Lightweight Adaptive Filtering for Efficient Learning and Updating of Probabilistic Models

Antonio Filieri
University of Stuttgart
Stuttgart, Germany

Lars Grunske
University of Stuttgart
Stuttgart, Germany

Alberto Leva
Politecnico di Milano
Milan, Italy

***Abstract*—Adaptive software systems are designed to cope with unpredictable and evolving usage behaviors and environmental conditions. For these systems reasoning mechanisms are needed to drive evolution, which are usually based on models capturing relevant aspects of the running software. The continuous update of these models in evolving environments requires efficient learning procedures, having low overhead and being robust to changes. Most of the available approaches achieve one of these goals at the price of the other. In this paper we propose a lightweight adaptive filter to accurately learn time-varying transition probabilities of discrete time Markov models, which provides robustness to noise and fast adaptation to changes with a very low overhead. A formal stability, unbiasedness and consistency assessment of the learning approach is provided, as well as an experimental comparison with state-of-the-art alternatives.**

I. Introduction

Non-functional properties such as reliability, performance, or energy consumption are a central factor in the design of software systems, moving from the niche of critical systems to everyday software. Probabilistic quantitative properties, which are able to characterize the uncertainty and unpredictability of external phenomena affecting software behavior from the interaction with the users to the contention on accessing physical resources. For this reason, significant research effort has been conducted in recent years about specification and verification of probabilistic quantitative properties [1–7]. These verification approaches commonly build upon convenient formal models able to capture the probabilistic nature of the described phenomena (e.g., Markov models or queuing networks).

However, most of these models are constructed at design time based on initial assumptions about the software and its execution environment. These assumptions might be invalidated by unforeseen changes the software may undergo during its execution [8, 9]. To handle this issue, probabilistic models need to be continually updated during runtime [10–13] to provide a current view on the running systems, supporting also the runtime verification of the desired properties.

In general, designing time efficient and accurate algorithms to keep a probabilistic model continuously updated during runtime is an open problem, deeply investigated by the Software Engineering community [12–15]. An early example is the Kami approach [12] that uses a Bayesian estimator to learn transition probabilities of Discrete-Time Markov Chains (DTMCs). However, the longer a Kami estimator runs the higher the effect of the historical data is on the estimation. Thus KAMI is producing inaccurate results once the probabilities change. The authors of the initial Kami approach have also noticed this and have extended their approach with a change point detection algorithm [16], which resets the estimation once the observed transition probabilities have significantly changed. Adding a change point detection method to Kami significantly increases the robustness towards change, however it comes at the cost of an increased runtime overhead. The Cove approach [17] enhances Kami's Bayesian estimator by adding an aging mechanism to forget old information. Cove results are thus more robust to changes, however the intrinsic noise filtering capability of the original Bayesian estimator is weakened by the aging mechanism, leading to more noisy estimates. Cove has been extended with a procedure to automatically set an optimal aging factor [14]. In the area of performance tracking, Kalman filters are configured and used to estimate performance measures and to keep them updated at runtime [15]. Kalman filters are well known for their ability to reduce input noise and to provide smooth estimates. However, this comes at the price of slower responses to abrupt changes. A good trade-off between these two aspects usually requires a non-trivial tuning of the algorithm's parameters.

Trading off noise-rejection, prompt reaction to changes, and computational overhead remains an open problem. In this paper we propose a novel lightweight adaptive filter to learn and continuously update the transition probabilities of a DTMC that:

- is specifically designed to *improve the trade-off* between smooth estimation and prompt reaction to changes
- is equipped with an *online auto-tuning procedure* to robustly discriminate between actual changes and outliers
- is *provably stable, unbiased, and consistent*, with a formal quantification of its convergence time and its noise filtering strength
- requires a *negligible runtime computational overhead.*

We implemented our approach in Python [18] and formally proved (cp. Section IV) that the algorithm satisfies the desired properties. We further performed a preliminary experimental evaluation (cp. Section VI) with common input data patterns to highlight the strengths and weaknesses. Additionally, we applied the algorithm to learn the operational profile of a large case study [19] to underpin these results (cp. Section VII).

II. Background

This section briefly recalls essential background concepts for our approach. In Section II-A a formal definition of Discrete-Time Markov Chains (DTMCs) is provided. In Section II-B we will introduce basic definitions and assumptions about statistical inference for DTMCs.

A. Discrete Time Markov Models

A Discrete-Time Markov Chain is a state-transition system where the choices among successor states are governed by a probability distribution. Formally, a DTMC is a tuple (S, s_0, P, L, AP) [20], where S is a (finite) set of states, $s_0 \in S$ is the initial state, $P : S \times S \to [0, 1]$ is a stochastic matrix, AP is a set of atomic propositions, and $L : S \to 2^{AP}$ is a labeling function that associates to each state the set of atomic propositions that are true in that state. An element p_{ij} of the Matrix P represents the transition probability from state s_i to state s_j, i.e. the probability of going from state s_i to state s_j in exactly one step.

The probability of moving from s_i to s_j in exactly two steps can be computed as $\sum_{s_x \in S} p_{ix} \cdot p_{xj}$, that is the sum of the probabilities of all the paths originating in s_i, ending in s_j, and having exactly one intermediate state. The previous sum is, by definition, the entry (i, j) of the power matrix P^2. Similarly, the probability of reaching s_j from s_i in exactly k steps is the entry (i, j) of matrix P^k. As a natural generalization, the matrix $P^0 \equiv I$ represents the probability of moving from state s_i to state s_j in zero steps, i.e., 1 if $s_i = s_j$, 0 otherwise.

Since P is a stochastic matrix, the sum of the elements for each of its rows has to be 1. Formally, each row i of P identifies a *categorical distribution* [21]. Furthermore, thanks to the Markov property [22], these categorical distributions are pairwise probabilistically independent, since the choice of the next state only depends on the current one. This property will be exploited in the next section to support the definition of a localized learning approach of the transition matrix P.

B. Statistical Learning for DTMCs

The identification of DTMC models from the observation of a running system is a well-known statistical problem [23, 24], with relevant applications in many disciplines [22], including software engineering [25–32].

In this paper we focus on learning the transition probability matrix P of a DTMC, assuming its structure does not change, i.e., only transition probabilities may be unknown or subject to change [29, 33, 34]. Thanks to the Markov property, this problem can be reduced to the independent learning of n independent categorical distribution, where n is the number of states composing the DTMC. This simplifies both the monitoring and the learning tasks.

Several approaches have been proposed in literature, including maximum likelihood estimators [23] and Bayesian estimators [13, 35]. The latter have recently gained more relevance for online learning thanks to the (usually) faster convergence and the ability to embed expert knowledge in the form of an assumed *prior* next state distribution [12–14, 17]. Despite their ability to estimate the actual transition probabilities, even in presence of noisy observations, for time-invariant processes, most statistical approaches fail to promptly react to changes in the transition probability. This leads to slow convergence time after a change and, consequently, poor accuracy and reliability of the estimates.

III. Learning Through Filters

In this section we will introduce our online learning approach for DTMCs based on filtering. The input to our systems is a sequence of measures representing the average transition frequency from a state s_i to each state s_j over a period of observation. Calling k the k-th period of observation (also referred to as *time step*), the average transition frequencies $p^m_{ij}(k)$ at time step k are defined as $n_{ij}(k)/\sum_x n_{ix}(k)$, where $n_{ij}(k)$ is the number of transitions from s_i to s_j occurred at time step k. Since those counts are obtained by monitoring the system for a limited time, we assume the observed frequencies to include an additive zero-mean noise component, accounting for both the uncertainty of the sampling procedure and possible issues with the monitoring infrastructure (e.g., communication delays). The values of the noise for each time step are assumed to be independent among one another and, approximately, normally distributed, with unknown variance [36, 37].

For the considerations stated in Section II-B, we will instantiate a filter for each state, aiming at learning its next state distribution. A similar approach is followed by most state-of-the-art approaches for learning DTMCs [12–14].

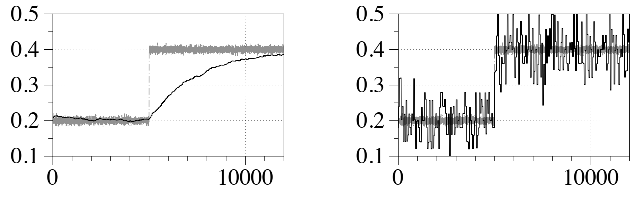
To describe our approach, let us first focus in Section III-A on the estimation of a scalar parameter not subject to any constraint. In Section III-B, we will extend the approach to cope with multiple dependent variables, whose sum has to be equal to a given value. This extension is needed to handle the structural dependencies among transitions of a DTMC originating from the same state. Finally, in Section III-C, we will introduce an online auto-tuning procedure to automatically adapt the change point detection mechanism of the filter to cope with changing and unpredictable operation scenarios.

A. Learning a Scalar Measure

The goal of our learning procedure is to estimate an unknown, time-varying probability vector $p(k)$ from the (noisy) measurements $p^m(k)$. The output of the filter will be an estimate $\widehat{p}(k)$ of $p(k)$. The simplest viable filter for our purpose is a unity-gain, first-order discrete-time filter [38], whose dynamics is described in Equation (1):

$$\widehat{p}(k) = a \cdot \widehat{p}(k-1) + (1-a) \cdot p^m(k-1), \qquad 0 < a < 1 \quad (1)$$

For this filter high values of a (i.e., close to 1) provide good noise filtering and smoothing, which is desirable to estimate a stationary probability from noisy observation. However, the tracking of abrupt (e.g., stepwise) variations of $p(k)$ would be very slow. On the other hand, small values of a (i.e., close to 0) would promptly follow abrupt variations of $p(k)$ but at the price of poor noise filtering. An example of such behavior is shown in Figure 1. Ideally, we would like to have



(a) $a = 0.98$ (b) $a = 0.02$

Fig. 1. Noise filtering versus fast change tracking for the filter of Equation (1).

a close to 1 when the $p(k)$ is stationary, in order to obtain a smooth estimation, while a close to 0 whenever a change in $p(k)$ occurs. Our goal in the remainder of this section is to introduce our strategy for the dynamic adaptation of a, based on the behavior of $p(k)$, as we can observe it through the measurement $p^m(k)$.

The strategy we propose is to control a in the range between two asymptotic bounds a_{lo} and a_{hi}. For a stable and non-oscillatory behavior of the estimation process, we require $0 < a_{lo} < a_{hi} < 1$ (we will come back to *stability* later). We want our adaptation mechanism to drive a toward a_{hi} when the process is stationary, and toward a_{lo} in presence of a change. To distinguish these two situations, we propose to consider the observed probability distribution of the input measurements $p^m(k)$. Indeed, being produced by monitoring the process for a limited amount of time (i.e., a time step), these measurements usually show a certain degree of dispersion around the actual value $p(k)$. Let us assume for now we can quantify such dispersion, and use it to define a threshold $e_{thr} \geq 0$ such that an input measure distant more than e_{thr} from the current estimate $\widehat{p}(k)$ can be "likely" assumed to be the bad smell of a change point[1].

For smoothness reasons, let us define the value of $a(k)$ (the introduction of the adaptation mechanism made it time-dependent) as:

$$a(k) = a_0 + \Delta a f_a(e(k) - e_{thr}) \tag{2}$$

where $a_0 = 0.5(a_{hi} + a_{lo})$, $\Delta a > 0$ tunes the adaptation speed (the larger the faster; default $a_{hi} - a_{lo}$), and $e(k) = |p^m(k) - \widehat{p}(k-1)|$ and $f_a(\cdot)\, \Re \mapsto (-0.5, 0.5)$ is a continuous, differentiable, strictly monotonically decreasing function such that

$$\lim_{x \to -\infty} f_a(x) = 0.5, \quad f_a(0) = 0, \quad \lim_{x \to \infty} f_a(x) = -0.5. \tag{3}$$

To ensure that $a_{lo} < a(k) < a_{hi}\ \forall e_{mag}(k)$ we choose:

$$f_a(x) = -\frac{\arctan(\mu \cdot x)}{\pi} \tag{4}$$

where μ is a design parameter determining the gradient of $f_a(\cdot)$ around the origin, and, in turn, the "speed" of transition between the two asymptotic values ± 0.5. Notice that the definition of $f_a(\cdot)$ in terms of the $arctan(\cdot)$ function satisfies all the requirements stated above. Furthermore, the selection of $arctan(\cdot)$ is typical [39], when arbitrarily steep transitions between two values are required to be obtained through a continuous and continuously differentiable function. Alternative definitions of $f_a(\cdot)$ are possible, but their analysis is beyond the scope of this paper. Higher-order filters can also provide for a finer specification of the transition function, though with an increase computational complexity and, in general, less provable stability result. Our solution aims at achieving the simplest strategy suitable for solving our problem, and with the lowest possible computational overhead.

[1]This intuition is used in several statistical tests about the mean of a population and for the identification of outliers [21, 35].

Combining Equations (1) and (2), under the assumption that we have properly quantified e_{thr}, results in the nonlinear discrete-time dynamic equations of our learning filter:

$$\begin{cases} e(k) &= |p^m(k) - \widehat{p}(k-1)| \\ a(k) &= a_0 + \Delta a \cdot f_a(e(k) - e_{thr}) \\ \widehat{p}(k) &= a(k) \cdot \widehat{p}(k-1) + (1 - a(k)) \cdot p^m(k-1) \end{cases} \tag{5}$$

where in a mathematical sense p^m is the input, and $\widehat{p}$ both the state and the output of the dynamic system [37].

The filter in (5) is the core of our learning approach. In the next section we will extend it to estimate categorical distributions instead of scalar values, while in Section III-C we will formally describe the online adaptation mechanism that allows for automatically adjusting the value of e_{thr}, and, consequently, of $a(k)$.

B. Learning Categorical Distributions

The filter defined in the previous section can be used to estimate each single probability p_{ij} individually. However, the obtained estimates for each row of P would most likely not constitute correct categorical distributions (sum is not 1).

In order to ensure the estimation of correct categorical distribution of each state s_i, we will first estimate each probability p_{ij} individually and then applying a convenient "correction" procedure. This procedure minimizes the Euclidean norm of the distance between the vector $\widehat{\mathbf{p}}$ of the uncorrected estimates and the vector of the corrected one, subject to a unity sum constraint for the latter (and only positive probabilities by construction). Formally, let $\widehat{\mathbf{p}}$ and $\widehat{\mathbf{p^c}}$ be $\widehat{\mathbf{p}}(k) = [\widehat{p}_1(k) \ldots \widehat{p}_n(k)]'$ and $\widehat{\mathbf{p^c}}(k) = [\widehat{p^c}_1(k) \ldots \widehat{p^c}_n(k)]'$ Our correction procedure requires to solve the following optimization problem:

$$\begin{cases} \min\limits_{\widehat{\mathbf{p^c}}(k)} \left(\widehat{\mathbf{p^c}}(k) - \widehat{\mathbf{p}}(k)\right)' \left(\widehat{\mathbf{p^c}}(k) - \widehat{\mathbf{p}}(k)\right) \\ \text{subject to } \sum_{i=1}^{n} \widehat{p^c}_i(k) = 1 \end{cases} \tag{6}$$

Using the Lagrange multipliers method to solve this optimization problem [40], the Lagrangian of the problem is

$$\mathcal{L}(k) = \sum_{i=1}^{n} \left(\widehat{p^c}_i(k) - \widehat{p}_i(k)\right)^2 + \lambda \left(\sum_{i=1}^{n} \widehat{p^c}_i(k) - 1\right) \tag{7}$$

and solving the corresponding Karush, Kuhn, and Tucker (KKT) equations [41] leads to the (affine) correction formula

$$\widehat{\mathbf{p^c}}(k) = F_c \cdot \widehat{\mathbf{p}}(k) + H_c \tag{8}$$

where
$$F_c = I_n - \frac{\mathbf{1}_{n \times n}}{n}, \quad H_c = \frac{\mathbf{1}_{n \times 1}}{n}, \tag{9}$$

and the symbol $\mathbf{1}_{p \times c}$ denoting a $p \times c$ matrix with unity elements, and I_n being the identity matrix of order n.

In the remainder of this paper, we will always refer to the corrected estimator for the transition probabilities of a DTMC (i.e. $\widehat{p}_{ij} \triangleq \widehat{p^c}_{ij}$), unless otherwise specified.

C. Online Filter Adaptation

The core element of the online adaptation mechanism is the dynamic correction of the parameter e_{thr}. This parameter has to capture the dispersion of an input measurement $p^m(k)$ around the actual value it is measuring $p(k)$.

An effective and sequentially computable index of the dispersion of a probability distribution is its variance. An efficient and numerically stable algorithm for the sequential (i.e., sample by sample) estimation of the variance has been proposed by Knuth [42] and reported in Algorithm 1.

```
n=0; mean=0; m2 = 0;
for x in data do
    n = n+1;
    delta = x - mean;
    mean = mean + delta/n;
    m2 = m2 + delta*(x - mean);
end
variance = m2/(n-1);
```

Algorithm 1: Knuth's algorithm for seq. variance estimation.

We adapt Knuth's algorithm by executing the body of the loop for each incoming measurement $p^m(k)$, and updating the input variance (σ^2). This way we can efficiently keep our dispersion index updated after every new measure is gathered.

In order to compute e_{thr}, we assume the input measurements to have Gaussian distribution centered around the actual measure and with variance σ^2 [36, 37]. We then want to decide for each incoming measurement if it can be reasonably explained under this assumption or not. To do so, we operate similarly to a statistical hypothesis test on the mean of the input distribution. Our hypothesis is that the actual mean of the input distribution is $\widehat{p}(k-1)$, i.e., the latest estimate from our filter, and try to decide whether the measurement $p^m(k)$ is far enough from $\widehat{p}(k-1)$ to contradict our hypothesis. Since the variance of the input distribution has been estimated, we should refer in this case to a t-test [21]. However, assuming enough measurements have been retrieved (as a rule of thumb, at least 30), we can safely use a z-test [21]. After deciding a confidence value α, we can decide whether the last measurement retrieved should be considered "explainable" with the current estimation or the outcome of a change in the process (it may also be the case of an outlier, but we will discuss how to handle this situation later on). In particular, we assume a change point occurred if

$$\frac{|p^m(k) - \widehat{p}(k-1)|}{\sigma} \geq z_{1-\frac{\alpha}{2}} \tag{10}$$

where $\sigma = \sqrt{\sigma^2}$ is the standard deviation of the input distribution. Equation (10) can be straightforwardly used to adjust the value of e_{thr} after a new sample has been gathered:

$$e_{thr}(k) = \sigma(k) \cdot z_{1-\frac{\alpha}{2}(k)} \tag{11}$$

The decision about α constitutes a trade-off between a quick response to smooth changes in the process and robustness to measurement outliers. Particularly relevant for this decision is the (average) number of transitions observed in each time step: a small number will likely lead to noisy measurements and increases the likelihood of outliers; in such case a very small value of α is recommended (see also Section VI).

D. On the Filter Configuration

The filter introduced in Section III-A has four configuration parameters: μ, a_{lo}, a_{hi}, and α. While the role of α has been already discussed in the previous section, we will briefly (and informally) discuss here how the values of the others affect the performance of the filter. Some addition formal considerations will be provided in Section IV.

Intuitively, there are two extreme operating conditions for a filter: the probability to be estimated is stationary, or it is undergoing an abrupt change. In the first case, we aim at obtaining a "clean" and smooth estimate. This means we would like a to be close to 1. If the input measurements come from a stationary distribution they will most likely be recognized as "compatible" with the current estimate (see Section III-C). This will drive the value of a (asymptotically) to a_{hi}. Thus, setting a_{hi} close to 1 will provide smooth estimates of a stationary probability. Analogous considerations can be stated for the case of abrupt changes, when the value of a will be moved toward its lower bound a_{lo}.

However, while it is quite safe to set a_{hi} very close to 1, bringing a_{lo} close to 0 reduces the filter's robustness to outliers. Indeed, α allows to set a threshold to decide if an input measurement is not compatible with the current estimate. However, there is always the chance that an extreme measurement is over the threshold and brings a towards a_{lo}. Thus, a more conservative choice of a_{lo} is recommended if the input measures are known to have a high variance. A rule of thumb to decide such value is that $1-a_{lo}$ can be considered as the maximum degree of trust on a new input measurement. However, setting $a_{lo} > .3$ may produce a sensible slow down in change point tracking, and should be done carefully.

The value of μ determines how "fast" to move from a_{lo} to a_{hi} and vice versa. High values of μ are recommended (≥ 1000) to obtain prompt switches when a change occurs.

Finally, a hidden parameter to consider is the duration of a time step. Indeed, longer time steps may allow to collect more events within the period, which in turn improves the quality of the input measurements (which are the transition frequencies computed over each time step). However, since the filter updates its estimates every time step, extending their duration may slow down the filter reaction to changes.

IV. Formal Assessment

The proposed adaptive filter requires a constant number of operations each time step, as easily observable from Equation (5) and the filter adaptation procedure in Algorithm 1. Furthermore, the operations are mostly elementary floating point operations, and only a minimal amount of memory is required, making our approach suitable to be executed even on low-power devices (e.g., embedded systems). An empirical estimation of the actual execution time on a general purpose computer will be shown in Section VI.

However, the low computational overhead was only one of our goals. In this section we will formally prove several other critical properties of our approach. The proofs will provide a theoretical guarantee about its applicability and the quality of its results. An empirical assessment based on selected case studies and the comparison with state of the art alternatives will also be provided in Section VI.

In Section IV-A, we will prove the stability of our system, i.e., its ability to converge to a steady state equilibrium for every constant value of the input. In Section IV-B we will prove that our filter is an unbiased and consistent estimator of the transition probabilities we aim at learning. In Section IV-C, we will precisely quantify the ability of our filter to reduce the noise in the input measurements (i.e., their variance, under the weak assumption of Gaussian distribution) as a function of the filter's configuration parameters. Finally, in Section IV-D we will assess the settling time required for our estimates to converge after a change as occurred.

A. Stability

A dynamic system is asymptotically stable if there exists an equilibrium point to which the system tends; i.e., for any given constant input, the output converges to a specific value (within a convenient accuracy) regardless of the initial state [37]. As time tends to infinity, the distance to the equilibrium point has to tend to zero.

Our filter is formally defined by the dynamic system of Equation (5). In the following we will prove that the filter always converges to an equilibrium value, regardless its initial conditions and for all the (valid) values of the input measures.

Let (5) be subject to the constant input $p^m(k) = \overline{p}^m$. The corresponding equilibrium value $\overline{\widehat{p}}$ can be obtained by computing the fixed point solutions of the dynamic system [37]:

$$\overline{\widehat{p}} = \overline{a} \cdot \overline{\widehat{p}} + (1 - \overline{a}) \cdot \overline{p}^m \tag{12}$$

where $\overline{a}$ is the equilibrium value for $a(k)$. This yields the unique solution

$$\overline{\widehat{p}} = \overline{p}^m \tag{13}$$

since the case $\overline{a} = 1$ is excluded by construction (Equation (1)). Also, since at the computed equilibrium $\overline{e} = 0$ (because of Equation (13)), we can compute the value of $\overline{a}$ as follows:

$$\overline{a} = a_0 + \Delta a f_a(-e_{thr}) \tag{14}$$

Hence, for any $\overline{p}^m$ there is one equilibrium with $\overline{\widehat{p}} = \overline{p}^m$, and $\overline{a}$ given by (14). To prove the stability of all equilibria (i.e., the stability of the filter in every operating condition), we can analyze the response of the system to a deviation from such equilibrium [37]. We define the system output variation with respect to the equilibrium value as

$$\widehat{p}^\delta(k) \triangleq \widehat{p}(k) - \overline{\widehat{p}} \tag{15}$$

Combining Equation (15) with the last equation in (5):

$$\widehat{p}^\delta(k) = a(k) \cdot \widehat{p}^\delta(k-1) + (1 - a(k)) \cdot (p^m(k-1) - \overline{\widehat{p}}) \tag{16}$$

Defining now the input variation with respect to the equilibrium value as

$$p^{m^\delta}(k) \triangleq p^m(k) - \overline{p}^m \tag{17}$$

we have

$$\widehat{p}^\delta(k) = a(k) \cdot \widehat{p}^\delta(k-1) + (1 - a(k)) \cdot p^{m^\delta}(k-1). \tag{18}$$

Furthermore, exploiting again the equilibrium,

$$e(k) = |p^{m^\delta}(k) + \overline{p}_m - (\widehat{p}^\delta(k-1) + \overline{\widehat{p}})| = |p^{m^\delta}(k) - \widehat{p}^\delta(k-1)| \tag{19}$$

and the estimator can be rewritten in input/output variational form [37] as

$$\begin{cases} a(k) & = a_0 + \Delta a \cdot f_a(|p^{m^\delta}(k) - \widehat{p}^\delta(k-1)| - e_{thr}) \\ \widehat{p}^\delta(k) & = a(k) \cdot \widehat{p}^\delta(k-1) + (1 - a(k)) \cdot p^{m^\delta}(k-1) \end{cases} \tag{20}$$

For the purpose of the equilibrium stability, it is required to study the motion of (20) under the constant input corresponding to the equilibrium, i.e., with $p^{m^\delta}(k) = 0$. This means analyzing the system

$$\begin{cases} a(k) & = a_0 + \Delta a \cdot f_a(|\widehat{p}^\delta(k-1)| - e_{thr}) \\ \widehat{p}^\delta(k) & = a(k) \cdot \widehat{p}^\delta(k-1) \end{cases} \tag{21}$$

Given the bounds on $a(k)$ inherent to $f_a(\cdot)$, $\widehat{r}^\delta(k)$ in (21) eventually converges to zero irrespectively of its initial value. Thus, all the equilibria of the dynamic system in Equation (5) are globally asymptotically stable.

The stability proof guarantees the applicability of our learning approach under any possible (valid) input measurements, in particular to learn the transition probability of any DTMC.

B. Unbiasedness and Consistency

Denoting with p^o the true (constant) value of the measure to estimate, from Equation (13) and the stability proof provided in the previous section it follows that

$$\lim_{k \to \infty} \widehat{p}(k) = p^o \qquad \forall\, \widehat{p}(0) \tag{22}$$

Hence, we can state

$$\lim_{k \to \infty} E[\widehat{p}(k)] = p^o \text{ and } \lim_{k \to \infty} E[(\widehat{p}(k) - p^o)^2] = 0 \tag{23}$$

Thus the estimator is (asymptotically) unbiased and consistent.

C. Variance of the Estimate

Consider the case where the input measurements provided to our filter are a realization of a white Gaussian noise input, i.e., a Gaussian distribution with mean 0 and variance σ_w^2 (i.e. the simplest distribution allowing to arbitrarily set its variance). For the sake of simplicity, assume a to be a constant value. The ratio of the output variance over the input variance σ_w^2 is the $\mathcal{H}_2$ norm of the estimator [37], i.e., of the linear, time-invariant dynamic system with input $p^m(k)$ and output $\widehat{p}(k)$.

The Z-Transform of the filter's transfer function [37] is:

$$G(z) = \frac{1 - a}{z - a} \tag{24}$$

Its $\mathcal{H}_2$ norm can be expressed as

$$||G||_2 = \sqrt{tr\left(\sum_{k=0}^{\infty} g(k) \cdot g^T(k)\right)} \tag{25}$$

where $g(k)$ is the system's impulse response, i.e., the inverse $\mathbb{Z}$ transform of its transfer function [37]. In our case, then

$$g(k) = \mathbb{Z}^{-1}\left[\frac{1 - a}{z - a}\right] = (1 - a) \cdot a^{-k} \tag{26}$$

which leads to

$$||G||_2 = \sqrt{\sum_{k=0}^{\infty} (1 - a)^2 \cdot a^{-2k}} = \sqrt{\frac{(1 - a)^2}{1 - a^2}} \tag{27}$$

Note that $||G||_2$ tends to 1^- and 0^+ when a tends to 0^+ and 1^-, respectively. This means that the output variance is never greater than that of the input, and is reduced by higher values of a. This is in line with the intuitive considerations on the impact of large and small values of a provided in Section III-A.

D. Convergence Time

For an intuitive analysis, assume $p^m(k)$ undergoes a step from zero to one, and that the convergence time k_c is taken as the number of steps required to drive the estimation error magnitude down to the same threshold e_{thr} used for switching from the "fast tracking" to the "sharp filtering" modes of the system, by moving a towards a_{lo} and a_{hi}, respectively. This immediately leads to determine k_c as the minimum value of k, s.t. $a_{lo}^k < e_{thr}$,

$$k_c = \left\lceil \frac{\log\, e_{thr}}{\log\, a_{lo}} \right\rceil \tag{28}$$

V. Related Work

Markov model learning and inferring of transition probabilities of a DTMC has been widely used in different domains [23, 24]. Two additional requirements for Software Engineering applications are the possibility of embedding experts or domain knowledge and the ability of performing the estimation online, by continuously improving on the *prior* knowledge initially assumed.

One of the first approaches providing these features is Kami [12, 13]. This approach implements an established Bayesian estimator to learn the transition probabilities of a DTMC online. It requires a low computational overhead and provides high accuracy and noise filtering for the estimation of stationary processes. However, it provides slow responses in presence of changes [13]. The same authors [16] faced the problem of change point detection, again following a Bayesian approach. The resulting technique is designed to operate offline on recorded execution traces. It is quite accurate in identifying change points, however it involves the use of a Gibbs sampling techniques to compute the posterior change point distribution [43]. Such randomized method requires a large number of operations for each change point probe. This requires a relatively high computational power, which might be too expensive to be deployed on many embedded system. The execution time is orders of magnitudes higher than the original approach.

In [15], the authors propose an approach for the continuous tracking of time-varying parameters of performance models. The approach is based on (Extended) Kalman filters [37] and is able to estimate also correlated parameters, to take into account nonlinear constraints among their values, and to embed a prior knowledge about the distribution to estimate. However, as reported by the authors, Kalman filters provide their optimal performance when the model describing the temporal evolution and the dependencies among parameters is linear. Despite having been proposed in the domain of performance, the approach of [15] can be easily adapted to also learn the (time-varying) transition probabilities of a DTMC. The simplest way is to use a Kalman filter to estimate each single transition probability and then to apply the correction strategy we introduced in Section III-B for the transitions originating from the same state. The configuration of the Kalman filters requires specifying two parameters: the measurement error covariance R and the disturbance error covariance Q [15]. If we estimate each single transition probability independently, the two matrices reduce to two scalar values r and q, representing the variance of the measurement error and the variance of the disturbance error. Informally, a high value of r means a poor information from measures, while a high value of q means high drift expected for the parameters' estimates. By tuning these two parameters it is possible to define a tradeoff between noise filtering (thus smoother estimates) and quick reaction to changes in the estimated probabilities.

Another approach based on Bayesian estimation that aims at overcoming the limitations of Kami in presence of changing transition probabilities is the *Cove* approach [10, 14, 17]. The basic intuition is to scale of each input measurement with an aging factor that gives more relevance to recent observations [17]. In presence of a change, this input aging allows to quickly discard old information and to give more relevance to the new ones. The configuration of this approach requires the specification of a prior knowledge for the distribution to estimate, the confidence $c_0 > 0$ on such an initial prior, and the value of a parameter α_c which determines the aging factors: a input measurement which has been observed t time steps ago it will be discounted by a factor in the order of α_c^{-t}. Cove has been extended with a procedure to automatically adjust the values of c_0 and α_c [14]. In the special case for the aging factor ($\alpha_c = 1$) Cove reduces to Kami.

VI. Experimental Evaluation

In this section we report on the experimental evaluation of our lightweight adaptive filtering approach to learn transition probabilities of a DTMC. We will benchmark it against related approaches with respect to (i) the estimation accuracy and (ii) the time required for the estimations. Following from the discussion in Section V, we selected for comparison the two algorithms by Calinescu et al. [14] and Zheng et al. [15]. They will be referred to as *Cove* and *Kalman* respectively.

The three approaches will be compared in this section on six different change patterns. On the first part of the comparison we will consider a selected case of each pattern for which visualizing the behavior of the estimates and assessing their accuracy. The scope of the comparison will be then extended to a set of $7,000$ execution traces composed by both randomized realizations of each pattern and combination thereof. Finally, we will report on the computation time required by the three approaches and discuss possible threats to validity.

Accuracy metrics. As accuracy metrics the Mean Average Relative Error (MARE) [44] (similar to [15, 45]) will be used:

$$\text{MARE} \;=\; \frac{1}{n}\sum_{i=1}^{n}\left|\frac{\hat{p}(i) - p(i)}{p(i)}\right| . \tag{29}$$

where $\hat{p}(i)$ represents the estimate at time i, $p(i)$ the actual value to estimate, and n is the number of points.

Experimental settings. We implemented all the approaches in Python (v2.7) and executed the experiments on a quad-core Intel(R) Xeon(R) CPU E31220 @3.10GHz with 32Gb of memory and running an Ubuntu Server 12.04.4 64bit. The memory consumption of the three approaches was negligible. Our implementation is available at [18].

The three algorithms are compared on their performance on estimating the next state distribution of a state of a DTMC (i.e., a row of its transition matrix). The input traces are composed by sequences of $30,000$ events. Each event is an integer number identifying the destination state of the taken transition. This destination state is randomly selected according to known (time-varying) transition probabilities. For the sake of readability of the experiments, the number of reachable states has been set to 3. This choice does not affect the assessment of the accuracy of the estimates for none of the approaches, since the accuracy of the estimates is not directly affected by the number of transitions. The comparison of the execution times of the three approaches is also not effected by this choice, since such time grows linearly with the number of reachable states for all the approaches. Thus the comparison is straightforwardly generalizable to larger problems.

Explorative Study. Figures 3a to 3f report how the three approaches perform with respect to six change patterns. Before going into details of these figure, we briefly discuss the configuration settings for each of the three approaches.

Tools configuration. Concerning *Cove*, we tried both the static parametrization proposed [17] and the adaptive one introduced in [14]. The latter provides for an online tuning to trade accuracy for change reaction time. However, since the inter-arrival time of the events in these settings is constant, the online tuning method does not provide significant improvement on the optimal static configuration proposed in [17]. For this reason we report here the result of the superior static parameter settings with $c_0 = 1$ and $\alpha_c = 1.01$.

For *Kalman*, we set the parameter q to the square of the maximum change to be captured in a single time step, as recommended in [15], which in our case yields $q = 0.2^2$ (notably, Kalman filters are in general robust to bad initial values of q by continuously tuning their behavior on incoming data). Concerning the value of r just general recommendations have been proposed in [15]. Coherently with them, we set $r = q^{-1}$. This is also common practice in the area of control [38]. The *Kalman* approach [15] has been developed for tracking performance measures and is not directly applicable to the constraints of categorical probability distributions for DTMCs. Consequently, we used it to estimate each single transition probability and then applied the same correction procedure we introduced for *LAF* in Section III-B.

Concerning *LAF*, we configured it to have prompt reactions to changes ($\mu = 10000$), to consider as outlier a measure having maximum probability to occur $\alpha = 10^{-4}$, and to provide relatively good filtering capabilities, both while the estimated probability is stationary ($a_{hi} = 0.9$) and during its changes ($a_{lo} = 0.3$). Notice that higher values of a_{hi} would improve the noise reduction capabilities of *LAF*, but slow down its convergence after a change. On the other hand, lower values of a_{lo} would make the reaction to changes more drastic, which has the counterpart of increasing the risk of erroneously following possible outliers.

Finally, while *Cove* can update its estimates after each new event, *Kalman* and *LAF* update theirs every time step. For this reason we defined a time step to occur every 75 events. This value is relatively small. However, a large time step may slow down the reaction to changes (see Section III-D).

Experimental results. We executed and compared the three approaches over six input data patterns that are commonly used to evaluated the related approaches [14, 15]: *Noisy, Step, Ramp, Square wave, Triangle wave* and *Outlier*. By covering these input data patterns we are stressing different aspects of the learning problem. The accuracy results (MARE) for a single run of all the input patterns are reported in Table I. For readability, we report in Figures 3a to 3f only the estimates of the probability of moving toward one target state. For all the plots in Figures 3a to 3f, a dashed grey line represents the actual transition probability to estimate, while the continuous black line represents its estimate. In the following we discuss each of the six transition patterns and the performance of the three approaches.

TABLE I
SIX-PATTERNS BENCHMARK.

	LAF	Kalman	Cove
Noisy	4.41%	4.54%	11.70%
Step	4.19%	8.51%	8.65%
Ramp	4.28%	7.04%	8.27%
Square	6.54%	21.47%	8.50%
Triangle	7.79%	12.84%	8.16%
Outlier	3.68%	3.78%	8.56%

Noisy. For this case (Figure 3a), we do not sample from the actual stationary transition probabilities but we add them a white noise with standard deviation .01. For *LAF* and *Kalman* this white noise adds to the unavoidable measurement noise. After an initial transitory, both *LAF* and *Kalman* roughly converge to the actual mean value of the estimated probability (0.2) and provide similar values for the MARE index. In this situation, *LAF* has not perceived any significant change in the measures and is thus operation as a low pass filter with pole in a_{hi}. Hence, increasing a_{hi} would lead to a slower initial convergence, but smoother estimate, as it is for *Kalman*, whose optimality properties in this scenario are well studied [38] (the worst accuracy of *Kalman* is mostly due to the initial slow convergence). *Cove* converges almost immediately, but with a poor filtering of the input noise.

Step. In the step change pattern the estimated probability suddenly changes from 0.2 to 0.4. *Kalman* provides the smoothest estimates, though at the price of a slower reaction to the change. On the other hand, *LAF* promptly reacts to the change. Notice on Figure 3b the exponential convergence toward the new estimate, as expected from the stability proof and the settling time assessment of Section IV. The settling time can be improved by reducing the value of a_{lo}. However, too small values of it may lead to overshooting due to an overreaction. While *Cove* reacts immediately to the change, its estimates keep being noisy. Notably, in this case *LAF* is about two times more accurate than the others.

Ramp. The estimate transition probability moves here linearly from .2 to .4 in $10,000$ time points. This situation is particularly stressful for the change-point detection mechanism of *LAF*, whose e_{thr} gets continuously updated until the variance estimator converges to the new steady value. The accumulation of the errors of the internal variance estimator and the main *LAF* filter might lead to false positives during or right after the ramp (as in Figure 3c around time $26,000$). In this cases, a not too small value of a_{lo} (as a rule of thumb between .2 and .35) may reduce the deviation after the false detection and allow for a faster recovery, as evident from the figure. As expected from [15, 38, 45] *Kalman* can cope reasonably well with smooth changes, however it is slower than *LAF*, which leverages change reaction to perform step-shaped cuts of the estimate error (see Figure 3c around time $15,000$ and $19,000$). *Cove* reacted immediately to the change and has been able to follow the ramp, though with the usual noise. Also in this case *LAF* is about two times more accurate than the others.

Square wave. The square wave amplifies the issues relate to the step change, by allowing for a shorter learning time before each change. While *Kalman* suffers a slow convergence rate, *LAF* and *Cove* follow the changes, again with *Cove* producing a more noisy output. Under this scenario, the accuracy of *LAF* is about three times higher than *Kalman*, while producing a smoother estimate than *Cove*.

Triangle wave. In this case, smooth changes between two probability values alternates periodically. With respect to the case of ramp, the slopes are steeper, requiring a faster convergence to the estimators. *Cove* can follow the changes as quickly as for the ramp case. *Kalman* provides smooth estimates, but its convergence time is too long and fails to follow the repeated changes. On the other hand, *LAF* copes with the continuous changes by combining a shorter convergence rate of the adaptive low-pass filter with occasional step-shaped error cuts triggered by the change point detection mechanism (as already observed for the ramp). However, as for the case of the ramp, the continuously changing distribution may lead to the accumulation of internal estimation errors that increase the chance of false positives (see time $20,000$ of Figure 3e), whose effects are recovered by $a_{lo} = .3$.

Outlier. In the last case we artificially introduced an outlier with the duration of 25 events. *LAF* and *Kalman* show a negligible reaction to the outlier. This is both due to the filtering actions of the two and to the fact that, operating on a time window, the impact of the outlier is already reduced by the preliminary computation the window's transition frequencies. Despite the triggering of a false change detection, $a_{lo} = .3$ keeps the filter robust to outliers, making it achieve an accuracy slightly higher then *Kalman*, whose effective in filtering outliers is known [38]. Notice that, as for the case of *Noisy*, the main loss of accuracy of *Kalman* is due to the initial convergence. The very fast reaction to change of *Cove* made it quickly follow the outlier, though recovering to the correct estimate right after.

General comments. As final remarks, we noticed that *Cove* provides a very fast reaction to changes, which make their presence almost irrelevant as for the impact on the MARE. This comes however at the price of having noisy estimates. To obtain a similar behavior with *LAF*, both a_{lo} and a_{hi} have to be set to very low values: this way *LAF* will approximate a low pass filter with a very small pole, which, looking at the equations, would behave similarly to *Cove*. A well-known problem of statistical estimation for probability values is the difficulty of catching rare events, i.e. with probability close to 0 (or close to 1, since this implies another transition probability has to be very small). This issue is present for the three approaches. The stability prove of the two filters *LAF* and *Kalman* guarantees that they will eventually converge to the estimated probability, however, for such extreme cases the convergence time might be longer.

Randomized pattern instances. To further investigate the accuracy of our filter, we generated for each pattern a set of $1,000$ random instances and analyzed the performance of the three approaches on this broader set of problems. Concerning the generation of the random instances: *Noisy* and *Outlier* require to generate a baseline stationary distribution and, respectively, the standard deviation of the noise (sampled between .001 and .1) and the duration of the outlier (we take as amplitude half of the maximum gap allowed by the baseline distribution); all the other patterns require to define two distributions and, for the square and triangle wave patterns, the period of the wave ($30,000/n$ with $n \in [2, 15]$). The MAREs of the three approaches are reported in the first six boxplots in Figure 3g. Notably, the results for all the patterns resemble the accuracies reported in Table I for the exploratory study, thus the behavior of the three approaches in each of the six change patterns does not depend, on average, on the characteristics of the specific instance of such pattern.

Finally, the last box of Figure 3g (*VarMix*) shows the accuracy of the three approaches on long traces (from $50,000$ to $500,000$ events) obtained by sequentially combining multiple random instances of the six patterns. The duration and of each instance and their oerder are randomized as well. The results of *VarMix* confirm the earlier results of the single patterns. In particular, *Kalman* suffers from the presence of fast changes, while the MARE of *Cove* is not significantly affected by these changes, but by the high noise of its estimates.

Runtime overhead. On average over $6,000$ runs, *LAF*, *Kalman*, and *Cove* required 50, 80, and 126 ms to process $30,000$ events. Consequently, *LAF* reduces the runtime overhead of *Kalman*, and *Cove*. Notice that *LAF* and *Kalman* update their estimates every time window, while *Cove* updates every new measure.

Threats to validity. A threat to external validity is the use of predefined input data patterns for the comparison of the approaches and the ability to generalize these results to common traces of realistic software systems. We have selected these inputs inline with common theory, common practice in control theory to stress the response of dynamic systems (Step, Ramp), and of filters in particular (Noisy, Periodic, Outlier), and the related approaches [14, 15, 37] and could observe similar patterns also in QoS data sets of web services [46] and web systems (cp. next Section VII). Consequently, we argue that a good performance for these basic input data types will also results in a good general performance.

The threats to internal validity include obviously the selection of the parameters c_0, α_c, q and r for the related approaches and the number of events per time step. We have specifically selected these parameters with defaults that are also defined in the original papers. Furthermore, parameter sweeps other the range of these parameters confirmed that they where good choices for our experiment. Another threat to internal validity is the implementation of the related approaches and measurement environment. As can be seen from the experimental setup we tried to avoid systematic measurement errors and for the implementation we did follow the instructions for the algorithms provided in the original literature.

VII. Application to a Realistic Problem

A common application of DTMC learning is to learn users behavior for a software system (e.g., [47]). This problem can be refined to the scenario of estimating the probability of a user browsing from one webpage to another capturing online log events. To evaluate suitability of *LAF* in this scenario we took the open logs of the World Cup 98 website [19].

The logs spread over a period of about 3 months. The website is composed of a total of over 32000 pages. We mapped every webpage to a unique integer identifier. Each line of the log includes a unique client id. To identify a client session we set a timeout of 30 minutes after the last occurrence of the client id to consider the corresponding user disconnected. A session is thus described by the sequence of pairs [time,pageId] visited by a client. During a session, the client is expected to move from one page to another following navigation links. For demonstration purpose, we show in Figure 2 the online estimation of the transition probability from the page "/english/teams/teambio160.htm" to the page "/english/competition/statistics.htm".

The monitored webpage has been active for about 33 days for a total of 2835186 recorded transitions. We reduced the granularity of the observations by applying a sliding window of 500 seconds. Excluding the windows where no events occurred, 14852 windows have been processed.

The gray line represents the transition frequency observed during the corresponding window reported on the x-axis. The black thicker line represents the value of the *LAF* estimate.

Since we do not know the real value to be estimated, it is hard to evaluate the ability of *LAF* to capture the average transition probability other than by visual inspection (indeed, a proper computation of MARE would require to know the

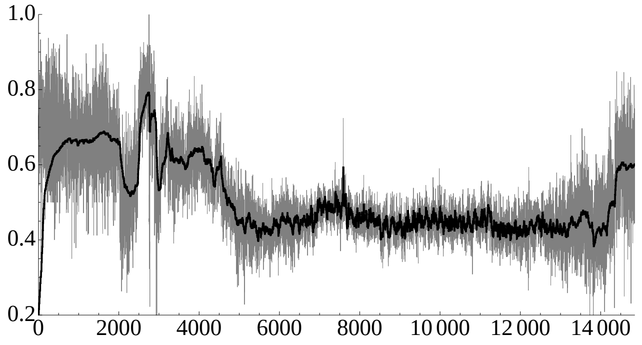


Fig. 2. Estimate of a transition probability from the Worldcup98 case study (x-axis in time windows).

actual transition probability of whom the observed transition frequencies are realizations). However, it is easy to recognize in Figure 2 the occurrence of several patterns we analyzed previously in this section for which a deep quantitative investigation has been provided. The execution time (over 5 runs) to estimate the transition probabilities from the observed state (5 destination states) have been 1703, 2177, and 15953 ms for *LAF*, *Kalman*, and *Cove* respectively. Since *Cove* operates per transition, its execution time is higher than *LAF* and *Kalman*, which instead updates their estimates every 75 transitions and performed inline with the results on the benchmark patterns. Overall, with the new algorithm *LAF* we have been able to process the data for this realistic case and we could confirm the results of the experimental evaluation.

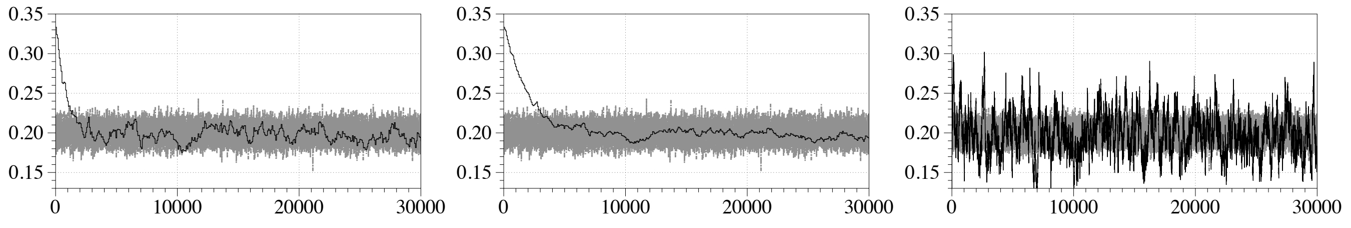
VIII. Conclusions

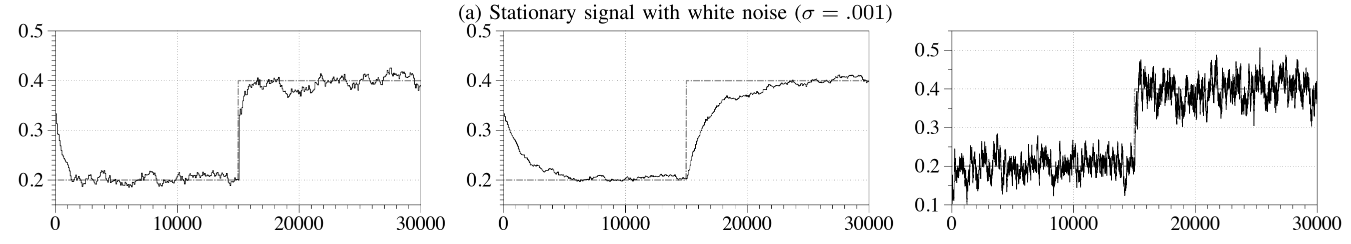
In this work we presented a lightweight adaptive filter for online learning of the transition matrix of a DTMC. We proved it is stable and provides an unbiased and consistent estimate of the transition probabilities. We also quantified its ability to reduce the variance of noisy input measurements and its convergence time after a change has occurred. The filter introduces a minimal computational overhead, being able to process $30,000$ events in about 0.05 seconds on a general purpose computer. Its memory demand does not depend on the number of events to be processed, and it is fairly negligible. The experimental results show an high accuracy of the obtained estimates.

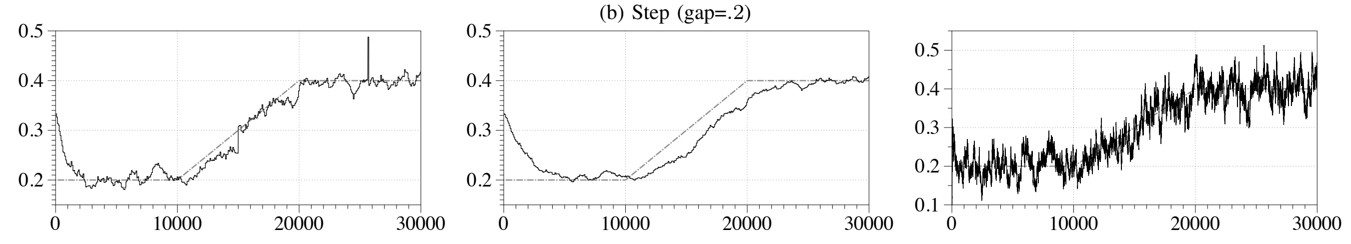
We plan to extend this work along several directions. First, we will expand the developed algorithm to other probabilistic quality evaluation models, including queueing networks for performance analysis, and integrate it into a general framework for continual verification [10]. Second, we plan to increase the order of the filter to further improve its ability of trading off reaction to changes versus robustness to outliers. Finally, we plan to investigate the combination of LAF with forecasting techniques [46, 48] for proactive problem detection.

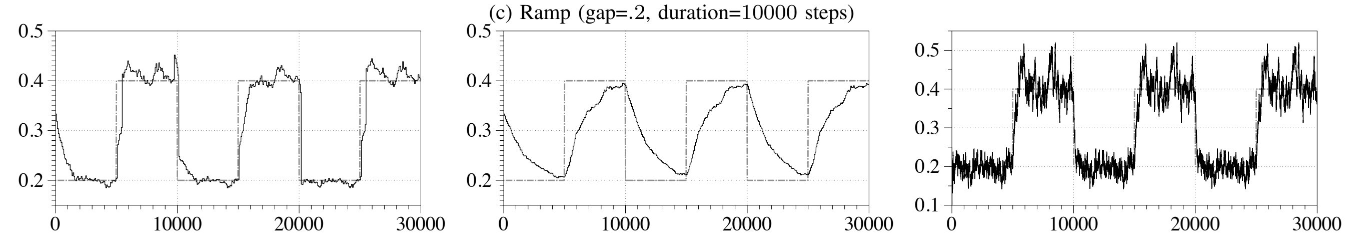
Acknowledgements

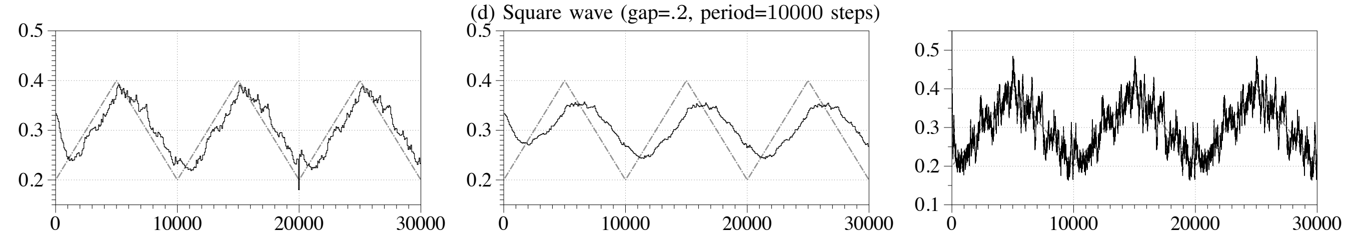
This work has been partially supported by the DFG (German Research Foundation) under the Priority Programme SPP1593: Design For Future - Managed Software Evolution.

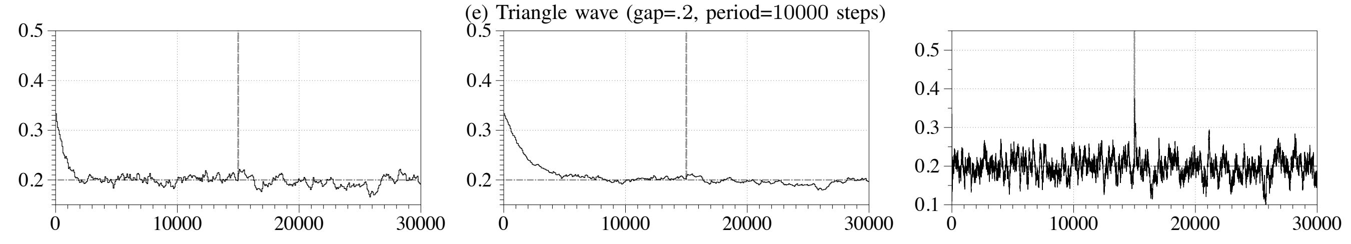

(a) Stationary signal with white noise ($\sigma = .001$)

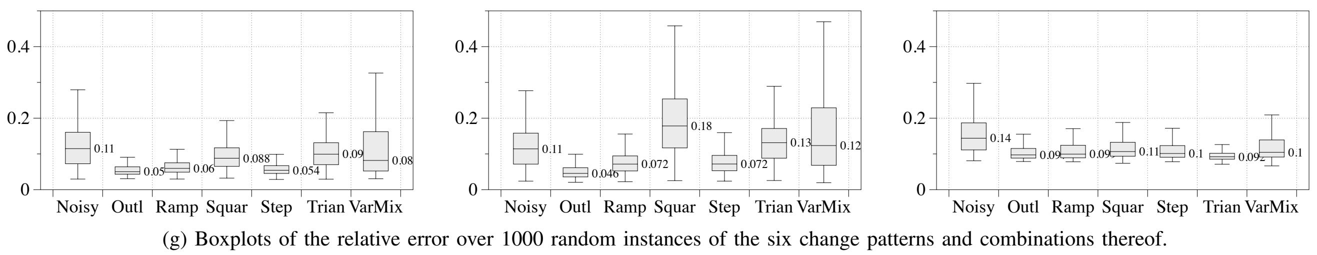

(b) Step (gap=.2)


(c) Ramp (gap=.2, duration=10000 steps)


(d) Square wave (gap=.2, period=10000 steps)


(e) Triangle wave (gap=.2, period=10000 steps)


(f) Outlier (gap=.4, duration=25 steps)



(g) Boxplots of the relative error over 1000 random instances of the six change patterns and combinations thereof.

Fig. 3. First six rows: estimates of the probability of moving toward the first state obtained by *LAF* (left column), *Kalman* (central column), and *Cove* (right column). Last row: IQR boxplots of the relative errors obtained over 1000 random instances of the six patterns and combinations thereof.

References

[1] A. Aziz, K. Sanwal, V. Singhal, and R. Brayton, "Model-checking continuous-time markov chains," *ACM Trans. Comput. Log.*, vol. 1, no. 1, pp. 162–170, Jul 2000 DOI:10.1145/343369.343402

[2] C. Baier, J.-P. Katoen, and H. Hermanns, "Approximate symbolic model checking of continuous-time markov chains," in *Proceedings of the 10th International Conference on Concurrency Theory (CONCUR 1999)*, vol. 1664. Springer, 1999, pp. 146–161 DOI:10.1007/3-540-48320-9_12

[3] C. Baier, B. R. Haverkort, H. Hermanns, and J.-P. Katoen, "Model-checking algorithms for continuous-time markov chains," *IEEE Trans. Softw. Eng.*, vol. 29, no. 6, pp. 524–541, 2003 DOI:10.1109/TSE.2003.1205180

[4] A. Bianco and L. de Alfaro, "Model checking of probabilistic and nondeterministic systems," in *Proceedings of the 15th Conference on Foundations of Software Technology and Theoretical Comp. Science (FSTTCS)*, vol. 1026. Springer, 1995, pp. 499–513 DOI:10.1109/MC.2009.326

[5] H. Hermanns, J.-P. Katoen, J. Meyer-Kayser, and M. Siegle, "A tool for model-checking Markov chains," *International Journal on Software Tools for Technology Transfer (STTT)*, vol. 4, no. 2, pp. 153–172, Feb 2003 DOI:10.1007/s100090100072

[6] M. Z. Kwiatkowska, G. Norman, and D. Parker, "Probabilistic symbolic model checking with prism: a hybrid approach," *Int. Journal on Software Tools for Technology Transfer (STTT)*, vol. 6, no. 2, pp. 128–142, Aug 2004 DOI:10.1007/s10009-004-0140-2

[7] L. Grunske, "Specification patterns for probabilistic quality properties," in *30th International Conference on Software Engineering (ICSE 2008)*. ACM, 2008, pp. 31–40 DOI:10.1145/1368088.1368094

[8] L. Baresi, E. D. Nitto, and C. Ghezzi, "Towards open-world software: Issues and challenges," ser. SEW. IEEE, 2006, pp. 249–252 DOI:10.1109/MC.2006.362

[9] B. H. C. Cheng, R. de Lemos, H. Giese, P. Inverardi, J. Magee, J. Andersson, B. Becker, N. Bencomo, Y. Brun, B. Cukic, G. D. M. Serugendo, S. Dustdar, A. Finkelstein, C. Gacek, K. Geihs, V. Grassi, G. Karsai, H. M. Kienle, J. Kramer, M. Litoiu, S. Malek, R. Mirandola, H. A. Müller, S. Park, M. Shaw, M. Tichy, M. Tivoli, D. Weyns, and J. Whittle, "Software engineering for self-adaptive systems: A research roadmap," in *Software Engineering for Self-Adaptive Systems*, ser. LNCS, vol. 5525. Springer, 2009, pp. 1–26.

[10] R. Calinescu, K. Johnson, Y. Rafiq, S. Gerasimou, G. C. Silva, and S. N. Pehlivanov, "Continual verification of non-functional properties in cloud-based systems," in *Proceedings of the 5th International Workshop Non-functional Properties in Modeling: Analysis, Languages and Processes, Miami, USA, September 29, 2013*, vol. 1074, 2013, pp. 1–5.

[11] G. S. Blair, N. Bencomo, and R. B. France, "Models@ run.time," *IEEE Computer*, vol. 42, no. 10, pp. 22–27, 2009 DOI:10.1109/MC.2009.326

[12] 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 2009)*. IEEE, 2009, pp. 111–121 DOI:10.1109/ICSE.2009.5070513

[13] A. Filieri, C. Ghezzi, and G. Tamburrelli, "A formal approach to adaptive software: continuous assurance of non-functional requirements," *Formal Aspects of Computing*, vol. 24, pp. 163–186, 2012 DOI:10.1007/s00165-011-0207-2

[14] R. Calinescu, Y. Rafiq, K. Johnson, and M. E. Bakir, "Adaptive model learning for continual verification of non-functional properties," in *ACM/SPEC International Conference on Performance Engineering, ICPE'14, Dublin, Ireland, March 22-26, 2014*. ACM, 2014, pp. 87–98 DOI:10.1145/2568088.2568094

[15] T. Zheng, C. M. Woodside, and M. Litoiu, "Performance model estimation and tracking using optimal filters," *IEEE Trans. Softw. Eng.*, vol. 34, no. 3, pp. 391–406, 2008 DOI:10.1109/TSE.2008.30

[16] I. Epifani, C. Ghezzi, and G. Tamburrelli, "Change-point detection for black-box services," in *Proceedings of the 18th ACM SIGSOFT International Symposium on Foundations of Software Engineering (FSE 2010)*. ACM, 2010, pp. 227–236 DOI:10.1145/1882291.1882326

[17] R. Calinescu, K. Johnson, and Y. Rafiq, "Using observation ageing to improve markovian model learning in qos engineering," in *Second Joint WOSP/SIPEW International Conference on Performance Engineering (ICPE'11)*. ACM, 2011, pp. 505–510 DOI:10.1145/1958746.1958823

[18] A. Filieri, L. Grunske, and A. Leva, "Lightweight adaptive filtering for dtmc learning: supplementary material," http://www.antonio.filieri.name/publications/preprints/2015-icse/.

[19] Hewlett-Packard Labs, "Worldcup98 web logs," http://ita.ee.lbl.gov/html/contrib/WorldCup.html.

[20] C. Baier and J.-P. Katoen, *Principles of Model Checking*. MIT Press, 2008.

[21] W. Pestman, *Mathematical Statistics*, ser. De Gruyter Textbook. De Gruyter, 2009.

[22] S. Ross, *Stochastic processes*, ser. Wiley series in probability and statistics: Probability and statistics. Wiley, 1996.

[23] T. W. Anderson and L. A. Goodman, "Statistical inference about markov chains," *The Annals of Mathematical Statistics*, vol. 28, no. 1, pp. 89–110, 1957.

[24] C. Chatfield, "Statistical inference regarding markov chain models," *J. R. Stat. Soc.*, vol. 22, no. 1, pp. 7–20, 1973.

[25] A. Immonen and E. Niemel, "Survey of reliability and availability prediction methods from the viewpoint of software architecture," *SoSyM*, vol. 7, no. 1, pp. 49–65, 2008 DOI:10.1007/s10270-006-0040-x

[26] R. C. Cheung, "A user-oriented software reliability model," *IEEE Trans. Softw. Eng.*, vol. 6, no. 2, pp. 118—125, 1980 DOI:10.1109/TSE.1980.234477

[27] A. Filieri, C. Ghezzi, V. Grassi, and R. Mirandola, "Reliability analysis of component-based systems with multiple failure modes," *Component-Based Software Engineering (CBSE 2010)*, pp. 1–20, 2010 DOI:10.1007/978-3-642-13238-4_1

[28] M. Kwiatkowska, "Quantitative verification: models techniques and tools," in *Proceedings of the the 6th joint meeting of the European software engineering conference and the ACM SIGSOFT symposium on The foundations of software engineering (ESEC/FSE 2007)*. ACM, 2007, pp. 449—458 DOI:10.1145/1287624.1287688

[29] A. Filieri, C. Ghezzi, and G. Tamburrelli, "Run-time efficient probabilistic model checking," in *Proceedings of the 33rd International Conference on Software Engineering (ICSE 2011)*. ACM, 2011, pp. 341–350 DOI:10.1145/1985793.1985840

[30] R. Calinescu, L. Grunske, M. Z. Kwiatkowska, R. Mirandola, and G. Tamburrelli, "Dynamic qos management and optimization in service-based systems," *IEEE Trans. Softw. Eng.*, vol. 37, no. 3, pp. 387–409, 2011 DOI:10.1109/TSE.2010.92

[31] A. Filieri, C. Ghezzi, A. Leva, and M. Maggio, "Self-adaptive software meets control theory: A preliminary approach supporting reliability requirements," in *Proceedings of the 26th IEEE/ACM International Conference on Automated Software Engineering (ASE 2011)*. IEEE CS, 2011, pp. 283–292 DOI:10.1109/ASE.2011.6100064

[32] ——, "Reliability-driven dynamic binding via feedback control," in *Software Engineering for Adaptive and Self-Managing Systems (SEAMS), 2012 ICSE Workshop on*, june 2012, pp. 43—52 DOI:10.1109/SEAMS.2012.6224390

[33] A. Filieri and C. Ghezzi, "Further steps towards efficient runtime verification: Handling probabilistic cost models," in *Proceedings of the First International Workshop on Formal Methods in Software Engineering: Rigorous and Agile Approaches*, ser. FormSERA '12. IEEE Press, 2012, pp. 2–8.

[34] I. Meedeniya and L. Grunske, "An efficient method for architecture-based reliability evaluation for evolving systems with changing parameters," in *IEEE 21st International Symposium on Software Reliability Engineering, (ISSRE 2010)*. IEEE CS, 2010, pp. 229–238 DOI:10.1109/ISSRE.2010.19

[35] C. Robert, *The Bayesian Choice: From Decision-Theoretic Foundations to Computational Implementation*, ser. Springer Texts in Statistics. Springer, 2007.

[36] A. Law, *Simulation Modeling and Analysis*, ser. McGraw-Hill series in industrial engineering and management science. McGraw-Hill Education, 2014.

[37] K. Åström and B. Wittenmark, *Computer-Controlled Systems: Theory and Design*. Dover Publications, 2013.

[38] W. Levine, *The Control Handbook*, ser. Electrical Engineering Handbook. Taylor & Francis, 2010.

[39] F. Cellier and E. Kofman, *Continuous System Simulation*. Springer, 2006.

[40] K. Ito and K. Kunisch, *Lagrange Multiplier Approach to Variational Problems and Applications*, ser. Advances in Design and Control. Society for Industrial and Applied Mathematics, 2008.

[41] H. W. Kuhn and A. W. Tucker, "Nonlinear programming," in *Proc. 2^{nd} Berkeley Symposium on Mathematical Statistics and Probabilistics*. Berkeley: University of California Press, 1951, pp. 481–492.

[42] D. Knuth, *The Art of Computer Programming: Seminumerical algorithms*, ser. Addison-Wesley series in computer science and information processing. Addison-Wesley, 1981.

[43] C. Robert and G. Casella, *Monte Carlo Statistical Methods*, ser. Springer Texts in Statistics. Springer, 2010.

[44] C. Chatfield, *The Analysis of Time Series: An Introduction*, ser. Chapman & Hall/CRC Texts in Statistical Science. Taylor & Francis, 2013.

[45] A. Filieri, H. Hoffmann, and M. Maggio, "Automated design of self-adaptive software with control-theoretical formal guarantees," in *Proceedings of the 36th International Conference on Software Engineering (ICSE 2014)*, ser. ICSE. ACM, 2014, pp. 299–310 DOI:10.1145/2568225.2568272

[46] A. Amin, L. Grunske, and A. Colman, "An automated approach to forecasting qos attributes based on linear and non-linear time series modeling," in *Proceedings of the 27th IEEE/ACM International Conference on Automated Software Engineering (ASE 2012)*. IEEE, Sept 2012, pp. 130–139 DOI:10.1145/2351676.2351695

[47] C. Ghezzi, M. Pezzè, M. Sama, and G. Tamburrelli, "Mining behavior models from user-intensive web applications," in *Proceedings of the 36th International Conference on Software Engineering (ICSE 2014)*. ACM, 2014, pp. 277–287 DOI:10.1145/2568225.2568234

[48] A. Amin, A. Colman, and L. Grunske, "An approach to forecasting qos attributes of web services based on arima and garch models," in *Proceedings of the IEEE 19th International Conference on Web Services (ICWS 2012)*. IEEE, June 2012, pp. 74–81 DOI:10.1109/ICWS.2012.37