Using Observation Ageing to Improve Markovian Model Learning in QoS Engineering

[Work in Progress]

Radu Calinescu, Kenneth Johnson and Yasmin Rafiq
Computer Science Research Group, Aston University, Birmingham B4 7ET, UK
{R.C.Calinescu,K.H.A.Johnson,rafiqy}@aston.ac.uk

If I am going to have a true memory, there are a thousand things that must first be forgotten.

Thomas Merton

ABSTRACT

Markovian models are widely used to analyse quality-of-service properties of both system designs and deployed systems. Thanks to the emergence of probabilistic model checkers, this analysis can be performed with high accuracy. However, its usefulness is heavily dependent on how well the model captures the actual behaviour of the analysed system. Our work addresses this problem for a class of Markovian models termed *discrete-time Markov chains* (DTMCs). We propose a new Bayesian technique for learning the state transition probabilities of DTMCs based on observations of the modelled system. Unlike existing approaches, our technique weighs observations based on their age, to account for the fact that older observations are less relevant than more recent ones. A case study from the area of bioinformatics workflows demonstrates the effectiveness of the technique in scenarios where the model parameters change over time.

Categories and Subject Descriptors

D.2.8 [**Software Engineering**]: Metrics; G.3 [**Probability and statistics**]: Markov processes, Reliability and life testing

General Terms

Algorithms, Measurement, Reliability, Theory

1. INTRODUCTION

Markovian models are increasingly used to model and analyse quality-of-service (QoS) properties of technical systems. The explanation for this trend is twofold. Firstly, the use of technical systems in safety- and business-critical applications is on the rise, and so is the importance of ensuring that these systems comply with strict performance and reliability requirements. Secondly, the last decade has brought the advent of powerful software tools that assist system developers in building Markovian models, and automate model analysis. These tools are termed *probabilistic model checkers*. They take as input a formal system description expressed in a high-level modelling language, and convert it into a Markovian chain that they then use to analyse user-specified QoS properties. Example of such properties include the probability that a fault occurs within a specified time period, and the expected response time of a web service under a given workload. Widely used probabilistic model checkers include PRISM[7], MRMC[9] and Ymer[14].

Probabilistic model checking can be employed effectively at several stages in the life cycle of a technical system. During system design, the technique can be used to analyse alternative solutions and identify those that satisfy the envisaged QoS requirements without having to build and test potentially expensive system prototypes. For existing systems, probabilistic modelling and analysis can be used to verify whether QoS requirements are achieved or remedial action is needed to ensure compliance. A repository of case studies that fit these descriptions and spawn a broad range of application domains is available from the PRISM web site.[1] More recently, probabilistic model checking has been used to guide self-optimisation in autonomic IT systems during their execution stage [2, 3, 4].

Alternative techniques for analysing the QoS properties of technical systems include simulation and testing. However, these techniques can only examine a finite (and often small) number of scenarios that the system may operate in. As many systems are associated with a very large or even infinite number of scenarios, the results produced by simulation and testing are approximate and cannot guarantee compliance with QoS requirements. In contrast, probabilistic model checking performs an exhaustive analysis of the considered QoS properties, producing precise results that guarantee or disprove each analysed property irrefutably [10].

There is one proviso to the above statement: the usefulness of probabilistic model checking depends on how accurate the analysed Markovian models are. Any inaccuracies in the structure or parameters of a model will unavoidably lead to assessments that do not reflect the actual QoS compliance of the real-world system. Getting the model structure wrong invalidates completely the analysis, but is less likely to happen—at least when the model is built by an

[1] `http://www.prismmodelchecker.org/casestudies`

ICPE'11, March 14–16, 2011, Karlsruhe, Germany.

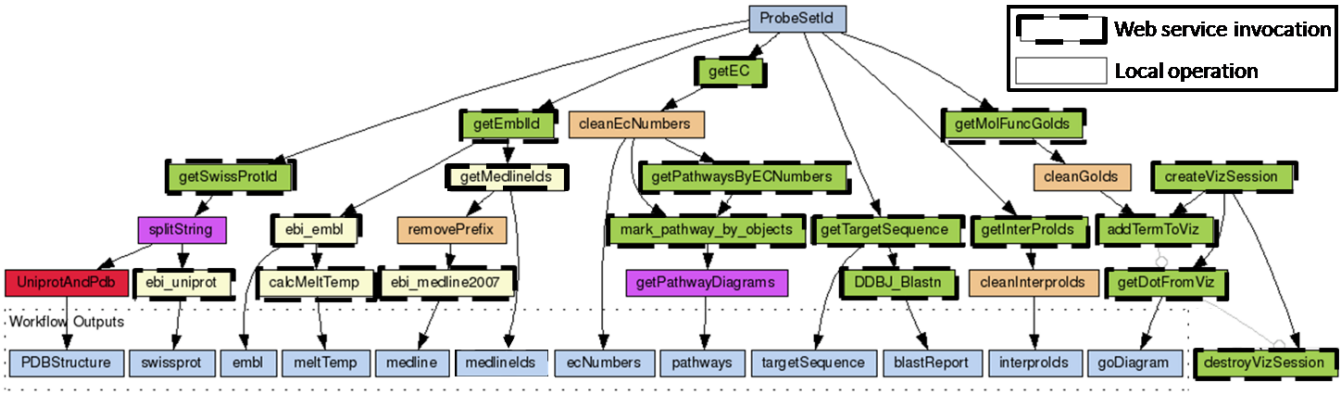

Figure 1: Bioinformatics workflow taken from `http://www.myexperiment.org/workflows/28.html`

experienced user of the tool. Using the wrong parameter values, however, could easily be encountered even in models developed by expert tool users, as Markovian model parameters are state transition probabilities or rates that are difficult to measure accurately. In fact, these parameters often vary during the lifetime of a system, as a result of changes in system workload, environment and internal state. When this is the case, no one set of parameter values exists that can be used to fully and accurately analyse system QoS compliance. What is required instead is to continually and accurately *learn* the parameter values as the system is evolving, and to carry out the assessment each time they change.

We propose a new approach to learning the values of discrete-time Markov chain (DTMC) parameters based on *observations of the modelled system behaviour*. Note that it is also possible to learn DTMC parameters through *analysing the system specification, code, workload and/or allocated resources*. The work described in this paper, however, focuses on scenarios in which this information is unavailable, e.g., because the modelled system components are services provided by third parties.

Previous work on observation-based estimation of DTMC parameters [5] has used standard Bayesian learning that works well when these parameters are unknown but constant. However, this technique places equal importance on all observations of the modelled system, irrespective of how recent or old they are. Therefore, the learning approach in [5] is less suitable for the frequently encountered scenarios in which parameters change over time. In this case, the most recent observations of the system are the most relevant.

The main contribution of this paper is to extend the learning technique from [5] by introducing an *ageing coefficient* that is used to weight each observation with its age. That is, the older the observation, the less relevant it becomes. The other contributions are an evaluation of the new Bayesian learning technique using a case study from the area of bioinformatics workflows, and an analysis of how to choose its ageing coefficient.

The paper outline is as follows. Section 2 provides the necessary definitions and an outline of the standard Bayesian learning method we have extended. Section 3 gives a formal description of the new technique, and Section 4 evaluates its effectiveness in the context of a real-world case study. Section 5 describes related work. Section 6 summaries our results and provides directions for further research.

2. BACKGROUND

2.1 Probabilistic model checking of DTMCs

For the purpose of QoS engineering, a DTMC is represented as a tuple

$$M = (S, s_1, P, L), \tag{1}$$

where: $S = \{s_1, \ldots, s_n\}$ is a finite set of $n \geq 1$ states; s_1 is the initial state; P is an $n \times n$ transition probability matrix; and $L : S \to 2^{AP}$ is a labelling function which assigns a set of atomic propositions from AP to each state in S. The element p_{ij} from P represents the probability of transitioning to state j from state i, $1 \leq i, j \leq n$, and $\sum_{j=1}^{n} p_{ij} = 1$.

Probabilistic model checkers operate on Markovian models expressed in a high-level, state-based language. Given a DTMC description in this language, the low-level representation (1) is derived automatically. Our work uses the probabilistic model checker PRISM [7], which supports the analysis of DTMC properties specified in a reward-augmented version of probabilistic computational tree logic (PCTL) [6].

To illustrate the process, we consider a real-world bioinformatics workflow taken from the Taverna repository of scientific workflows myExperiment[2] (Figure 1). Taverna [8] is a workflow engine widely used by research communities from bioinformatics, astronomy and social sciences. The workflow in Figure 1 has been used in studies of an autoimmune disease that represents the most common cause of hyperthyroidism in young people and children (i.e., the *Graves disease*). This workflow was chosen for our case study because it is one of the most complex and most used workflows from the repository (it invokes 18 different web services running at four research centres in the UK and Japan; and had a download count of 166 when last checked by the authors).

A fragment of the PRISM model for this workflow is shown in Figure 2, where $p1$ to $p18$ represent *a priori* estimates of the probabilities that the 18 web service invocations complete successfully, and a PRISM module is used to model each web service. An additional module (`Workflow`) is used to model the workflow as a whole, thus enabling the specification of a reliability QoS requirement such as "the workflow must complete successfully with a probability of at least 0.95" as the PCTL reachability property:

$$\text{"}init\text{"} \Rightarrow P_{\geq 0.95}[\mathrm{F}\ wf = \texttt{SUCC}] \tag{2}$$

[2] `http://www.myexperiment.org`

```
dtmc

const double p1=0.999;
const double p2=0.998;
...
const int SUCC=1;
const int FAIL=2;

module getEmblld
 getEmblld : [0..2] init 0;
 [] (getEmblld=0) -> p1:(getEmblld'=SUCC) +
                     (1-p1):(getEmblld'=FAIL);
endmodule

module Sebi_embl
  ebi_embl : [0..2] init 0;
  [] ebi_embl=0 & (getEmblId=SUCC)-> p2:(ebi_embl'=SUCC) +
                                     (1-p2):(ebi_embl'=FAIL);
  [] (ebi_embl=0) & (getEmblld=FAIL) -> 1:(ebi_embl'=FAIL);
endmodule
...
module WorkFlow
  wf : [0..2] init 0; // 0 - init; 1 - success; 2 - fail
  [] (wf=0) & (ebi_uniprot=FAIL | calcMeltTemp=FAIL | ...
                     | getDotFromViz=FAIL) -> 1:(wf'=FAIL);
  [] (wf=0) & (ebi_uniprot=SUCC & calcMeltTemp=SUCC & ...
                    & getDotFromViz=SUCC) -> 1: (wf'=SUCC);
endmodule
```

Figure 2: PRISM model for the workflow in Figure 1

2.2 DTMC parameter learning

A technique for learning the state transition probabilities p_{ij} for the DTMC (1) when only *a priori* estimates p^0_{ij}, $1 \leq i, j \leq n$, are available initially is presented in [5], and successfully used in the context of workflow QoS management in [2]. This Bayesian learning technique is applicable when the analysed system is operational, and its state transitions are monitored. Suppose that, as a result of this monitoring, we observe $N_i > 0$ transitions from state s_i to other states in S, for each $1 \leq i \leq n$. Also, assume that the k-th observation of a transition from state s_i to another state, $1 \leq k \leq N_i$, is a transition to state s_{j_k}, and that

$$\sigma^k_{ij} = \begin{cases} 1 & \text{if } j = j_k, \\ 0 & \text{otherwise.} \end{cases} \tag{3}$$

Note that $\sum_{j=1}^{n} \sigma^k_{ij} = 1$ since, for each observation k, the system will transition from state s_i to precisely one state from S. The work presented in [2] uses theoretical results from [13] and the Bayes' rule to derive the *updating rule* for estimating the probability p_{ij} after the observation of the k-th transition from state s_i to another state in S as

$$p^k_{ij} = \frac{c^0_i}{c^0_i + k} p^0_{ij} + \frac{k}{c^0_i + k} \frac{\sum_{l=1}^{k} \sigma^l_{ij}}{k}, \tag{4}$$

for $1 \leq k \leq N_i$. The *smoothing parameter* $c^0_i \geq 1$ quantifies the confidence in the accuracy of the a priory estimates p^0_{ij}, $1 \leq i, j \leq n$. For a description of the steps involved in deriving this result the reader is refered to [5].

3. DTMC PARAMETER LEARNING WITH OBSERVATION AGEING

The updating rule (4) was shown [2, 5] to be effective in scenarios where the actual probability p_{ij} differs from the a priori estimate p^0_{ij}, but is a constant. However, in many scenarios involving real-world systems, p_{ij} is prone to changing dynamically. For such scenarios, rule (4) is slow to detect requirement violations or—as will be shown in Section 4—may even fail to detect them when they are short lived.

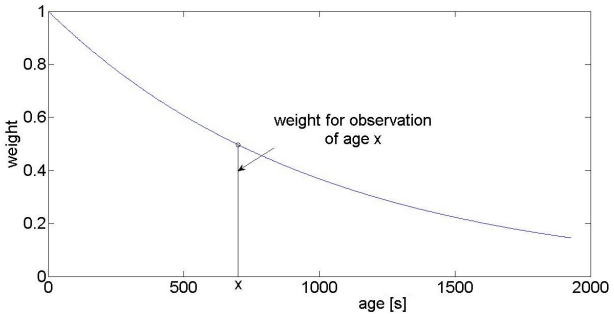

Figure 3: The ageing function α^{-age} is used to weigh observations (shown for $\alpha = 1.001$)

Our extended DTMC parameter learning technique overcomes this limitation by weighting the $k > 0$ observations from the updating rule (4) based on their "age". To achieve this, the extended technique timestamps each observation σ^k_{ij} from eqs. (3)–(4) with the time instant t^k when the observation was made.[3] We assume that the updating rule (4) is applied as soon as the k-th observation is made, i.e., at time moment t_k. Therefore, when the updating rule is applied, the age of a generic observation l, $1 \leq l \leq k$, is precisely

$$age_l = t_k - t_l. \tag{5}$$

To reflect the decreasing importance of observations as they become older in a dynamic scenario, we associate a weight

$$w_l = \alpha^{-age_l} = \frac{1}{\alpha^{t_k - t_l}} \tag{6}$$

with each observation l, $1 \leq l \leq k$, where $\alpha \geq 1$ represents the observation *ageing coefficient* (Figure 3). As shown later in this section, the choice of a negative exponential function as the ageing function is motivated by the ease with which it allows the application of the new updating rule.

The extended updating rule is then obtained by multiplying each σ^l_{ij} term from (4) by its associated weight (6):

$$p^k_{ij} = \frac{c^0_i}{c^0_i + k} p^0_{ij} + \frac{k}{c^0_i + k} \frac{\sum_{l=1}^{k} w_l \sigma^l_{ij}}{\sum_{l=1}^{k} w_l}, \tag{7}$$

for $1 \leq k \leq N_i$. Note that the denominator for the last part of this updating rule was adjusted to $\sum_{l=1}^{k} w_l$ in order to satisfy the invariant $\sum_{j=1}^{n} p^k_{ij} = 1$:

$$\begin{aligned} \sum_{j=1}^{n} p^k_{ij} &= \frac{c^0_i}{c^0_i + k} \sum_{j=1}^{n} p^0_{ij} + \frac{k}{c^0_i + k} \sum_{j=1}^{n} \frac{\sum_{l=1}^{k} w_l \sigma^l_{ij}}{\sum_{l=1}^{k} w_l} = \\ &= \frac{c^0_i}{c^0_i + k} \times 1 + \frac{k}{c^0_i + k} \frac{\sum_{l=1}^{k} \left(w_l \sum_{j=1}^{n} \sigma^l_{ij} \right)}{\sum_{l=1}^{k} w_l} = \\ &= \frac{c^0_i}{c^0_i + k} + \frac{k}{c^0_i + k} \frac{\sum_{l=1}^{k} (w_l \times 1)}{\sum_{l=1}^{k} w_l} = \frac{c^0_i}{c^0_i + k} + \frac{k}{c^0_i + k} = 1. \end{aligned}$$

[3] Recording this additional information typically requires only a small extension to the monitoring part of a system.

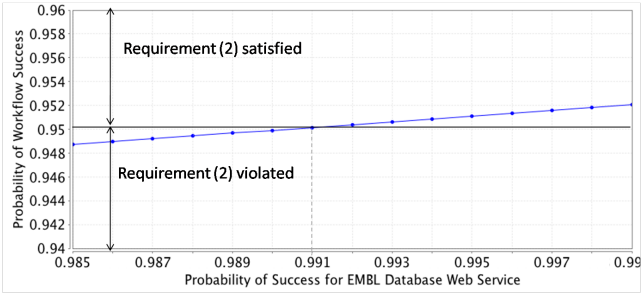

Figure 4: PRISM analysis of the workflow compliance with requirement (2)

Finally, note that selecting an ageing coefficient $\alpha = 1$ makes all weights $w_l = 1$, thus reducing the extended updating rule (7) to the base updating rule in (4). This property represents another advantage of using the ageing function in Figure 3.

Analysis of the extended learning algorithm.

As mentioned earlier, the choice of an exponential negative ageing function simplifies the application of the updating rule (7). Thus, the first term (i.e., $\frac{c_i^0}{c_i^0+k} p_{ij}^0$) and the multiplicative factor $\frac{k}{c_i^0+k}$ can both be calculated in constant, $O(1)$ time. To analyse the complexity of calculating the remaining part of the rule, we introduce the notation $f_{ij}^k = \sum_{l=1}^{k} w_l \sigma_{ij}^l$ and $g_{ij}^k = \sum_{l=1}^{k} w_l$, and observe that:

$$f_{ij}^k = \sum_{l=1}^{k} w_l \sigma_{ij}^l = \sum_{l=1}^{k} \frac{\sigma_{ij}^l}{\alpha^{t_k - t_l}} = \frac{\sigma_{ij}^k}{\alpha^{t_k - t_k}} + \sum_{l=1}^{k-1} \frac{\sigma_{ij}^l}{\alpha^{t_k - t_l}} =$$

$$= \sigma_{ij}^k + \sum_{l=1}^{k-1} \frac{\sigma_{ij}^l}{\alpha^{t_k - t_{k-1}} \alpha^{t_{k-1} - t_l}} = \sigma_{ij}^k + \frac{f_{ij}^{k-1}}{\alpha^{t_k - t_{k-1}}}$$

and, following a similar proof,

$$g_{ij}^k = 1 + \frac{g_{ij}^{k-1}}{\alpha^{t_k - t_{k-1}}}.$$

As a result, the part $\frac{f_{ij}^k}{g_{ij}^k} = \frac{\sum_{l=1}^{k} w_l \sigma_{ij}^l}{\sum_{l=1}^{k} w_l}$ from rule (7) can also be calculated in $O(1)$ time, by using the recursive definitions above for $k \geq 2$, and $f_{ij}^1 = \sigma_{ij}^1$ and $g_{ij}^1 = 1$ for $k = 1$. Furthermore, calculating the right-hand side of (7) at step k does not require the technique to maintain a record of all k observations (i.e., of σ_{ij}^l and t_l for all $1 \leq l \leq k$). The only observation-related values required to carry out the calculation at step k are t_{k-1}, f_{ij}^{k-1}, g_{ij}^{k-1}, σ_{ij}^k and t_k. Thus, the memory complexity of the learning algorithm is also $O(1)$.

4. EXPERIMENTS AND RESULTS

The DTMC parameter learning algorithm in Section 3 was validated through the simulation of a wide range of scenarios for the bioinformatics workflow from Figure 1. This section presents a subset of these scenarios that involves learning the probability of a successful invocation of the web service **ebi_embl** from the workflow, based on an initial estimate and on observations obtained through monitoring the service. This service was selected because its execution involves complex bioinformatics database operations that expose its invocation to variations in performance and reliability.

Assuming that the probabilities of all other service invocations were fixed, the probabilistic model checker PRISM was used to analyse the impact of changes in the success probability of service **ebi_embl** on the compliance of the workflow with requirement (2). Figure 4 depicts the result of this analysis, showing that the workflow satisfies the requirement if and only if service **ebi_embl** has a probability of success of at least $pRequired = 0.991$.[4]

Figure 5 presents the experimental results for three scenarios involving dynamic changes to the actual probability of success $pActual$ for the simulated web service over $30,000$s of simulated time. Each scenario corresponds to a different $pActual$ change pattern, and the experimental results were obtained through averaging the results of 100 Matlab simulations of each of the three patterns. The observation timestamps t_k, $1 \leq k \leq N_i$, were selected from a Poisson distribution with a mean inter-arrival time of $1/\lambda = 1s$, and the observations σ_{ij}^k, $1 \leq k \leq N_i$, were taken from a Bernoulli distribution with parameter $pActual$. The smoothing parameter was fixed at $c_i^0 = 50$, and the *a priori* probability was set to 0.991, i.e., the minimum probability for which requirement (2) is satisfied. Simulations were carried out for a large number of values of the ageing coefficient α, though, in the interest of readability, the diagrams in Figure 5 show the results for only two of these values, i.e., $\alpha = 1$ (which corresponds to the base learning algorithm) and $\alpha = 1.001$.

In each scenario from Figure 5, the probability of success for the simulated service starts and ends at a value $pActual = 0.991$ that ensures compliance with requirement (2). However, this probability drops to a value $pActual = 0.99$ that violates the requirement during a time interval whose start and duration are different for each scenario. The time intervals a, a', b and b' marked on each scenario denote time intervals when the estimated probability of a successful service invocation does not sufficiently reflect the actual probability $pActual$. This leads to an erroneous verification result of requirement (2). Thus, a and a' are the time intervals during which the requirement violation is undetected for $\alpha = 1.001$ and $\alpha = 1$, respectively; and b and b' represent time intervals during which the requirement is satisfied but wrongly classified as violated for $\alpha = 1.001$ and $\alpha = 1$, respectively.

A comparison of the lenghts of time intervals a and a', and of b and b' shows the our learning method (which corresponds to $\alpha = 1.001$) provides a more precise estimation of $pActual$ than the base Bayesian learning algorithm. In particular, Scenario 3 shows that our method provides a good estimate even for a short degradation in the probability of success for the simulated service; in contrast, the base method is unable to detect this degradation.

Finally, the results from scenarios 1 and 2 show that the effectiveness of our method does not depend on the timing of the changes in the value of $pActual$—the time intervals a for the two scenarios are of similar length, and so are the time intervals b. In contrast, when the base method is used, the length of the time interval a' is much longer for scenario 1, in which $pActual$ had the same value (i.e., 0.992) for a

[4]The DTMC PRISM model and the Matlab simulation code for the case study are available at http://www1.aston.ac.uk/eas/staff/dr-radu-calinescu/dtmc-model-and-simulation-code.

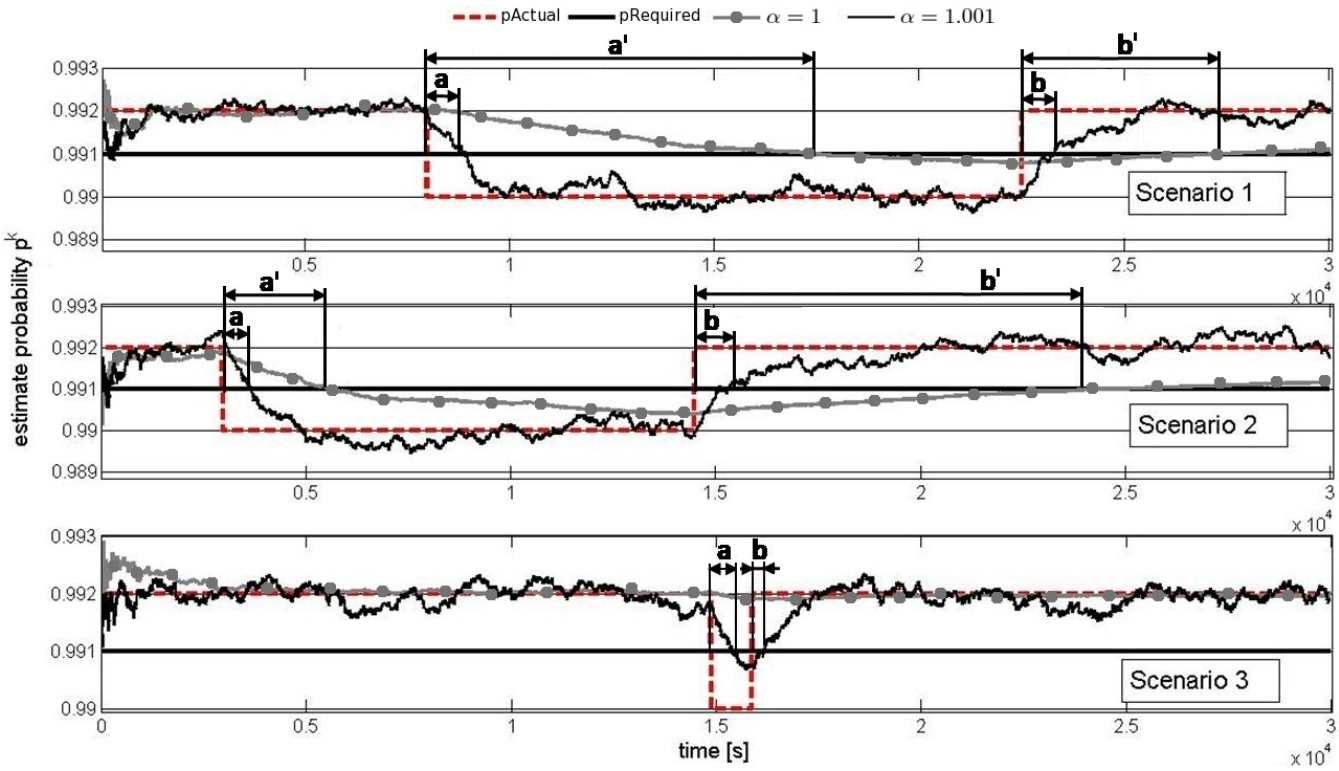

Figure 5: Experimental results contrasting the effectiveness of the observation-ageing and base learning techniques in scenarios when the actual probability of success *pActual* changes dynamically.

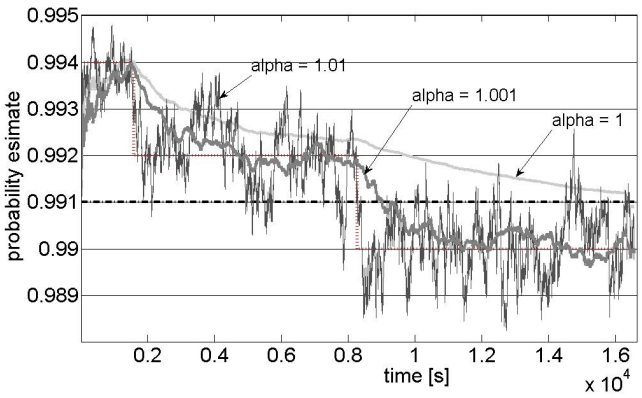

Figure 6: Effect of different choices for the ageing coefficient α

longer period of time; and the length of b' is much longer in scenario 2, in which *pActual* maintained a value of 0.99 for an extended period of time.

Choosing the ageing coefficient α.

The value of the ageing coefficient α determines the age range of the observations that contribute effectively to the most significant $d > 0$ digits of the probability estimate p_{ij}^k in (7). If too few of the observations σ_{ij}^l, $1 \leq l \leq k$, are within this age range, then the probability estimate p_{ij}^k will depend too heavily on the very recent past and will oscillate as shown in Figure 6.

To avoid this undesirable effect, more of the observations σ_{ij}^l must contribute to the d most significant digits of p_{ij}^k, i.e., have a coefficient that is larger than 10^{-d} in (7):

$$\frac{k}{c_i^0 + k} \frac{w_l}{\sum_{l=1}^k w_l} \geq 10^{-d}. \quad (8)$$

Assuming that the mean distance between successive observations is $\mu > 0$, then, for large values of k, $\frac{k}{c_i^0+k} \approx 1$ and $\sum_{l=1}^k w_l = \sum_{l=1}^k \frac{1}{\alpha^{t_k - t_l}} \approx \sum_{l=1}^k \frac{1}{\alpha^{(k-l)\mu}} = 1 + \frac{1}{\alpha^\mu} + \frac{1}{\alpha^{2\mu}} + \cdots + \frac{1}{\alpha^{(k-1)\mu}} = \left(1 - \frac{1}{\alpha^{k\mu}}\right) / \left(1 - \frac{1}{\alpha^\mu}\right) \approx 1/\left(1 - \frac{1}{\alpha^\mu}\right) = \frac{\alpha^\mu}{\alpha^\mu - 1}$, so (8) becomes $w_l / \left(\frac{\alpha^\mu}{\alpha^\mu - 1}\right) \approx \frac{\alpha^\mu - 1}{\alpha^{(k-l)\mu}\alpha^\mu} \geq 10^{-d}$. The observations that satisfy this inequality contribute to the d most significant digits of the probability estimate p_{ij}^k. To ensure that at least $m > 0$ observations are taken into account, the inequality must be satisfied for $k - l = m$ (Figure 7).

5. RELATED WORK

Significant research has focused on monitoring performance and reliability properties of technical systems [11, 12], and on modelling and analysing these properties formally [1]. However, to the best of our knowledge, very few approaches have considered combining techniques from the two research areas [5, 15, 16]. The approach in [5] is described in detail in Section 2.2, and the way in which its ability to handle change is improved by our approach to learning DTMC parameters is presented in Section 3. The approach in [15, 16] uses Kalman filter estimators to update the parameters of queueing-network performance models. Our results complement this approach, as they target DTMC reliability models.

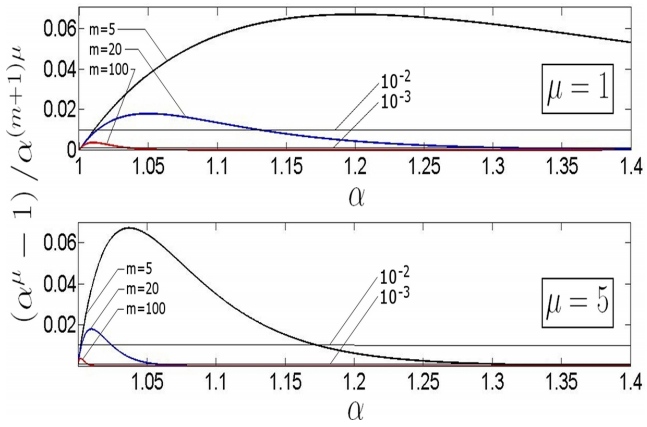

Figure 7: Choosing the ageing coefficient: α values for which $(\alpha^{\mu}-1)/\alpha^{(m+1)\mu} \geq 10^{-d}$ ensure that the m most recent observations contribute to the d most significant digits of the probability estimate

6. CONCLUSION AND FUTURE WORK

The work in progress described in this paper addresses the growing need for techniques capable of learning the parameters of the models used in QoS engineering from run-time observations of the analysed system. The Bayesian learning technique introduced in the paper estimates the parameters of discrete-time Markov chains for systems operating in changing environments, and achieves high accuracy by weighing the importance of individual observations based on their age.

Our future work aims to evaluate the effectiveness of observation ageing in learning the parameters of service-based systems deployed on virtual machines running within cloud data centres. We envisage that in this environment service QoS parameters will vary not only with the service workload, but also with changes within other applications that are using the same physical resources as the analysed service. As a further step in exploiting the new technique, we are considering integrating it into the QoS management and optimisations framework from [2].

7. ACKNOWLEDGEMENTS

This work was partly supported by the UK Engineering and Physical Sciences Research Council grant EP/H042644/1.

8. REFERENCES

[1] S. Balsamo, A. Di Marco, P. Inverardi, and M. Simeoni. Model-based performance prediction in software development: A survey. *IEEE Trans. Softw. Eng.*, 30:295–310, May 2004.

[2] R. Calinescu, L. Grunske, M. Kwiatkowska, R. Mirandola, and G. Tamburrelli. Dynamic QoS management and optimisation in service-based systems. *IEEE Transactions on Software Engineering*, 99(PrePrints), 2010.

[3] R. Calinescu and S. Kikuchi. Formal methods @ runtime. In *Modelling, Development and Verification of Adaptive Computer Systems*, Lecture Notes in Computer Science. Springer, 2011. To appear.

[4] R. Calinescu and M. Kwiatkowska. Using quantitative analysis to implement autonomic IT systems. In *Proceedings of the 31st International Conference on Software Engineering (ICSE'09)*, pages 100–110, 2009.

[5] I. Epifani, C. Ghezzi, R. Mirandola, and G. Tamburrelli. Model evolution by run-time parameter adaptation. In *Proceedings of the 31st International Conference on Software Engineering*, ICSE '09, pages 111–121, Washington, DC, USA, 2009. IEEE Computer Society.

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

[7] A. Hinton, M. Kwiatkowska, G. Norman, and D. Parker. PRISM: A tool for automatic verification of probabilistic systems. In H. Hermanns and J. Palsberg, editors, *Tools and Algorithms for the Construction and Analysis of Systems*, volume 3920 of *Lecture Notes in Computer Science*, pages 441–444. Springer Berlin / Heidelberg, 2006.

[8] D. Hull, K. Wolstencroft, R. Stevens, C. Goble, M. Pocock, P. Li, and T. Oinn. Taverna: a tool for building and running workflows of services. *Nucleic Acids Research*, 34(Web Server issue):729–732, July 2006.

[9] J.-P. Katoen, M. Khattri, and I. S. Zapreev. A Markov reward model checker. In *Quantitative Evaluation of Systems*, pages 243–244, Los Alamitos, 2005. IEEE Computer Society.

[10] M. Kwiatkowska. Quantitative verification: Models, techniques and tools. In *Proc. 6th Joint Meeting of the European Software Engineering Conference and the ACM SIGSOFT Symposium on the Foundations of Software Engineering (ESEC/FSE)*, pages 449–458. ACM Press, September 2007.

[11] R. Pietrantuono, S. Russo, and K. S. Trivedi. Online monitoring of software system reliability. *European Dependable Computing Conference*, 0:209–218, 2010.

[12] F. Raimondi, J. Skene, and W. Emmerich. Efficient online monitoring of web-service SLAs. In *Proceedings of the 16th ACM SIGSOFT International Symposium on Foundations of software engineering*, SIGSOFT '08/FSE-16, pages 170–180, New York, NY, USA, 2008. ACM.

[13] C. C. Strelioff, J. P. Crutchfield, and A. W. Hübler. Inferring Markov chains: Bayesian estimation, model comparison, entropy rate, and out-of-class modeling. *Phys. Rev. E*, 76(1):011106, Jul 2007.

[14] H. L. S. Younes. Ymer: A statistical model checker. In K. Etessami et al., editors, *Computer Aided Verification*, volume 3576 of *LNCS*, pages 429–433. Springer, Berlin, 2005.

[15] T. Zheng, M. Woodside, and M. Litoiu. Performance model estimation and tracking using optimal filters. *IEEE Transactions on Software Engineering*, 34(3):391–406, 2008.

[16] T. Zheng, J. Yang, M. Woodside, M. Litoiu, and G. Iszlai. Tracking time-varying parameters in software systems with extended Kalman filters. In *Proceedings of the 2005 conference of the Centre for Advanced Studies on Collaborative research*, CASCON '05, pages 334–345. IBM Press, 2005.