosen for the reduced model $R = 5$ as we experimentally found that this value of R yields a tight confidence interval. We generated 1000 simulations to estimate every $\mu_n(s_0)$ with a confidence level for the simulation of 10^{-6}.

[1]Here, K means Kilobyte, M means Megabyte and G means Gigabyte.

Fox-Glynn algorithm. We plotted in figure 1 the curves $\mu_n(s)$, $\frac{e^{-\lambda}\lambda^n}{n!}$ and $\frac{e^{-\lambda}\lambda^n}{n!}\mu_n(s)$ for the tandem queues with two queues and $\lambda = 100$ with logarithmic scale. The quantity which we estimate is $\sum_{n=0}^{\infty} \frac{e^{-\lambda}\lambda^n}{n!}\mu_n(s)$. We observe that for $n < 50$, $\mu_n(s_0) = 0$ whereas the Poisson probability for such a n is not null. Therefore a left truncation of $n^- = 50$ on the Fox-Glynn does not produce any error. On the right part of the Poisson distribution we notice that after the maximum ($n = 100$) the curve decreases while the curve of μ_n increases. Thus the maximum of the product is shifted to the right compared to the maximum of the Poisson probabilities. In order to get a confidence interval of $10^{-1}\mu_\tau(s_0)$ a big enough right truncation index is required. We choose a right truncation on the index $n^+ = 206$ in order to bound the error by 10^{-10} in the Fox-Glynn algorithm.

Analysis of confidence interval. We collected our results with respective time and space consumption for the three algorithms and PRISM in table II. We also computed the value μ with a confidence level of 0.001 estimated with method described in [10]. The overall confidence level is then equal to $(206-50)\times10^{-6}+0.001 = 156\cdot10^{-6}+0.001 = 0.001156$ using formula (2) from [13]. In all experiments, the width of the confidence interval is ten times smaller than the estimated value. Moreover when the numerical computation terminates, the result belongs to the confidence interval. With our choice of truncation indices, the contribution of the right truncation of the Poisson distribution to the length of the confidence interval is several magnitude orders less than the contribution associated with the statistical estimations. So in order to reduce this length, we should increase the number of simulations letting unchanged the truncation index n^+.

Analysis of numerical and statistical PRISM. We compare our method to numerical and statistical model checking done by PRISM. Due to the rarity of the considered event the statistical approach always fails returning 0. We observe that for small models ($k \leq 4$), PRISM numerical model checker is faster and uses less memory than COSMOS. For $k = 5$, our method is 10 times faster and uses up to 28 times less memory. For $k \geq 6$, PRISM crashes due to a lack of memory.

Comparison of the three methods. While the empirical storage behaviour of the three methods follows the theoretical study, memory does not constitute a bottleneck until $k = 8$. For this value, memory required by method 1 is too important. In order for method 2 to fail, farther time horizons must be chosen.

VI. CONCLUSION

We proposed a method of statistical model checking in order to compute with accuracy a tiny probability associated with a timed temporal formula on a CTMC. We obtain a true confidence interval bounding this value. We have developed a theoretical framework justifying the validity of a confidence interval and ensuring the reduction of the

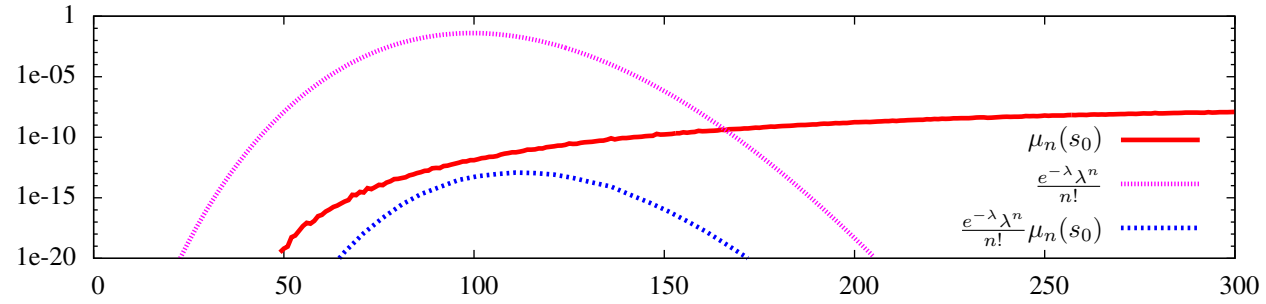

Figure 1. Repartition of Poisson and $\mu_n(s)$ probabilities

Table II
EXPERIMENTAL RESULTS FOR THE TANDEM QUEUES

k	Size of $\mathcal{C}$	numerical PRISM			Cosmos												
								Method 1			Method 2			Method 3			
		T (s)	Mem	$\mu_\tau(s_0)$	$\mu_\tau(s_0)$	$\mu_\infty(s_0)$	Conf. Int.	T_{pre}	T_{sim}	Mem	T_{pre}	T_{sim}	Mem	T_{pre}	T_{sim}	Mem
2	2601	0.021	156K	1.996e-13	1.993e-13	3.764e-8	1.732e-14	≈ 0	68	140M	≈ 0	69	140M	≈ 0	73	158M
3	132651	1.36	4.3M	1.694e-12	1.692e-12	9.196e-7	1.271e-13	≈ 0	144	202M	≈ 0	141	200M	≈ 0	137	200M
4	6765201	107	168M	9.381e-12	9.392e-12	1.524e-5	4.997e-13	1	243	259M	2	246	239M	1	250	237M
5	$\approx$345e+6	5306	8400M	3.941e-11	3.941e-11	2.290e-4	1.725e-12	7	501	439M	7	538	310M	7	561	300M
6	$\approx$17e+9	Out of Memory			1.355e-10	2.355e-3	4.031e-12	57	2577	1347M	57	2278	509M	54	2470	448M
7	$\approx$897e+9				4.013e-10	8.391e-3	9.998e-12	415	33262	7039M	487	31942	1581M	387	33087	1213M
8	$\approx$45e+12				1.051e-09	0.088	2.757e-11	Out of Memory			3030	261050	7502M	2896	267357	5157M

variance. As the memory requirements (which depend on the time horizon) put a kurb on the efficiency of the method, we propose three algorithms with a different trade-off between time and space. We have implemented these algorithms in the statistical model checker COSMOS and we have done experiments on several examples. We detailed one of them in the paper.

We plan to go further in several directions. Our first goal is to deal with infinite models whose reduction yields an infinite one and more expressive language logical formula. Finally we aim at defining formalisms on which the reduced model can be automatically produced.

REFERENCES

[1] C. Baier, B. Haverkort, H. Hermanns, and J.-P. Katoen, "Model checking continuous-time markov chains by transient analysis," 2000.

[2] M. Kwiatkowska, G. Norman, and D. Parker, "Stochastic model checking," in *SFM'07*, ser. LNCS, vol. 4486. Springer, 2007, pp. 220–270.

[3] A. Legay, B. Delahaye, and S. Bensalem, "Statistical model checking: an overview," in *RV 10*. Springer, 2010, pp. 122–135.

[4] P. Ballarini, H. Djafri, M. Duflot, S. Haddad, and N. Pekergin, "HASL: An expressive language for statistical verification of stochastic models," in *VALUETOOLS'11*, Cachan, France, May 2011, pp. 306–315.

[5] G. Chiola, G. Franceschinis, R. Gaeta, and M. Ribaudo, "GreatSPN 1.7: Graphical editor and analyzer for timed and stochastic Petri nets," *Perform. Eval.*, vol. 24, no. 1-2, pp. 47–68, 1995.

[6] M. Kwiatkowska, G. Norman, and D. Parker, "PRISM: Probabilistic symbolic model checker," in *Computer Performance Evaluation: Modelling Techniques and Tools*, ser. LNCS. Springer, 2002, vol. 2324, pp. 113–140.

[7] J. Bengtsson, K. G. Larsen, F. Larsson, P. Pettersson, and W. Yi, "UPPAAL - a tool suite for automatic verification of real-time systems," in *Hybrid Systems*, 1995, pp. 232–243.

[8] H. Younes, "Ymer: A statistical model checker," in *Computer Aided Verification*, ser. LNCS. Springer, 2005, vol. 3576, pp. 171–179.

[9] G. Rubino and B. Tuffin, *Rare Event Simulation using Monte Carlo Methods*. Wiley, 2009.

[10] B. Barbot, S. Haddad, and C. Picaronny, "Coupling and importance sampling for statistical model checking," in *TACAS*, ser. LNCS. Springer, Mar. 2012, pp. 331–346.

[11] A. Jensen, "Markoff chains as an aid in the study of markoff processes," *Skand. Aktuarietidskr*, 1953.

[12] T. Lindvall, *Lectures on the coupling method*. Dover, 2002.

[13] B. Barbot, S. Haddad, and C. Picaronny, "Importance sampling for model checking of continuous-time Markov chains," Laboratoire Spécification et Vérification, ENS Cachan, France, Research Report LSV-12-08, May 2012.

[14] B. L. Fox and P. W. Glynn, "Computing poisson probabilities," *Commun. ACM*, vol. 31, no. 4, pp. 440–445, 1988.

[15] D. N. Jansen, "Understanding Fox and Glynn's "computing poisson probabilities"," Nijmegen: Radboud Universiteit, Tech. Rep. ICIS-R11001, 2011.

[16] L. Kleinrock, *Queueing Systems*. Wiley Interscience, 1976, vol. II: Computer Applications.