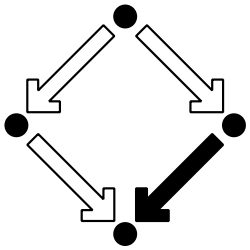
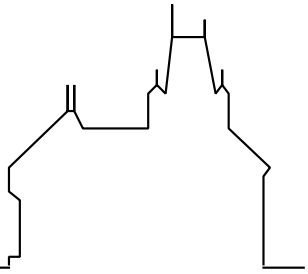


RISC-Linz

Research Institute for Symbolic Computation
Johannes Kepler University
A-4040 Linz, Austria, Europe



Applying Statistical Model Checking to the Analysis of a Retrial Queueing System with Constant Time Impatience

Wolfgang SCHREINER and János SZTRIK

(September 2019)

RISC-Linz Report Series No. 19-12

Supported by: Austrian-Hungarian Bilateral Cooperation in Science and Technology project 2017-2.2.4-TeT-AT-2017-00010 and the Aktion sterreich-Ungarn project 101u7.

Applying Statistical Model Checking to the Analysis of a Retrial Queueing System with Constant Time Impatience*

Wolfgang Schreiner
Research Institute for Symbolic Computation (RISC)
Johannes Kepler University, Linz, Austria
Wolfgang.Schreiner@risc.jku.at

János Sztrik
Department of Informatics Systems and Networks, Faculty of Informatics
University of Debrecen, Debrecen, Hungary
sztrik.janos@inf.unideb.hu

September 16, 2019

Abstract

We report on experiments with the automated analysis of a finite-source queueing system with an unreliable server where clients abort unserved requests after some maximum (constant) amount of time. In earlier work, we created a CTMC model of this system where the constant time bound was approximated by an Erlang distribution and analyzed this model with the probabilistic model checker PRISM. However, due to state space explosion only instances of the system with a very small number of clients could be analyzed. In this paper, we extend the results to larger systems by applying the statistical model checking capabilities of PRISM where results are approximated by sampling a large number of simulation runs. In particular, we address the change from the analysis of the steady state behavior of the system to the analysis of finite runs of the system and the trade-off between the accuracy of the results and the computational efforts needed to derive them.

*Supported by the Aktion Österreich-Ungarn project 101öu7.

Contents

1 Introduction

This paper builds upon the work presented in our previous paper "On the Probabilistic Model Checking of a Retrial Queueing System with Unreliable Server, Collision, and Constant Time Impatience" [3]. In that work, we applied the probabilistic model checker PRISM [1, 2] to model a certain kind of retrial queueing system; however, due to the problem of "state space" explosion, the analysis was only possible for models of very small size. In this paper, we investigate how the technique of *statistical model checking* can be applied to also analyze larger model instances.

In more detail, we consider a system with the following characteristics (see Figure 1): There are N clients that request being served at rate λ each (here and in the following "rate" means the parameter of an exponential distribution). A server processes client requests at rate μ; after a client has been served, it may start another service request. If a client finds upon a newly generated request the server busy with serving another client, a *collision* occurs and both clients move to an *orbit*. A client in the orbit retries its service request at rate ν; also a retrial may cause a collision in the server, which moves the currently served client into the orbit. If a client in the orbit does not find the server ready to accept its service request after T time units, it *aborts* the current request and becomes ready to start another one. If the server is idle, it may *fail* with rate γ_0. If the server is busy, it may fail with rate γ_1; this moves the currently served client to the orbit. While being in the failure state, the server does not serve clients; and all newly arriving requests go into the orbit. A failed server is repaired with rate γ_2 and then becomes ready to serve clients again.

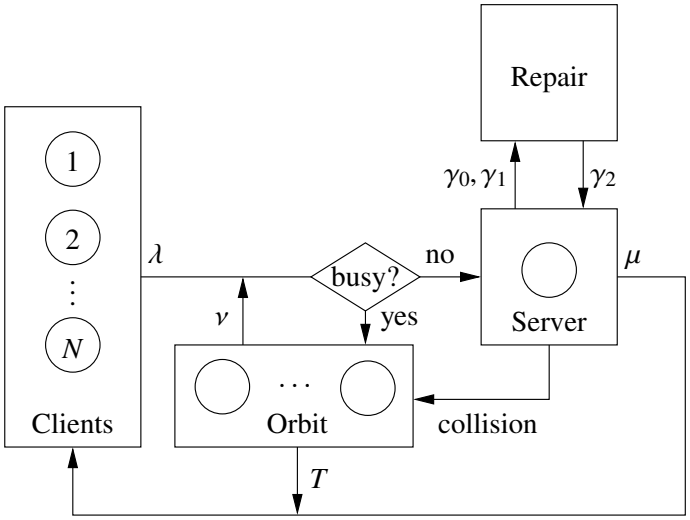


Figure 1: The System Model

In [3] we described this system in PRISM by a CTMC model. Since in such a model all delays can be only described as exponential distributions, the main challenge was to deal with the constant impatience time T. The solution was to equip the model with "alarm clocks", one per client. Each such alarm clock is an automaton that triggers an action after a sequence of

$K > 0$ transitions each of which is exponentially distributed with mean time T/K; consequently the time after which the alarm "rings" follows an Erlang distribution with mean time T and shape K. The special case $K = 1$ represents the exponential distribution; the larger K is, the more "deterministically" the time variable has value T.

The trade-off, however, is that such an alarm clock for each of the N clients causes a "blow-up" of the the size of the model by a factor K^N, with correspondingly harsh consequences on the time and space needed for analyzing the model. Consequently, in [3] we were only able to apply the PRISM model checker to models with $N \leq 6$ clients and Erlang distributions with shape $K \leq 9$. Actually, our analysis determined that for $K = 5$ (which still yields a rather crude approximation of a constant delay) the results did not differ significantly any more from those achieved with higher values of K such that we content ourselves with $K = 5$ for most of the analysis. However, the maximum system size $N = 6$ imposed a serious limitation on the expressiveness of the results.

In this paper we investigate how to overcome this limitation by applying a feature of PRISM that we have not used so far, that of *statistical model checking*. Normally PRISM analyzes a system by constructing and analyzing a probabilistic model, essentially by solving a linear equation systems whose dimension is the number of states of the system; the analysis is thus subject to the problem of state space explosion. However, one may also apply PRISM's simulation capabilities to automatically determine a quantity of interest by statistical *sampling*: without constructing the probabilistic model, PRISM may randomly generate from the system description a large number of (finite) runs of the system, determine the quantity in each of these runs, and compute from these samples an approximately correct result; the accuracy of the approximation is ensured by an automatic analysis of its confidence interval. While the quality of the result determined by statistical model checking thus depends on the number of system runs considered, each run can be computed with comparatively moderate time and space requirements that are independent of the size of the state space,

The goal of this paper is to experimentally determine whether and how well this technique of statistical model checking is also applicable to our problem. For this purpose, we investigate in Section 2 how to adapt the PRISM queries we have used in [3] to be amenable to statistical model checking, we analyze the accuracy of this technique applied to our system, and finally produce experimental results. Section 3 presents our conclusions and outlines directions of further work.

2 Statistical Model Checking the System

Throughout this paper we consider the PRISM model listed in the appendix of [3], but generalized to a maximum of $N = 50$ clients (the language of PRISM does not allow the parameterized construction of system models with N components, thus the model has to be manually adapted, which is a tedious but conceptually simple task).

2.1 CSL Queries for Statistical Model Checking

The main quantity that we will consider in this paper is the "service probability" *PE* of a client, i.e., the probability that an initiated request is indeed successfully served (i.e., not aborted due to collision or impatience). This quantity was determined in [3] by the following CSL queries:

```
"NE": R{"NumExit"}=?  [ S ] ;
"NA": R{"NumAbort"}=? [ S ] ;
"PE": 1/(1+("NA"/"NE"));
```

Here *NE* and *NA* denote the "steady state" quantities of the reward structures `NumExit` respectively `NumAbort`. These rewards denote in a run the number of "exit" transitions of the clients after a successful service respectively the number of unsuccessful "abort" transitions after a premature interruption of the service; from these the probability *PE* can be calculated.

As a minor nuisance, in the "statistical model checking" mode (triggered in the menu of a formula by the entry "Simulate" rather than "Verify" respectively by the option "Use Simulation" in "New Experiment"), PRISM does not support the calculation of *PE*. Thus the values of *NE* and *NA* have to be individually computed and the results exported to an external tool for visualization (this is why the diagrams in this paper have a visual style different from that in [3]).

As a more fundamental problem, statistical model checking cannot compute steady state rewards but only rewards accumulated in finite runs. We thus modify above queries as follows:

```
const double M;
"NE": R{"NumExit"}=?  [ C <= M ] ;
"NA": R{"NumAbort"}=? [ C <= M ] ;
```

In these queries we compute the "cumulative reward" up to time bound M. Provided that M is sufficiently large, the value of $1/(1+NA/NE)$ approximates the corresponding quantity computed by the steady state rewards. In our subsequent experiments (where, e.g., every client requests service every 100 time units and a service is performed in 16 time units) a value of $M = 2000$ is used, which produces results that do not significantly from those produced with larger values.

2.2 The Accuracy of Statistical Model Checking

We use the queries described above to perform our experiments with the same parameter values as given in [3]. However, before performing the actual analysis, we are interested in ensuring the accuracy of the computed results.

For this, PRISM supports (among others) the *confidence interval* method, which depends on the parameters

- interval (half-)width w,
- confidence α, and
- number of samples S

with the following interpretation: if the statistical model checking repeatedly computes an estimation x' for an unknown quantity, in a fraction of $1-\alpha$ of these computations, the actual value x of this quantity is in the interval $[x'-w, x'+w]$. It should be noted that the parameter α (confusingly called "confidence" by PRISM) is the fraction of computations for which the actual value is *not* in the interval.

In PRISM the user can fix two of these parameters, from which PRISM determines the third parameter such that above interpretation holds. In particular,

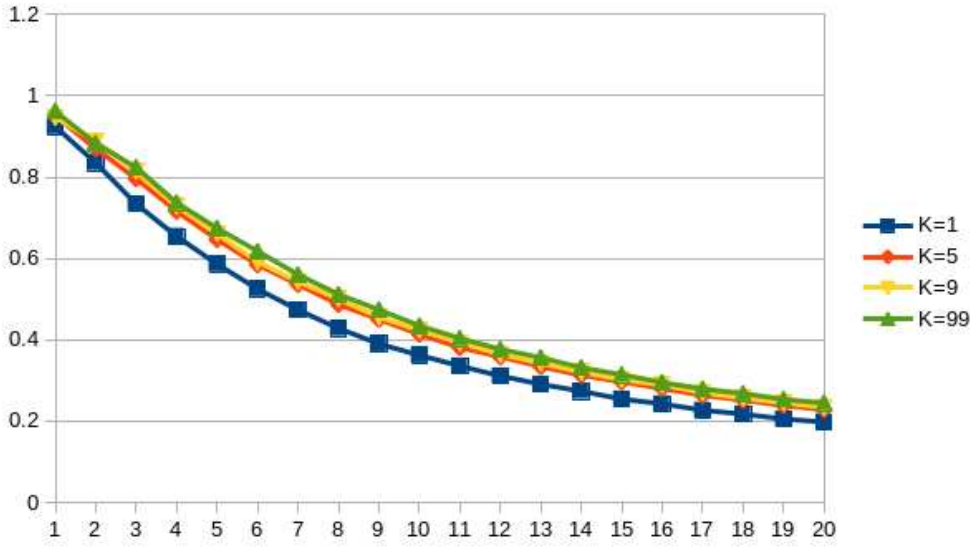


Figure 2: Service Probability for $1 \leq N \leq 20$ Clients ($w = 0.5$, $\alpha = 0.1$)

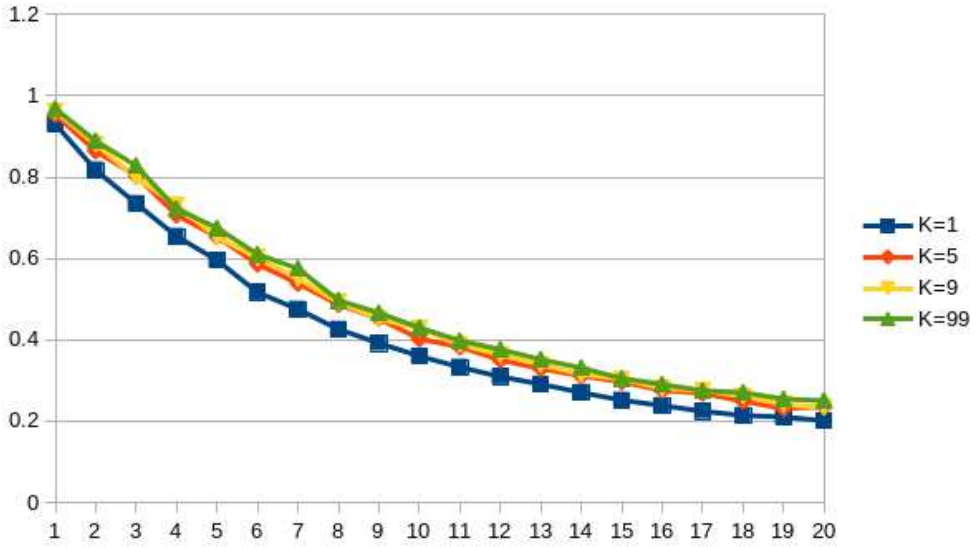


Figure 3: Service Probability for $1 \leq N \leq 20$ Clients ($S = 50$)

- if we fix w and α, PRISM automatically performs the statistical model checking with the necessary number of samples S;
- if we fix S and w, PRISM reports the half-width w of the confidence interval;
- if we fix S and α, PRISM reports the confidence α in the interval.

Figure 2 demonstrates the results of statistically model checking our system for $1 \leq N \leq 20$ clients and varying Erlang shapes K with $\alpha = 0.1$ (i.e., a confidence of 90%) and $w = 0.5$. Thus PRISM computes here very accurate results with a reasonably high confidence; these results very well match those that were derived in [3] for $1 \leq N \leq 6$ by exact model checking. Also as in [3], we note that the results for $K = 1$ (corresponding to the exponential distribution) differ substantially from those for the larger values of K, but for $K \geq 5$, the values the values are very similar. In particular, the values for $K = 9$ and $K = 99$ (which very closely approximates a constant distribution) do not significantly differ from each other.

The downside of above check is that for larger values of N we require a very high number of runs to achieve the required level of confidence. For instance, in the case of $K = 5$, checking $N = 1$ client requires $S \approx 150$ runs, but checking $N = 20$ clients requires $S \approx 650$ runs. Since larger values of K also require more transitions to perform a run up to time bound M, the checking time grows correspondingly: with $N = 12$, a check for $K = 12$ takes about two minutes, while a check for $K = 99$ takes about eight minutes; a check for $K = 99$ and $N = 20$ takes about 1000 runs and 25 minutes time.

Thus while narrow confidence intervals and high confidence levels are theoretically attractive, they pragmatically lead to high checking times. Thus we also investigate how the overall results are practically affected by the choice of a fixed number S of system runs with possibly wide confidence intervals respectively low confidence values but also with much smaller checking times. For instance, Figure 3 demonstrates the results for $S = 50$ system runs. Clearly the curves are not as "smooth" as those depicted in Figure 2, which illustrates the higher uncertainty of the results. However, all in all they nevertheless show not only very much the same qualities (in particular the large difference between the case $K = 1$ and the other cases which are again very similar), but also that the absolute quantities are not far off. This is a quite appealing result considering that each data point has been computed in a much shorter time (considerably less than a minute).

2.3 Analyzing Large Models by Statistical Model Checking

Based on the results of above investigations, we will now analyze larger instances of our system with $5 \leq N \leq 50$ clients and Erlang shape $K = 9$ using a fixed number of $S = 100$ system runs; manual inspection determines that these values yield with 90% confidence a half-width w of the confidence intervals that is about 1–3% of the approximated result. Furthermore, the Erlang shape $K = 9$ represents a significantly better approximation of constant time impatience than the value $K = 5$ we used in [3]; as Figure 2 demonstrates, we can expect that the derived results do not differ significantly any more from those for a system with "true" constant time impatience.

In detail, we have derived the following results (see Figure 4):

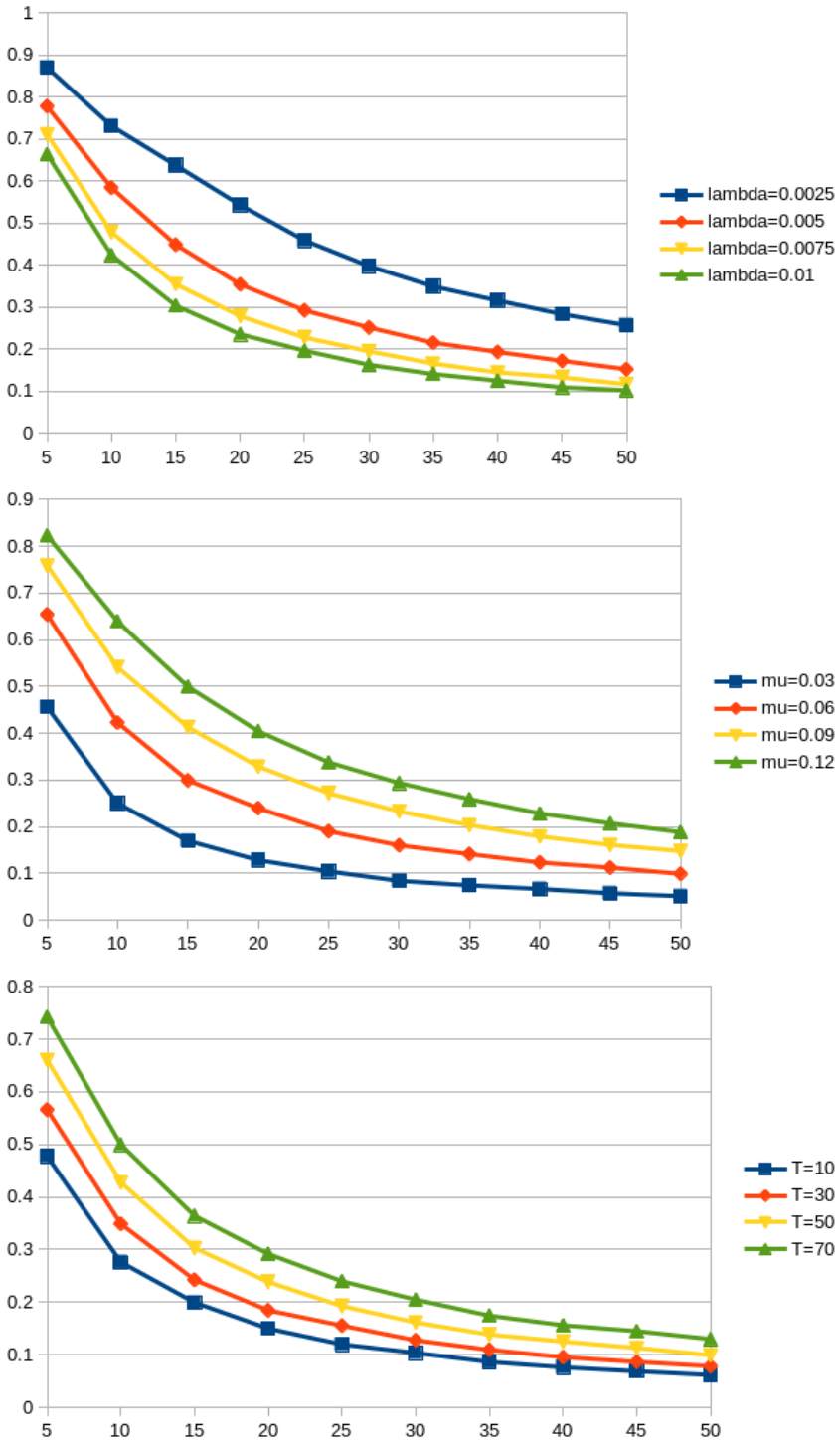


Figure 4: Service Probability PE for $K = 9$ with $5 \leq N \leq 50$ Clients ($S = 100$)

- The upper diagram in the figure determines the effect of the request rate λ on the service probability PE; we see that higher request rates lead to lower service probability due to the larger number of collisions; for $\lambda \geq 0.0075$, the results start to converge.
- The middle diagram determines the effect of the service rate μ on PE; we see that higher service rates lead to higher service probability due to the smaller number of collisions; for $N \geq 50$, the results start to converge.
- The lower diagram determines the effect of the impatience time T on PE; we see that larger impatience time lead to lower service probability due to the smaller number of premature aborts; for $N \geq 50$, the results start to converge.

All curves show "smooth" shapes, which indicates that the inaccuracies of statistical sampling do not affect the results in a major way.

Figure 5 presents (for the same parameters as in Figure 4) the average time W that a client that is successfully served has spent in the system (waiting time in orbit plus service time, also considering its failed service attempts). This time is determined by application of Little's law as $K/(\lambda \cdot (N - K) \cdot PE)$ where the number K of clients in the system (orbit or server) is determined by a CSL query for a corresponding reward structure. The results generally exhibit the expected qualities:

- The higher the arrival rate λ, the higher the time is that a client spends in the system due to the increased number of collisions.
- The higher the service rate μ, the lower the time is due the reduced number of collisions.
- The lower the timeout bound T is, the lower the time is due to more clients leaving the system without having been served.

However, the curves are not so "smooth" any more, which demonstrates a somewhat greater uncertainty in their accuracy (W depends in a non-linear way on approximations of three values NE, NA, and K); here a larger sample number than $S = 100$ (which however already yields a computation time of up to 5 minutes per query) would be beneficial.

3 Conclusions

We have investigated in this paper how large instances of a CTMC model of a retrial queuing system with constant time impatience can be analyzed in PRISM. The previously experienced limits on model sizes imposed by exact model checking are now mitigated by statistical model checking where the derived results are approximated with reasonable accuracy.

The use of PRISM, rather than a manually crafted simulation program, for this kind of analysis provides two core advantages: First, the builtin confidence interval analysis allows to easily determine the accuracy of the analysis respectively to adapt the number of samples computed to ensure a certain level of accuracy. Second, the same model can be used to derive (for smaller model instances) exact results for the steady state behavior of the system; this may be used to validate the approximations derived from analyzing finite runs of the system.

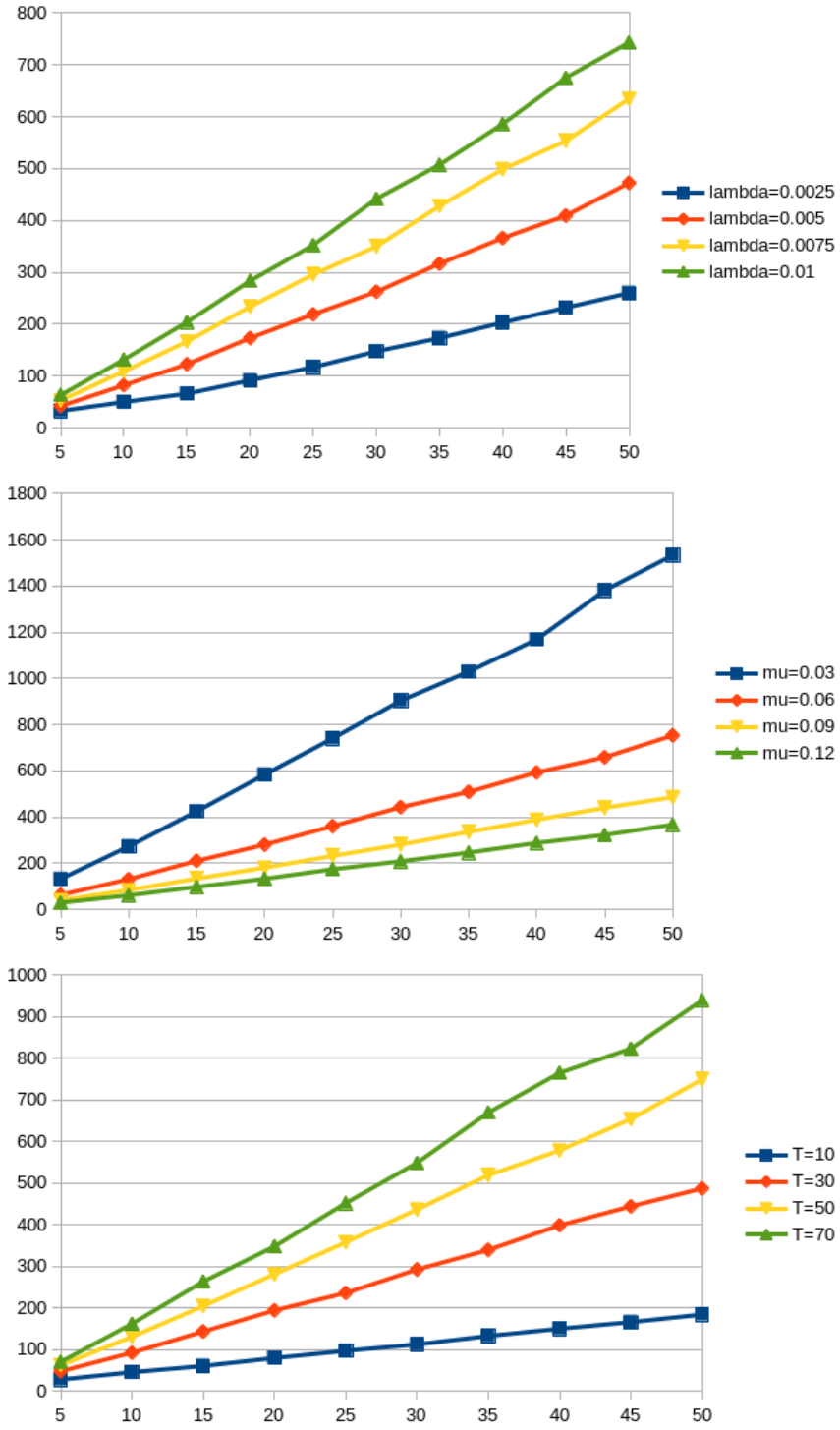


Figure 5: Spending Time W for $K = 9$ with $5 \leq N \leq 50$ Clients ($S = 100$)

All in all the results demonstrate that the statistical model checking features of PRISM may be attractive for the kind of problems we have addressed in this paper. Nevertheless it must be noted that statistical model checking in PRISM also has its limitations: In particular it only supports a limited amount of queries, essentially the computation of rewards accumulated over finite runs (respectively of probabilities that these runs satisfy certain properties). Thus the analysis of the "steady state" behavior of such a model has to be approximated by the analysis of long runs of this model (this, however, is true for all simulation-based approaches).

To further validate these results, we plan to also derive corresponding results from manually crafted simulation programs which may also employ *true* constant impatience rather than the Erlang-based approximation of constant impatience; it will be interesting to compare the accuracy and the cost/benefit ratio of the PRISM approach to such an explicit programming approach. On the long term, these experiments shall pave the way to an analytic analysis of retrial queuing system with constant time impatience.

References

[1] M. Kwiatkowska, G. Norman, and D. Parker. "PRISM 4.0: Verification of Probabilistic Real-time Systems". In: *Proc. 23rd International Conference on Computer Aided Verification (CAV'11)*. Ed. by G. Gopalakrishnan and S. Qadeer. Vol. 6806. Lecture Notes in Computer Science. Springer, 2011, pp. 585–591. DOI: 10.1007/978-3-642-22110-1_47.

[2] David A. Parker, ed. *PRISM — Probabilistic Symbolic Model Checker*. http://www.prismmodelchecker.org. Department of Computer Science, University of Oxford, UK. 2013.

[3] Wolfgang Schreiner and János Sztrik. *On the Probabilistic Model Checking of a Retrial Queueing System with Unreliable Server, Collision, and Constant Time Impatience*. Tech. rep. Johannes Kepler University, Linz, Austria: Research Institute for Symbolic Computation (RISC), July 2019. URL: https://www.risc.jku.at/publications/download/risc_5954/main.pdf.