Adaptive Runtime Estimate of Task Execution Times using Bayesian Modeling

Anna Friebe
Mälardalen University
Västerås, Sweden
anna.friebe@mdh.se

Filip Marković
Mälardalen University
Västerås, Sweden
filip.markovic@mdh.se

Alessandro V. Papadopoulos
Mälardalen University
Västerås, Sweden
alessandro.papadopoulos@mdh.se

Thomas Nolte
Mälardalen University
Västerås, Sweden
thomas.nolte@mdh.se

***Abstract*—In the recent works that analyzed execution-time variation of real-time tasks, it was shown that such variation may conform to regular behavior. This regularity may arise from multiple sources, e.g., due to periodic changes in hardware or program state, program structure, inter-task dependence or inter-task interference. Such complexity can be better captured by a Markov Model, compared to the common approach of assuming independent and identically distributed random variables. However, despite the regularity that may be described with a Markov model, over time, the execution times may change, due to irregular changes in input, hardware state, or program state. In this paper, we propose a Bayesian approach to adapt the emission distributions of the Markov Model at runtime, in order to account for such irregular variation. A preprocessing step determines the number of states and the transition matrix of the Markov Model from a portion of the execution time sequence. In the preprocessing step, segments of the execution time trace with similar properties are identified and combined into clusters. At runtime, the proposed method switches between these clusters based on a Generalized Likelihood Ratio (GLR). Using a Bayesian approach, clusters are updated and emission distributions estimated. New clusters can be identified and clusters can be merged at runtime. The time complexity of the online step is O(N^2 + NC) where N is the number of states in the Hidden Markov Model (HMM) that is fixed after the preprocessing step, and C is the number of clusters.**

***Index Terms*—Real-time systems, Hidden Markov Model, Bayesian Analysis, Probabilistic Timing Analysis**

I. Introduction

The characterization of the execution time of a real-time task is an important step towards analyzing the schedulability of a real-time system. The execution-time characterization usually focuses on the Worst-Case Execution Time (WCET), allowing for the analysis of hard real-time guarantees (for more details see [1], [2]). On the other hand, hardware acceleration features, multi-core systems [3] and complex, interacting tasks, e.g., in mixed criticality systems [4], [5], pose several challenges in achieving tight bounds on the WCET and Worst-Case Response-Time (WCRT) [2].

In the recent decades, probabilistic approaches have been proposed in relation to execution time estimates. The main purpose of the probabilistic approaches is to derive a more realistic distribution of the execution-time values, lowering upon the over-provisioning when one considers the worst-case values, while still considering Quality of Service (QoS) [6] or Quality of Control (QoC) [7]. The majority of this work considers estimating the probabilistic WCET (pWCET) distribution, that upper-bounds the execution time distributions of all valid scenarios and feasible sequences of repeated program execution [8]–[10]. Measurement-based techniques based on Extreme Value Theory (EVT) [11]–[13] require that extreme values of the execution time distribution are independent and that the measurements contain samples from the worst case distribution [14], [15]. As an upper bound, the pWCET may still be very pessimistic compared to the average execution time, and compared to the upper bound of the execution time distributions of scenarios that are valid in a more limited context of task execution that involves hardware and software state as well as input.

In some cases the entire distribution may be relevant, and not only the upper bound based on the distribution's tail. One such case is where QoS/ QoC adaptation can be utilized. Tasks can allow for different QoS/ QoC levels as proposed by Lu et al. [16]. In a robotic application, the robot's speed can be adjusted to allow for a lower frequency control loop. In these cases, the deadline miss probability can be kept sufficiently low with task adaptation. This can also relax the requirement to capture samples from the most extreme conditions in the analysis stage, provided the adaptation options are satisfactory.

In this paper, we address the problem of runtime estimation of execution-time distribution, analyzing the execution trace of a task **at runtime**. More specifically, in a preprocessing step, the number of states and the transition matrix of the Hidden Markov Model (HMM) are derived from a portion of the execution time sequence. Segments within this sequence that are similar are identified. Here we use the Generalized Likelihood Ratio (GLR), a measure for the likelihood that the segments are generated from the same HMM. Similar segments are combined into clusters, to form HMMs with differing emission distributions. At runtime, the algorithm switches between these HMMs depending on similarity with the current segment of the execution time trace. New HMMs can be created and the emission distributions updated. The complexity of the proposed runtime adaptive algorithm for the estimation of task distributions is $\mathcal{O}(N^2 + NC)$ where

This work was supported by the Swedish Research Council (VR) via the project "Practical Probabilistic Timing Analysis of Real-Time Systems (PARIS)", and by the Knowledge Foundation (KKS) via the project FIESTA.

N is the number of states in the HMM that is fixed after the preprocessing step, and C is the number of clusters. The proposed approach has the potential for being used for the assessment of several real-time system properties, but such an investigation is beyond the scope of this work.

The remainder of this paper is organized as follows. Related work is outlined in Section II. In Section III, we describe the system-model assumptions, along with definitions and mathematical background used in the paper. Then, in Section IV we describe the derivation of the initial HMM in the preprocessing step, which is followed by Section V, the description of the method for online model-parameter adaptation. The evaluation is described in Section VI, and the paper is concluded in Section VII.

II. Related work

Two major surveys on the Probabilistic Timing Analysis [8] and the Probabilistic Schedulability Analysis [17] of real-time systems have been conducted by Davis and Cucu-Grosjean, while a taxonomy and survey on pWCET analysis and associated methods was provided by Cazorla et al. [10]. We further describe the state-of the art in measurement-based methods, where the contributions of this paper fall into.

Measurement-Based Probabilistic Timing Analysis was introduced by Cucu-Grosjean [18], based on previous work related to the use of Extreme Value Theory (EVT) [11]–[13]. EVT is applied to find the pWCET, an upper bound on the probability of exceeding each possible execution time value. Methods based on EVT require that extreme values of the execution time distribution are independent [14], [15]. EVT-based techniques have also been applied in order to estimate upper bounds on response time distributions [19]–[22].

Moving from extreme values, and focusing on estimates of the full execution time distribution, the distribution of a visual task in a robotic application has been modeled as a HMM with discrete emission distributions by Frías et al. [23], [24]. HMMs can capture the regularity and dependability in the task execution, that may arise from different sources, e.g., sensed input, periodic nature of task interactions, or the algorithms being used in the tasks.

Friebe et al. [25] proposed an approach to estimate the execution time distribution using HMMs with Gaussian emission distributions, and proposed an automatic way of estimating the number of states in HMM from the execution trace. The methods from [25] are utilized for HMM fitting in the preprocessing step.

In all these approaches, the structure of the HMM and the emission distributions are learned in an offline phase, based on existing logged data. However, although in the base case the execution times may be characterized with a Markov Model, throughout the task's life cycle the model accuracy may deteriorate due to different irregularities such as changes in input, hardware, or program state. If the observations used for fitting the HMM are not fully representative of the runtime observations, the model may also be inaccurate. In Frías et al. [24], two separate experiments are performed, for a clean and a noisy track respectively, and these give rise to two different Markov Models, with notably different bandwidth requirements. In order to apply these methods for tasks where the context affecting the execution time distribution may change, an adaptive approach is necessary. In this work, we assume that a preprocessing phase is conducted where the HMM fitting is performed as in previous work [25], but we propose a runtime Bayesian adaptation method to continuously refine the execution time model based on the new observations.

Lu et al. [16] propose a Feedback Control Real-Time Scheduling (FCS) architecture, including a Monitor, a Controller and a QoS Actuator. An adaptive estimate of the execution time distribution could allow for the Monitor to predict the deadline miss probability rather than measuring the past deadline miss ratio. To the best of our knowledge, no adaptive runtime estimates of execution time distributions have previously been proposed.

In the proposed method, segments of the execution time trace are considered, and the similarity measure is defined considering the HMM as a whole. An alternative could have been to consider each execution time sample separately, and adapt and add states to a single HMM while updating the transition matrix. This could be be achieved by considering novelty detection such as in Gruhl et al. [26] in combination with a HMM update mechanism. We hypothesize that the context affecting a task's execution time distribution can change suddenly, and that the task can be affected in a similar way in several segments during the execution. Therefore we have chosen to consider execution time segments, and to enable switching betweeen clusters. In the proposed Bayesian model, the emission distributions are Gaussian with unknown mean and precision. An alternative could have been to model the emission distributions as Gaussian with unknown mean but fix the precision estimates in the preprocessing step. Due to the risk of underestimating the variance and thus the tail width, this option was not chosen.

III. System Model and Definitions

In this section, a task model with irregular execution-time variability is outlined. A Bayesian model for estimating the execution-time distribution from observations is described. A measure of similarity, Generalized Likelihood Ratio (GLR), for the Bayesian models is presented. This measure is used in the preprocessing and adaptive steps to determine points where the execution-time distribution changes, and to find similar segments of the execution-time trace.

Notation. We denote sequences with parentheses, $()$, and sets with braces, $\{\}$. The estimate of a quantity x is indicated in the following as $\hat{x}$. Table I lists the main symbols used in the paper.

A. Task model

We consider a periodic task, that generates a sequence of jobs. The sequence of execution times of the jobs is $cs = (c_1, c_2, \ldots, c_t)$. We assume that the execution-time sequence

Notation	Description
$cs = (c_1, c_2, \ldots, c_t)$	Execution-time sequence
N	Number of states in the Markov Model
$\mathcal{M} = \{m_1, m_2, \ldots, m_N\}$	Set of Markov states
$\mathcal{P}$	$N \times N$ state transition matrix
$\mathcal{C} = \{C_1, C_2, \ldots, C_N\}$	Set of execution-time distributions
s_j	Contiguous segment of cs
$\{\mu_{nj}, \sigma^2_{nj}\}$	Mean and variance of state m_n in segment s_j
$S_k = \{s_j, \ldots\}$	Cluster, set of segments
γ_{ni}	Occupancy probability of state m_n in segment s_i
$\hat{\mathbf{a}}[0]_{jn}$	Estimated number of observations in state m_n and segment s_j
$\hat{\mathbf{a}}[1]_{jn}$	Estimated sum of observations in state m_n and segment s_j
$\hat{\mathbf{a}}[2]_{jn}$	Estimated sum of squared observations in state m_n and segment s_j
$\hat{\mu}_{nk}$	Estimated mean of state n and cluster S_k
$\hat{\nu}_{nk}$	Estimated variance of state n and cluster S_k
$NG(\mu, \lambda)$	Normal-Gamma distribution
$\mu_0, \kappa_0, \alpha_0, \beta 0$	Prior hyperparameters of NG
$\mu_L, \kappa_L, \alpha_L, \beta_L$	Posterior hyperparameters of NG
$D = (x_1, \ldots, x_L)$	Observation sequence of length L
$\mathrm{GLR}(S_k, S_l)$	Generalized Likelihood Ratio between clusters S_k and S_l
ℓ_k	Log-likelihood of cluster S_k
$nPseudoObs$	Number of pseudo observations in prior construction
$\mathcal{GP}(m(x), k(x, x'))$	Gaussian process with mean m and kernel k

TABLE I: Important notation used in this work.

cs can be characterized by a Markov model, described by the set $\{\mathcal{M}, \mathcal{P}, \mathcal{C}\}$, where

- $\mathcal{M} = \{m_1, m_2, \ldots, m_N\}$ is the set of N states, with m_n, $n \in \mathbb{N}$.
- $\mathcal{P}$ is the $N \times N$ state transition matrix, where the element $p_{a,b}$ represents the conditional probability $\mathbb{P}(X_{i+1} = m_b | X_i = m_a)$ of being in state m_b at round $i+1$, given that at round i the state is m_a.
- $\mathcal{C} = \{C_1, C_2, \ldots, C_N\}$ is the set of execution-time distributions, or emission distributions related to respective state. In this paper, these are modeled as Gaussian distributions with mean μ_n, and variance σ^2_n, i.e., $C_n \sim \mathcal{N}(\mu_n, \sigma^2_n)$.

In the sequence cs, we assume that N and $\mathcal{P}$ remain unchanged, but at a finite number of points in the sequence, referred to as points of cluster change, the parameters $\{\mu_n, \sigma^2_n\}$ may take new (different) values. For this purpose we introduce another index, j, to explicitly indicate the dependency on time. The parts of the sequence where $\{\mu_n, \sigma^2_n\}$ remain constant are referred to as *segments* s_j. In each segment s_j, the mean and variance are denoted $\{\mu_{nj}, \sigma^2_{nj}\}$. A set $S_k = \{s_j, \ldots\}$ of non-adjacent segments with the same values of $\{\mu_{nj}, \sigma^2_{nj}\}$ are referred to as a *cluster*. An illustration is shown in Fig. 1.

B. Estimating sufficient statistics

The Markov Model $\{\mathcal{M}, \mathcal{P}, \mathcal{C}\}$ can be estimated by the use of the Forward-Backward algorithm [27] in combination with the Expectation Maximization algorithm [28]. The number of states N can be determined as described in Section IV. Given this information and an execution time sequence or segment, the state occupancy probabilities γ_{ni} can be obtained for each state m_n and execution time observation cs_i using the Forward-Backward algorithm. The occupancy probabilities are used to calculate sufficient statistics (presented in [25], [29]) for each segment s_j and state n. Sufficient statistics are a compact way of storing the information needed to estimate the Gaussian emission distribution of each state within the segment, and are also used when updating the Bayesian model. The sufficient statistics for a Gaussian distribution are: (i) $\hat{\mathbf{a}}[0]$, an estimate of the number of observations in the state, (ii) $\hat{\mathbf{a}}[1]$, an estimate of the sum of the observations in the state, and (iii) $\hat{\mathbf{a}}[2]$, an estimate of the sum of the squared observations in the state.

$$\hat{\mathbf{a}}[0]_{jn} = \sum_{i, c_i \in s_j} \gamma_{ni}, \tag{1}$$

$$\hat{\mathbf{a}}[1]_{jn} = \sum_{i, c_i \in s_j} c_i \gamma_{ni}, \tag{2}$$

$$\hat{\mathbf{a}}[2]_{jn} = \sum_{i, c_i \in s_j} c_i^2 \gamma_{ni}. \tag{3}$$

C. Bayesian model

In a Bayesian approach, a conjugate distribution is a distribution where the posterior probability $p(\Theta|D)$ of the parameter Θ given observations D, takes the same functional form as the prior distribution $p(\Theta)$ [30]. For a Gaussian probability distribution with unknown mean μ and precision $\lambda = 1/\sigma^2$, the conjugate distribution is a Normal-Gamma distribution [31], denoted as $NG(\mu, \lambda)$. When we have a prior distribution of μ and λ as given by

$$\begin{aligned} p(\mu, \lambda) = NG(\mu, \lambda | \mu_0, \kappa_0, \alpha_0, \beta_0) \triangleq \\ \mathcal{N}(\mu | \mu_0, (\kappa_0 \lambda)^{-1}) Ga(\lambda | \alpha_0, rate = \beta_0), \end{aligned} \tag{4}$$

and observations $D = (x_1, \ldots, x_L)$, the posterior probability distribution can be computed as:

$$p(\mu, \lambda | D) = NG(\mu, \lambda | \mu_L, \kappa_L, \alpha_L, \beta_L), \tag{5}$$

$$\mu_L = \frac{\kappa_0 \mu_0 + \sum_{i=1}^{L} x_i}{\kappa_0 + L}, \tag{6}$$

$$\kappa_L = \kappa_0 + L, \tag{7}$$

$$\alpha_L = \alpha_0 + L/2, \tag{8}$$

$$\beta_L = \beta_0 + \frac{1}{2} \left(\sum_{i=1}^{L} (x_i - \bar{x})^2 + \frac{\kappa_0 L (\bar{x} - \mu_0)^2}{(\kappa_0 + L)} \right). \tag{9}$$

Given that we are not certain of which state each observation c_i belongs to, we rewrite Eqs. (6) to (9), using the sufficient statistics in Eqs. (1) to (3), yielding:

$$\mu_L = \frac{\kappa_0 \mu_0 + \hat{\mathbf{a}}[1]}{\kappa_0 + \hat{\mathbf{a}}[0]}, \tag{10}$$

$$\kappa_L = \kappa_0 + \hat{\mathbf{a}}[0], \tag{11}$$

$$\alpha_L = \alpha_0 + \hat{\mathbf{a}}[0]/2, \tag{12}$$

$$\beta_L = \beta_0 + \frac{1}{2} \left(\hat{\mathbf{a}}[2] - \frac{\hat{\mathbf{a}}[1]^2}{\hat{\mathbf{a}}[0]} + \frac{\kappa_0 \hat{\mathbf{a}}[0] (\frac{\hat{\mathbf{a}}[1]}{\hat{\mathbf{a}}[0]} - \mu_0)^2}{(\kappa_0 + \hat{\mathbf{a}}[0])} \right), \tag{13}$$

where we excluded the indices j, n for readability, and we used $\hat{\mathbf{a}}[0]_{jn}$ to estimate L, $\hat{\mathbf{a}}[1]_{jn}$ to estimate $L\bar{x}$ and $\hat{\mathbf{a}}[2]_{jn}$

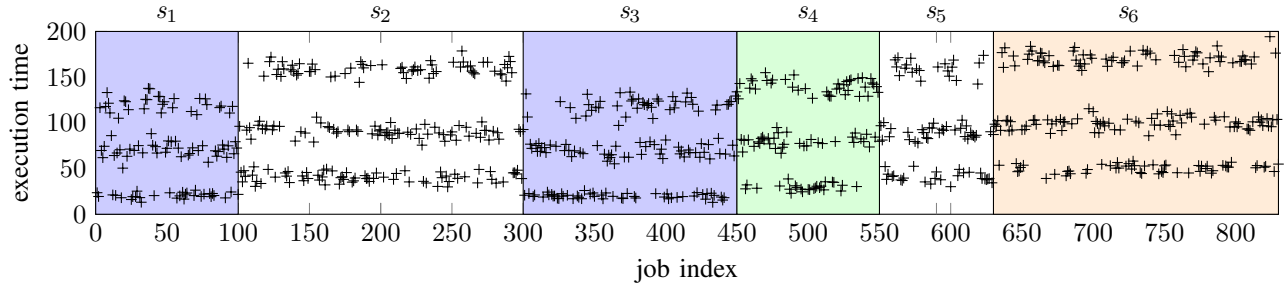

Fig. 1: An execution time sequence separated into six *segments* ($s_1, s_2, s_3, s_4, s_5, s_6$) and four *clusters* ($S_1, S_2, S_3, S_4$), where $S_1 = \{s_1, s_3\}$, $S_2 = \{s_2, s_5\}$, $S_3 = \{s_4\}$, and $S_4 = \{s_6\}$.

to estimate $L\bar{x}^2$. Eqs. (10) to (13) describe the Normal Gamma parameters for a state and segment.

The initial hyperparameters can be conceptualized as derived from a number of pseudo-observations, with the mean μ_0 derived from κ_0 observations and the precision λ_0 derived from $2\alpha_0$ observations with mean μ_0 and sum of squared deviations $2\beta_0$.

Since the Normal-Gamma distribution is the conjugate distribution, segments can be added and the posterior hyperparameters updated incrementally by substituting the existing posterior hyperparameters with the prior hyperparameters. In this way the posterior Normal-Gamma distribution for a cluster is constructed by incrementally updating the hyperparameters for each segment in the cluster. Similarly, segments can be removed from the estimate by updating the hyperparameters according to:

$$\mu_L = \frac{\kappa_0\mu_0 - \hat{\mathbf{a}}[1]}{\kappa_0 - \hat{\mathbf{a}}[0]}, \tag{14}$$

$$\kappa_L = \kappa_0 - \hat{\mathbf{a}}[0], \tag{15}$$

$$\alpha_L = \alpha_0 - \hat{\mathbf{a}}[0]/2, \tag{16}$$

$$\beta_L = \beta_0 - \frac{1}{2}\left(\hat{\mathbf{a}}[2] - \frac{\hat{\mathbf{a}}[1]^2}{\hat{\mathbf{a}}[0]} + \frac{\kappa_0\hat{\mathbf{a}}[0](\frac{\hat{\mathbf{a}}[1]}{\hat{\mathbf{a}}[0]} - \mu_0)^2}{(\kappa_0 - \hat{\mathbf{a}}[0])}\right). \tag{17}$$

Note that the posterior predictive distribution for a single point prediction is the Student's t-distribution as described by Murphy [31]:

$$p(x|D) = t_{2\alpha_L}\left(\frac{\beta_L(\kappa_l + 1)}{\alpha_L\kappa_L}\right). \tag{18}$$

D. GLR between sets of segments

The GLR of two sets of observations can be used to determine whether the sets are likely to belong to a joint distribution or if it is more likely that they belong to distinct distributions. This is done by calculating the ratio of the probability of the observations under the joint model and the product of the probabilities of the observations under distinct models. The GLR measure is central to the proposed approach and used in the preprocessing as well as the adaptive step.

The GLR between two execution time segment sets S_k and S_l, and the union of the sets given as $S_{k\cup l}$, using log-likelihoods (indicated with ℓ), is [32]:

$$\mathrm{GLR}(S_k, S_l) = \ell_{k\cup l} - (\ell_k + \ell_l). \tag{19}$$

The posterior predictive distribution for m new observations given the Normal-Gamma prior is given in Murphy [31] as

$$p(D_{new}|D) = \frac{\Gamma(\alpha_{n+m})}{\Gamma(\alpha_n)}\frac{\beta_n^{\alpha_n}}{\beta_{n+m}^{\alpha_{n+m}}}\sqrt{\frac{\kappa_n}{\kappa_{n+m}}}(2\pi)^{m/2}. \tag{20}$$

were Γ represents the gamma function.

We use the same prior distribution for the two segment sets and for the union of the sets. Evaluating the posterior probability of the observations in a state of a segment set S_k under the posterior predictive distribution calculated from the same observations gives the log-likelihood using estimates from Eqs. (11) to (13) as:

$$\begin{aligned}\ell_k = {} & \log\Gamma(\alpha_{2k}) - \log\Gamma(\alpha_k) + \alpha_k\log\beta_k - \\ & \alpha_{2k}\log\beta_{2k} + \frac{1}{2}(\kappa_k - \kappa_{2k}) + \frac{\hat{\mathbf{a}}[0]_k}{2}2\pi.\end{aligned} \tag{21}$$

The last terms cancel out in the GLR, and the resulting equation for two segment sets and a state is:

$$\begin{aligned}\mathrm{GLR}(S_k, S_l) = {} & \log\Gamma(\alpha_{2(k\cup l)}) - \log\Gamma(\alpha_{(k\cup l)}) + \alpha_{k\cup l}\log\beta_{k\cup l} \\ & - \alpha_{2(k\cup l)}\log\beta_{2(k\cup l)} + \frac{1}{2}(\kappa_{k\cup l} - \kappa_{2(k\cup l)}) \\ & - \log\Gamma(\alpha_{2k}) + \log\Gamma(\alpha_k) - \alpha_k\log\beta_k \\ & + \alpha_{2k}\log\beta_{2k} - \frac{1}{2}(\kappa_k - \kappa_{2k}) \\ & - \log\Gamma(\alpha_{2l}) + \log\Gamma(\alpha_l) - \alpha_l\log\beta_l \\ & + \alpha_{2l}\log\beta_{2l} - \frac{1}{2}(\kappa_l - \kappa_{2l})\end{aligned} \tag{22}$$

The GLR of two segment sets is estimated as the sum of the GLRs over the states. We use GLRs based on log-likelihoods, so this is equivalent to multiplication of the GLR measure as described by Liu and Kubala [32].

IV. Preprocessing step

The preprocessing step is performed on an execution time sequence where we expect to capture the regular variation of execution times. The preprocessing step identifies:

1) The number of states N and transition matrix $\mathcal{P}$ of the cluster HMMs.
2) The segments and clusters within this execution time sequence.
3) The sufficient statistics for the HMM states of each cluster.

A HMM is fitted to the execution time sequence, using the tree-based cross validation approach described in [25]. This fitting process provides the number of states and transision matrix.

The normal distribution parameters μ, σ in the HMM from the preprocessing step are used in combination with a number of pseudo observations, $nPseudoObs$, to create initial prior Normal-Gamma distributions as:

$$\mu_0 = \mu, \tag{23}$$
$$\kappa_0 = nPseudoObs, \tag{24}$$
$$\alpha_0 = \frac{nPseudoObs}{2}, \tag{25}$$
$$\beta_0 = \alpha_0 \cdot \sigma^2. \tag{26}$$

The number $nPseudoObs$ is chosen for each state in relation to the stationary probability of the state.

A. Finding points of cluster change

For a sequence of execution time observations, we want to find the set of points where the model parameters change.

1) Finding one point of model change: Initially, we consider the simpler problem of finding one point of model change in a sequence. Given a starting index x_{start} and a stopping index x_{stop} within the sequence, we aim to find the index x_{split} that minimizes the $GLR(x)$, defined as

$$\mathrm{GLR}(x) = \mathrm{GLR}(\{s_{x-}\}, \{s_{x+}\}) \tag{27}$$
$$s_{x-} = (c_{x_{start}}, c_{x_{start}+1}, \ldots, c_x) \tag{28}$$
$$s_{x+} = (c_{x+1}, c_{x_{split}+1}, \ldots, c_{x_{stop}}) \tag{29}$$
$$x_{split} = \arg\min_x \mathrm{GLR}(x) \tag{30}$$

where the segments s_{x-} and s_{x+} indicate the segments before and after x. The optimization is performed Bayesian Optimization as implemented in the Python library GpyOpt[1], where BayesianOptimization is configured with a Radial Basis Function kernel and Expected Improvement as acquisition function. Posterior predictive Student's t-distributions are derived for s_{x-}, s_{x-} and for their union. The log-likelihoods for these segments are calculated by applying the Forward-Backward algorithm. The transition matrix $\mathcal{P}$ in taken from the fitted HMM, but the posterior predictive distributions are used as emission distributions.

If the resulting $\mathrm{GLR}(x_{split})$ is lower than a given GLR_{limit}, x_{split} is considered to be a point of model change.

[1]https://sheffieldml.github.io/GPyOpt/

2) Finding several points of model change: In order to find several points of model change within an execution time sequence cs, we apply the method described in Section IV-A1 for the entire sequence, $x_{start} = 1$, $x_{stop} = t$. Recursively, the method is applied for the sequences with $x_{start} = 1$, $x_{stop} = x_{split}$ and $x_{start} = x_{split} + 1$, $x_{stop} = t$, and further, similarly to a binary search approach, until one of the following stopping criteria are met:

1) The resulting $\mathrm{GLR}(x_{split})$ is above the given GLR_{limit};
2) The length of the subsequence is below a minimum length between splitting points.

B. Segment clustering

The segments are clustered into sets using an approach similar to the Leader-Follower Clustering described by Duda et al. [33]. The longest segment is added as the first cluster. For each segment, in order of decreasing segment length, the cluster that gives the maximum GLR is found as outlined in Section III-D. If the GLR between the segment and the closest cluster is large enough, the segment is merged into the cluster, otherwise a new cluster is created. The threshold has been set to 10 times the threshold used in finding the points of model change.

V. Online model adaptation

In the runtime process, a sliding window is considered. A simplified flowchart of the algorithm is available in Fig. 2.

The sliding window has a length of $T = a \cdot step$. Here, a and $step$ are integers, and $step$ is the size of the sliding window movement at each step. We assume that a starting cluster at the beginning of the window is known. The sliding window hyperparameters are estimated by calculating the posterior distribution using Eqs. (10) to (13) with the initial prior distribution. Sufficient statistics $\hat{\mathbf{a}}[0]$, $\hat{\mathbf{a}}[1]$ and $\hat{\mathbf{a}}[2]$ are derived using the Forward-Backward algorithm [27]. The emission distributions for the states are generalized Student's t-distributions, the posterior predictive distributions of the cluster at the start of the window as given in Eq. (18). The prior distribution is chosen as outlined in Section IV, Eqs. (23) to (26).

A number of GLR thresholds are used in the process, to determine whether a cluster change shall be made, if a new cluster shall be created, or if the current cluster shall be merged with another. In addition we use different thresholds for preprocessing clusters compared to newly created clusters, where we require a closer match with newly created clusters. One reason for this is that new clusters can be dominated by the prior distribution, and for this reason are more likely to have a high GLR when compared to each other. We base all thresholds on the threshold used for finding points of model change in the preprocessing step. Different multiplicative factors are applied, according to Table II.

A. Determining if there is a cluster change in the window

The GLR is estimated of the sliding window and the starting cluster distributions. If this is below a threshold

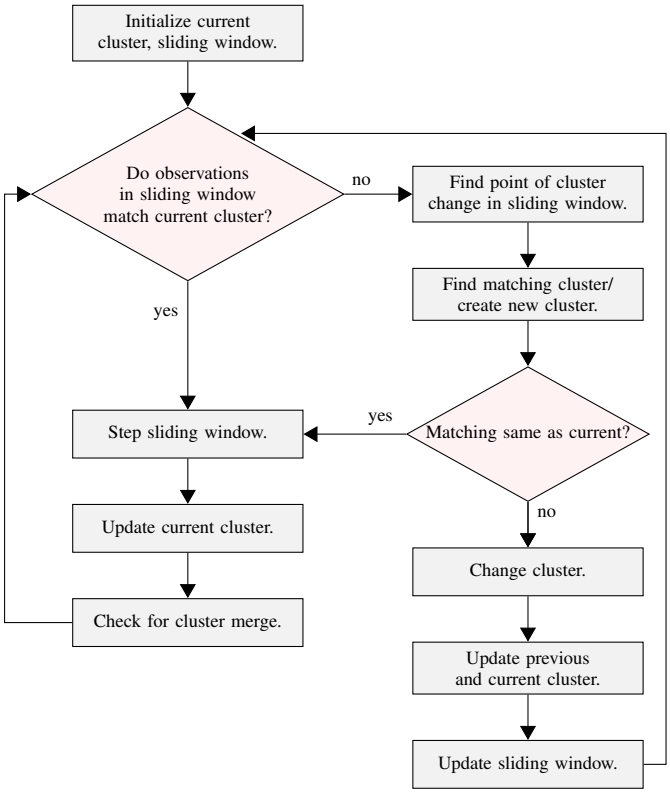

Fig. 2: Simplified flowchart of the adaptive process. The process continues until the task is terminated and there are no more observations to process.

Threshold	Purpose	Factor
slidingLimit	Check if cluster change in slidingwindow	1
newClusterLimit	Create new cluster	2
changePreLimit	Change to a preprocessing cluster	1
mergeClusterPreLimit	Merge current with preprocessing cluster	1.5
mergeClusterLimit	Merge current with closest cluster	1

TABLE II: Thresholds used in the adaptive process and the multiplicative factor to the threshold used in finding points of model change in the preprocessing step.

$slidingLimit$, we move into the right column of Fig. 2. A segment $clusterFindSegment$ of length T around the endpoint of the sliding window is considered. A Normal-Gamma distribution for this segment is calculated using Eqs. (10) to (13) and the initial prior distribution. Sufficient statistics are derived with the Forward-Backward algorithm using the generalized Student's t-distribution as the posterior predictive of the initial prior distribution. The point of cluster change and the cluster at the end of the sliding window are determined as outlined in Algorithm 1.

Determining the point of cluster change: The sliding window is divided into chunks that are considered from the endpoints of the sliding window. The GLR of the starting cluster with the first chunk is compared to the GLR of $closestClusterAll$ with the last chunk. Iteratively, the chunk with the highest GLR is added to the start or end sections of the sliding window, and the next chunk from the appropriate side is considered, until all chunks are added to either side.

B. Updating the sliding window and clusters

If a cluster change has taken place, we continue downwards in the right column of Fig. 2. A new sliding window is created from the endpoint of the current, using posterior student distributions of the new cluster to calculate the sufficient statistics. Posterior distributions and sufficient statistics for the previous and new clusters are updated with the sliding window segments prior and after the cluster changing point.

If there is no need for cluster change, we proceed in the left column of Fig. 2. The sliding window is advanced with the step size. The distributions are updated by removing the sufficient statistics for the step no longer in the sliding window, and adding those for the new step, according to Eqs. (10) to (13) and Eqs. (14) to (17). The updated cluster is compared with the other existing clusters. If the GLR of the current cluster and the closest cluster is large enough the clusters are merged.

C. Complexity analysis

The computation of sufficient statistics with the Forward-Backward method has a time complexity of $\mathcal{O}(N^2L)$, where N is the number of states and L is the length of the considered section. For each window, L is bounded by $2T$, as we may need to calculate sufficient statistics for the $clusterFindSegment$ when a cluster change is considered and for a new sliding window in the event of cluster change.

The GLR calculations are summed over the states, and we find the maximum GLR among all clusters, resulting in a total time complexity of $\mathcal{O}(NC)$, where N is the number of states and C is the number of clusters.

The total time complexity of the adaptive step is $\mathcal{O}(N^2 + NC)$, where N is the number of states in the HMM, fixed after the preprocessing step, and C is the number of clusters.

VI. Evaluation

A. Goal of the evaluation

In the following experiments[2], we first generated the execution samples according to the predefined ground truth model, and then we performed the proposed method on the execution samples in order to estimate the posterior distribution. The goal of the experiments was to investigate the accuracy of the estimated posterior distribution having the ground truth model as the reference. By using synthetic data in the evaluations we can make comparisons to the ground truth distributions. Comparisons are made by calculating the Kullback-Leibler (KL) divergence, as will be further outlined below.

In the experiments, we distinguish the two main steps, the preprocessing step of the method – where the initial execution sample is analyzed in an offline manner – and the adaptive process—where the estimated parameters, from the

[2]Code and data are available online https://github.com/annafriebe/AdaptiveETBayes.

Algorithm 1 Pseudocode describing the process of finding the potential point of change and the new cluster.

Input Current cluster at the beginning of the sliding window, preprocessing and adaptive clusters, sliding window and cluster finding segment with Normal-Gamma distributions.

Output Point of cluster change and current cluster at end of sliding window.

```
function CLUSTERCHANGE(clusters, preClusters, clusterFindSegment,slidingWindow, currentCluster)
    closestClusterAll ← arg max_{c∈clusters} GLR(c, clusterFindSegment)
    potentialChangePoint ← FINDCHANGEPOINT(slidingWindow, currentCluster, closestClusterAll)
    testEndSegment ← slidingWindow[potentialChangePoint:end]
    testNGAll ← POSTERIORNG(closestClusterAll, testEndSegmentNG)
    testGLRAll ← GLR(closestClusterAll, testNGAll)
    if testGLRAll < newClusterLimit then
        newCluster ← CREATECLUSTER(testEndSegment)
        return potentialChangePoint, newCluster
    end if
    closestClusterPre ← arg max_{c∈preClusters} GLR(c, clusterFindSegment)
    testNGPre ← POSTERIORNG(closestClusterPre, testEndSegmentNG)
    testGLRPre ← GLR(closestClusterPre, testNGPre)
    if testGLRPre > changePreLimit then
        return potentialChangePoint, closestClusterPre
    end if
    return potentialChangePoint, closestClusterAll
end function
```

preprocessing step, are adaptively modified online in order to account for the changes in the ground truth model over time. For the preprocessing step, we compared the estimated posterior distributions after the clustering process to the ground truth distributions. For the adaptive process, we compared the estimated posterior distributions during the adaptive process to the known generative distributions. Three versions of the adaptive process are evaluated.

1) The full algorithm with clusters created, adapted and merged. We refer to this version as EST_FP.
2) The online algorithm with cluster adaptation, but without creation and merging of clusters. We refer to this version as EST_NCM.
3) The online algorithm without creating, adapting or merging clusters, only switching between the clusters resulting from the preprocessing stage. We refer to this version as EST_SP.

To evaluate the similarity between the posterior estimates and the ground truth distributions, the KL divergence is calculated. The KL divergence is an asymmetric measure of the difference from a distribution Q to another reference distribution P, with continuous probability density functions $q(x)$ and $p(x)$ respectively, defined as

$$D_{KL}(P||Q) = \int_{-\infty}^{\infty} p(x) \log \frac{p(x)}{q(x)}\, dx. \tag{31}$$

The KL divergence was chosen as it quantifies the information lost when moving from the ground truth distribution to the estimated distribution. The GLR is not suitable for this evaluation because it is based on likelihoods of observations.

The KL divergence is numerically approximated from the estimated posterior distribution to the ground truth distribution in the range $[0, 150]$. The posterior distribution is constructed by weighting the generalized student's t distributions of each state with the stationary probabilities of the states in the fitted HMM. The ground truth normal distributions are similarly weighted with the known stationary probabilities of states.

B. Generation of sequences from the ground truth model

Execution sequences are generated from the ground truth model defined as a three state Markov model, where transition probabilities between states are in the range 0.1-0.8. Each sequence is constructed from five clusters, and each cluster is constructed from the segments of length within interval $[50, 300]$. One of the clusters does not appear until after the first 1000 job indices, which means it is not in the preprocessing section. The execution time samples for each cluster and its respective segments, are generated according to a three state Markov Model, such that each state is characterized by a Gaussian emission distribution with a mean randomly generated from one of the three following uniform ranges $[25, 50]$, $[65, 80]$ and $[95, 120]$ respectively and standard deviations within the range $[2, 6]$. The cluster means are ordered, so that if the mean of a state in cluster A is lower than the mean of the same state in cluster B, $\mu_{nA} < \mu_{nB}$, then the same relation applies to the other states' means in these clusters. The reason for this is that points of model change are not as accurately found when the state means of two clusters move in opposite directions. This is likely due to the construction of the GLR of segments as the sum over the states.

C. Results

1) Preprocessing step: In Fig. 3 we show means and standard deviation of the estimate, i.e. the posterior generalized student's t distributions of the resulting clusters as black lines. We also show the means and standard deviations of the ground truth clusters as red lines. For sequence 2, four states are identified, and the state with the lowest stationary probability is displayed in cyan. In Fig. 3, points of model change are also visible. KL divergence measures along the sequences are displayed, in black for the preprocessing section. Mean KL divergence measures for each cluster and for the preprocessing section are displayed in Table III.

2) Online adaptive process: The means and standard deviations of the posterior generalized student's t distributions

Sequence	1	2	3	4
Cluster 1	0.111	0.054	0.158	0.109
Cluster 2	0.149	1.663	0.045	0.036
Cluster 3	0.051	0.323	0.081	0.580
Cluster 4	0.151	0.067	0.116	0.174
All clusters	0.107	0.156	0.085	0.107

TABLE III: KL divergence measures for the preprocessing process.

during the adaptive process are displayed in blue in Fig. 3 for four sequences. These are shown in relation to the known means and standard deviations of the normal distributions in the true clusters in red. For sequence 2, the HMM identification finds four states, and the state with the lowest stationary probability is displayed in cyan. In Fig. 3 the points of model change are also visible.

The left column displays the result when applying EST_FP, the full process, to four test sequences. Creation of new clusters is indicated with black vertical lines, and merging of clusters is marked with red vertical lines. The middle column shows the result when applying EST_NCM, without creation and merging of clusters, but with adaptive cluster updates for the same sequences. The right column displays the result when applying EST_SP, with only switching between the preprocessing clusters.

The KL divergence from the distribution constructed from the posterior generalized student's t distributions to the distribution constructed from ground truth Gaussian distributions is calculated. When constructing the distributions, the emission distributions are weighted with the estimated and known stationary probabilities respectively. The KL divergence is calculated for each job index in each sequence for the three versions of the adaptive process. Results are displayed in Fig. 3. Means are calculated for each ground truth cluster and for the entire adaptive part of the sequence, and presented in Table IV. In sequences 1, 3 and 4, EST_SP has a better average fit (lower average KL divergence measure), as can be seen in the "All clusters" row of the tables. We also look at the average KL divergence of clusters not appearing in the preprocessing portion, that is Cluster 5 for all sequences, and for sequence 2 additionally Cluster 2. Here we see that EST_FP has lower KL divergence measures than EST_SP in four out of the five new clusters. For the EST_NCM, the KL divergence is lower for all five new clusters. EST_FP and EST_NCM appear to be roughly equivalent for new clusters, with the EST_NCM having lower KL divergence scores in three out of five new clusters.

D. Discussion

The KL divergence in the adaptive section is in the range of 2-10 times larger than in the preprocessing section in our experiments, for all three versions of the adaptive process. A larger KL divergence is expected from a less computationally expensive approach.

The fact that EST_SP performs better than the versions with cluster updates for clusters available at the preprocessing step indicates that there is some deterioration of the estimates, possibly due to erroneous estimates of the points of cluster change.

Sequence no.	Cluster no.	EST_FP	EST_NCM	EST_SP
1	1	0.347	0.248	0.277
	2	0.276	0.276	0.310
	3	N/A	N/A	N/A
	4	0.770	0.786	0.442
	5	0.411	0.266	0.359
	All clusters	0.459	0.401	0.346
2	1	0.173	0.173	0.217
	2	0.263	0.261	0.681
	3	0.493	0.493	0.539
	4	0.340	0.342	0.110
	5	0.355	0.362	0.400
	All clusters	0.297	0.297	0.392
3	1	0.460	0.608	0.478
	2	0.506	0.257	0.139
	3	1.142	1.180	0.808
	4	0.285	0.282	0.159
	5	0.270	0.503	0.512
	All clusters	0.688	0.739	0.536
4	1	0.241	0.242	0.215
	2	0.347	0.281	0.046
	3	0.498	0.502	0.620
	4	0.414	0.417	0.171
	5	0.631	0.627	0.760
	All clusters	0.418	0.405	0.373

TABLE IV: KL divergence measures for different sequences and clusters.

It can be noted that in some segments, the estimated means of the states with the highest and lowest means tend to move towards the middle state in the three state HMM, coinciding with a higher standard deviation. This is likely due to samples generated by the middle state distribution resulting in occupancy probabilities significantly higher than zero for an additional state. When these samples are weighted into the sufficient statistics of the lower or higher state, the posterior distribution is affected in this manner.

The choice of prior distribution has a similar influence on the posterior estimates. In the proposed method, the prior distribution is based on the HMM fitted to the preprocessing section. For portions of the execution time trace that deviate significantly from the preprocessing section, the posterior estimates will have a mean that is drawn towards the prior mean, and a variance that is overestimated due to the prior pseudo observations acting as outliers.

E. Limitations and future evaluation goals

The main limitation of the evaluation is that it has been performed with synthetic data, where the execution time samples are generated from ground-truth distributions with instantaneous cluster changes at specified points in time. The main reason for this choice was the controllable experiment setup where the ground truth model is known. One of the sensitive design choices of the experiment is evident in the generation of the ordered means within the clusters, which should be generalised in the future evaluations. Also, at the

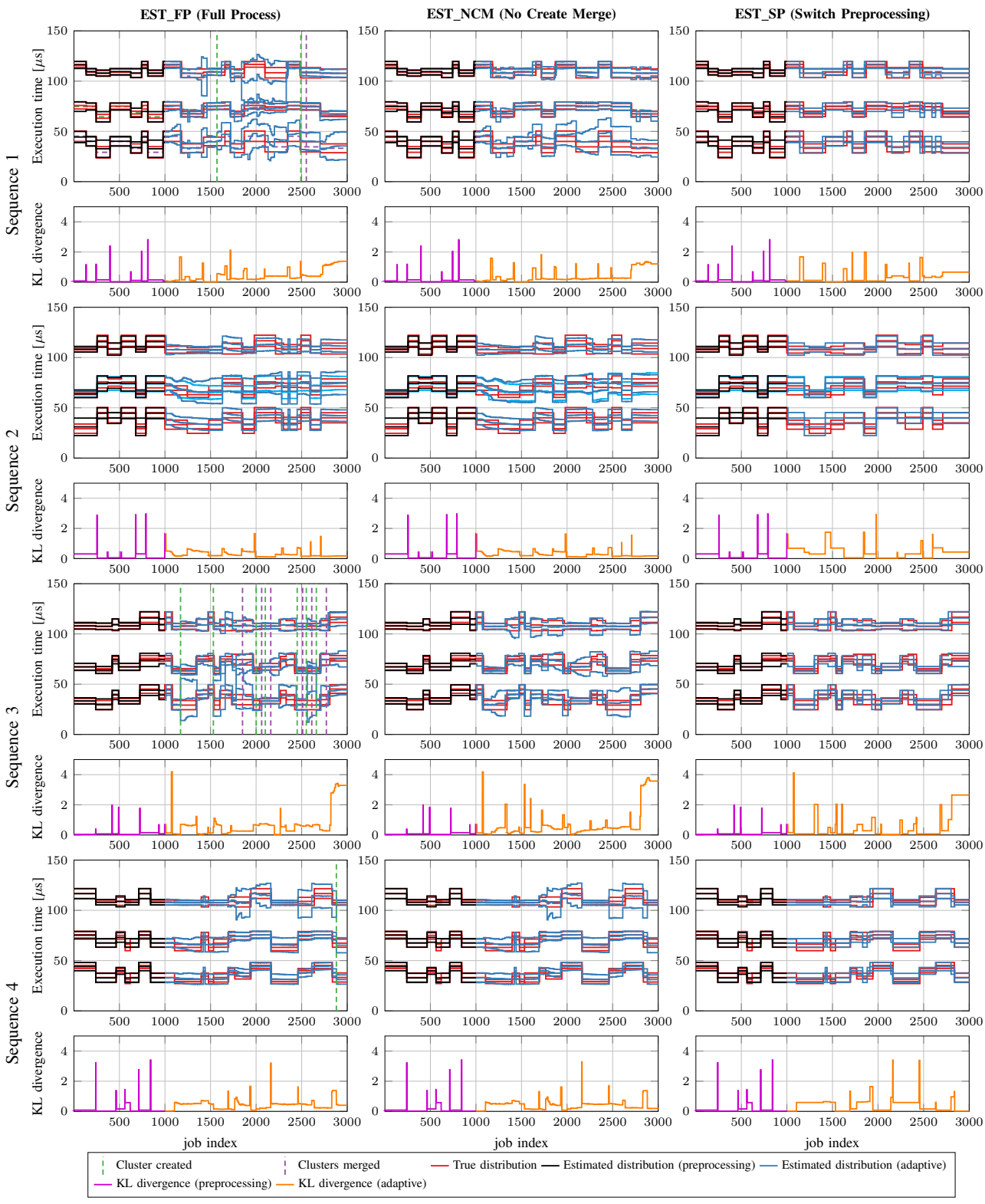

Fig. 3: True and predicted distributions for four sequences, with the three different versions of the process. KL divergence measures along the sequences are displayed.

moment we cannot be certain that the results are valid for more realistic use cases and this will be addressed in the future work.

VII. Conclusion and Future work

In this paper, we proposed a method to adjust at runtime an HMM aimed at characterizing the execution time of a task, with a limited time complexity. The posterior execution-time distributions obtained through the proposed approach could be used to assess several real-time properties of a system, e.g., estimating the deadline miss probabilities, but further investigations are needed, and devoted to future work.

The results from the evaluated synthetic test cases indicate that the proposed method is capable of adapting the estimates at runtime, such that the estimated distribution tracks the ground truth distribution used to generate the execution time samples. The similarity between the estimated and ground truth distributions are evaluated by calculating the Kullback-Leibler divergence. In some cases we can see biased means and increasing standard deviations in the posterior distribution. Future work will investigate the possibility of introducing regularization to limit the increase in the standard deviation. Furthermore, a more extensive evaluation will be performed on real applications, e.g., computer vision, robotics, or control use cases, to better assess the ability of the proposed approach to provide meaningful information on the execution time distributions of complex real-time applications.

References

[1] R. Wilhelm, J. Engblom, A. Ermedahl, N. Holsti, S. Thesing, D. Whalley, G. Bernat, C. Ferdinand, R. Heckmann, T. Mitra *et al.*, "The worst-case execution-time problem—overview of methods and survey of tools," *ACM TECS*, vol. 7, no. 3, p. 36, 2008.

[2] M. Joseph and P. Pandya, "Finding response times in a real-time system," *The Computer Journal*, vol. 29, no. 5, pp. 390–395, 1986.

[3] A. Löfwenmark and S. Nadjm-Tehrani, "Fault and timing analysis in critical multi-core systems: A survey with an avionics perspective," *Journal of Syst. Architecture*, vol. 87, pp. 1–11, 2018.

[4] A. Burns and R. I. Davis, "A survey of research into mixed criticality systems," *ACM Comput. Surv.*, vol. 50, no. 6, 2017.

[5] R. Wilhelm, "Mixed Feelings About Mixed Criticality," in *Int. W. on Worst-Case Exec. Time Analysis (WCET)*, vol. 63, 2018, pp. 1:1–1:9.

[6] D. D. Clark, S. Shenker, and L. Zhang, "Supporting real-time applications in an integrated services packet network: Architecture and mechanism," *SIGCOMM Comput. Commun. Rev.*, vol. 22, no. 4, p. 14–26, 1992.

[7] P. Martí, J. M. Fuertes, G. Fohler, and K. Ramamritham, "Improving quality-of-control using flexible timing constraints: metric and scheduling," in *IEEE Real-Time Syst. Symp. (RTSS)*, 2002, pp. 91–100.

[8] R. I. Davis and L. Cucu-Grosjean, "A survey of probabilistic timing analysis techniques for Real-Time systems," *Leibniz Trans. Emb. Syst.*, vol. 6, no. 1, pp. 03–1–03:60, May 2019.

[9] J. L. Diaz, J. M. Lopez, M. Garcia, A. M. Campos, K. Kim, and L. L. Bello, "Pessimism in the stochastic analysis of real-time systems: Concept and applications," in *IEEE Int. Real-Time Syst. Symp. (RTSS)*, 2004, pp. 197–207.

[10] F. J. Cazorla, L. Kosmidis, E. Mezzetti, C. Hernandez, J. Abella, and T. Vardanega, "Probabilistic worst-case timing analysis: Taxonomy and comprehensive survey," *ACM Comput. Surv.*, vol. 52, no. 1, 2019.

[11] A. Burns and S. Edgar, "Predicting computation time for advanced processor architectures," in *Euromicro Conf. on Real-Time Syst. (ECRTS)*, 2000, pp. 89–96.

[12] S. Edgar and A. Burns, "Statistical analysis of wcet for scheduling," in *IEEE Real-Time Syst. Symp. (RTSS)*, 2001, pp. 215–224.

[13] D. Griffin and A. Burns, "Realism in statistical analysis of worst case execution times," in *Int. W. on Worst-Case Exec. Time Analysis (WCET)*, 2010.

[14] L. Santinelli, J. Morio, G. Dufour, and D. Jacquemart, "On the sustainability of the extreme value theory for wcet estimation," in *Int. W. on Worst-Case Exec. Time Analysis (WCET)*, 2014.

[15] M. Leadbetter, G. Lindgren, and H. Rootzén, "Conditions for the convergence in distribution of maxima of stationary normal processes," *Stochastic Proc. and their Appl.*, vol. 8, no. 2, pp. 131–139, 1978.

[16] C. Lu, J. A. Stankovic, S. H. Son, and G. Tao, "Feedback control real-time scheduling: Framework, modeling, and algorithms," *Real-Time Systems*, vol. 23, no. 1, pp. 85–126, 2002.

[17] R. I. Davis and L. Cucu-Grosjean, "A survey of probabilistic schedulability analysis techniques for Real-Time systems," *LITES: Leibniz Trans. Emb. Syst.*, pp. 1–53, 2019.

[18] L. Cucu-Grosjean, L. Santinelli, M. Houston, C. Lo, T. Vardanega, L. Kosmidis, J. Abella, E. Mezzetti, E. Quiñones, and F. J. Cazorla, "Measurement-Based probabilistic timing analysis for multi-path programs," in *Euromicro Conf. on Real-Time Syst. (ECRTS)*, 2012, pp. 91–101.

[19] Y. Lu, T. Nolte, J. Kraft, and C. Norstrom, "Statistical-based response-time analysis of systems with execution dependencies between tasks," in *IEEE Int. Conf. on Eng. of Complex Comp. Syst.*, 2010, pp. 169–179.

[20] ——, "A statistical approach to response-time analysis of complex embedded real-time systems," in *IEEE Int. Conf. on embedded and real-time computing systems and applications*, 2010, pp. 153–160.

[21] Y. Lu, T. Nolte, I. Bate, and L. Cucu-Grosjean, "A statistical response-time analysis of complex real-time embedded systems by using timing traces," in *IEEE Int. Symp. on Ind. and Emb. Syst.*, 2011, pp. 43–46.

[22] ——, "A statistical response-time analysis of real-time embedded systems," in *IEEE Real-Time Syst. Symp. (RTSS)*, 2012, pp. 351–362.

[23] L. Abeni, D. Fontanelli, L. Palopoli, and B. V. Frías, "A Markovian model for the computation time of real-time applications," in *IEEE Int. Instrumentation and Measurement Tech. Conf. (I2MTC)*, 2017, pp. 1–6.

[24] B. V. Frías, L. Palopoli, L. Abeni, and D. Fontanelli, "Probabilistic real-time guarantees: There is life beyond the iid assumption," in *IEEE Real-Time and Embedded Technology and Applications Symp. (RTAS)*, 2017, pp. 175–186.

[25] A. Friebe, A. V. Papadopoulos, and T. Nolte, "Identification and validation of markov models with continuous emission distributions for execution times," in *IEEE Int. Conf. on Emb. and Real-Time Comp. Syst. and Appl. (RTCSA)*, 2020, pp. 1–10.

[26] C. Gruhl, J. Schmeiβing, S. Tomforde, and B. Sick, "Normal-wishart clustering for novelty detection," in *2020 IEEE International Conference on Autonomic Computing and Self-Organizing Systems Companion (ACSOS-C)*. IEEE, 2020, pp. 64–69.

[27] L. R. Rabiner, "A tutorial on hidden markov models and selected applications in speech recognition," *Proc. the IEEE*, vol. 77, no. 2, pp. 257–286, 1989.

[28] T. K. Moon, "The expectation-maximization algorithm," *IEEE Signal processing magazine*, vol. 13, no. 6, pp. 47–60, 1996.

[29] T. Shinozaki, "HMM state clustering based on efficient cross-validation," in *IEEE Int. Conf. on Acoustics Speech and Signal Proc.*, vol. 1, 2006.

[30] C. M. Bishop, *Pattern recognition and machine learning*. springer, 2006.

[31] K. P. Murphy, "Conjugate Bayesian analysis of the Gaussian distribution," University of British Columbia, Tech. Rep., 2007. [Online]. Available: https://www.cs.ubc.ca/~murphyk/Papers/bayesGauss.pdf

[32] D. Liu and F. Kubala, "Online speaker clustering," in *IEEE Int. Conf. on Acoustics, Speech, and Signal Processing*, vol. 1, 2004, pp. 333–336.

[33] R. O. Duda, P. E. Hart *et al.*, *Pattern classification and scene analysis*. Wiley New York, 1973, vol. 3.