Mälardalen University Press Licentiate Theses
No. 325

TIMING AND SCHEDULABILITY ANALYSIS OF REAL-TIME SYSTEMS USING HIDDEN MARKOV MODELS

Anna Friebe

2022

School of Innovation, Design and Engineering

ISBN 978-91-7485-555-5
ISSN 1651-9256
Printed by E-Print AB, Stockholm, Sweden

Till Monica och Robert.

Andas lugnt… En okänd blå materia är fastnaglad vid stolarna.
Guldnitarna flög in med oerhörd hastighet
och tvärstannade
som om de aldrig varit annat än stillhet.

Tomas Tranströmer

Acknowledgements

I would like to express gratitude to my supervisors Thomas Nolte, Alessandro Papadopoulos, and Filip Marković. Thank you for fruitful research discussions, for your invaluable input on this journey, and for your kind patience with my blunders and frustration. 5/5, I can warmly recommend these supervisors to any aspiring PhD student in the field. I want to thank Björn Lisper for reviewing the licentiate thesis and proposal, and thank the anonymous reviewers of the included papers for their feedback.

The work in this thesis was supported by the Swedish Research Council (Vetenskapsrådet) via the project "Practical Probabilistic Timing Analysis of Real-Time Systems (PARIS)". Thank you for providing an opportunity to explore this research direction. I also wish to thank the anonymous reviewers for helpful feedback on the submitted papers.

Thank you all co-workers at IDT, for the warm and open atmosphere, for all discussions at lunches and fikas.

Most of all, I would like to thank my family for your love and support. My children Rasmus and Ida, my husband Jimmy, my brothers Jonas and Anders. Last but not least I want to thank my parents Monica and Robert, who always supported me and to whom I dedicate this thesis. Tack för allt, jag älskar er!

Anna Friebe
Västerås, April 2022

Abstract

In real-time systems functional requirements are coupled to timing requirements, a specified event needs to occur at the appropriate time.

In order to ensure that timing requirements are fulfilled, there are two main approaches, static and measurement-based. The static approach relies on modeling the hardware and software and calculating upper bounds for the timing behavior. On the other hand, measurement-based approaches use timing data collected from the system to estimate the timing behavior.

The usability of static and measurement-based approaches is limited in many modern systems due to the increased complexity of hardware and software architectures. Static approaches to timing and schedulability analysis are often infeasible due to their complexity. Measurement-based approaches require that design-time measurements are representative of the timing behavior at run-time, which is problematic to ensure in many cases. Designing systems that guarantee the timing requirements without excessive resource overprovisioning is a challenge.

A Hidden Markov Model (HMM) describes a system where the behavior is state-dependent. In this thesis, we model the execution time distribution of a periodic task as an HMM where the states are associated with continuous emission distributions. By modeling the execution times in this manner with a limited number of parameters, a step is taken on the path toward tracking and controlling timing properties at runtime.

We present a framework for parameter identification of an HMM with Gaussian emission distributions from timing traces, and validation of the identified models. In evaluated cases, the parameterized models are valid in relation to timing traces.

For cases where design-time measurements are not representative of the system at runtime we present a method for the online adaptive update of the emission distributions of an HMM. Evaluation with synthetic data shows that the estimate tracks the ground truth distribution.

A method for estimating the deadline miss probability for a task with execution times modeled by an HMM with Gaussian emission distributions, in a Constant Bandwidth Server (CBS), is proposed. The method is evaluated with simulation and for a synthetic task with a known Markov Chain structure running on real hardware.

Sammanfattning

I realtidssystem är funktionella krav kopplade till tidskrav – en viss händelse måste inträffa vid rätt tid. För att försäkra sig om att tidskrav är uppfyllda finns två huvudsakliga metoder – statisk eller mätningsbaserad. En statisk analys baseras på modeller av hårdvara och mjukvara, och beräknar en övre gräns för tidsbeteendet. Mätningsbaserade analyser använder insamlat data från systemet för att uppskatta tidsbeteendet. Användbarheten av både statiska och mätningsbaserade metoder är begränsad i många moderna system eftersom komplexiteten hos hårdvara och mjukvara ökat. Statiska metoder är ofta omöjliga att genomföra på grund av komplexiteten. För mätningsbaserade metoder krävs att mätningarna som insamlats vid design är representativa för tidsbeteendet i drift, vilket är svårt att garantera i många fall. Att designa system som garanterar tidskraven utan överdriven resurstilldelning är en utmaning. En Hidden Markov Model (HMM) beskriver ett system med beteende som är tillståndsberoende. I denna avhandling modellerar vi exekveringstidens fördelning hos en periodisk task (uppgift) som en HMM där tillstånden är kopplade till kontinuerliga emissionsfördelningar. Genom att modellera exekveringstiderna på detta vis med ett begränsat antal parametrar, tar vi ett steg på vägen mot att följa och kontrollera tidsbeteendet i drift. Vi presenterar ett ramverk för parameteridentifiering för en HMM med Gaussiska emissionsfördelningar från tidsdata, och validering av de identifierade modellerna. De parametriserade modellerna är giltiga i relation till tidsdata i de fall som utvärderats. För fall när mätningar vid design inte är representativa för systemet i drift presenterar vi en metod för direkt adaptiv uppdatering av emissionsfördelningarna i en HMM. Utvärdering med syntetiska data visar att uppskattningen följer den sanna fördelningen. En metod föreslås för att uppskatta sannolikheten för att missa en deadline när exekveringstiden modelleras som en HMM med Gaussiska emissionsfördelningar hos en task i en Constant Bandwidth Server (CBS). Metoden utvärderas med simulering och med syntetiska program med känd Markov-struktur som körs på verklig hårdvara.

List of Publications

Papers Included in This Thesis[1]

Paper A: Anna Friebe, Alessandro V. Papadopoulos, Thomas Nolte. *Identification and Validation of Markov Models with Continuous Emission Distributions for Execution Times.* In IEEE International Conference on Embedded and Real-Time Computing Systems and Applications (RTCSA), 2020.

Paper B: Anna Friebe, Filip Marković, Alessandro V. Papadopoulos, Thomas Nolte. *Adaptive Runtime Estimate of Task Execution Times using Bayesian Modeling.* In IEEE International Conference on Embedded and Real-Time Computing Systems and Applications (RTCSA), 2021.

Paper C: Anna Friebe, Filip Marković, Alessandro V. Papadopoulos, Thomas Nolte. *Estimating Deadline Miss Probabilities of Continuous-Emission Markov Chain Tasks* Under review.

[1]The included papers have been reformatted to comply with the thesis layout.

Other Relevant Publications[2]

Paper X: Jonathan Thörn, Najda Vidimlic, Anna Friebe, Alessandro V. Papadopoulos, and Thomas Nolte. *Work-In-Progress: Probabilistic Timing Analysis of a Periodic Task on a Microcontroller* In IEEE International Conference on Emerging Technologies and Factory Automation (ETFA), 2019.

Paper Y: Anna Friebe, Alessandro V. Papadopoulos, and Thomas Nolte. *Work-In-Progress: Validation of Probabilistic Timing Models of a Periodic Task with Interference – A Case Study* In IEEE Real-Time systems Symposium (RTSS), 2019.

Paper Z: Manuel F. Silva, Anna Friebe, Benedita Malheiro, Pedro Guedes, Paulo Ferreira, and Matias Waller. *Rigid Wing Sailboats: A State of the Art Survey* In Ocean Engineering, 2019.

[2]Not included in this thesis.

Contents

I

Thesis

Chapter 1

Introduction

In real-time systems, we must consider requirements on timing properties, in addition to functional requirements. It is of importance to have the correct behavior *at the appropriate time*.

In Safety-Critical Systems (SCS), such as a car or a radiation treatment system, malfunction can lead to loss of life or serious injury. The functionality of for example braking, steering, and radiation delivery systems needs to be computed correctly and the result of the computation needs to be available at a certain time. The brake needs to be applied timely after the pedal is pushed. Likewise, the radiation needs to be delivered to the tissue for an exact amount of time - to cause damage to the tumor, but allow the surrounding tissue to repair. In *hard real-time systems*, failures to meet timing requirements, e.g., deadline misses, are considered to be system failures. The notion of hard real-time systems applies to critical timing requirements in SCS.

There are also systems where timing requirements affect the operation in less critical manners. For example, in video streaming or other multimedia applications, failures to meet a deadline can cause a deterioration of the user experience or Quality of Service (QoS) [20]. In robotic or process control systems, failures to meet a deadline can affect the Quality of Control (QoC) [49]. These systems are referred to as *soft* real-time systems, and these are the systems we consider in our research.

1.1 Motivation

In timing analysis, the execution times of tasks or programs are mathematically modeled and analyzed. Schedulability analysis refers to analysis of the response time of functionality that may involve one or several tasks. In conventional timing and schedulability analysis, the Worst-Case Execution Time (WCET) [69], and Worst-Case Response Time (WCRT) of tasks are estimated or upper bounded. Static Analysis, Dynamic/ Measurement-Based Analysis, or Hybrid Analysis are utilized to retrieve the estimates or bounds. The results are used to ensure that the system processor resources are sufficient and that all deadlines will be met.

Today's complex systems are equipped with hardware to increase the computation speed. Examples of such hardware are pipelines, branch prediction, out-of-order execution, and caches or scratchpads. In addition, today's systems often have multiple cores and allow for more complex software and mixed-criticality systems. Introduction of these acceleration features generally decreases the average execution and response times, but does not necessarily have a large impact on the worst-case. This causes additional variation in execution times and response times, and contributes to difficulties in achieving tight bounds on the WCET and WCRT using conventional analysis techniques [42, 68, 13]. The complex systems also make it increasingly difficult to ensure and prove that measurement-based and hybrid techniques provide valid results [26], as the measurements need to be representative of the execution times at run-time. These issues may lead to significant resource overprovisioning when designing systems, alternatively a situation where there are no guarantees that timing requirements can be met.

One path forward is an adaptive approach where the system has several QoS levels. For the lowest QoS level, guarantees on timing requirements are derived. Operation at higher QoS levels is permitted, as long as some run-time constraints on timing requirements are fulfilled. For example, a robotic arm may be designed to move slowly in an environment with moving obstacles, but allowed to proceed quickly when the space is clear.

1.2 Probabilistic Timing Analysis

In many cases, probabilistic guarantees of distribution properties may be sufficient, i.e., that the deadline miss probability is sufficiently small, or some other requirement on the distribution of deadline misses that can guarantee the QoS. In a recent survey of 120 industry practitioners in real-time systems [6], only 15% answered that the most time-critical component in their systems can never miss a deadline.

In probabilistic analysis, instead of considering one run of the program - the one with the longest execution time - we consider a sequence of many runs with feasible input values and initial hardware states. The scalar WCET - the upper bound of the worst-case execution time of runs - is replaced with a probability distribution, the probabilistic Worst-Case Execution Time (pWCET). This name opens for misinterpretation, a more suitable name would be probabilistic Execution Time Exceedance Bound. An illustration is given in Figure 1.1. The x-axis represents execution time, and for each execution time et, $0 \leq et \leq WCET$, the distribution is an upper bound on the probability of exceeding et for every valid scenario. A scenario is a sequence of executions of the program that consists of a feasible repeated execution of the program. While the WCET upper bounds all possible execution times for the program, the pWCET upper bounds all possible execution time distributions for the program for all valid scenarios. The pWCET upper bounds all execution time distributions as visualized in Figure 1.1, and using the definition of upper bound from Davis and Cucu-Grosjean [26] based on Diaz et al. [28]:

Definition 1.2.1. *The probability distribution of a random variable X is greater than or equal to (i.e. upper bounds) that of another random variable Y (denoted by $X \geq Y$) if the Cumulative Distribution Function (CDF) of X is never above that of Y, $F_X(t) \leq F_Y(t)$ for all t. Alternatively, X is greater than or equal to Y if the 1-CDF of X is never below that of Y, $1 - F_X(t) \geq 1 - F_Y(t)$, for all t.*

Many probabilistic analysis methods assume that execution times are independent and identically distributed. For estimates of the tail of the probabilistic Worst-Case Execution Times pWCET based on Extreme Value Theory, it is sufficient that there is stationarity [36] or that extreme values of the distribution are independent [61].

In many cases, such as in soft real-time systems, a picture of the entire execution time distribution is useful. The pWCET distribution may be very pessimistic in relation to average execution time distributions. An assumption that execution times are independent and identically distributed, i.i.d., may not always be valid. Frías et al. [31] have shown that in some robotics applications, such as image processing for lane detection, a Markov Chain model of the execution times is more appropriate. Here, within a part of the track, images to be processed are similar to each other, which causes execution times to be similar for these jobs.

A Markov Chain model may also be appropriate when the execution times vary in a time-dependent manner due to other reasons than input varying with some regularity. This can be for example due to a regular change in the internal state of the program, or the system state being affected directly or indirectly by other processes in a recurrent manner.

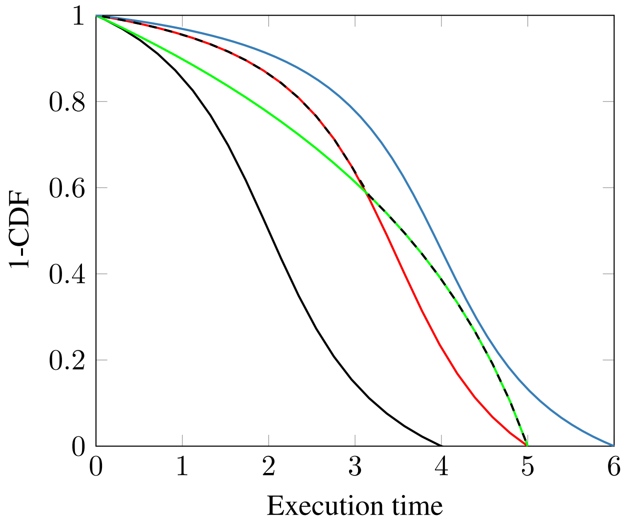


Figure 1.1: 1-CDF of a program's execution time under three scenarios are shown in black, red, and green. The precise pWCET distribution as the tight upper bound on these distributions is dashed, and an upper bounding pWCET is displayed in blue. The figure is inspired by Figure 2 in [26].

1.3 Problem Formulation

It is a challenge to enable sufficiently accurate timing and schedulability analysis of today's complex systems. Multicore [42] and mixed-criticality systems [68, 13], as well as fog and edge computing [64, 63, 22], require new methods for ensuring sound timing and schedulability estimates along with functional integrity and limited overprovisioning.

Measurement-based probabilistic timing analysis [14, 30, 32, 23] is a practical approach to estimating the pWCET distribution based on samples from the tail of the distribution. This approach is safe given some assumptions are fulfilled. Given that only extreme values from the tail are used to estimate the pWCET, in many cases, this estimate will be pessimistic compared to common scenarios. This is needed in safety-critical systems where standards require a specific bound on the deadline miss probability. However, methods that consider the entire timing distribution can allow for system design without excessive overprovisioning. These kinds of approaches are of particular interest in the case of soft real-time applications where requirements on QoS or QoC are considered.

Timing and schedulability analysis of systems where execution time distributions change irregularly over time due to input or system state changes is another challenge. Measurement-based probabilistic timing analysis based on Extreme Value Theory requires that the sampled distribution is stationary or that extreme values are independent [61]. In addition, measurement-based analysis requires that the measurements used at analysis include observations from the worst-case timing behavior of the system at run-time.

Chapter 2

Background and Related Work

In this section, background and related work are provided on real-time systems, Hidden Markov Models (HMMs) and probabilistic timing analysis.

2.1 Task Model

A task is a concept used to represent a piece of software - a program or a thread. A task consists of one or more jobs. In this thesis the task model is periodic, that is, the jobs j_i of a task τ have arrival times a_i separated by a period T. The arrival time a_{i+1} of job j_{i+1} can be written as $a_{i+1} = a_i + T$ when we assume no jitter in the arrival times.

2.2 Real-Time Systems

In a *hard* real-time system there is a requirement that no job misses its deadline. In *soft* and *firm* real-time systems on the other hand, some deadlines can be missed. The result of a job that missed its deadline is still useful in a soft real-time system, although the result may be less valuable than a result delivered on time. In a firm real-time system, a result delivered after a job's deadline is of no use [16]. The concept of weakly hard real-time systems has been introduced, where bounds on the amount of missed and met deadlines during a window of time are formally specified. [9]. In this thesis we are concerned with soft real-time systems, although some of the methods derived may be useful also in firm and weakly hard contexts.

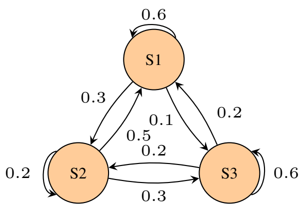


Figure 2.1: An example of a three state Markov Model with transition probabilities.

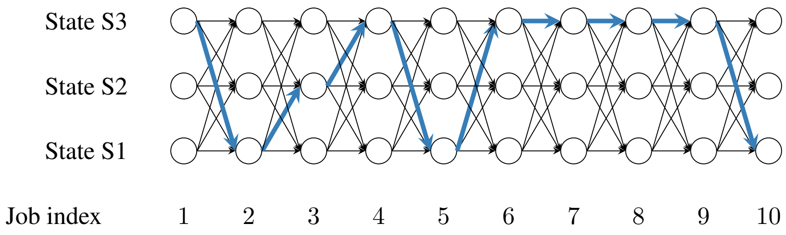


Figure 2.2: A possible state sequence from the Markov Model in Figure 2.1.

2.3 Hidden Markov Models

A Markov Model is a system characterized by a number of states and probabilities for transitioning between the states. The current state depends only on the state in the previous step and transition probabilities, not on events that occurred earlier. This lack of memory is referred to as the Markov property. An example of a three-state Markov Model with transition probabilities is shown in Figure 2.1.

In a Hidden Markov Model (HMM), the model state is not directly observed, instead we can observe an outcome that depends on the model state. As an example, we assume that a task in a real-time system releases jobs sequentially. The task is associated with different states, that may be dependent on the program state or interference from other tasks. Each state has a certain probability distribution of execution times, that we refer to as the emission distribution. We can observe the sequential execution times of the task's jobs, but not the underlying states directly. One possible state sequence of the Markov Model in Figure 2.1 can be the sequence shown in Figure 2.2. A possible execution time sequence associated with this state sequence is seen in Figure 2.3. In this example each state is associated with a Gaussian emission distribution, with mean and standard deviation of $S1$ as 20 and 2, of $S2$ as 30 and 3, and of $S3$ as 40 and 4.

A tutorial on HMMs with some applications in speech recognition is [58].

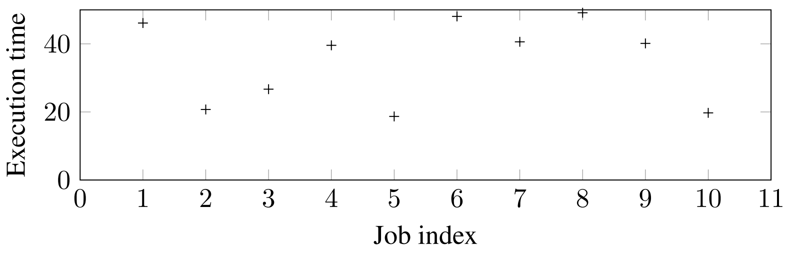


Figure 2.3: A possible execution time sequence from the state sequence in Figure 2.2.

2.4 Probabilistic Timing Analysis

A thorough overview of probabilistic timing analysis techniques is provided by Davis and Cucu-Grosjean in [26]. A recent taxonomy and survey on Probabilistic Worst-Case Execution Time analysis is provided by Cazorla et al. [17].

2.4.1 Static Probabilistic Timing Analysis

Work in this area applies static analysis tools on the code and a model of the hardware timing behavior of the platform to obtain the pWCET distribution. The majority of the work here considers models of set-associative and fully associative random replacement caches. Quinones et al. [57] showed that for some cases with programs displaying a cache risk pattern, random replacement gives better results and lower variability over memory layouts compared to Least Recently Used (LRU) replacement. Altmeyer et al. [7] provide an analysis of randomized caches, considering reuse distance, associativity and contention. They show that random replacement is preferable when the number of accessed memory blocks in a loop is larger than the cache associativity, and LRU otherwise. Analysis using random replacement caches has been extended to the multi-path case [39, 38]. Chen and Beltrame [18] perform timing analysis of a system with a random replacement cache by using an adaptive Markov model. In addition to static probabilistic timing analysis with random replacement caches, there is also work related to input and branch probabilities, and probabilities of faults.

2.4.2 Measurement-Based Probabilistic Timing Analysis

Methods in this area estimate the pWCET by applying statistical techniques to observations of execution time measurements. The theoretical basis is in Extreme Value Theory (EVT).

Burns, Edgar, and Griffin initiated this research direction in several papers [14, 30, 32]. Measurement-Based Probabilistic Timing Analysis was then introduced by Cucu-Grosjean et al. [23] in 2012, as a statistically sound method based on Extreme Value Theory to estimate the pWCET from execution time measurements.

Santinelli et al. [61] point to earlier work by Leadbetter et al. [36] to show that stationarity and extremal independence is sufficient for the application of EVT and that independent and identically distributed observations are not required.

2.5 Probabilistic Schedulability Analysis

A thorough overview of probabilistic schedulability analysis techniques is provided by Davis and Cucu-Grosjean in [25], accompanying the survey on timing analysis mentioned above.

2.5.1 Probabilistic Response Time Analysis

Probabilistic Response Time Analysis is used to calculate the response time distribution of jobs, and in this manner estimate the probability of a deadline miss.

Diaz et al. in 2002 [27] presented response time analysis for a system with periodic tasks where random variables describe execution times. Here, the worst-case processor utilization can exceed 1, since a backlog is considered at the end of the hyperperiod. They show that the backlog is a Markov chain. In 2004 they also provided properties needed to achieve a safe over-approximation [28].

Ivers and Ernst [34] in 2009 showed that when execution times are dependent, it is not sufficient to use the per-program pWCET distributions in convolution to obtain a safe over-estimate of the response time distribution. Instead, they propose a method with probability boxes, bounding the response time distribution in presence of unknown dependencies.

In 2013 Maxim and Cucu-Grosjean [50] proved that for fixed priority, constrained deadline tasks and preemptive scheduling, and provided jobs are aborted when their deadline is missed, the pWCRT distribution can be found by synchronous release of the first jobs of each task. Here, execution times, deadlines, and interarrival times are independent random variables.

Most work in this direction uses convolutions in the response time analysis. The worst case computational complexity of a single discrete linear convolution is quadratic. Taking a reasonable number of tasks into account causes state explosion and tractability issues. Downsampling reduces the number of values of the discrete distribution used in the calculations and improves tractability at the expense of pessimism. Random sampling was proposed by Refaat et al. [59]. Maxim et al. [52] compared downsampling by uniform spacing, domain quantization and a downsampling method reducing pessimism. Marković et al. [48] presented a method for optimal pessimism-minimizing downsampling, and a downsampling algorithm by uniform spacing in the probability domain. Analytical bounds [19, 66] are more efficient compared to convolution-based methods, but add more pessimism. Von der Brüggen et al. have proposed methods for approximating deadline miss probabilities efficiently by using multinomial distributions [66]. The same group have proposed a method for approximating the Worst Case Deadline Failure Probability (WCDFP), that is an upper bound on the probability of any job of a task to miss its deadline, considering dependencies among a bounded number of jobs [67].

In 2021, Bozhko et al. applied Monte Carlo simulation to estimate the WCDFP [10] for fixed-priority preemptive scheduling, execution times being independent random variables.

2.5.2 Statistical Response Time Analysis

Similarly as in Measurement-Based Probabilistic Timing Analysis for pWCET estimates, Extreme Value Theory, EVT, has also been applied in order to estimate the response time distributions. The most important work in this line of research has been performed by Lu et al. [46, 45, 43, 44]. Maxim et al. have shown that the application of EVT gives sound over-estimates, where fitting distributions to the observations directly does not [53].

2.5.3 Probabilistic Analysis of Server-based Systems

Abeni et al. have published a significant amount of work related to server-based systems. In a server-based system, temporal isolation of tasks is achieved by partitioning the processor resource. The Constant Bandwidth Server (CBS) was introduced in 1998 [1], and probabilistic deadlines for Quality of Service guarantees were introduced in 1999 [2], both by Abeni and Buttazzo. The same group has considered execution times [3, 56] and interarrival times [5, 47, 56] modeled with probability distributions.

Mills and Anderson [54] analyze sporadic tasks with stochastic execution times under a server-based scheduler. Tardiness and response time bounds are derived, considering dependencies within but not across time windows.

Bounds grow with increased time windows. Liu, Mills and Anderson proposed an alternative use of independence thresholds [41], where execution times above a certain threshold are independent.

Frías et al. [31, 4] have published work regarding execution time models for tasks where execution times display dependencies due to slowly changing input data. They have shown that for a robotic image processing task in line following, modeling execution times as a Hidden Markov Model is appropriate. In this work, discrete emission distributions for the different states are used. That is, each state is associated with a probability distribution of execution times, with a finite number of execution times/ number of clock cycles having certain probabilities. The deadline miss probability under CBS is estimated for the Hidden Markov Model and compared to an i.i.d. case. The calculated probabilities are compared to experimental results with CBS as implemented in the Linux SCHED_DEADLINE scheduling policy. The experiments show that with an i.i.d. assumption of execution times, the probability of respecting the deadline is overestimated, i.e. the estimate is optimistic. The estimates based on the identified Hidden Markov Model, on the other hand, are very close to the experimental results. The software tool PROSIT has been developed to enable these analysis methods [65].

2.5.4 Real-Time Queueing Theory

Real-Time Queueing Theory is an area where queue lengths and lead-times, the remaining time until the deadline, are analyzed mathematically. Lehozcky [37] introduced the concept in 1996, building upon work on queuing theory that started in the 1950s. For systems with high utilization, Real-Time Queueing Theory can be applied to calculate the lead-time process for tasks in the queue under specific queueing/ scheduling policies. This information can be used to calculate the deadline miss probability. The work was mathematically formalized in 2001 [29]. Kruk et al. provided a similar analysis for systems, where jobs are discarded when their deadline is missed [35]. They show that Earliest Deadline First (EDF) scheduling minimizes the amount of discarded work. The amount of missed deadlines 1-2 orders of magnitude lower when discarding work at deadlines given by analysis and supported by simulations. In Real-Time Queueing Theory arrival times, computation times and deadlines are independent random variables.

2.5.5 Probabilities from Faults

Schedulability analysis accounting for probabilities of faults and recovery computations have been performed, this line of research was initiated by Burns [15]. There has been a significant amount of work-related to faults and recovery in the context of the Controller Area Network (CAN) [55, 11, 12, 24, 8].

2.5.6 Mixed-Criticality Systems

Some work has recently been done on applying probabilistic methods in the analysis of Mixed-Criticality Systems, for example by Maxim et al. [51]. The interest in this area is increasing.

2.6 Relation to the Work Presented in this Thesis

The work in this thesis builds particularly upon the work of Frías et al. [31, 4]. The discrete execution time distributions related to the states of the HMM in this related work are replaced with continuous, Gaussian distributions in the work presented in this thesis. The reasons for selecting this distribution are mainly the efficient representation, simplicity and tractability. Due to these factors it is likely better suited for adaptive approaches compared to a discrete distribution. In our work, a more automated method for determining a suitable number of states is provided. In addition, we present an adaptive approach, that allows for incorporating irregular changes to the execution time distributions into the model. We also propose a method for estimating the deadline miss probability from a task described by an HMM with Gaussian execution time distributions under CBS.

Chapter 3

Research Goals

The research goals are distilled from the problem formulation. This regards the need for new practical approaches to timing analysis and the observation that the entire execution time distribution is of interest for applications with QoS or QoC concerns. It also includes the need to address systems where there can be irregular changes in the execution time distribution due to changes in input or system state, and that measurements used for analysis are not always fully representative of the system at runtime.

The main research goal is to find new practical approaches to timing analysis in applications with QoS or QoC concerns, by modeling the entire execution time distribution using a suitable probabilistic framework and utilizing this framework for schedulability analysis. More specifically, the individual research goals are:

- **RG1**: To model the execution time distribution of a periodic task as an HMM with continuous emission distributions and propose a method for parameter identification for the HMM using execution time traces.
- **RG2**: To define a method for empirical validation of the task execution time HMM, using execution time traces.
- **RG3**: To propose a method for the online update of the emission distributions of the task execution time HMM to enable adaptive models in dynamic systems.
- **RG4**: To estimate the deadline miss probability under reservation-based scheduling of a periodic task with execution times modeled by an HMM with continuous emission distributions.

Chapter 4

Research Process and Methods

The main motivation of the work is to move towards practically useful timing analysis for systems with QoS or QoC concerns. While probabilistic analysis methods give a thorough theoretical underpinning to the work, the use of timing measurements from tasks running on real systems connects theory to the real world.

An illustration of the research process is shown in Figure 4.1. The figure is adapted from Holz et al. [33]. A problem is identified and formulated. The evaluation strategy is outlined, and data is collected and/ or generated. The constructive step is based upon theoretical grounds from related work and theoretical proofs in applicable cases. Collected data can be utilized in the constructive step where methods are developed and models parameterized or adapted, as well as in the evaluation step. Depending on the problem, evaluation is performed with data from simulation, synthetic test programs on real systems, realistic programs, or a triangulation using several of these alternatives. The iterative nature of the process is shown, where the evaluation of the research in relation to one research goal gives rise to new problem formulations and research goals.

The basis of the research is a realist constructivist approach [21], where knowledge is created in interaction with real-world observations.

Evaluation is performed as case studies, according to the steps outlined by Runesson and Höst [60].

1. Case study design, definition of objectives, and planning of the case study.

2. Preparation for data collection.

3. Collecting evidence.

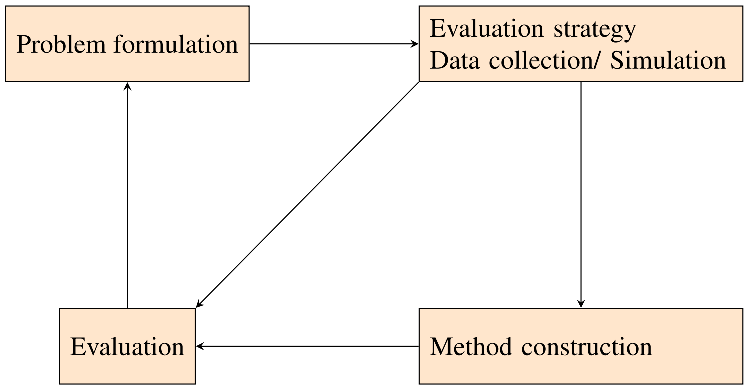


Figure 4.1: The main steps of the research process.

4. Analysis of collected data.

5. Reporting.

The objectives of the evaluations depend on the problem in question. In Paper A, a data consistency approach [40] is applied to evaluate the identified, parameterized model and compare data generated from the model with data collected from synthetic test programs and a realistic program. In Paper B simulated data is used in the evaluation, and the adapted distribution is compared to the ground truth using the Kullback-Leibler divergence. For evaluation of Paper C, triangulation is used with two different synthetic test programs and simulation.

4.1 Threats to Validity

We consider the validity types discussed in Wohlin et al. [70], namely **construct validity**, **internal validity**, **external validity** and **reliability**. Regarding **construct validity**, concepts such as execution time and response time are well defined and can be appropriately measured. It is worth noting that the validity of predictions will need to be defined clearly, and be interpreted under given conditions. On the matter of **internal validity**, there is a possibility that tracing affects the execution time and response time measurement through cache effects. This possibility needs to be taken into account when drawing conclusions on causality. Theoretical proofs are provided for the method presented in Paper C, under certain assumptions, and not in Paper A or B. The **external validity** is limited to types of applications and platforms fulfilling the assumptions or utilized in the evaluations. Any generalizations need to be made with caution. By providing software with the papers, we increase the reliability of our research. We increase the possibility for other researchers to apply the methods, and also to spot weaknesses and errors.

Chapter 5

Thesis Contributions

In this section, the contributions and included papers are described. An overview of the connections of the research goals to the papers, with an illustration of the process, is provided in Figure 5.1.

5.1 Contributions

The thesis contributions are described here and mapped to the research goals in Table 5.1.

- C1: A framework for model identification and validation of execution time HMMs with continuous emission distribution models from execution time sequences, including automatically determining a suitable number of states.
- C2: A method for online adaptation of execution time HMM emission distributions from new timing measurements.
- C3: A method for estimating deadline miss probabilities for periodic tasks with execution times modeled by HMMs with Gaussian emission distributions, in a CBS.

C1: A Framework for Model Identification and Validation of Execution Time HMMs with Continuous Emission Distribution Models from Execution Time Sequences, Including Automatically Determining a Suitable Number of States.

As a first step, we evaluate the validity of a HMM with continuous, Gaussian emission distributions. This contribution is provided in Paper A. A framework

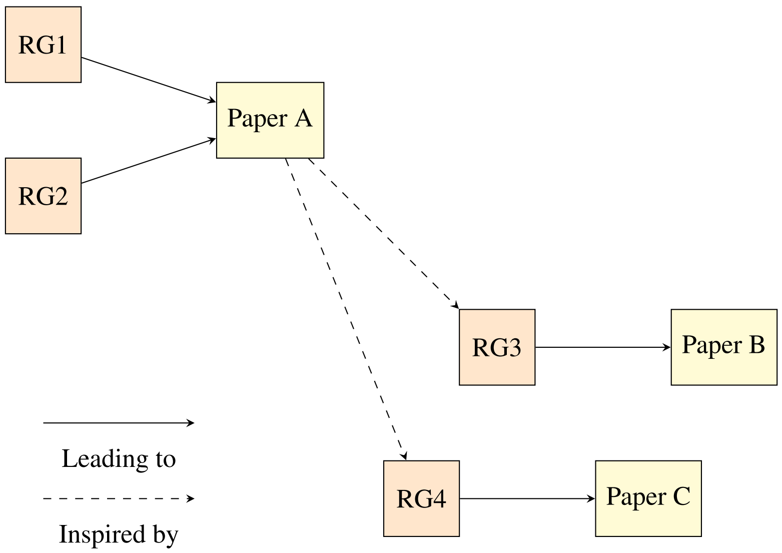


Figure 5.1: Illustration of the connection between research goals and papers.

Table 5.1: Contributions C1 through C3 address research goals RG1 through RG4 in this way.

	RG1	RG2	RG3	RG4
C1	X	X		
C2			X	
C3				X

for identification of a HMM with Gaussian emission distributions from execution time traces is presented, in order to model the execution times of a periodic task. This includes automatically determining a suitable number of states with a tree-based cross-validation approach [62]. The identified HMMs are validated using a data consistency approach [40], where data generated from the identified model are compared with timing traces using a dispersion based statistic. For the evaluated cases, a simple test program with a known Markov chain structure and a video compression test case, we show that the identified HMMs are valid models with respect to the timing trace observations. This reinforces conclusions from previous work that HMMs can be valid models for execution times [31, 4], and extends it to continuous emission distributions. Source code and data are available[1]. Preliminary tests show that the identified model in the video decompression test case is not valid for different video in-

[1]https://github.com/annafriebe/MarkovChainETFramework

put. This is consistent with results from [31, 4] that indicate that different input give rise to different HMMs and tasks with different resource requirements.

C2: A Method for Online Adaptation of Execution Time HMM Emission Distributions from new Timing Measurements.

As indicated in Paper A and previously in [31, 4], different input can give rise to different HMMs. This is likely also the case with other causes of irregular variation in the execution times, such as for example some interference from other processes. Since the methods for identification of the HMMs are not suited for runtime purposes, we propose online adaptation of the emission distributions as outlined in Paper B. We assume that the number of states and the transition probabilities are not changing. We also assume that the emission distributions change at some time points, but remain the same in segments of the execution time sequence between those points of change. Segments can have the same or similar emission distributions, and in this case they are combined into a cluster. The number of states and the transition matrix are identified in a preprocessing step, along with a number of segments and clusters of a preprocessing part of the execution time sequence. In an adaptive process, points of model change are detected, and new segments are identified. New segments are determined to be part of an existing cluster or starting a new cluster. A Bayesian approach is used to model the probability distribution of the mean and variance of each segment. From the Bayesian model a Generalized Likelihood Ratio (GLR) is defined, that gives the probability that the observations of two segments or clusters belong to a joint distribution as opposed to distinct distributions. This GLR measure is central and used to find the points of model change and to combine segments into clusters in both the preprocessing step and the adaptive process. The evaluation in Paper B is performed with synthetic data. This gives us a ground truth distribution that can be used for comparison with the posterior predictive distribution. It also ensures that the assumptions are met - that the number of states and the transition matrix remain the same, and that the emission distributions of the states change at certain points in time. The comparison is performed by calculating the Kullback-Leibler (KL) divergence from the estimated posterior predictive distribution to the ground truth distribution. Source code and data are available[2].

[2]https://github.com/annafriebe/AdaptiveETBayes

C3: A Method for Estimating Deadline Miss Probabilities for Periodic Tasks with Execution Times Modeled by HMMs with Gaussian Emission Distributions, in a CBS.

We want to use the identified HMM of the execution times to estimate the workload distribution and deadline miss probabilities. These estimates are needed for schedulability analysis, and for decisions on resource assignments and adaptation. Therefore we propose a method for estimating the deadline miss probabilities for these tasks in a CBS, as described in Paper C. In the proposed method, a workload accumulation scheme is considered from a point of workload depletion up until a certain number of task periods, N. The deadline miss probability along each workload accumulation sequence up until length N is upper bounded, and the probability of taking each of these is also upper bounded. Along with an upper bound of the probability of accumulation sequences longer than N, a bound on the deadline miss probability is constructed. As an alternative to bounding the probability of longer accumulation sequences, a method for estimating this is provided. Workload accumulation sequences with the same number of visits in all states are combined, thus reducing the complexity. The number of task periods included, N is increased up until a maximum number, or until the workload depletion bounds deteriorate. Accumulation sequences are used to estimate the the deadline miss probabilities of each state and overall. Theoretical proofs are provided that an upper bound of the deadline miss probability is obtained in the case of Gaussian emission distributions, and with a bound on the probability of carrying over workload into the second period of workload accumulation.

Evaluation is performed for two simple test programs with the same transition probabilities, one with Gaussian emission distributions and one with exponential emission distributions. Timing traces are retrieved and used for estimating means and standard deviations of the Gaussian emission distribution of the model. The transition probabilities are known. The deadline miss probability bounds and estimates are compared with simulation and with the deadline miss ratio when running the program in the Linux CBS implementation. The bound adds significant pessimism compared to the estimates. In all cases with Gaussian emission distributions, the estimates also upper bound the results from simulation and experiments. The bounds and estimates are tighter for the state with the highest deadline miss probability, and tighter for cases with lower utilization and shorter relative deadlines. In one evaluated test case with exponential distributions, the resulting estimate is optimistic for the state with the highest deadline miss probability. The overall deadline miss probability estimate still upper bounds the result of simulation and experiments.

Source code and data will be made available upon acceptance of the paper.

5.2 Overview of Included Papers

In this section, abstracts and contributions of the papers included in the thesis are listed. The papers are mapped to the contributions in Table 5.2.

Table 5.2: Papers A through C align with contributions C1 through C3 in this way.

	C1	C2	C3
Paper A	X		
Paper B		X	
Paper C			X

5.2.1 Paper A

Title: Identification and Validation of Markov Models with Continuous Emission Distributions for Execution Times
Authors: Anna Friebe, Alessandro V. Papadopoulos, and Thomas Nolte
Status: Published at RTCSA 2020.
Abstract: It has been shown that in some robotic applications, where the execution times cannot be assumed to be independent and identically distributed, a Markov Chain with discrete emission distributions can be an appropriate model. In this paper, we investigate whether execution times can be modeled as a Markov Chain with continuous Gaussian emission distributions. The main advantage of this approach is that the concept of distance is naturally incorporated. We propose a framework based on Hidden Markov Model (HMM) methods that 1) identifies the number of states in the Markov Model from observations and fits the Markov Model to observations, and 2) validates the proposed model with respect to observations. Specifically, we apply a tree-based cross-validation approach to automatically find a suitable number of states in the Markov model. The estimated models are validated against observations, using a data consistency approach based on log-likelihood distributions under the proposed model. The framework is evaluated using two test cases executed on a Raspberry Pi Model 3B+ single-board computer running Arch Linux ARM patched with PREEMPT_RT. The first is a simple test program where execution times intentionally vary according to a Markov model, and the second is a video decompression using the `ffmpeg` program. The results show that in

these cases the framework identifies Markov Chains with Gaussian emission distributions that are valid models with respect to the observations.
Personal Contributions I have been the main author and driver of the work. Planning of the paper and evaluation has been performed jointly with the co-authors. I have developed the software, performed experiments and written the manuscript draft that has been improved in collaboration with the co-authors.

5.2.2 Paper B

Title: Adaptive Runtime Estimate of Task Execution Times using Bayesian Modeling
Authors: Anna Friebe, Filip Marković, Alessandro V. Papadopoulos, and Thomas Nolte
Status: Published at RTCSA 2021.
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.
Personal Contributions: I have been the main author and driver of the work. Planning of the paper and evaluation has been performed with the co-authors. I have developed the software, performed the experiments and written the

draft of the paper that has been improved in collaboration with the co-authors.

5.2.3 Paper C

Title: Estimating Deadline Miss Probabilities of Continuous-Emission Markov Chain Tasks.
Authors: Anna Friebe, Filip Marković, Alessandro V. Papadopoulos, and Thomas Nolte
Status: Under review.
Abstract: Estimating the response times of real-time tasks and applications is important for the analysis and implementation of real-time systems. Probabilistic approaches have gained attention over the past decade, as they provide a modeling framework that allows for less pessimism in the analysis of real-time systems. Among the different proposed approaches, Markov chains have been shown to be effective for the analysis of real-time systems, in particular, in the estimate of the pending workload probability distribution and of the deadline miss probability. However, this has been analyzed only for discrete emission distributions, but not for continuous ones. In this paper, we propose a method for analyzing the workload probability distribution and bounding the deadline miss probability for a task executing in a Constant Bandwidth Server, where execution times are described by a Markov model with Gaussian emission distributions. In the evaluation, deadline miss probability bounds and estimates are derived with a workload accumulation scheme. The results are compared to simulation and measured deadline miss ratios from tasks under the Linux Constant Bandwidth Server implementation SCHED_DEADLINE.
Personal Contributions: I have been the main author and driver of the work. Planning of the paper and evaluation has been performed jointly with the co-authors. I have provided the proofs, developed the software, performed the evaluation and written the draft of the paper that has been improved in collaboration with the co-authors.

Chapter 6

Conclusions and Future Work

The main research goal is to find new practical approaches to timing analysis in applications with QoS or QoC concerns, by modeling the entire execution time distribution using a suitable probabilistic framework and utilizing this framework for schedulability analysis.

We have shown that an HMM with Gaussian emission distributions is suitable for modeling the execution times of certain tasks with regular execution time variation, such as a video decompression task. We have proposed a method for estimating the deadline miss probability of such an HMM and a method for runtime adaptation of the emission distributions. The work presented in this thesis provides a step towards the use of HMMs with Gaussian emission distributions to approximate the deadline miss probability in adaptive real-time systems. It should be noted that some links in the chain are not complete in the work presented in this thesis. The posterior emission distributions from the adaptive method are not Gaussian distributions, but Student's t-distributions. Thus it is not suitable for feeding directly into the method for deadline miss estimate. The deadline miss estimate does not take into account emission distributions that change during the accumulation sequence. On the contrary, the main idea relies on emission distributions that remain the same for each state. Thus, the methods proposed in this thesis need to be complemented. One way to close this gap could be to estimate a Gaussian distribution from the Normal-Gamma distributions during the accumulation sequence, that with a sufficient probability overestimates the resulting deadline miss probability.

There is some need for further development of the methods. In particular, regularization should be introduced in the adaptive method to avoid excessive growth of the estimated variance when states' emission distributions are close to each other.

More extensive evaluation of the methods for realistic use cases should be performed. The hypothesis that continuous distributions are more robust with a small number of observations compared to discrete distributions needs to be tested.

Bibliography

[1] Luca Abeni and Giorgio Buttazzo. Integrating multimedia applications in hard real-time systems. In *IEEE Real-Time Systems Symp. (RTSS)*, pages 4–13, 1998.

[2] Luca Abeni and Giorgio Buttazzo. Qos guarantee using probabilistic deadlines. In *Euromicro Conf. on Real-Time Systems (ECRTS)*, pages 242–249, 1999.

[3] Luca Abeni and Giorgio Buttazzo. Stochastic analysis of a reservation based system. In *Int. Workshop on Parallel and Distributed Real-Time Systems*, volume 1, 2001.

[4] Luca Abeni, Daniele Fontanelli, Luigi Palopoli, and Bernardo Villalba Frías. A markovian model for the computation time of real-time applications. In *IEEE international instrumentation and measurement technology conference (I2MTC)*, pages 1–6, 2017.

[5] Luca Abeni, Nicola Manica, and Luigi Palopoli. Efficient and robust probabilistic guarantees for real-time tasks. *Journal of Systems and Software*, 85(5):1147–1156, 2012.

[6] Benny Akesson, Mitra Nasri, Geoffrey Nelissen, Sebastian Altmeyer, and Robert I Davis. A comprehensive survey of industry practice in real-time systems. *Tech. Rep.*, 2020.

[7] Sebastian Altmeyer, Liliana Cucu-Grosjean, and Robert I Davis. Static probabilistic timing analysis for real-time systems using random replacement caches. *Real-Time Systems*, 51(1):77–123, 2015.

[8] Philip Axer, Maurice Sebastian, and Rolf Ernst. Probabilistic response time bound for can messages with arbitrary deadlines. In *2012 Design, Automation & Test in Europe Conference & Exhibition (DATE)*, pages 1114–1117. IEEE, 2012.

[9] Guillem Bernat, Alan Burns, and Albert Liamosi. Weakly hard real-time systems. *IEEE transactions on Computers*, 50(4):308–321, 2001.

[10] Sergey Bozhko, Georg von der Brüggen, and Björn B. Brandenburg. Monte carlo response-time analysis. In *IEEE Real-Time Syst. Symp. (RTSS)*, pages 342–355, 2021.

[11] Ian Broster, Alan Burns, and Guillermo Rodríguez-Navas. Probabilistic analysis of can with faults. In *IEEE Real-Time Syst. Symp. (RTSS)*, pages 269–278. IEEE, 2002.

[12] Ian Broster, Alan Burns, and Guillermo Rodriguez-Navas. Timing analysis of real-time communication under electromagnetic interference. *Real-Time Systems*, 30(1-2):55–81, 2005.

[13] Alan Burns and Robert I. Davis. A survey of research into mixed criticality systems. *ACM Comput. Surv.*, 50(6), 2017.

[14] Alan Burns and Stewart Edgar. Predicting computation time for advanced processor architectures. In *Euromicro Conf. on Real-Time Systems (ECRTS)*, pages 89–96, 2000.

[15] Alan Burns, Sasikumar Punnekkat, Lorenzo Strigini, and David R Wright. *Probabilistic scheduling guarantees for fault-tolerant real-time systems*. IEEE, 1999.

[16] Giorgio C. Buttazzo and Marco Caccamo. Minimizing aperiodic response times in a firm real-time environment. *IEEE Transactions on Software Engineering*, 25(1):22–32, 1999.

[17] Francisco J. Cazorla, Leonidas Kosmidis, Enrico Mezzetti, Carles Hernandez, Jaume Abella, and Tullio Vardanega. Probabilistic worst-case timing analysis: Taxonomy and comprehensive survey. *ACM Computing Surveys (CSUR)*, 52(1), 2019.

[18] Chao Chen and Giovanni Beltrame. An adaptive markov model for the timing analysis of probabilistic caches. *ACM Trans. Design Automation of Electronic Systems (TODAES)*, 23(1):1–24, 2017.

[19] Kuan-Hsun Chen and Jian-Jia Chen. Probabilistic schedulability tests for uniprocessor fixed-priority scheduling under soft errors. In *IEEE Int. Symp. on Industrial and Embedded Systems (SIES)*, pages 1–8. IEEE, 2017.

[20] David D. Clark, Scott Shenker, and Lixia Zhang. Supporting real-time applications in an integrated services packet network: Architecture and mechanism. *SIGCOMM Comput. Commun. Rev.*, 22(4):14–26, 1992.

[21] Gordana Dodig Crnkovic. Constructive research and info-computational knowledge generation. In *Model-Based Reasoning in Science and Technology*, pages 359–380. Springer, 2010.

[22] Tommaso Cucinotta, Luca Abeni, Mauro Marinoni, Alessio Balsini, and Carlo Vitucci. Reducing temporal interference in private clouds through real-time containers. In *IEEE International Conference on Edge Computing (EDGE)*, pages 124–131, 2019.

[23] Liliana Cucu-Grosjean, Luca Santinelli, Michael Houston, Code Lo, Tullio Vardanega, Leonidas Kosmidis, Jaume Abella, Enrico Mezzetti, Eduardo Quiñones, and Francisco J Cazorla. Measurement-Based probabilistic timing analysis for multi-path programs. In *Euromicro Conf. on Real-Time Systems (ECRTS)*, pages 91–101, 2012.

[24] Robert I Davis and Alan Burns. Robust priority assignment for messages on controller area network (can). *Real-Time Systems*, 41(2):152–180, 2009.

[25] Robert Ian Davis and Liliana Cucu-Grosjean. A survey of probabilistic schedulability analysis techniques for Real-Time systems. *LITES: Leibniz Transactions on Embedded Systems*, pages 1–53, 2019.

[26] Robert Ian Davis and Liliana Cucu-Grosjean. A survey of probabilistic timing analysis techniques for Real-Time systems. *LITES: Leibniz Transactions on Embedded Systems*, 6(1):03–1–03:60, 2019.

[27] José Luis Díaz, Daniel F García, Kanghee Kim, Chang-Gun Lee, L Lo Bello, José María López, Sang Lyul Min, and Orazio Mirabella. Stochastic analysis of periodic real-time systems. In *IEEE Real-Time Systems Symp. (RTSS)*, pages 289–300, 2002.

[28] Jose Luis Diaz, Jose Maria Lopez, Manuel Garcia, Antonio M Campos, Kanghee Kim, and Lucia Lo Bello. Pessimism in the stochastic analysis of real-time systems: Concept and applications. In *IEEE Real-Time Systems Symp. (RTSS)*, pages 197–207, 2004.

[29] Bogdan Doytchinov, John Lehoczky, and Steven Shreve. Real-time queues in heavy traffic with earliest-deadline-first queue discipline. *Annals of Applied Probability*, pages 332–378, 2001.

[30] Stewart Edgar and Alan Burns. Statistical analysis of wcet for scheduling. In *IEEE Real-Time Systems Symp. (RTSS)*, pages 215–224, 2001.

[31] Bernardo Villalba Frias, Luigi Palopoli, Luca Abeni, and Daniele Fontanelli. Probabilistic real-time guarantees: There is life beyond the i.i.d. assumption. In *IEEE Real-Time and Embedded Technology and Applications Symposium (RTAS)*, pages 175–186, 2017.

[32] David Griffin and Alan Burns. Realism in statistical analysis of worst case execution times. In *Int. Workshop on Worst-Case Execution Time Analysis (WCET)*, 2010.

[33] Hilary J Holz, Anne Applin, Bruria Haberman, Donald Joyce, Helen Purchase, and Catherine Reed. Research methods in computing: What are they, and how should we teach them? In *Working group reports on ITiCSE on Innovation and technology in computer science education*, pages 96–114. Association for Computing Machinery, 2006.

[34] Matthias Ivers and Rolf Ernst. Probabilistic network loads with dependencies and the effect on queue sojourn times. In *International Conference on Heterogeneous Networking for Quality, Reliability, Security and Robustness*, pages 280–296. Springer, 2009.

[35] Łukasz Kruk, John Lehoczky, Kavita Ramanan, Steven Shreve, et al. Heavy traffic analysis for edf queues with reneging. *The Annals of Applied Probability*, 21(2):484–545, 2011.

[36] M Ross Leadbetter, Georg Lindgren, and Holger Rootzén. Conditions for the convergence in distribution of maxima of stationary normal processes. *Stochastic Processes and their Applications*, 8(2):131–139, 1978.

[37] John P Lehoczky. Real-time queueing theory. In *IEEE Real-Time Systems Symp. (RTSS)*, pages 186–195, 1996.

[38] Benjamin Lesage, David Griffin, Sebastian Altmeyer, Liliana Cucu-Grosjean, and Robert I Davis. On the analysis of random replacement caches using static probabilistic timing methods for multi-path programs. *Real-Time Systems*, 54(2):307–388, 2018.

[39] Benjamin Lesage, David Griffin, Sebastian Altmeyer, and Robert I Davis. Static probabilistic timing analysis for multi-path programs. In *IEEE Real-Time Systems Symp. (RTSS)*, pages 361–372, 2015.

[40] Andreas Lindholm, Dave Zachariah, Petre Stoica, and Thomas B Schön. Data consistency approach to model validation. *IEEE Access*, 7:59788–59796, 2019.

[41] Rui Liu, Alex F Mills, and James H Anderson. Independence thresholds: Balancing tractability and practicality in soft real-time stochastic analysis. In *IEEE Real-Time Syst. Symp. (RTSS)*, pages 314–323, 2014.

[42] Andreas Löfwenmark and Simin Nadjm-Tehrani. Fault and timing analysis in critical multi-core systems: A survey with an avionics perspective. *Journal of Systems Architecture*, 87:1–11, 2018.

[43] Yue Lu, Thomas Nolte, Iain Bate, and Liliana Cucu-Grosjean. A statistical response-time analysis of complex real-time embedded systems by using timing traces. In *IEEE Int. Symp. on Industrial and Embedded Systems (SIES)*, pages 43–46, 2011.

[44] Yue Lu, Thomas Nolte, Iain Bate, and Liliana Cucu-Grosjean. A statistical response-time analysis of real-time embedded systems. In *IEEE Real-Time Systems Symp. (RTSS)*, pages 351–362, 2012.

[45] Yue Lu, Thomas Nolte, Johan Kraft, and Christer Norstrom. 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 (RTCSA)*, pages 153–160, 2010.

[46] Yue Lu, Thomas Nolte, Johan Kraft, and Christer Norstrom. Statistical-based response-time analysis of systems with execution dependencies between tasks. In *IEEE Int. Conf. on Engineering of Complex Computer Systems*, pages 169–179, 2010.

[47] Nicola Manica, Luigi Palopoli, and Luca Abeni. Numerically efficient probabilistic guarantees for resource reservations. In *IEEE Int. Conf. on Emerging Technologies & Factory Automation (ETFA)*, pages 1–8, 2012.

[48] Filip Marković, Alessandro Vittorio Papadopoulos, and Thomas Nolte. On the convolution efficiency for probabilistic analysis of real-time systems. In *Euromicro Conf. on Real-Time Systems (ECRTS)*. Schloss Dagstuhl-Leibniz-Zentrum für Informatik, 2021.

[49] Pau Martí, Josep M Fuertes, Gerhard Fohler, and Krithi Ramamritham. Improving quality-of-control using flexible timing constraints: metric and scheduling. In *IEEE Real-Time Systems Symp. (RTSS)*, pages 91–100, 2002.

[50] Dorin Maxim and Liliana Cucu-Grosjean. Response time analysis for fixed-priority tasks with multiple probabilistic parameters. In *IEEE Real-Time Systems Symp. (RTSS)*, pages 224–235. IEEE, 2013.

[51] Dorin Maxim, Robert I Davis, Liliana Cucu-Grosjean, and Arvind Easwaran. Probabilistic analysis for mixed criticality systems using fixed

priority preemptive scheduling. In *International Conference on Real Time and Networks Systems (RTNS)*, pages 237–246, 2017.

[52] Dorin Maxim, Mike Houston, Luca Santinelli, Guillem Bernat, Robert I Davis, and Liliana Cucu-Grosjean. Re-sampling for statistical timing analysis of real-time systems. In *International Conference on Real Time and Networks Systems (RTNS)*, pages 111–120, 2012.

[53] Dorin Maxim, Frank Soboczenski, Iain Bate, and Eduardo Tovar. Study of the reliability of statistical timing analysis for real-time systems. In *International Conference on Real Time and Networks Systems (RTNS)*, pages 55–64, 2015.

[54] Alex F. Mills and James H. Anderson. A multiprocessor server-based scheduler for soft real-time tasks with stochastic execution demand. In *IEEE Int. Conf. on Embedded and Real-Time Computing Systems and Applications (RTCSA)*, pages 207–217, 2011.

[55] Nicolas Navet, Y-Q Song, and Françoise Simonot. Worst-case deadline failure probability in real-time applications distributed over controller area network. *Journal of systems Architecture*, 46(7):607–617, 2000.

[56] Luigi Palopoli, Daniele Fontanelli, Luca Abeni, and Bernardo Villalba Frias. An analytical solution for probabilistic guarantees of reservation based soft real-time systems. *IEEE Trans. Parallel and Distributed Systems*, 27(3):640–653, 2015.

[57] Eduardo Quinones, Emery D Berger, Guillem Bernat, and Francisco J Cazorla. Using randomized caches in probabilistic real-time systems. In *Euromicro Conf. on Real-Time Systems (ECRTS)*, pages 129–138, 2009.

[58] Lawrence R. Rabiner. A tutorial on hidden markov models and selected applications in speech recognition. *Proc. the IEEE*, 77(2):257–286, 1989.

[59] Khaled S Refaat and Pierre-Emmanuel Hladik. Efficient stochastic analysis of real-time systems via random sampling. In *Euromicro Conf. on Real-Time Systems (ECRTS)*, pages 175–183. IEEE, 2010.

[60] Per Runeson and Martin Höst. Guidelines for conducting and reporting case study research in software engineering. *Empirical software engineering*, 14(2):131, 2009.

[61] Luca Santinelli, Jérôme Morio, Guillaume Dufour, and Damien Jacquemart. On the sustainability of the extreme value theory for wcet estimation. In *Int. Workshop on Worst-Case Execution Time Analysis (WCET)*, 2014.

[62] Takahiro Shinozaki. Hmm state clustering based on efficient cross-validation. In *2006 IEEE International Conference on Acoustics Speech and Signal Processing Proceedings*, volume 1, 2006.

[63] Václav Struhár, Moris Behnam, Mohammad Ashjaei, and Alessandro Vittorio Papadopoulos. Real-time containers: A survey. In *Workshop on Fog Computing and the IoT (Fog-IoT)*, volume 80, pages 7:1–7:9, 2020.

[64] Aida Čaušević, Alessandro Vittorio Papadopoulos, and Marjan Sirjani. Towards a framework for safe and secure adaptive collaborative systems. In *IEEE Annual Computer Software and Applications Conf. (COMPSAC)*, volume 2, pages 165–170, 2019.

[65] Bernardo Villalba Frías, Luigi Palopoli, Luca Abeni, and Daniele Fontanelli. The prosit tool: Toward the optimal design of probabilistic soft real-time systems. *Software: Practice and Experience*, 48(11):1940–1967, 2018.

[66] Georg von der Brüggen, Nico Piatkowski, Kuan-Hsun Chen, Jian-Jia Chen, and Katharina Morik. Efficiently approximating the probability of deadline misses in real-time systems. In *Euromicro Conf. on Real-Time Systems (ECRTS)*. Schloss Dagstuhl-Leibniz-Zentrum fuer Informatik, 2018.

[67] Georg von der Brüggen, Nico Piatkowski, Kuan-Hsun Chen, Jian-Jia Chen, Katharina Morik, and Björn B Brandenburg. Efficiently approximating the worst-case deadline failure probability under EDF. In *IEEE Real-Time Syst. Symp. (RTSS)*, pages 214–226, 2021.

[68] Reinhard Wilhelm. Mixed Feelings About Mixed Criticality (Invited Paper). In *Int. Workshop on Worst-Case Execution Time Analysis (WCET)*, volume 63, pages 1:1–1:9, 2018.

[69] Reinhard Wilhelm, Jakob Engblom, Andreas Ermedahl, Niklas Holsti, Stephan Thesing, David Whalley, Guillem Bernat, Christian Ferdinand, Reinhold Heckmann, Tulika Mitra, et al. The worst-case execution-time problem—overview of methods and survey of tools. *ACM TECS*, 7(3):36, 2008.

[70] Claes Wohlin, Per Runeson, Martin Höst, Magnus C Ohlsson, Björn Regnell, and Anders Wesslén. *Experimentation in software engineering*. Springer Science & Business Media, 2012.

II

Included Papers

Chapter 7

Paper A
Identification and Validation of Markov Models with Continuous Emission Distributions for Execution Times.

Anna Friebe, Alessandro V. Papadopoulos, and Thomas Nolte. In IEEE International Conference on Embedded and Real-Time Computing Systems and Applications (RTCSA), 2020.

Abstract

It has been shown that in some robotic applications, where the execution times cannot be assumed to be independent and identically distributed, a Markov Chain with discrete emission distributions can be an appropriate model. In this paper we investigate whether execution times can be modeled as a Markov Chain with continuous Gaussian emission distributions. The main advantage of this approach is that the concept of distance is naturally incorporated. We propose a framework based on Hidden Markov Model (HMM) methods that 1) identifies the number of states in the Markov Model from observations and fits the Markov Model to observations, and 2) validates the proposed model with respect to observations. Specifically, we apply a tree-based cross-validation approach to automatically find a suitable number of states in the Markov model. The estimated models are validated against observations, using a data consistency approach based on log likelihood distributions under the proposed model. The framework is evaluated using two test cases executed on a Raspberry Pi Model 3B+ single-board computer running Arch Linux ARM patched with PREEMPT_RT. The first is a simple test program where execution times intentionally vary according to a Markov model, and the second is a video decompression using the `ffmpeg` program. The results show that in these cases the framework identifies Markov Chains with Gaussian emission distributions that are valid models with respect to the observations.

7.1 Introduction

In real-time systems requirements on timing properties must be considered, in addition to functional requirements, i.e., it is of importance to have the correct behavior *at the appropriate time*. Real-time requirements range from safety critical timing requirements of *hard real-time systems* found in applications of aeronautics, automotive and medical device systems to *soft real-time systems*, e.g., common in multimedia applications. Failure to meet hard real-time requirements may result in a disaster and/or loss of human life whereas failure to meet a soft real-time requirement can cause a deterioration of the Quality of Service (QoS) [11] such as in a video playback or affect the Quality of Control (QoC) [34] of a robot's motion planning.

It is a challenge to enable sufficiently accurate timing analysis of today's complex systems. Multicore processors [28] and mixed-criticality systems [7, 43], as well as fog and edge computing capabilities [41, 40, 12], require new methods for ensuring sound timing estimates along with functional integrity and limited over-provisioning of computational resources and bandwidth for communication. Practical timing analysis methods that consider the entire timing distribution, as opposed to only the tail of the distribution, can allow for system design without excessive over-provisioning. Taking the entire distribution into account is of particular interest primarily in the case of soft real-time applications where requirements on QoS or QoC are considered.

Frías et al. have shown that computation times of a computer vision application in a robotic system can be described as a Markov Model [20, 4]. Inspired by the work of Frías et al., in this paper we investigate the following research question: How can the execution time distribution of a task be faithfully modeled in a probabilistic framework? In particular:

1. Can execution time distributions be suitably modeled as a Markov Chain, where each state is associated with a Gaussian emission distribution?

2. How can one estimate the model parameters from timing measurements of the task's jobs?

Our main motivation for exploring *continuous emission distributions* is that they naturally include a concept of distance. Two execution time measurements that are similar are more likely to originate from the same state, compared to measurements of different magnitude. In the standard methods for Markov models with discrete emission distributions, each execution time value is treated as a label and the distance information is lost. By using continuous distributions, a model likely to provide a reasonable estimate from a smaller amount of observations. Although each execution time is a discrete number of clock cycles, realistic tasks on today's processors often result in a large number of possible values, that can be closely approximated by a continuous distribution. In order to develop methods that can be of practical use in schedulability analysis, we estimate model parameters from observations.

In this paper, we present an automated framework that estimates and validates an execution time distribution model from observations. The proposed model is a Markov Model with a Gaussian emission distribution associated with each state. More precisely, we propose an HMM, as we observe the execution times, but the states cannot be directly observed. Firstly, in **step 1** we identify the number of states for the HMM, and fit the model to the observations. A *tree-based cross-validation approach* [39] is adopted for identifying the number of states. We estimate the parameters for the Gaussian distributions and the transition matrix by applying the *Expectation-Maximization algorithm* [35], initialized with values resulting from the tree-based cross-validation. In **step 2** we validate the estimated model using observations. Here, we adopt a *data consistency approach* [27], and derive methods for application of this approach to Markov Chains using outputs from the *Forward-backward algorithm.*

A set of probabilistic techniques are selected and combined in the framework, to enable identification and validation of the HMM. The methods are applied to two test cases, a test program with a known Markov Chain behavior, and a video decompression program treated as a black box. The results are presented and discussed. Further investigation is needed to evaluate the applications where the framework and the specific techniques of each step are most suitable, and for what applications other techniques are better for one or several of the proposed steps.

The rest of the paper is structured as follows. Section 7.2 presents the related work, followed by Section 7.3 that presents the task model. Section 7.4 presents the proposed framework. Sections 7.5 and 7.6 discuss the experimental results, Finally, Section 7.7 concludes the paper and highlight directions for future work.

7.2 Related Work

Cucu-Grosjean and Davis have recently provided thorough surveys of the literature on probabilistic methods in Timing Analysis [15], Response Time Analysis, analysis of server-based systems, Real-Time Queuing Theory, system analysis with fault modeling and Mixed Criticality Systems [14]. Cazorla et al. provide a taxonomy and a survey on the methods used in Probabilistic

Worst-Case Execution Time Analysis [9]. The authors also emphasize the fact that while measurement-based approaches may allow analysis of a black-box system, the results are only reliable if the analysis data are representative with respect to the operational environment. That is, all sources of variation in execution times or latencies that are relevant for the result need to be contributing to the variation in the data. Otherwise their effects need to be upper-bounded or accounted for in other ways.

In the area of static probabilistic timing analysis, many works consider models for set-associative or fully associative caches. Quinones et al. [37] showed that for some cases with programs displaying a cache risk pattern, random replacement gives better results and lower variability compared to Least Recently Used (LRU) replacement. Altmeyer et al. [6] provide analysis considering reuse distance, associativity and contention. Analysis using random replacement caches has been extended to the multi-path case [26, 25]. Chen and Beltrame [10] perform timing analysis for single-path programs on systems with evict-on-miss random replacement caches by using an adaptive Markov model.

Measurement-Based Probabilistic Timing Analysis methods estimate the pWCET by applying statistical techniques to observations of execution time measurements. While WCET is a scalar value – the upper bound of the worst case execution time of runs – the pWCET is a probability distribution representing the upper bound on the probability of exceeding each execution time value in valid scenarios of repeated runs of the program. The theoretical basis is in Extreme Value Theory (EVT). The first work in this direction was by Burns, Edgar and Griffin [8, 19, 21]. Measurement-Based Probabilistic Timing Analysis was then introduced by Cucu-Grosjean et al. [13] in 2012.

Probabilistic Response Time Analysis is used to calculate the response time distribution of jobs, and in this manner estimate the probability of a deadline miss. Diaz et al. [16] presented response time analysis for a system with periodic tasks where random variables describe execution times. Here, the worst-case processor utilisation can exceed 1, since a backlog is considered at the end of the hyperperiod. They show that the backlog is a Markov chain. In [17] they also provided properties needed to achieve a safe over-approximation. More recent, the system model was extended by Kaczynski et al. [22] to also allow for systems with aperiodic tasks.

Similarly as in Measurement-Based Probabilistic Timing Analysis for pWCET estimates, EVT has also been applied in order to estimate response time distributions. The majority of the work in this line of research, Statistical Response Time Analysis, has been performed by Lu et al. [32, 31, 29, 30].

Real-Time Queueing Theory is an area where queue lengths and waiting times are analyzed mathematically. Lehozcky [24] introduced the concept in 1996, building upon work on queuing theory that started in the 1950s. Doytchinov provided a mathematical formalization in [18].

Probabilistic analysis has also been applied in analysis of server-based systems. Buttazzo and Abeni introduced the Constant Bandwidth Server (CBS) [1], and probabilistic deadlines for Quality of Service guarantees [2]. The same group has considered execution times [3, 36] and interarrival times [5, 33, 36] modeled with probability distributions.

Frías et al. [20, 4] have published work regarding execution time models for tasks where execution times display dependencies due to slowly changing input data. They have shown that for a robotic image processing task in line following, modeling execution times as a Hidden Markov Model is appropriate. In this work, discrete emission distributions for the different states are used. The deadline miss probability under CBS/Earliest Deadline First (EDF) is estimated for the Hidden Markov Model and compared to an assumption of independent and identically distributed random variables. The calculated probabilities are compared to experimental results with CBS/EDF as implemented in the Linux SCHED_DEADLINE scheduling policy. The experiments show that with an independent and identically distributed (i.i.d.) assumption of execution times, the probability of respecting the deadline is overestimated, i.e., the estimate is optimistic. The estimates based on the identified Hidden Markov Model, on the other hand, are very close to the experimental results.

7.3 Task Model

In this paper, we consider a periodic task τ consisting of a sequence of periodic jobs J_i, $i \in \mathbb{N}$, with period T. Each job has an execution time $c_i \in \mathbb{R}$.

We model the execution time distribution of the task according to an adapted version of the Markov Computation Time Model (MCTM) in Frías et al. [20]. The model is described by the set $\{\mathcal{M}, \mathcal{P}, \mathcal{C}\}$, where

- $\mathcal{M} = \{m_1, m_2, \dots, m_N\}$ is the set of N states, $m_n, n \in \mathbb{N}$.
- $\mathcal{P}$ 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.
- $\mathcal{C} = \{C_1, C_2, \dots, C_N\}$ is the set of execution time distributions, or emission distributions related to respective state. In this paper, these are modelled as Gaussian distributions with mean μ_n, and variance σ_n^2, i.e., $C_n \sim \mathcal{N}(\mu_n, \sigma_n^2)$.

7.4 Framework

In this section, we present and describe the framework that we have developed for the identification and validation of the probabilistic model. Specifically, the framework consists of the following steps:

1. Firstly, we apply the tree-based cross-validation approach [39] described in Section 7.4.1 to identify the number of states in the HMM from the observations. An HMM with the identified number of states is fitted to the observations, according to the Expectation-Maximization [35] algorithm, using the likelihoods obtained with the Forward-backward algorithm [38]. The Gaussian distribution parameters and the transition matrix used as a starting point for the optimization is given by the outputs of the tree-based cross-validation.

2. Secondly, the obtained model is validated using a data consistency approach [27] described in Section 7.4.2. Here we derive expressions using outputs from the Forward-backward algorithm for application of the data consistency model validation.

In the following subsections we describe these steps on model identification and validation in more detail.

7.4.1 Tree-Based Cross-Validation

In general, the number of states N is not known a priori, and must be identified, for example based on logged data. In order to identify a number of states N_{opt} that allows for capturing the execution sequence properties without overfitting, a tree-based cross-validation approach is applied, as described in Shinozaki [39]. The execution time sequence $cs = \{c_1, c_2, \ldots, c_{NS}\}$ consisting of execution times from $NS \in \mathbb{N}$ jobs, is split into M non-overlapping folds cs_f with index f.

$$cs = \cup_{f=1}^{M} cs_f$$
$$cs_f \cap cs_g = \emptyset, \quad f \neq g$$

For each fold with index f, we also define the complement cs_f^c, the remaining folds:

$$cs_f^c = \cup_{f \neq g} cs_g$$

For each fold an MCTM with $N > N_{\text{opt}}$ states is fitted to the remaining folds cs_f^c. The initial values of means and standard deviations for the emission distributions are given by k-means clustering with $k = N$.

The occupancy probability γ_{ni} is the probability of being in state n at round i, given the observation sequence cs_f, where $c_i \in cs_f$ and the model parameters retrieved from fitting to cs^c_f. The occupancy probabilities for each state index n and observation round i are calculated with the Viterbi algorithm [38].

Statistics containing all information from the sample needed for parameter value estimates for a statistical model are sufficient for the parameter. By calculating the sufficient statistics we can store the needed information from a sample in a compact manner. Sufficient statistics for likelihood estimates of a Markov chain model with Gaussian emission distribution are $a0$, $a1$ and $a2$ [39]. These are calculated for each fold index f and state index n:

$$a0_{fn} = \sum_{i, c_i \in cs_f} \gamma_{ni}$$

$$a1_{fn} = \sum_{i, c_i \in cs_f} c_i \gamma_{ni}$$

$$a2_{fn} = \sum_{i, c_i \in cs_f} c_i^2 \gamma_{ni}$$

For a set or cluster s of states, the maximum likelihood mean μ and variance ν for a fold index f can be calculated from the sufficient statistics from remaining folds:

$$\mu_{fs} = \frac{\sum_{g \neq f} \sum_{m_n \in s} a1_{gn}}{\sum_{g \neq f} \sum_{m_n \in s} a0_{gn}} \tag{7.1}$$

$$\nu_{fs} = \frac{\sum_{g \neq f} \sum_{m_n \in s} a2_{gn}}{\sum_{g \neq f} \sum_{m_n \in s} a0_{gn}} - \mu_{fs}^2 \tag{7.2}$$

These are then used to calculate a likelihood per fold index f and cluster s:

$$L_{fs} = -\frac{1}{2} \times \sum_{m_n \in s} \left(\ln(2\pi\nu_{fs}) a0_{fn} + \frac{a2_{fn} - 2\mu_{fs} a1_{fn} + \mu_{fs}^2 a0_{fn}}{\nu_{fs}} \right) \tag{7.3}$$

The likelihood for a cluster is then calculated by summation of the likelihoods of each fold index.

$$L_s = \sum_{f=1}^{M} L_{fs} \tag{7.4}$$

A tree is created, and initially all states are placed in a cluster in the root node. The cross validated likelihood is calculated for the tree consisting of only this cluster. Attempts are made to split the leaf nodes of the tree, so that the node's cluster is split into two clusters. The possible ways of splitting the states in a tree node are:

1. 2-means clustering of 2D data points of mean and standard deviation of each state is performed, and the resulting split is evaluated.

2. The states are ordered with respect to increasing mean, and each possible split along the ordered states is evaluated.

3. The modes are ordered with respect to increasing standard deviation, and each possible split along the ordered states is evaluated.

The split that gives the greatest increase in likelihood is selected. Nodes are split as long as the cross validated likelihood of the subtree is increased by the split, or until there is only one state in the tree node. The node splitting process is a greedy algorithm that may lead to a local maximum. Pseudo code is provided in Algorithm 7.1.

When a suitable number of states has been found, an MCTM with this number of states is fitted to the execution time samples, starting from initial values taken from the node clusters.

7.4.2 Data Consistency Model Validation

We evaluate whether the fitted model as described in Section 7.4.1,is valid, with respect to observations. Thus, we generate samples from the model and using a data consistency approach [27] we compare the generated samples to observations. If the observed data is consistent with data generated from the model, the model can be used in schedulability analysis.

The model validation can be performed with the same observations used for the model estimate, to evaluate whether the model can capture the properties of the observations. The evaluation can also be performed with observations from other runs of the program, to evaluate whether the model and parameters are valid in these cases, for different inputs or different hardware states.

The data consistency approach we apply is described by Lindholm et al. [27]. The log-likelihood under the proposed model is estimated for samples generated from the model and for the observed samples. Using a log-likelihood based statistic, an estimate is calculated of the probability of generating the observed sample or a sample with higher dispersion, from the evaluated model. This is equivalent to the probability that we reject the model on the basis of the observed data being overdispersed, assuming that the data

Algorithm 7.1 Pseudo code describing the tree cluster splitting process. The likelihood increase for possible splits of a cluster are calculated using the pre-computed sufficient statistics.

```
1:  function TREECLUSTERSPLITTING($suffStats$, $N$)
2:      $tree \leftarrow createNode()$
3:      $tree.states \leftarrow [1 : N]$                ▷ Add all states to the root cluster
4:      $tree.[leftStates, rightStates, advantage] \leftarrow calcSplitAdvantage(tree, suffStats)$    ▷ Find the best split
5:      while $(tree.advantage > 0)$ and $(nLeafNodes(tree) \leq N)$ do    ▷ While the likelihood increases, and we can split leaves
6:          for $node \in leaves(tree)$ do
7:              if $node.advantage > 0$ then
8:                  $node.leftChild \leftarrow createNode()$    ▷ Add new leaf nodes and split the state cluster
9:                  $node.leftChild.states \leftarrow node.leftStates$
10:                 $node.rightChild \leftarrow createNode()$
11:                 $node.rightChild.states \leftarrow node.rightStates$
12:                 $node.states \leftarrow \emptyset$
13:             end if
14:         end for
15:         for $node \in leaves(tree)$ do
16:             $node.[leftStates, rightStates, advantage] \leftarrow calcSplitAdvantage(node, suffStats)$    ▷ Find the best split of the leaf
17:         end for
18:         for $node \in tree; post-order$ do
19:             $node.advantage \leftarrow maxAdvantageChildren(node)$    ▷ Move the highest likelihood increase to the root
20:         end for
21:     end while
22:     return $tree$                ▷ Return the tree with $N_{opt}$ leaf clusters
23: end function
```

is generated from the model. Another way of describing it is that the model is underdispersed compared with the observations. This probability of falsely rejecting the model or Probability of False Alarm due to underdispersion (PFA_u) is similar to the p-value concept in hypothesis evaluation. While the p-value is the probability of obtaining the test results or more extreme values assuming the null hypothesis is correct, the PFA_u is the probability of obtaining data with at least the observed variability, assuming they are generated from the proposed model.

We denote the observed execution times with an underline, as $\underline{\mathbf{c}}$. In our case, we evaluate a single model with a probability distribution $p(\mathbf{c}|\mathcal{M}_*, \mathcal{P}_*, \mathcal{C}_*)$, where $\mathcal{M}_*, \mathcal{P}_*, \mathcal{C}_*$ are the parameters of the fitted MCTM.

Using the model, we generate trajectories denoted with tilde $\tilde{\mathbf{c}} \sim p(\mathbf{c}|\mathcal{M}_*, \mathcal{P}_*, \mathcal{C}_*)$. Using $c_{1:t}$ to denote the samples at rounds 1 to t from the trajectory, the conditional likelihood of an execution time measurement in a trajectory under the model is:

$$p_t = p(c_t|c_{1:t-1}, \mathcal{M}_*, \mathcal{P}_*, \mathcal{C}_*) = \frac{p(c_{1:t}|\mathcal{M}_*, \mathcal{P}_*, \mathcal{C}_*)}{p(c_{1:t-1}|\mathcal{M}_*, \mathcal{P}_*, \mathcal{C}_*)}$$

This can be calculated from the scaling factors resulting from the Forward-backward algorithm. We denote these as sca_i. From Rabiner [38] we have the probability of the observations expressed in terms of the scaling factors:

$$p(c_{1:t}|\mathcal{M}_*, \mathcal{P}_*, \mathcal{C}_*) = \frac{1}{\prod_{i=1}^{t} sca_i}$$

From this it is clear that the conditional probability can be written as:

$$p_t = p(c_t|c_{1:t-1}, \mathcal{M}_*, \mathcal{P}_*, \mathcal{C}_*) = \frac{1}{sca_t}$$

The conditional log-likelihood of a data point is:

$$z_t \triangleq \ln p(c_t|c_{1:t-1}, \mathcal{M}_*, \mathcal{P}_*, \mathcal{C}_*) = -\ln sca_t \tag{7.5}$$

Conditional probabilities of outputs for each state separately can be estimated using the transition matrix and the scaled forward variables $\hat{\alpha}$:

$$\begin{aligned} &p(c_t, X_t = j|c_{1:t-1}, \mathcal{M}_*, \mathcal{P}_*, \mathcal{C}_*) \\ &= p(c_t|X_t = j) \sum_{k=1}^{N} p_{k,j} p(X_{t-1} = k|c_{1:t-1}, \mathcal{M}_*, \mathcal{P}_*, \mathcal{C}_*) \\ &= \frac{1}{\sigma_j\sqrt{2\pi}} e^{-\frac{(c_t-\mu_j)^2}{2\sigma^2}} \sum_{k=1}^{N} p_{k,j} \hat{\alpha}_{k,t-1} \end{aligned}$$

From Rabiner [38] we have that:

$$\begin{aligned}
\alpha_{k,t} &= p(c_{1:t}, X_t = k|\mathcal{M}_*, \mathcal{P}_*, \mathcal{C}_*) \\
\hat{\alpha}_{k,t} &= \alpha_{k,t} \prod_{i=1}^{t} sca_i = \frac{p(c_{1:t}, X_t = k|\mathcal{M}_*, \mathcal{P}_*, \mathcal{C}_*)}{p(c_{1:t}|\mathcal{M}_*, \mathcal{P}_*, \mathcal{C}_*)} \\
&= p(X_t = k|c_{1:t}, \mathcal{M}_*, \mathcal{P}_*, \mathcal{C}_*)
\end{aligned}$$

The conditional log-likelihood of each state and data point is:

$$\begin{aligned}
z_{t,j} &\triangleq \ln p(c_t|X_t = j, c_{1:t-1}, \mathcal{M}_*, \mathcal{P}_*, \mathcal{C}_*) \\
&= -\ln \sigma_j - \frac{\ln 2\pi}{2} - \frac{(c_t - \mu_j)^2}{2\sigma^2} + \ln \sum_{k=1}^{N} p_{k,j} \hat{\alpha}_{k,t-1} \qquad (7.6)
\end{aligned}$$

We denote the mean of the log-likelihood of data points in generated trajectories as $\mathbb{E}[\tilde{z}_t]$ and the variance as $\mathbf{Var}[\tilde{z}_t]$. The test statistic T for a trajectory is defined:

$$T(\mathbf{c}; \mathcal{M}_*, \mathcal{P}_*, \mathcal{C}_*) = \frac{1}{n} \sum_{t=1}^{n} \frac{z_t - \mathbb{E}[\tilde{z}_t]}{\mathbf{Var}[\tilde{z}_t]} \qquad (7.7)$$

T statistics are defined similarly for each state by replacing z_t with $z_{t,j}$. S is defined as the random event of a generated sample resulting in a higher T-statistic than the observed one:

$$S(\tilde{\mathbf{c}}, \underline{\mathbf{c}}) : T(\tilde{\mathbf{c}}; \mathcal{M}_*, \mathcal{P}_*, \mathcal{C}_*) > T(\underline{\mathbf{c}}; \mathcal{M}_*, \mathcal{P}_*, \mathcal{C}_*)$$

When the probability of S, $\mathbb{P}_{\tilde{\mathbf{c}}|\mathcal{M}_*, \mathcal{P}_*, \mathcal{C}_*}(S(\tilde{\mathbf{c}}, \underline{\mathbf{c}}))$ is close to 0 or close to 1, it indicates that the observed data is inconsistent with the proposed model.

Lindholm et al. define PFA_u, the probability of falsely rejecting a model due to under-dispersion of the generated log likelihoods as:

$$PFA_u \triangleq \mathbb{P}_{\tilde{\mathbf{c}}|\mathcal{M}_*, \mathcal{P}_*, \mathcal{C}_*}(S(\tilde{\mathbf{c}}, \underline{\mathbf{c}})) \qquad (7.8)$$

Lindholm et al. also define the probability of falsely rejecting the model due to under- or overdispersion as:

$$PFA = \min(PFA_u, 1 - PFA_u)$$

However, in our work, we use PFA_u, as an under-dispersion of data generated from the proposed model indicates that the model is optimistic with regard to tail estimates. We note that values of PFA_u that are close to 1 also indicate model inconsistency, but that this relates to over-dispersion of data generated from the model.

Pseudo code for the data consistency approach is given in Algorithm 7.2.

Algorithm 7.2 Pseudo code describing the data consistency validation process.

```
1: function DATACONSISTENCYVALIDATION($\mathcal{M}_*, \mathcal{P}_*, \mathcal{C}_*, \underline{c}$)
2:     for $i \in 1 : M'$ do
3:         $traj1 \leftarrow generateTraj(\mathcal{M}_*, \mathcal{P}_*, \mathcal{C}_*, M', length(\underline{c}))$    ▷ Generate $M'$ trajectories of the same length as observations.
4:         $simZ1[i] \leftarrow calcZ(traj1, \mathcal{M}_*, \mathcal{P}_*, \mathcal{C}_*)$    ▷ Calculate log likelihoods $z_t$ and $N$ $z_{t,j}$ for the samples as in Equations (7.5) and (7.6).
5:     end for
6:     $EZ \leftarrow mean(simZ1)$    ▷ Estimate $\mathbb{E}[z_t]$ and $N$ $\mathbb{E}[z_{t,j}]$ across $M'$ values for each round $t$.
7:     $VarZ \leftarrow var(simZ1)$    ▷ Estimate $\mathbf{Var}[z_t]$ and $N$ $\mathbf{Var}[z_{t,j}]$ across $M'$ values for each round $t$.
8:     for $i \in 1 : M$ do
9:         $traj2 \leftarrow generateTraj(\mathcal{M}_*, \mathcal{P}_*, \mathcal{C}_*, M, length(\underline{c}))$    ▷ Generate $M$ trajectories of the same length as observations.
10:        $simZ2[i] \leftarrow calcZ(traj2, \mathcal{M}_*, \mathcal{P}_*, \mathcal{C}_*)$    ▷ Calculate log likelihoods $z_t$ and $N$ $z_{t,j}$ for the samples as in Equations (7.5) and (7.6).
11:        $Tsim[i] \leftarrow calcT(simZ2, EZ, VarZ)$ ▷ Calculate $M \times (N+1)$ $T$s for $z_t$ and $z_{t,j}$ as in Equation (7.7) from simulated trajectories.
12:    end for
13:    $obsZ \leftarrow calcZ(\underline{c}, \mathcal{M}_*, \mathcal{P}_*, \mathcal{C}_*)$    ▷ Calculate log likelihoods $z_t$ and $N$ $z_{t,j}$ for the samples as in Equations (7.5) and (7.6).
14:    $Tobs \leftarrow calcT(obsZ, EZ, VarZ)$ ▷ Calculate $N+1$ $T$s for $z_t$ and $z_{t,j}$ as in Equation (7.7) from observations.
15:    $PFA_u \leftarrow count(Tsim > Tobs)/M$    ▷ Estimate the probability of S for the entire model and per state.
16:    return $PFA_u$
17: end function
```

7.5 Evaluation

7.5.1 Test Setup

A Raspberry Pi 3B+ single board computer with quad-core 1.4 GHz BCM2837B0 is utilized in the tests. Arch Linux ARM kernel 4.14.87 with `PREEMPT_RT` patch 4.14.87-49 is configured with fully preemptible kernel and timer frequency of 100Hz. The SD card low latency mode and `dwc_otg` FIQ are disabled. A test program is pinned to a core that is isolated from load-balancing and scheduling algorithms. The scaling governor is set to performance for all cores and USB is disabled during the run. The program is run in user space with FIFO scheduling and maximum priority. The `ftrace` utility `trace-cmd` is used to log release (`sched_wakeup`) and scheduling (`sched_switch`) events, and to generate trace reports with nanosecond precision from the trace logs.

The model identification and validation framework is applied offline using the recorded traces.

7.5.2 Implementation

The tree-based cross validation approach described in Section 7.4.1 and the data consistency criterion model validation described in Section 7.4.2 are implemented in R[1], utilizing the R packages `depmixS4` [42] and `data.tree`. Evaluation code as well as test programs and scripts are available online [2].

Four folds are used for the cross validation. As described in Section 7.4.1, the MCTM is fitted to three folds, and the sufficient statistics using the fitted model are calculated for the remaining fold. The occupancy probabilities are determined by application of the Viterbi algorithm. A new MCTM with the number of states given by the tree-based cross validation is created, and initialized with the means and variances from the clusters, as given in Equations (7.1) and (7.2) averaged over all folds. This model is then fitted to the entire training set, and the fitted model is validated with the data consistency criterion.

Values of the probability of false alarm due to underdispersion, PFA_u, are estimated for the entire model using z_t as in Equation (7.5), and for each state in the model using $z_{t,j}$ as in Equation (7.6). First, 100 trajectories are generated for estimation of the mean and variance of the log likelihood. The trajectories are generated using the simulate function in `depmixS4`, and log

[1]`https://www.r-project.org/`

[2]https://github.com/annafriebe/MarkovChainETFramework

likelihoods are retrieved from `depmixS4`'s forward and scaling variable resulting from the Forward-backward algorithm. Second, 100 new trajectories are generated for calculation of T values as given by Equation (7.7). Referring to Algorithm 7.2, both M' and M are set to 100.

7.5.3 Markov Chain Test Program

In a first test, a program with a known Markov Chain behavior is evaluated. The test program contains a state machine with three states. The program keeps an array of 100 integers, initialized from a random uniform distribution in the range [0, 4711]. It executes a job periodically at a 5ms interval. In the job, a state transition is performed, given the following transition matrix:

$$\mathcal{P} = \begin{pmatrix} 0.7 & 0.1 & 0.2 \\ 0.5 & 0.1 & 0.4 \\ 0.5 & 0.2 & 0.3 \end{pmatrix} \tag{7.9}$$

Depending on the current state, elements in the array are increased with 43 and a modulo operation with 4711 is performed. The first state has the shortest average execution time, the second state the middle and the third state the longest average execution time.

Logs are created from 21 runs of the program, one is used for model parameter estimation, and 20 in the model evaluation. In each run, the task releases 10 000 jobs. A python script is used to calculate the execution time for each job. The steady state is considered, so the logs from the first 250 jobs and the last executed job instance are excluded. The execution times of the first 250 are slightly lower, due to the program always starting in state 1 and possibly due to the system state. The last job's execution time is much longer due to produced status output before termination.

The execution time sequence used for estimating models is displayed in Figure 7.1.

The tree based cross-validation approach is applied with 8 initial states to the training execution time sequence. The fitted model has six remaining states. The means and standard deviations of the states and PFA_u values are displayed in Table 7.1, and the estimated transition matrix is given by:

$$\mathcal{P} = \begin{pmatrix} 0.51 & 0.18 & 0.08 & 0.02 & 0.19 & 0.007 \\ 0.45 & 0.27 & 0.05 & 0.04 & 0.18 & 0.005 \\ 0.36 & 0.14 & 0.07 & 0.047 & 0.38 & 0.005 \\ 0.31 & 0.18 & 0.07 & 3.7 \times 10^{-5} & 0.43 & 0.012 \\ 0.34 & 0.15 & 0.15 & 0.06 & 0.30 & 0.008 \\ 0.12 & 0.45 & 0.02 & 0.18 & 0.12 & 0.11 \end{pmatrix} \tag{7.10}$$

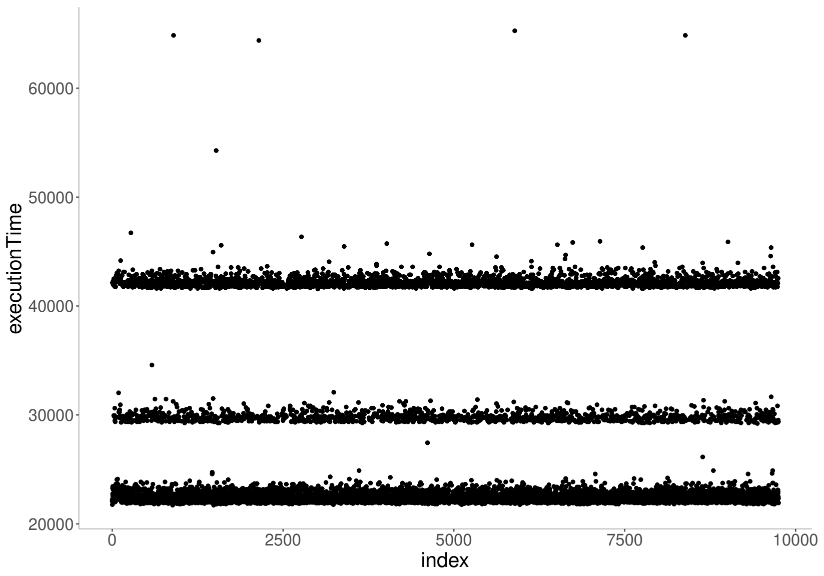


Figure 7.1: The execution time sequence from the Markov Chain Test program used for estimating models. Times are given in nanoseconds.

Based on the means and standard deviations from Table 7.1, we can see that states 1 and 2 represent the program state with the lowest mean execution time, states 3 and 4 represent the middle program state and states 5 and 6 represent the highest program state and state 6 also some outliers. If we sum the values of columns 1-2, 3-4 and 5-6 for each row in Equation (7.10), we see that for rows 1-5 they sum up to values similar to the corresponding transition probabilities in 7.9.

We see in Table 7.1 that the model is valid for all but one of the test sequences (test 8).

7.5.4 Video Decompression

A video is generated with images from the Tears of Steel open movie project[3]. The video is created with `ffmpeg` from frames 5000–8999 of the `1080bis-png` images[4]. The frame rate is set to 25 fps.

A trace is logged during decoding of the video with `ffmpeg` in native frame rate. The sequence of execution times from the decoding is displayed in Figure 7.2. We consider the steady state, therefore the first 250 and the

[3]https://mango.blender.org/

[4]https://media.xiph.org/tearsofsteel/tearsofsteel-1080bis-png/

Table 7.1: State means and standard deviations (in nanoseconds) and corresponding PFA_u values for the model estimated from the training sequence.

State	**1**	**2**	**3**	**4**	**5**	**6**	**All**
mean	22240	22859	29652	30248	42185	41203	NA
stddev	222	420	221	409	383	9 190	NA
PFA_u							
test 1	0.22	0.21	0.18	0.18	0.94	0.44	0.31
test 2	0.52	0.49	0.33	0.33	0.14	0.56	0.87
test 3	0.38	0.37	0.24	0.24	0.43	0.05	0.02
test 4	0.28	0.27	0.19	0.18	0.53	0.20	0.19
test 5	0.36	0.36	0.22	0.22	0.42	0.22	0.24
test 6	0.14	0.14	0.12	0.14	0.75	0.10	0.06
test 7	0.26	0.26	0.18	0.18	0.67	0.38	0.19
test 8	0.00	0.00	0.01	0.01	0.58	0.00	0.00
test 9	0.58	0.58	0.43	0.39	0.01	0.44	0.89
test 10	0.55	0.53	0.42	0.40	0.02	0.62	0.92
test 11	0.27	0.26	0.19	0.20	0.76	0.24	0.17
test 12	0.43	0.43	0.29	0.26	0.04	0.31	0.41
test 13	0.48	0.47	0.25	0.23	0.03	0.31	0.50
test 14	0.74	0.73	0.50	0.46	0.01	0.69	0.99
test 15	0.28	0.28	0.19	0.20	0.58	0.31	0.14
test 16	0.29	0.28	0.21	0.21	0.41	0.14	0.56
test 17	0.37	0.37	0.21	0.21	0.09	0.13	0.76
test 18	0.36	0.36	0.24	0.23	0.37	0.12	0.41
test 19	0.28	0.28	0.21	0.21	0.53	0.59	0.61
test 20	0.19	0.20	0.18	0.19	0.81	0.16	0.11
train	0.40	0.39	0.28	0.26	0.45	0.48	0.94

last 50 execution time measurements have been discarded, as outliers are seen on visual inspection. From the figures it is clear that the execution times are separated in distinct groups, the lower with execution times below 0.15 ms, accounting for approximately 57% of the samples, a slightly higher with execution times below 1 ms and a peak at around 0.45 ms, accounting for about 22% of the samples, a higher and more varying 10 ms accounting for approximately 19% of the samples, and the high with execution times above 22.5 ms accounting for less than 2% of the samples. These groups are considered as macrostates, and the transition probabilities between the states are displayed in Figure 7.3. The execution time sequence has been separated outside of the framework and the transition probabilities in Figure 7.3 are estimated directly from the sequence. The framework analysis is applied to each of these groups separately. The PFA_u values are with respect to the sequence used for model estimation.

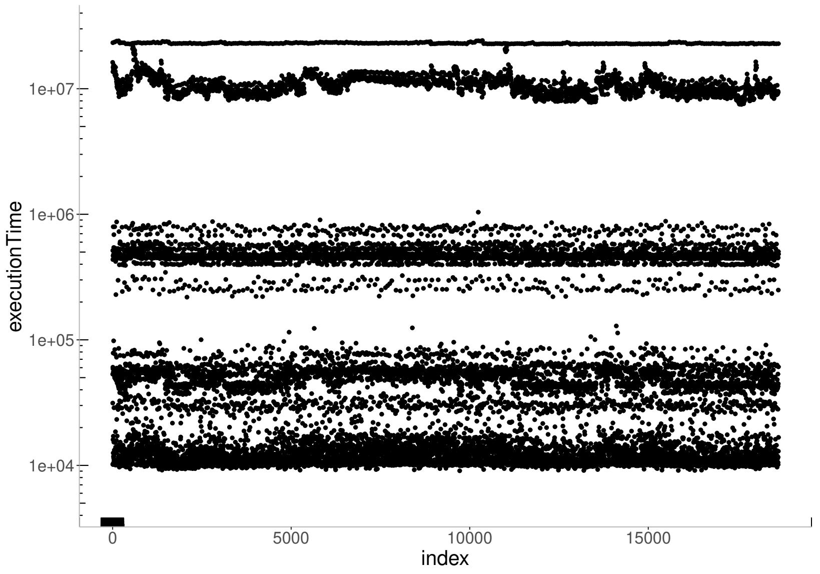


Figure 7.2

7.5.4.1 Macro state 1: Execution times below 0.15 ms

The execution times $c_i < 0.15$ ms are extracted from the video decompression log, resulting in a log of 10 538 samples. A Markov chain model is identified in the first two steps of the framework - the tree based cross validation approach described in Section 7.4.1, and fitting to the observations. The initial number of states is 20, and the resulting Markov model has 13 states. The data consistency criterion PFA_u for each state and for the entire model is calculated for the observations. The features and PFA_u values for the model is given in Table 7.2.

7.5.4.2 Macro state 2: Execution times in the range between 0.15 and 1 ms

The execution times $0.15 \leq c_i < 1$ ms consist of 4165 samples. The tree-based cross validation with 20 states in the initial Markov Model is applied and the resulting Markov Chain has 13 states. The state means and standard deviations of the estimated model, and the associated PFA_u values, are displayed in Table 7.3.

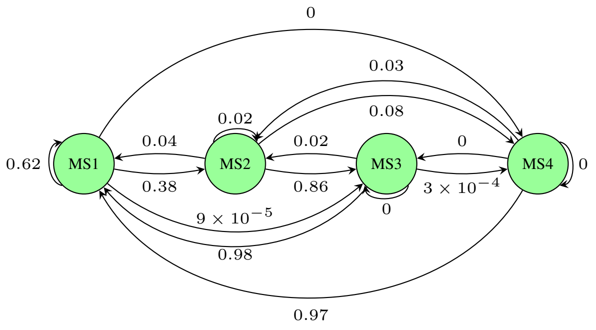


Figure 7.3: Estimated transition probabilities between the macro states.

Table 7.2: State means and standard deviations (in nanoseconds) and corresponding PFA_u values with the estimated model for macrostate 1.

State	**1**	**2**	**3**	**4**	**5**	**6**	**7**	**8**	**9**	**10**	**11**	**12**	**13**	**All**
mean	70 662	76 370	10 082	10 671	10 920	15 954	43 703	54 343	28 470	12 927	31 295	62 083	39 794	NA
stddev	15 236	2 360	230	640	578	2 585	1 069	4 095	1 908	1 034	3 024	2 159	2 065	NA
PFA_u	0.51	0.45	0.24	0.24	0.24	0.22	0.06	0.22	0.17	0.24	0.23	0.47	0.10	0.02

Table 7.2: State means and standard deviations (in nanoseconds) and corresponding PFA_u values with the estimated model for macrostate 1 (continued).

7.5.4.3 Macro state 3: Execution times in the range between 1 and 22.5 ms

The execution times $1 \leq c_i < 22.5$ ms are 3598 samples. The tree-based cross validation does not generally find a solution in this case - for many starting values the `depmixS4` fit function is unable to complete the expectation maximization step. Starting from 24 initial states, a solution with 14 states is found. The state means and standard deviations and corresponding PFA_u values for the model are listed in Table 7.4.

7.5.4.4 Macro state 4: Execution times above 22.5 ms

346 observations from the execution time sequence belong in macro state 4, $c_i \geq 22.5$ ms. The tree-based cross validation starting with 15 states identifies

Table 7.3: State means and standard deviations (in nanoseconds) and corresponding PFA_u values with the estimated model for macrostate 2.

State	1	2	3	4	5	6	7
mean	465 766	418 676	252 165	446 819	552 131	530 612	399 345
stddev	9 790	9 832	13 083	4 442	33 902	5 549	3 868
PFA_u	0.35	0.45	0.46	0.39	0.18	0.20	0.48

Table 7.3: State means and standard deviations (in nanoseconds) and corresponding PFA_u values with the estimated model for macrostate 2 (continued).

State	8	9	10	11	12	13	All
mean	773 326	681 703	481 032	301 732	586 839	452 460	NA
stddev	37 217	9 397	13 513	13 543	13 245	4 530	NA
PFA_u	0.24	0.20	0.31	0.46	0.17	0.37	0.37

a model with 8 states. The state means and standard deviations and corresponding PFA_u values for the model estimated from the execution time sequence are shown in Table 7.5.

7.6 Discussion

The evaluation allows us to conclude that a Hidden Markov Model with Gaussian emission distributions can be appropriate to model execution time sequence data, and that the proposed framework can be used to identify and validate such a model.

The analysis of the Markov Chain test program shows that the methods can be used to estimate the number of modes, the transition matrix, means and standard deviations to fit the model. While the test program is constructed to display Markov Chain properties, we show that the execution time distributions in each state can be modeled by a combination of modes with Gaussian emission distributions. Compared to the video decompression test, the program has a simple structure and a small memory footprint.

We also note that a Hidden Markov Model with Gaussian emission distributions appears to be valid in relation to the execution time sequences in the video decompression test.

The `depmixS4` methods used in the tree-based cross validation step are somewhat sensitive to the initial number of states, and the step may fail if this number is too large or too small. These methods can also fail if there are significant gaps between the execution time values, which is why the video

Table 7.4: State means and standard deviations (in nanoseconds) and corresponding PFA_u values with the estimated model for macrostate 3.

State	**1**	**2**	**3**	**4**	**5**	**6**	**7**
mean	12 146 116	9 907 489	9 408 568	11 763 973	8 753 004	13 172 000	13 071 808
stddev	237 788	253 316	284 388	481 965	306 717	328 076	636 583
PFA_u	0.07	0.16	0.17	0.08	0.17	0.03	0.07

Table 7.4: State means and standard deviations (in nanoseconds) and corresponding PFA_u values with the estimated model for macrostate 3 (continued).

State	**8**	**9**	**10**	**11**	**12**	**13**	**14**	**All**
mean	11 726 350	15 321 999	10 722 598	10 652 123	8 223 196	10 074 260	11 246 543	NA
stddev	276 281	2 759 114	288 845	456 994	332 875	286 002	314 261	NA
PFA_u	0.08	0.45	0.14	0.18	0.00	0.16	0.07	0.80

Table 7.5: State means and standard deviations (in nanoseconds) and corresponding PFA_u values with the estimated model for macrostate 4.

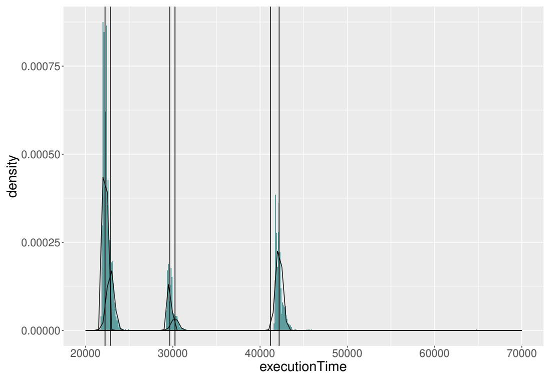
State	**1**	**2**	**3**	**4**	**5**	**6**	**7**	**8**	**All**
mean	22 979 652	22 881 488	23 863 163	22 786 818	22 733 574	23 365 667	23 094 934	23 198 326	NA
stddev	57 727	682 112	197 495	36 255	66 411	140 413	54 839	110 609	NA
PFA_u	0.39	0.36	0.37	0.36	0.35	0.32	0.30	0.38	0.57

decompression sequence is separated into macrostates. The framework could be expanded to manage separation into macrostates and find a suitable initial number of states.

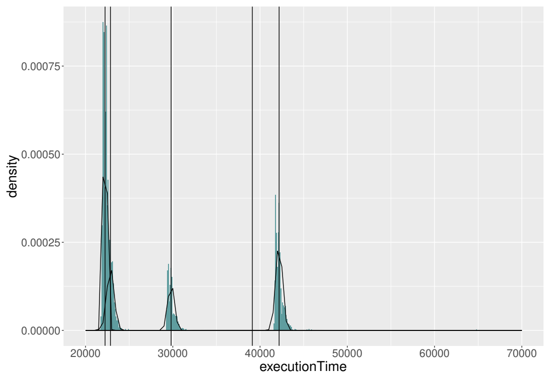
We also note that in some cases the number of states in the final models vary significantly. The heuristic algorithm for splitting the nodes could be adapted to evaluate a more exhaustive selection of possible splits, or replaced by an optimization algorithm such as for example simulated annealing [23].

Due to randomization utilized in many of the methods within the framework, different random seeds cause varying results. This is illustrated in Figure 7.4, where the Gaussian distributions of two models are visualized on top of a normalized histogram of the sequence they are estimated from. The Gaussian distributions are scaled with their respective stationary distribution probabilities.

We have conducted preliminary tests with the validation step performed with observations from running video decompression on another part of the "Tears of Steel" movie, that indicate that the identified model is not valid in this case. This may be due to input dependencies or cache related effects that



(a)



(b)

Figure 7.4: Normalized histograms of execution times (in nanoseconds) of the execution time sequence of the Markov chain test program. Two different estimated models from different random seeds are visualized with the Gaussian distributions of the states scaled with their respective stationary distribution probability, and their means displayed as vertical lines. In (a) we see the six state model from Section 7.5.3, and in (b) a five state model estimated with the framework applied to the same execution time sequence but initialized with another random seed.

cause the Gaussian distribution parameters and transition matrix to change over time and between runs.

Chen and Beltrame [10] show that effects of a random replacement cache can be described by an adaptive Markov Chain, and the ARM processor on the Raspberry Pi applies a pseudo-random cache replacement policy. Our methods derive a homogeneous Markov Chain, and if the model changes significantly during the sequence used for model parameter estimation, the model will not be valid, and this will be reflected in the resulting PFA_u values.

Finally, we note that cache-related jitter in our evaluations may be exaggerated by the `ftrace` process running simultaneously.

7.7 Conclusion and Future Work

This work proposed a measurement-based framework for probabilistic modeling of execution times of real-time applications. It presented an end-to-end workflow that first identifies the structure of a Markov Chain model and fits the probabilistic distributions to the collected execution time data, and finally validates the obtained model on the collected data based on a data consistency approach.

As with all measurement-based approaches, the application of this framework requires that the observations used at analysis are representative of the observations at runtime.

In order for the models to be useful in cases where full representativity of the observations at analysis time is not realistic to achieve, the methods described in this paper need to be complemented with (i) a method for providing a safe over-approximation of the execution time distribution, and (ii) a method for dynamically updating the model to reflect the effects on execution time patterns due to changes in input, program state or hardware state.

It is worth noticing that the proposed framework presents a consistent combination of different probabilistic tools, but it can include other techniques as alternatives. For example, the approach proposed in [20] for the identification of the Markov model can be used in the first step of the proposed framework, as an alternative method. Further investigation on the tradeoffs among different techniques is needed, and it is deferred to future work.

The framework could be further extended and automated, e.g., by specifying required limits on the PFA_u values. If these are too close to 0 or 1, one can reject the model. Attempts can be made to identify new models in an iterative manner, until we find a model that is not rejected, or we reach an iteration limit and deem the proposed model not consistent with the observations.

Finally, this paper focused on the single use case of video decompression. Other use cases will be analyzed in the future to better understand and investigate benefits and drawbacks of different probabilistic tools that can be included in this framework.

Bibliography

[1] Luca Abeni and Giorgio Buttazzo. Integrating multimedia applications in hard real-time systems. In *IEEE Real-Time Systems Symp. (RTSS)*, pages 4–13, 1998.

[2] Luca Abeni and Giorgio Buttazzo. Qos guarantee using probabilistic deadlines. In *Euromicro Conf. on Real-Time Systems (ECRTS)*, pages 242–249, 1999.

[3] Luca Abeni and Giorgio Buttazzo. Stochastic analysis of a reservation based system. In *Int. Workshop on Parallel and Distributed Real-Time Systems*, volume 1, 2001.

[4] Luca Abeni, Daniele Fontanelli, Luigi Palopoli, and Bernardo Villalba Frías. A markovian model for the computation time of real-time applications. In *IEEE international instrumentation and measurement technology conference (I2MTC)*, pages 1–6, 2017.

[5] Luca Abeni, Nicola Manica, and Luigi Palopoli. Efficient and robust probabilistic guarantees for real-time tasks. *Journal of Systems and Software*, 85(5):1147–1156, 2012.

[6] Sebastian Altmeyer, Liliana Cucu-Grosjean, and Robert I Davis. Static probabilistic timing analysis for real-time systems using random replacement caches. *Real-Time Systems*, 51(1):77–123, 2015.

[7] Alan Burns and Robert I. Davis. A survey of research into mixed criticality systems. *ACM Comput. Surv.*, 50(6), 2017.

[8] Alan Burns and Stewart Edgar. Predicting computation time for advanced processor architectures. In *Euromicro Conf. on Real-Time Systems (ECRTS)*, pages 89–96, 2000.

[9] Francisco J. Cazorla, Leonidas Kosmidis, Enrico Mezzetti, Carles Hernandez, Jaume Abella, and Tullio Vardanega. Probabilistic worst-case timing analysis: Taxonomy and comprehensive survey. *ACM Computing Surveys (CSUR)*, 52(1), 2019.

[10] Chao Chen and Giovanni Beltrame. An adaptive markov model for the timing analysis of probabilistic caches. *ACM Trans. Design Automation of Electronic Systems (TODAES)*, 23(1):1–24, 2017.

[11] David D. Clark, Scott Shenker, and Lixia Zhang. Supporting real-time applications in an integrated services packet network: Architecture and mechanism. *SIGCOMM Comput. Commun. Rev.*, 22(4):14–26, 1992.

[12] Tommaso Cucinotta, Luca Abeni, Mauro Marinoni, Alessio Balsini, and Carlo Vitucci. Reducing temporal interference in private clouds through real-time containers. In *IEEE International Conference on Edge Computing (EDGE)*, pages 124–131, 2019.

[13] Liliana Cucu-Grosjean, Luca Santinelli, Michael Houston, Code Lo, Tullio Vardanega, Leonidas Kosmidis, Jaume Abella, Enrico Mezzetti, Eduardo Quiñones, and Francisco J Cazorla. Measurement-Based probabilistic timing analysis for multi-path programs. In *Euromicro Conf. on Real-Time Systems (ECRTS)*, pages 91–101, 2012.

[14] Robert Ian Davis and Liliana Cucu-Grosjean. A survey of probabilistic schedulability analysis techniques for Real-Time systems. *LITES: Leibniz Transactions on Embedded Systems*, pages 1–53, 2019.

[15] Robert Ian Davis and Liliana Cucu-Grosjean. A survey of probabilistic timing analysis techniques for Real-Time systems. *LITES: Leibniz Transactions on Embedded Systems*, 6(1):03–1–03:60, 2019.

[16] José Luis Díaz, Daniel F García, Kanghee Kim, Chang-Gun Lee, L Lo Bello, José María López, Sang Lyul Min, and Orazio Mirabella. Stochastic analysis of periodic real-time systems. In *IEEE Real-Time Systems Symp. (RTSS)*, pages 289–300, 2002.

[17] Jose Luis Diaz, Jose Maria Lopez, Manuel Garcia, Antonio M Campos, Kanghee Kim, and Lucia Lo Bello. Pessimism in the stochastic analysis of real-time systems: Concept and applications. In *IEEE Real-Time Systems Symp. (RTSS)*, pages 197–207, 2004.

[18] Bogdan Doytchinov, John Lehoczky, and Steven Shreve. Real-time queues in heavy traffic with earliest-deadline-first queue discipline. *Annals of Applied Probability*, pages 332–378, 2001.

[19] Stewart Edgar and Alan Burns. Statistical analysis of wcet for scheduling. In *IEEE Real-Time Systems Symp. (RTSS)*, pages 215–224, 2001.

[20] Bernardo Villalba Frias, Luigi Palopoli, Luca Abeni, and Daniele Fontanelli. Probabilistic real-time guarantees: There is life beyond the i.i.d. assumption. In *IEEE Real-Time and Embedded Technology and Applications Symposium (RTAS)*, pages 175–186, 2017.

[21] David Griffin and Alan Burns. Realism in statistical analysis of worst case execution times. In *Int. Workshop on Worst-Case Execution Time Analysis (WCET)*, 2010.

[22] Giordano A Kaczynski, Lucia Lo Bello, and Thomas Nolte. Deriving exact stochastic response times of periodic tasks in hybrid priority-driven soft real-time systems. In *IEEE Int. Conf. on Emerging Technologies & Factory Automation (ETFA)*, pages 101–110, 2007.

[23] Scott Kirkpatrick, C Daniel Gelatt, and Mario P Vecchi. Optimization by simulated annealing. *science*, 220(4598):671–680, 1983.

[24] John P Lehoczky. Real-time queueing theory. In *IEEE Real-Time Systems Symp. (RTSS)*, pages 186–195, 1996.

[25] Benjamin Lesage, David Griffin, Sebastian Altmeyer, Liliana Cucu-Grosjean, and Robert I Davis. On the analysis of random replacement caches using static probabilistic timing methods for multi-path programs. *Real-Time Systems*, 54(2):307–388, 2018.

[26] Benjamin Lesage, David Griffin, Sebastian Altmeyer, and Robert I Davis. Static probabilistic timing analysis for multi-path programs. In *IEEE Real-Time Systems Symp. (RTSS)*, pages 361–372, 2015.

[27] Andreas Lindholm, Dave Zachariah, Petre Stoica, and Thomas B Schön. Data consistency approach to model validation. *IEEE Access*, 7:59788–59796, 2019.

[28] Andreas Löfwenmark and Simin Nadjm-Tehrani. Fault and timing analysis in critical multi-core systems: A survey with an avionics perspective. *Journal of Systems Architecture*, 87:1–11, 2018.

[29] Yue Lu, Thomas Nolte, Iain Bate, and Liliana Cucu-Grosjean. A statistical response-time analysis of complex real-time embedded systems by using timing traces. In *IEEE Int. Symp. on Industrial and Embedded Systems (SIES)*, pages 43–46, 2011.

[30] Yue Lu, Thomas Nolte, Iain Bate, and Liliana Cucu-Grosjean. A statistical response-time analysis of real-time embedded systems. In *IEEE Real-Time Systems Symp. (RTSS)*, pages 351–362, 2012.

[31] Yue Lu, Thomas Nolte, Johan Kraft, and Christer Norstrom. 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 (RTCSA)*, pages 153–160, 2010.

[32] Yue Lu, Thomas Nolte, Johan Kraft, and Christer Norstrom. Statistical-based response-time analysis of systems with execution dependencies between tasks. In *IEEE Int. Conf. on Engineering of Complex Computer Systems*, pages 169–179, 2010.

[33] Nicola Manica, Luigi Palopoli, and Luca Abeni. Numerically efficient probabilistic guarantees for resource reservations. In *IEEE Int. Conf. on Emerging Technologies & Factory Automation (ETFA)*, pages 1–8, 2012.

[34] Pau Martí, Josep M Fuertes, Gerhard Fohler, and Krithi Ramamritham. Improving quality-of-control using flexible timing constraints: metric and scheduling. In *IEEE Real-Time Systems Symp. (RTSS)*, pages 91–100, 2002.

[35] Todd K. Moon. The expectation-maximization algorithm. *IEEE Signal processing magazine*, 13(6):47–60, 1996.

[36] Luigi Palopoli, Daniele Fontanelli, Luca Abeni, and Bernardo Villalba Frias. An analytical solution for probabilistic guarantees of reservation based soft real-time systems. *IEEE Trans. Parallel and Distributed Systems*, 27(3):640–653, 2015.

[37] Eduardo Quinones, Emery D Berger, Guillem Bernat, and Francisco J Cazorla. Using randomized caches in probabilistic real-time systems. In *Euromicro Conf. on Real-Time Systems (ECRTS)*, pages 129–138, 2009.

[38] Lawrence R. Rabiner. A tutorial on hidden markov models and selected applications in speech recognition. *Proc. the IEEE*, 77(2):257–286, 1989.

[39] Takahiro Shinozaki. Hmm state clustering based on efficient cross-validation. In *2006 IEEE International Conference on Acoustics Speech and Signal Processing Proceedings*, volume 1, 2006.

[40] Václav Struhár, Moris Behnam, Mohammad Ashjaei, and Alessandro Vittorio Papadopoulos. Real-time containers: A survey. In *Workshop on Fog Computing and the IoT (Fog-IoT)*, volume 80, pages 7:1–7:9, 2020.

[41] Aida Čaušević, Alessandro Vittorio Papadopoulos, and Marjan Sirjani. Towards a framework for safe and secure adaptive collaborative systems. In *IEEE Annual Computer Software and Applications Conf. (COMPSAC)*, volume 2, pages 165–170, 2019.

[42] Ingmar Visser, Maarten Speekenbrink, et al. depmixs4: an R package for hidden markov models. *Journal of Statistical Software*, 36(7):1–21, 2010.

[43] Reinhard Wilhelm. Mixed Feelings About Mixed Criticality (Invited Paper). In *Int. Workshop on Worst-Case Execution Time Analysis (WCET)*, volume 63, pages 1:1–1:9, 2018.

Chapter 8

Paper B
Adaptive Runtime Estimate of Task Execution Times using Bayesian Modeling.

Anna Friebe, Filip Marković, Alessandro V. Papadopoulos, and Thomas Nolte.
In IEEE International Conference on Embedded and Real-Time Computing Systems and Applications (RTCSA), 2021.

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.

8.1 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 [33, 17]). On the other hand, hardware acceleration features, multi-core systems [20] and complex, interacting tasks, e.g., in mixed criticality systems [3, 32], pose several challenges in achieving tight bounds on the WCET and Worst-Case Response-Time (WCRT) [17].

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) [26]. 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 [9, 10, 5]. Measurement-based techniques based on Extreme Value Theory (EVT) [4, 12, 15] require that extreme values of the execution time distribution are independent and that the measurements contain samples from the worst case distribution [30, 18]. 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.[21]. 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 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 8.2. In Section 8.3, we describe the system-model assumptions, along with definitions and mathematical background used in the paper. Then, in Section 8.4 we describe the derivation of the initial HMM in the preprocessing step, which is followed by Section 8.5, the description of the method for online model-parameter adaptation. The evaluation is described in Section 8.6, and the paper is concluded in Section 8.7.

8.2 Related Work

Two major surveys on the Probabilistic Timing Analysis [9] and the Probabilistic Schedulability Analysis [8] 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. [5]. 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 [7], based on previous work related to the use of Extreme Value Theory (EVT) [4, 12, 15]. 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 [30, 18]. EVT-based techniques have also been applied in order to estimate upper bounds on response time distributions [25, 24, 22, 23].

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. [1, 13]. 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. [14] 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 [14] 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. [13], 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 [14], but we propose a runtime Bayesian adaptation method to continuously refine the execution time model based on the new observations.

Lu et al. [21] 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. [16] 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 distri-

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
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 8.1: Important notation used in this work.

butions 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.

8.3 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 8.1 lists the main symbols used in the paper.

8.3.1 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 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_n^2, i.e., $C_n \sim \mathcal{N}(\mu_n, \sigma_n^2)$.

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 $\{\mu_n, \sigma_n^2\}$ 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 $\{\mu_n, \sigma_n^2\}$ remain constant are referred to as *segments* s_j. In each segment s_j, the mean and variance are denoted $\{\mu_{nj}, \sigma_{nj}^2\}$. A set $S_k = \{s_j, \ldots\}$ of non-adjacent segments with the same values of $\{\mu_{nj}, \sigma_{nj}^2\}$ are referred to as a *cluster*. An illustration is shown in Figure 8.1.

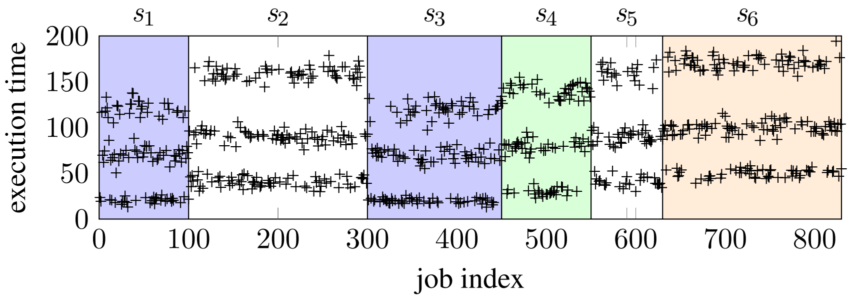


Figure 8.1: An execution time sequence separated into six *segments* $(s_1, s_2, s_3, s_4, s_5, s_6)$ and four *clusters* (S_1, S_2, S_3, S_4), where $S_1 = \{s_1, s_3\}$, $S_2 = \{s_2, s_5\}$, $S_3 = \{s_4\}$, and $S_4 = \{s_6\}$.

8.3.2 Estimating Sufficient Statistics

The Markov Model $\{\mathcal{M}, \mathcal{P}, \mathcal{C}\}$ can be estimated by the use of the Forward-Backward algorithm [29] in combination with the Expectation Maximization algorithm [27]. The number of states N can be determined as described in Section 8.4. 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 [14, 31]) 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.

$$\hat{\mathbf{a}}[0]_{jn} = \sum_{i, c_i \in s_j} \gamma_{ni}, \tag{8.1}$$

$$\hat{\mathbf{a}}[1]_{jn} = \sum_{i, c_i \in s_j} c_i \gamma_{ni}, \tag{8.2}$$

$$\hat{\mathbf{a}}[2]_{jn} = \sum_{i, c_i \in s_j} c_i^2 \gamma_{ni}. \tag{8.3}$$

8.3.3 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 $p(\Theta)$ [2]. For a Gaussian probability distribution with unknown mean μ and precision $\lambda = 1/\sigma^2$, the conjugate distribution is a Normal-Gamma distribution [28], denoted as $NG(\mu, \lambda)$. When we have a prior distribution of μ and λ as given by

$$\begin{aligned} p(\mu, \lambda) = NG(\mu, \lambda|\mu_0, \kappa_0, \alpha_0, \beta_0) &\triangleq \\ \mathcal{N}(\mu|\mu_0, (\kappa_0\lambda)^{-1}) Ga(\lambda|\alpha_0, rate = \beta_0)&, \end{aligned} \tag{8.4}$$

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

$$p(\mu, \lambda|D) = NG(\mu, \lambda|\mu_L, \kappa_L, \alpha_L, \beta_L), \tag{8.5}$$

$$\mu_L = \frac{\kappa_0\mu_0 + \sum_{i=1}^{L} x_i}{\kappa_0 + L}, \tag{8.6}$$

$$\kappa_L = \kappa_0 + L, \tag{8.7}$$

$$\alpha_L = \alpha_0 + L/2, \tag{8.8}$$

$$\beta_L = \beta_0 + \frac{1}{2}\left(\sum_{i=1}^{L}(x_i - \bar{x})^2 + \frac{\kappa_0 L(\bar{x} - \mu_0)^2}{(\kappa_0 + L)}\right). \tag{8.9}$$

Given that we are not certain of which state each observation c_i belongs to, we rewrite Equations (8.6) to (8.9), using the sufficient statistics in Equations (8.1) to (8.3), yielding:

$$\mu_L = \frac{\kappa_0\mu_0 + \hat{\mathbf{a}}[1]}{\kappa_0 + \hat{\mathbf{a}}[0]}, \tag{8.10}$$

$$\kappa_L = \kappa_0 + \hat{\mathbf{a}}[0], \tag{8.11}$$

$$\alpha_L = \alpha_0 + \hat{\mathbf{a}}[0]/2, \tag{8.12}$$

$$\beta_L = \beta_0 + \frac{1}{2}\left(\hat{\mathbf{a}}[2] - \frac{\hat{\mathbf{a}}[1]^2}{\hat{\mathbf{a}}[0]} + \frac{\kappa_0\hat{\mathbf{a}}[0](\frac{\hat{\mathbf{a}}[1]}{\hat{\mathbf{a}}[0]} - \mu_0)^2}{(\kappa_0 + \hat{\mathbf{a}}[0])}\right), \tag{8.13}$$

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 $\hat{\mathbf{a}}[2]_{jn}$ to estimate $L\bar{x}^2$. Equations (8.10) to (8.13) 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:

$$\mu_L = \frac{\kappa_0\mu_0 - \hat{\mathbf{a}}[1]}{\kappa_0 - \hat{\mathbf{a}}[0]}, \tag{8.14}$$

$$\kappa_L = \kappa_0 - \hat{\mathbf{a}}[0], \tag{8.15}$$

$$\alpha_L = \alpha_0 - \hat{\mathbf{a}}[0]/2, \tag{8.16}$$

$$\beta_L = \beta_0 - \frac{1}{2}\left(\hat{\mathbf{a}}[2] - \frac{\hat{\mathbf{a}}[1]^2}{\hat{\mathbf{a}}[0]} + \frac{\kappa_0\hat{\mathbf{a}}[0](\frac{\hat{\mathbf{a}}[1]}{\hat{\mathbf{a}}[0]} - \mu_0)^2}{(\kappa_0 - \hat{\mathbf{a}}[0])}\right). \tag{8.17}$$

Note that the posterior predictive distribution for a single point prediction is the Student's t-distribution as described by Murphy [28]:

$$p(x|D) = t_{2\alpha_L}\left(\frac{\beta_L(\kappa_l + 1)}{\alpha_L\kappa_L}\right). \tag{8.18}$$

8.3.4 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 [19]:

$$\mathrm{GLR}(S_k, S_l) = \ell_{k\cup l} - (\ell_k + \ell_l). \tag{8.19}$$

The posterior predictive distribution for m new observations given the Normal-Gamma prior is given in Murphy [28] as

$$p(D_{new}|D) = \frac{\Gamma(\alpha_{n+m})}{\Gamma(\alpha_n)}\frac{\beta_n^{\alpha_n}}{\beta_{n+m}^{\alpha_{n+m}}}\sqrt{\frac{\kappa_n}{\kappa_{n+m}}}(2\pi)^{m/2}. \tag{8.20}$$

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 Equations (8.11) to (8.13) as:

$$\begin{aligned}\ell_k = {} & \log\Gamma(\alpha_{2k}) - \log\Gamma(\alpha_k) + \alpha_k\log\beta_k - \\ & \alpha_{2k}\log\beta_{2k} + \frac{1}{2}(\kappa_k - \kappa_{2k}) + \frac{\hat{\mathbf{a}}[0]_k}{2}2\pi.\end{aligned} \tag{8.21}$$

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

$$\begin{aligned}\mathrm{GLR}(S_k, S_l) =& \log\Gamma(\alpha_{2(k\cup l)}) - \log\Gamma(\alpha_{(k\cup l)}) + \alpha_{k\cup l}\log\beta_{k\cup l}\\ &- \alpha_{2(k\cup l)}\log\beta_{2(k\cup l)} + \frac{1}{2}(\kappa_{k\cup l} - \kappa_{2(k\cup l)})\\ &- \log\Gamma(\alpha_{2k}) + \log\Gamma(\alpha_k) - \alpha_k\log\beta_k\\ &+ \alpha_{2k}\log\beta_{2k} - \frac{1}{2}(\kappa_k - \kappa_{2k})\\ &- \log\Gamma(\alpha_{2l}) + \log\Gamma(\alpha_l) - \alpha_l\log\beta_l\\ &+ \alpha_{2l}\log\beta_{2l} - \frac{1}{2}(\kappa_l - \kappa_{2l})\end{aligned} \tag{8.22}$$

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 [19].

8.4 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 [14]. 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:

$$\mu_0 = \mu, \tag{8.23}$$

$$\kappa_0 = nPseudoObs, \tag{8.24}$$

$$\alpha_0 = \frac{nPseudoObs}{2}, \tag{8.25}$$

$$\beta_0 = \alpha_0 \cdot \sigma^2. \tag{8.26}$$

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

8.4.1 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.

8.4.1.1 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

$$\mathrm{GLR}(x) = \mathrm{GLR}(\{s_{x-}\}, \{s_{x+}\}) \tag{8.27}$$

$$s_{x-} = (c_{x_{start}}, c_{x_{start}+1}, \ldots, c_x) \tag{8.28}$$

$$s_{x+} = (c_{x+1}, c_{x_{split}+1}, \ldots, c_{x_{stop}}) \tag{8.29}$$

$$x_{split} = \arg\min_x \mathrm{GLR}(x) \tag{8.30}$$

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.

8.4.1.2 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 8.4.1.1 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};

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

2. The length of the subsequence is below a minimum length between splitting points.

8.4.2 Segment Clustering

The segments are clustered into sets using an approach similar to the Leader-Follower Clustering described by Duda et al. [11]. 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 8.3.4. 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.

8.5 Online Model Adaptation

In the runtime process, a sliding window is considered. A simplified flowchart of the algorithm is available in Figure 8.2.

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 Equations (8.10) to (8.13) 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 [29]. 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 Equation (8.18). The prior distribution is chosen as outlined in Section 8.4, Equations (8.23) to (8.26).

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 8.2.

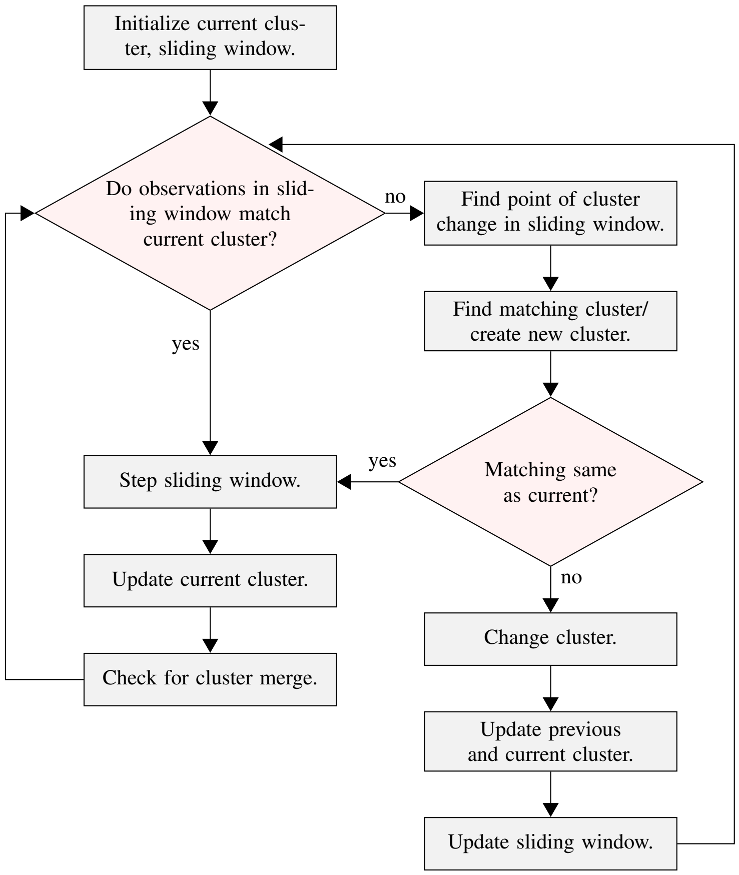

Initialize current cluster, sliding window.
Do observations in sliding window match current cluster?
no
Find point of cluster change in sliding window.
Find matching cluster/ create new cluster.
yes
Matching same as current?
yes
Step sliding window.
Update current cluster.
Check for cluster merge.
no
Change cluster.
Update previous and current cluster.
Update sliding window.

Figure 8.2: Simplified flowchart of the adaptive process. The process continues until the task is terminated and there are no more observations to process.

8.5.1 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 $slidingLimit$, we move into the right column of Figure 8.2. 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 Equations (8.10) to (8.13) 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 8.1.

Determining the Point of Cluster Change: The sliding window is divided

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

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.

Algorithm 8.1 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.

```
 1: function CLUSTERCHANGE(clusters, preClusters, clusterFindSegment,slidingWindow,
    currentCluster)
 2:     closestClusterAll ← $\arg\max_{c \in \text{clusters}} GLR(c, \text{clusterFindSegment})$
 3:     potentialChangePoint ← FINDCHANGEPOINT(slidingWindow, currentCluster, closest-
    ClusterAll)
 4:     testEndSegment ← slidingWindow[potentialChangePoint:end]
 5:     testNGAll ← POSTERIORNG(closestClusterAll, testEndSegmentNG)
 6:     testGLRAll ← $GLR$(closestClusterAll, testNGAll)
 7:     if testGLRAll < newClusterLimit then
 8:         newCluster ← CREATECLUSTER(testEndSegment)
 9:         return potentialChangePoint, newCluster
10:     end if
11:     closestClusterPre ← $\arg\max_{c \in \text{preClusters}} GLR(c, \text{clusterFindSegment})$
12:     testNGPre ← POSTERIORNG(closestClusterPre, testEndSegmentNG)
13:     testGLRPre ← $GLR$(closestClusterPre, testNGPre)
14:     if testGLRPre > changePreLimit then
15:         return potentialChangePoint, closestClusterPre
16:     end if
17:     return potentialChangePoint, closestClusterAll
18: end function
```

8.5.2 Updating the Sliding Window and Clusters

If a cluster change has taken place, we continue downwards in the right column of Figure 8.2. 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 Figure 8.2. 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 Equations (8.10) to (8.13) and Equations (8.14) to (8.17). 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.

8.5.3 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.

8.6 Evaluation

8.6.1 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 dis-

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

tribution. 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 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

$$D_{KL}(P||Q) = \int_{-\infty}^{\infty} p(x) \log \frac{p(x)}{q(x)}\, dx. \tag{8.31}$$

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.

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 8.3: KL divergence measures for the preprocessing process.

8.6.2 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.

8.6.3 Results

8.6.3.1 Preprocessing Step

In Figure 8.3 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 Figure 8.3, 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 8.3.

8.6.3.2 Online Adaptive Process

The means and standard deviations of the posterior generalized student's t distributions during the adaptive process are displayed in blue in Figure 8.3 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 Figure 8.3 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 Figure 8.3. Means are calculated for each ground truth cluster and for the entire adaptive part of the sequence, and presented in Table 8.4. 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.

8.6.4 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.

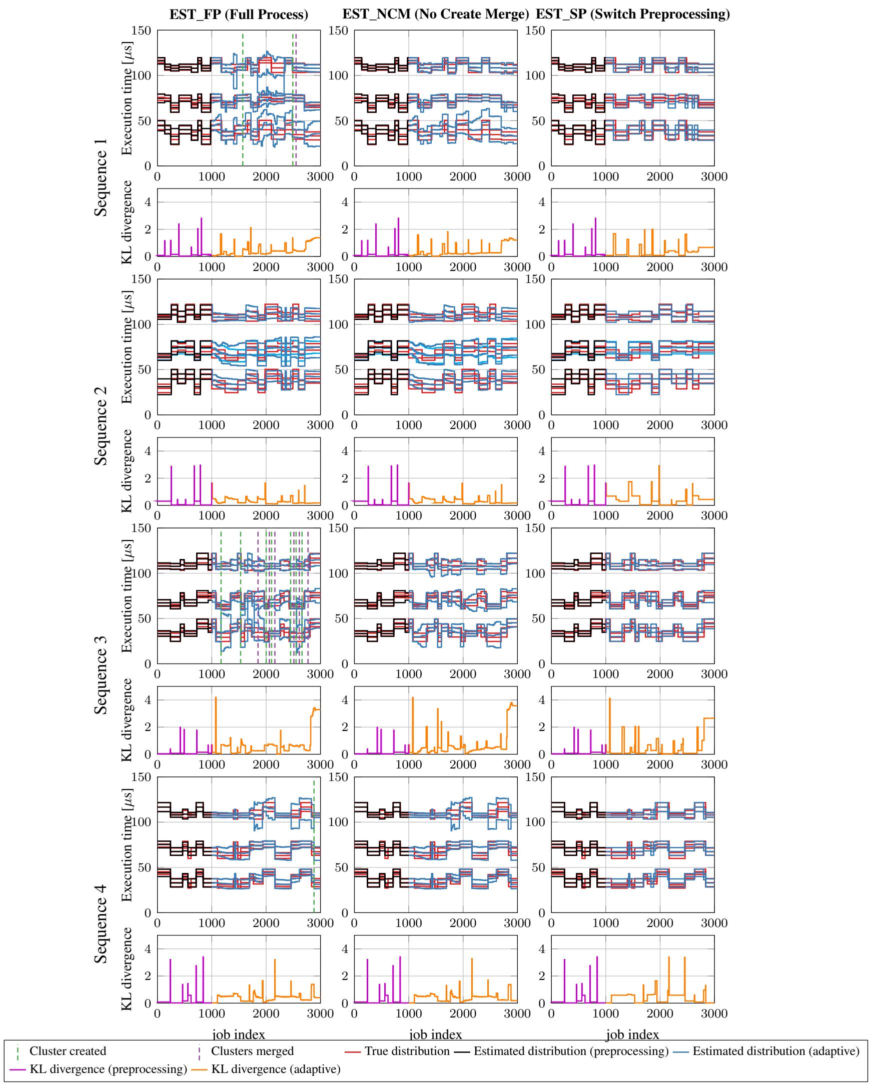


Figure 8.3: True and predicted distributions for four sequences, with the three different versions of the process. KL divergence measures along the sequences are displayed.

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 8.4: KL divergence measures for different sequences and clusters.

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.

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.

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

8.7 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.

Bibliography

[1] Luca Abeni, Daniele Fontanelli, Luigi Palopoli, and Bernardo Villalba Frías. A markovian model for the computation time of real-time applications. In *IEEE international instrumentation and measurement technology conference (I2MTC)*, pages 1–6, 2017.

[2] Christopher M. Bishop. *Pattern recognition and machine learning*. springer, 2006.

[3] Alan Burns and Robert I. Davis. A survey of research into mixed criticality systems. *ACM Computing Surveys (CSUR)*, 50(6), 2017.

[4] Alan Burns and Stewart Edgar. Predicting computation time for advanced processor architectures. In *Euromicro Conf. on Real-Time Systems (ECRTS)*, pages 89–96, 2000.

[5] Francisco J. Cazorla, Leonidas Kosmidis, Enrico Mezzetti, Carles Hernandez, Jaume Abella, and Tullio Vardanega. Probabilistic worst-case timing analysis: Taxonomy and comprehensive survey. *ACM Computing Surveys (CSUR)*, 52(1), 2019.

[6] David D. Clark, Scott Shenker, and Lixia Zhang. Supporting real-time applications in an integrated services packet network: Architecture and mechanism. *SIGCOMM Comput. Commun. Rev.*, 22(4):14–26, 1992.

[7] Liliana Cucu-Grosjean, Luca Santinelli, Michael Houston, Code Lo, Tullio Vardanega, Leonidas Kosmidis, Jaume Abella, Enrico Mezzetti, Eduardo Quiñones, and Francisco J Cazorla. Measurement-Based probabilistic timing analysis for multi-path programs. In *Euromicro Conf. on Real-Time Systems (ECRTS)*, pages 91–101, 2012.

[8] Robert Ian Davis and Liliana Cucu-Grosjean. A survey of probabilistic schedulability analysis techniques for Real-Time systems. *LITES: Leibniz Transactions on Embedded Systems*, pages 1–53, 2019.

[9] Robert Ian Davis and Liliana Cucu-Grosjean. A survey of probabilistic timing analysis techniques for Real-Time systems. *LITES: Leibniz Transactions on Embedded Systems*, 6(1):03–1–03:60, 2019.

[10] Jose Luis Diaz, Jose Maria Lopez, Manuel Garcia, Antonio M Campos, Kanghee Kim, and Lucia Lo Bello. Pessimism in the stochastic analysis of real-time systems: Concept and applications. In *IEEE Real-Time Systems Symp. (RTSS)*, pages 197–207, 2004.

[11] Richard O Duda, Peter E Hart, et al. *Pattern classification and scene analysis*, volume 3. Wiley New York, 1973.

[12] Stewart Edgar and Alan Burns. Statistical analysis of wcet for scheduling. In *IEEE Real-Time Systems Symp. (RTSS)*, pages 215–224, 2001.

[13] Bernardo Villalba Frias, Luigi Palopoli, Luca Abeni, and Daniele Fontanelli. Probabilistic real-time guarantees: There is life beyond the i.i.d. assumption. In *IEEE Real-Time and Embedded Technology and Applications Symposium (RTAS)*, pages 175–186, 2017.

[14] Anna Friebe, Alessandro V. Papadopoulos, and Thomas Nolte. Identification and validation of markov models with continuous emission distributions for execution times. In *IEEE Int. Conf. on Embedded and Real-Time Computing Systems and Applications (RTCSA)*, pages 1–10, 2020.

[15] David Griffin and Alan Burns. Realism in statistical analysis of worst case execution times. In *Int. Workshop on Worst-Case Execution Time Analysis (WCET)*, 2010.

[16] Christian Gruhl, Jörn Schmeißing, Sven Tomforde, and Bernhard Sick. Normal-wishart clustering for novelty detection. In *IEEE International Conference on Autonomic Computing and Self-Organizing Systems Companion (ACSOS-C)*, pages 64–69. IEEE, 2020.

[17] Mathai Joseph and Paritosh Pandya. Finding response times in a real-time system. *The Computer Journal*, 29(5):390–395, 1986.

[18] M Ross Leadbetter, Georg Lindgren, and Holger Rootzén. Conditions for the convergence in distribution of maxima of stationary normal processes. *Stochastic Processes and their Applications*, 8(2):131–139, 1978.

[19] D Liu and Francis Kubala. Online speaker clustering. In *IEEE Int. Conf. on Acoustics, Speech, and Signal Processing*, volume 1, pages 333–336, 2004.

[20] Andreas Löfwenmark and Simin Nadjm-Tehrani. Fault and timing analysis in critical multi-core systems: A survey with an avionics perspective. *Journal of Systems Architecture*, 87:1–11, 2018.

[21] Chenyang Lu, John A Stankovic, Sang H Son, and Gang Tao. Feedback control real-time scheduling: Framework, modeling, and algorithms. *Real-Time Systems*, 23(1):85–126, 2002.

[22] Yue Lu, Thomas Nolte, Iain Bate, and Liliana Cucu-Grosjean. A statistical response-time analysis of complex real-time embedded systems by using timing traces. In *IEEE Int. Symp. on Industrial and Embedded Systems (SIES)*, pages 43–46, 2011.

[23] Yue Lu, Thomas Nolte, Iain Bate, and Liliana Cucu-Grosjean. A statistical response-time analysis of real-time embedded systems. In *IEEE Real-Time Systems Symp. (RTSS)*, pages 351–362, 2012.

[24] Yue Lu, Thomas Nolte, Johan Kraft, and Christer Norstrom. 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 (RTCSA)*, pages 153–160, 2010.

[25] Yue Lu, Thomas Nolte, Johan Kraft, and Christer Norstrom. Statistical-based response-time analysis of systems with execution dependencies between tasks. In *IEEE Int. Conf. on Engineering of Complex Computer Systems*, pages 169–179, 2010.

[26] Pau Martí, Josep M Fuertes, Gerhard Fohler, and Krithi Ramamritham. Improving quality-of-control using flexible timing constraints: metric and scheduling. In *IEEE Real-Time Systems Symp. (RTSS)*, pages 91–100, 2002.

[27] Todd K. Moon. The expectation-maximization algorithm. *IEEE Signal processing magazine*, 13(6):47–60, 1996.

[28] Kevin P. Murphy. Conjugate Bayesian analysis of the Gaussian distribution. Technical report, University of British Columbia, 2007.

[29] Lawrence R. Rabiner. A tutorial on hidden markov models and selected applications in speech recognition. *Proc. the IEEE*, 77(2):257–286, 1989.

[30] Luca Santinelli, Jérôme Morio, Guillaume Dufour, and Damien Jacquemart. On the sustainability of the extreme value theory for wcet estimation. In *Int. Workshop on Worst-Case Execution Time Analysis (WCET)*, 2014.

[31] Takahiro Shinozaki. Hmm state clustering based on efficient cross-validation. In *2006 IEEE International Conference on Acoustics Speech and Signal Processing Proceedings*, volume 1, 2006.

[32] Reinhard Wilhelm. Mixed Feelings About Mixed Criticality. In *Int. Workshop on Worst-Case Execution Time Analysis (WCET)*, volume 63, pages 1:1–1:9, 2018.

[33] Reinhard Wilhelm, Jakob Engblom, Andreas Ermedahl, Niklas Holsti, Stephan Thesing, David Whalley, Guillem Bernat, Christian Ferdinand, Reinhold Heckmann, Tulika Mitra, et al. The worst-case execution-time

problem—overview of methods and survey of tools. *ACM TECS*, 7(3):36, 2008.

Chapter 9

Paper C
Estimating Deadline Miss Probabilities of Continuous-Emission Markov Chain Tasks.

Anna Friebe, Filip Marković, Alessandro V. Papadopoulos, and Thomas Nolte.
Under review.

Abstract

Estimating the response times of real-time tasks and applications is important for the analysis and implementation of real-time systems. Probabilistic approaches have gained attention over the past decade, as they provide a modeling framework that allows for less pessimism for the analysis of real-time systems. Among the different proposed approaches, Markov chains have been shown to be effective for the analysis of real-time systems, in particular, in the estimate of the pending workload probability distribution and of the deadline miss probability. However, this has been analyzed only for discrete emission distributions, but not for continuous ones. In this paper, we propose a method for analyzing the workload probability distribution and bounding the deadline miss probability for a task executing in a Constant Bandwidth Server, where execution times are described by a Markov model with Gaussian emission distributions. In the evaluation, deadline miss probability bounds and estimates are derived with a workload accumulation scheme. The results are compared to simulation and measured deadline miss ratios from tasks under the Linux Constant Bandwidth Server implementation SCHED_DEADLINE.

9.1 Introduction

Real-time systems are commonly characterized as hard or soft real-time systems. In a hard real-time system deadlines must always be met, but in a soft real-time system deadline misses can be tolerated to some extent. Deadline misses in a soft real-time system lead to a deterioration of the Quality of Service (QoS) [10] or Quality of Control (QoC) [28]. The number of deadline misses must be sufficiently low so that the QoS or QoC is retained at an acceptable level [8].

Hidden Markov Models (HMMs) have been utilized to model execution times in systems with dependencies, and where there is regularity in the variation of the execution times. In [5, 16], the authors have modeled execution times as Markov models with discrete emission distributions, including estimating the deadline miss probability under a Constant Bandwidth Server (CBS). Emission distributions have also been modeled as continuous Gaussian distributions [18, 17], with the advantage of potentially providing more robust estimates from a lower number of samples. Gaussian distributions also allow for representation with only two parameters, as opposed to the case where individual probabilities of each discrete execution time value are stored. The application of HMMs with continuous emission distributions has been limited to the estimation of the sole execution time [18].

This paper focuses on the problem of bounding and estimating the deadline miss probability of a real-time application, exploiting HMMs. In the literature, two concepts related to probabilistic deadlines are commonly used. The Deadline Miss Probability (DMP) is interpreted as the ratio of missed deadlines to the number of jobs in a long (tending to infinite) time interval. The Worst-Case Deadline Failure Probability (WCDFP) is interpreted as an upper bound on the probability of a deadline miss for any single job [12]. In this paper, we focus on the DMP as the long-run frequency interpretation, for the overall HMM and for each state separately.

More specifically, in this paper, we address the problem of upper bounding the workload distribution and deadline miss probability under CBS of a periodic task where execution times are modeled by a Markov chain with Gaussian emission distributions. We propose an iterative workload accumulation scheme, where workload distributions are accumulated sequentially over task periods. The scheme starts from a point of workload depletion, that is a task period with zero carry-in workload. The method provides an upper bound on the deadline miss probability in each state and overall under certain assumptions.

The method is evaluated by comparing the obtained results with the deadline miss ratio of tasks running under the Linux kernel implementation of CBS, `SCHED_DEADLINE` [22], and with results from simulation.

The paper is structured as follows. In Section 9.2, related work is discussed. The notation used in the paper and the system model is outlined in Section 9.3. The analysis for upper bounding the deadline miss probability is described in Section 9.4, and in Section 9.5 the parts are combined in an overall workload accumulation process. In Section 9.6 the evaluation is presented and results are provided. Conclusions and future work are discussed in Section 9.7.

9.2 Related Work

Davis and Cucu-Grosjean provide a comprehensive survey on probabilistic schedulability analysis techniques [12], along with a survey on probabilistic timing analysis [13].

Díaz et al. [14] presented a response time analysis for periodic tasks where execution times are independent random variables and showed that the backlog is a Markov chain.

Maxim and Cucu-Grosjean [29] showed that in systems where execution times, deadlines, and interarrival times are independent random variables, the Worst-Case Response Time (WCRT) can be found by synchronous release if deadlines are constrained and jobs are aborted when their deadline is missed.

Ivers and Ernst [19] addressed the case where execution times are dependent and proposed the use of Fréchet bounds and probability boxes.

Extreme Value Theory (EVT) has been applied in measurement-based statistical analysis of response times to find the probabilistic WCRT (pWCRT). This is an upper bound on the probability of exceeding a response time for every valid sequence of program executions and is based on finding the distribution of the extreme values, the distribution's tail. Most of the work in this regard has been done by Lu et al. [26, 25, 24]. Maxim et al. [30] have shown that the methods based on EVT provide sound results. EVT is applicable in cases of dependence, as long as there is stationarity [20] or extremal independence [33].

Real-time queuing theory [21] provides methods for analyzing the response time distribution specifically in the case of heavy traffic when utilization is close to 1.

Bozhko et al. [7] proposed a response time analysis with Monte Carlo simulation for fixed-priority preemptive scheduling with execution times as independent random variables.

Von der Brüggen et al. [35] provided a method for over-approximating the WCDFP under EDF for tasks with different execution modes. This includes derivation for acyclic task chain dependencies among a bounded number of subsequent jobs. The number of intervals considered is substantially reduced due to the observation that the probability of a deadline miss in an interval is bounded by the probability that the processor does not idle in the same interval.

Mills and Anderson [31] provide response time and tardiness bounds for soft real-time tasks with stochastic execution times, in a server-based scheduler. In this work, execution time dependence is considered within but not across time windows. A larger window leads to greater tardiness bounds. Liu, Mills, and Anderson [23] further proposed the use of independence thresholds, where independence is assumed for execution times exceeding a determined threshold value.

The CBS is described in Section 9.3. It was introduced by Abeni and Buttazzo [2], and used to obtain probabilistic deadlines for QoS guarantees [3]. Analysis under CBS has been performed with execution times [4, 32] and interarrival times [6, 27] modeled with probability distributions.

Tasks with dependent execution times have been modeled as Markov chains and been analyzed under CBS by Frías et al. [16, 5]. The steady-state response time distribution was calculated. The results were compared to running the task under Linux `SCHED_DEADLINE`. The time required for the analysis depends on the range of computation times, the number of states, and the resampling factor [34].

Execution times have been modeled as continuous Gaussian distributions in the context of emission distributions in a Markov chain [18, 17]. We are not aware of any work that analyzes this model in terms of response times or deadline miss probabilities. In this paper, we aim to bridge this gap and enable the use of an HMM with Gaussian emission distributions for schedulability analysis. Similarly as in the work of Frías et al. [16, 5], dependencies are explicit in the HMM, and the task is running in a CBS. The CBS provides isolation from other tasks on the system, so that the pending workload considered is carry-in workload from previous jobs of the same task, instead of workload from other tasks as in most work concerning response times. The choice of Gaussian distribution is partly based on simplicity and tractability. In [18] a HMM with Gaussian emission distributions was shown to be a valid model in a video decompression case. Modeling the execution times of each state as a Gaussian distribution may seem simplistic. However, several states with Gaussian distributions can be combined to form a more general distribution shape. In addition, if the means of the states' distributions differ significantly, the Markov Model transition probabilities may affect the response times to a greater extent than the state distribution shapes. As an example, a high likelihood of several consecutive jobs in the state with the longest execution times will lead to much longer response times, compared to a case where there is high likelihood that a job in this state is followed by a job in the state with the shortest execution times. Nevertheless, the use of the Gaussian distribution is a limitation of this work, and therefore the method is also evaluated for non-Gaussian distributions. Here, an exponential distribution is chosen. Exponential-tail distributions have been used in pWCET analysis[9, 1, 11], as the tail beyond a certain point is a safe upper bound of light-tailed distributions.

The iterative approach that we propose in this paper provides a bound/ estimate already after a few accumulation periods, while the method proposed by Frías et al. requires the calculation of the full steady state response time distribution.

9.3 System Model and Notation

The notation used in the paper is outlined in Table 9.1. We use the notation $\hat{x}$ to indicate the estimate of a variable X, and we use the superscripts *, $^\uparrow$, and $^\downarrow$ for the true values, upper, and lower bounds, respectively.

We will use the concept or upper bounding random variables, as defined in Definition 9.3.1.

Definition 9.3.1 (cf. [15, 13]). *Let $\mathcal{X}$ and $\mathcal{Y}$ be two random variables. We say that $\mathcal{X}$ is greater than or equal to $\mathcal{Y}$ (i.e., $\mathcal{X}$ upper bounds $\mathcal{Y}$), if the Cumulative Distribution Function (CDF) of $\mathcal{X}$ is never above that of $\mathcal{Y}$, and we denote this relation by $\mathcal{X} \geq \mathcal{Y}$.*

To upper bound workload distributions, we will use the partial Gaussian distribution, as defined in Definition 9.3.2. Let us consider a Gaussian $\mathcal{N}(\mu, \sigma^2)$ with probability density function $f(x|\mu, \sigma^2)$. $\Phi(x)$ is the cumulative density function of the standard normal distribution.

Definition 9.3.2. *We define a partial Gaussian distribution $\mathcal{N}^{tail}(\mu, \sigma^2, \alpha)$, originated from a Gaussian distribution $\mathcal{N}(\mu, \sigma^2)$, as:*

$$f^{tail}(x|\mu, \sigma^2, \alpha) = \begin{cases} 0, & x \leq \alpha \\ \frac{1}{\Phi(\frac{\mu-\alpha}{\sigma})} f(x|\mu, \sigma^2) & x > \alpha \end{cases} \tag{9.1}$$

In a partial Gaussian distribution, the probability for the elements of the Gaussian distribution lower than α are set to zero and the remaining is normalized so that the distribution integrates into one.

Table 9.1: Overview of notation used in this paper.

Symbol	**Description**
Basic notation	
T	Task period
J_i	Job at task period i
a_i	Arrival time of J_i
d_i	Absolute deadline of J_i
D	Relative deadline
P	Server period
Q	Server budget
n	Number of server periods in a task period
k	Number of server periods in a relative deadline
S	Number of Markov states
M	State transition matrix
N	Number of task periods in workload accumulation
Values of random variables	
c_i	Execution time of J_i
f_i	Finishing time of J_i
v_i	Workload at task period i
h	Accumulation sequence of state visits in Markov chain since workload depletion
$\tilde{h}$	Accumulation vector of the number of visits in each Markov state since workload depletion
Probability distributions and probabilities	
C	Execution time distribution
$\mathcal{V}_h, \mathcal{V}_{\tilde{h}}$	Workload distribution associated with an accumulation sequence or vector
$m_{i,j}$	Transition probability from state i to state j
$\xi(s)$	Stationary probability of being in s
$p_{in}(s, \tilde{h})$	Probability of entering s with $\tilde{h}$
$p_{co}(s, \tilde{h})$	Probability that $\tilde{h}$ in s carries workload to the next task period
$p_{wd}(s)$	Probability of workload depletion in s
p_{dm}	Deadline miss probability
$\beta(s)_N$	Probability of being in state s with h longer than N.

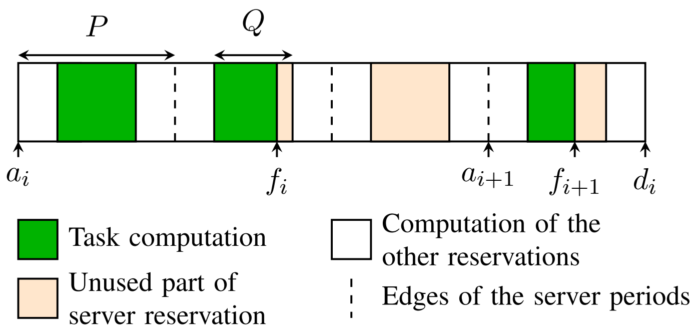
In the derivation of workload distributions, we use convolutions as defined in Definition 9.3.3.

Definition 9.3.3. *The convolution of f and g, denoted with the $*$ operator is:*

$$[f * g](z) = \int_{-\infty}^{\infty} f(z-x)g(x)\,\mathrm{d}x$$

9.3.1 Task Model

A real-time task τ consists of a sequence of jobs J_i, $i \in \mathbb{N}$. The arrival time of J_i is a_i. The tasks are periodic, with no jitter, i.e., $a_{i+1} = a_i + T$, with a_0 being



P
Q
a_i
f_i
a_{i+1}
f_{i+1}
d_i
Task computation
Computation of the other reservations
Unused part of server reservation
Edges of the server periods

Figure 9.1: An illustration of the task model and the CBS.

the arrival time of the first job. The execution time of J_i is c_i and its finishing time is f_i. The jobs may be preempted, and $f_i \geq a_i + c_i$. The execution time c_i is modeled as a random variable. The random variable $\mathcal{R}$ models the time from activation time to finish time of a job.

The deadline of a job J_i is $d_i = a_i + D$, where D is the relative deadline. Jobs are executed until completion, even when a deadline is missed. The relative deadline can be longer than the task period. We consider the probability of a deadline miss p_{dm}, that is the overall probability that a job finishes after the deadline, $p_{dm} = p(\mathcal{R} > D)$.

9.3.2 Scheduling Algorithm

The considered scheduling algorithm is reservation-based, namely the Constant Bandwidth Server (CBS). Each task has its own server. Within each server period P, the task is guaranteed to receive Q units of processing time. The fraction of the processing resource dedicated to this task, the bandwidth, is $B = Q/P$. We choose the server period so that it divides the task period evenly, i.e., $T = nP$, where n is a positive integer. We also define k, a positive integer that is the number of server periods in the relative deadline D, $D = kP$.

An illustration of the task model and CBS is shown in Figure 9.1. In this illustration, the task period is divided into three server periods, and the bandwidth is 0.5. As illustrated, the deadline of a job does not need to be within a task period from the arrival; the relative deadline may be longer than the period.

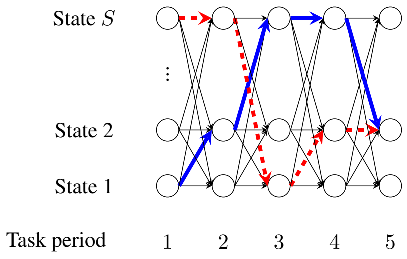


Figure 9.2: Illustration of the period by period workload accumulation sequence.

9.4 Execution Time Model and Analysis

9.4.1 Markov Chain Execution Times

In this section, we consider a task, where the execution time distribution is described by a Markov model characterized by the triplet $\langle \mathbb{S}, M, \mathbb{C} \rangle$. $\mathbb{S} = \{1, 2, \ldots, S\}$ is the set of S states, $S \in \mathbb{N}$. M is the $S \times S$ state transition matrix, where the element $m_{a,b}$ represents the conditional probability of being in state b at task period $i+1$, given that at task period i the state is a. $\mathbb{C} = \{C_1, C_2, \ldots, C_S\}$ is the set of execution time distributions, or *emission distributions* related to the respective state. These are modeled as Gaussian distributions with mean μ_s, and variance σ_s^2, i.e., $C_s \sim \mathcal{N}(\mu_s, \sigma_s^2)$.

9.4.2 Overview of the Proposed Approach

To upper bound the deadline miss probability of the task running under CBS, we propose a method based on a workload accumulation scheme. The main idea is outlined below, followed by the details in the remaining subsections.

In each task period, task τ is guaranteed nQ units of processing time. The pending workload at the i-th task period is denoted as v_i and defined as in [3]:

$$v_i = \underbrace{\max(0, v_{i-1} - nQ)}_{\text{carry-in workload}} + c_i \tag{9.2}$$

where the first term accounts for the previous workload, initially set to 0, and the first period is $v_1 = c_1$. An example of how the workload evolves according to the Markov chain model is shown in Figure 9.2. We start with zero initial pending workload and add one task period at a time of workload accumulation. In the figure, the dashed red and solid blue lines depict two possible workload accumulation sequences that are in state 2 at task period 5 from workload depletion. The *accumulation sequence* is modeled as a random variable $\mathcal{H}$ that can take the values of any possible workload accumulation path. In the example from Figure 9.2, in the dashed red path it takes the value $h = (S, S, 1, 2, 2)$.

Davis and Cucu-Grosjean [12] define the deadline miss probability for a task as the average deadline miss probability of task jobs during a hyperperiod. In this work, using the CBS allows us to discard pending workload from other tasks in the task set. The notion of hyperperiod is therefore irrelevant, and we define the Deadline Miss Probability *DMP* as the average deadline miss probability of the task's jobs. We do this by considering accumulation sequences. More specifically, we define the probability of a job arrival leading to the accumulation sequence h as $p_{in}(h)$. Since each job arrival leads to one specific accumulation sequence, the sum of $p_{in}(h)$ over all h equals 1. We define the conditional deadline miss probability for a job with accumulation sequence h as $p_{dm}(h)$. Then, the *DMP* is defined as the sum of the deadline miss probabilities for each accumulation sequence weighted with their respective probabilities:

$$\mathit{DMP} = \sum_{\forall h} p_{in}(h) p_{dm}(h) \tag{9.3}$$

Problem: *The sum of Equation* (9.3) *has a countably infinite number of terms. This paper investigates how to find a bound for DMP with a finite number of terms.*

In the remainder of this section, we will provide an upper bound on DMP by finding the upper bounds on p_{in} and p_{dm}. The process is divided into two steps. First, we compute the upper bounds on p_{in} and p_{dm} of accumulation sequences up to length N, thus approaching the true deadline miss probability. To make a safe bound, we then sum the p_{in} values in the remaining accumulation sequences of length $N + 1$ to infinity, assuming that p_{dm} for these periods is 1. This sum is referred to as β. This finally leads to the safe over-approximation of *DMP*.

The steps for deriving a bound on *DMP* are presented in this paper as follows:

Section 9.4.3: To determine upper bounds on p_{dm} and p_{in} in Equation (9.3) we need to find upper bounds on the pending workload distributions associated with each state and accumulation sequence. This is done in Equations (9.20) and (9.24).

Section 9.4.4: Bounds on p_{in} depend on the probability of carry-over workload p_{co} from the previous step and the transition probabilities m. In the first step of the accumulation process, p_{in} depends on the probability of workload depletion p_{wd} for each state. With p_{wd} propagating along the accumulation, each p_{in} is a linear combination of p_{wd} for the different states.

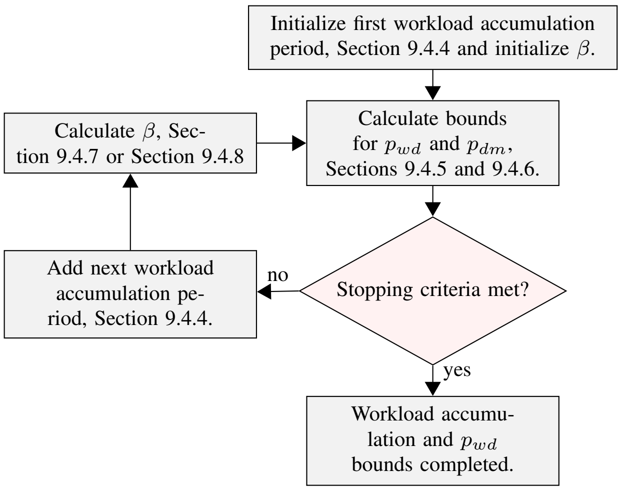

Initialize first workload accumulation period, Section 9.4.4 and initialize β.
Calculate bounds for p_{wd} and p_{dm}, Sections 9.4.5 and 9.4.6.
Calculate β, Section 9.4.7 or Section 9.4.8
Stopping criteria met?
no
Add next workload accumulation period, Section 9.4.4.
yes
Workload accumulation and p_{wd} bounds completed.

Figure 9.3: The workload accumulation process.

Section 9.4.5: Bounds on p_{wd} are derived, these rely on the sum of p_{in} in accumulation periods after N, denoted as β.

Section 9.4.6: In this section bounds on p_{dm} are presented, using the bounds on $\mathcal{V}$, p_{in} and β. The upper bound of p_{dm} for a state is defined in Equation (9.30).

Section 9.4.7: In this section, we derive a bound on β. β is the minimum of Equations (9.31) and (9.32), and is utilized for computing the lower bounds on p_{in}, p_{co} and finally $\mathcal{V}$.

Section 9.4.8: In this section, we derive an estimate of β as an alternative to the bound.

The parts are tied together in the iterative workload accumulation algorithm presented along with an example in Section 9.5. In Figure 9.3 the process is illustrated with reference to the different sections. Section 9.4.3 is not referenced in the figure as it is a basis for all the remaining sections.

9.4.3 Bounding the Conditional Pending Workload Distribution Associated with a Workload Accumulation Sequence

We seek upper and lower bounds of the conditional pending workload distribution conditioned on having a given accumulation sequence since the most recent point of workload depletion. As an example from Figure 9.2, we want to define the pending workload distribution in state 2 at task period 5, provided that the transitions since workload depletion have been along the path marked as dashed red, $h = (S, S, 1, 2, 2)$.

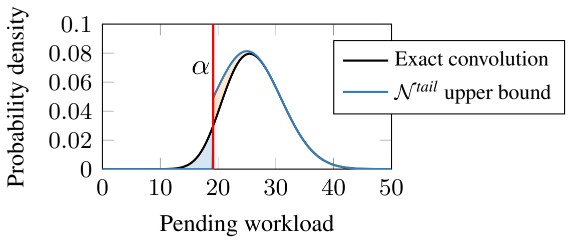


Figure 9.4: Illustration of a convolution result with an upper bounding partial Gaussian distribution.

We denote the conditional pending workload distribution, conditioned on a given accumulation sequence h as $\mathcal{V}_h$ with a probability density function $p(v|\mathcal{H} = h)$.

The lower and upper bounds for this conditional pending workload distribution depend only on the number of visits in each state in the accumulation sequence and are independent of their order. We model the accumulation vector as a random variable $\tilde{\mathcal{H}}$ that takes values as S-dimensional vectors of non-negative integer values, where each value represents the number of visits in a state. This means that the dashed red and the solid blue accumulation sequence lines in Figure 9.2 will contribute to the same accumulation vector at task period 5 since they both have the same number of visits in each state, that is $\tilde{h} = [1, 2, \ldots, 2]$. We define the operation $\tilde{h}[s]$ as taking the s-th element of $\tilde{h}$. We also define $\tilde{h}_{+s}$ as $\tilde{h}$ with the s-th element incremented by one, to simplify the notation of the accumulation vector in s with carry-in workload from $\tilde{h}$.

The number of possible bounded pending workload distributions of length N in a system with S states is $\binom{N+S-1}{N} = \frac{(N+S-1)!}{N!(S-1)!}$, as opposed to S^N which would be needed if ordering were taken into account. For a fixed number of states S, the number of distributions to consider increases with the number of periods considered as $\mathcal{O}(N^{S-1})$.

Recalling Definition 9.3.1, we derive an upper bound conditional pending workload distribution $\mathcal{V}^{\uparrow}_{\tilde{h}} \geq \mathcal{V}_h$.

In the following, we show that a *partial Gaussian distribution* (see Definition 9.3.2) upper bounds the conditional pending workload distribution. An illustration is in Figure 9.4, where the blue curve and red line upper bounds the black workload distribution, lower probability values (blue area) are moved to higher (orange area).

Theorem 9.1. *The conditional pending workload distribution associated with each state s and accumulation vector $\tilde{h}$ is upper bounded by*

$\mathcal{N}^{tail}(\mu(\tilde{h}), \sigma^2(\tilde{h}), \alpha(\tilde{h}, s))$.

We prove this by induction. For clarity we state a Lemma 9.2 for the base case, and Lemma 9.3 for the inductive step.

Lemma 9.2. *The partial Gaussian distribution* $\mathcal{N}^{tail}(\mu_s, \sigma_s^2, 0)$ *upper bounds the conditional pending workload distribution* $\mathcal{V}_{\tilde{h}}$ *in state* s *immediately after a point of workload depletion.*

Proof. In the first step after workload depletion, the conditional pending workload distribution $\mathcal{V}_h$ equals the execution time distribution of the entered state s. Excluding negative values and normalizing gives an upper bounding distribution, as probabilities are moved from lower workload values to higher. Thus, $\mathcal{N}^{tail}(\mu_s, \sigma_s^2, 0)$ is an upper bound. □

With non-zero carry-over workload in a transition from state s_p with accumulation vector $\tilde{h}$, and an upper bound on the workload distribution $\mathcal{N}^{tail}(\mu(\tilde{h}), \sigma^2(\tilde{h}), \alpha(\tilde{h}, s_p))$, into state s, we will show that the conditional pending workload distribution is upper bounded by the partial Gaussian distribution $\mathcal{N}^{tail}(\mu(\tilde{h}_{+s}), \sigma^2(\tilde{h}_{+s}), \alpha(\tilde{h}_{+s}, s))$. Below, in Equations (9.4) and (9.5) we define $\mu(\tilde{h}_{+s})$ and $\sigma^2(\tilde{h}_{+s})$. Equations (9.6) and (9.7) are used to simplify the expression of the starting value $\alpha(\tilde{h}_{+s}, s)$ of the resulting upper bounding distribution, defined in Equation (9.8). Here $sf^{-1}(q, \mu, \sigma^2)$ denotes the inverse survival function at q of a Gaussian distribution with mean μ, and variance σ^2. Equation (9.7) defines $K(\tilde{h}, s_p)$, the normalization factor needed for the conditional probability calculation. We perform a convolution with the upper bounding workload distribution in s_p with $\tilde{h}$ extending past the task period. $K(\tilde{h}, s_p)^{-1}$ is the integral of this part, to get a probability distribution integrating to one.

$$\mu(\tilde{h}_{+s}) = \mu_s + \sum_{i=1}^{S} \tilde{h}[i](\mu_i - nQ) \tag{9.4}$$

$$\sigma^2(\tilde{h}_{+s}) = \sigma_s^2 + \sum_{i=1}^{S} \tilde{h}[i]\sigma_i^2 \tag{9.5}$$

$$\alpha_\Delta(\tilde{h}, s_p) = \max(0, \alpha(\tilde{h}, s_p) - nQ) \tag{9.6}$$

$$K(\tilde{h}, s_p) = \left[\Phi\left(\frac{\mu(\tilde{h}) - nQ - \alpha_\Delta(\tilde{h}, s_p)}{\sigma(\tilde{h})}\right)\right]^{-1} \tag{9.7}$$

$$\alpha(\tilde{h}_{+s}, s) = \begin{cases} 0 & \tilde{h} = \mathbf{0} \\ sf^{-1}(\frac{1}{K(\tilde{h}, s_p)}, \mu(\tilde{h}_{+s}), \sigma^2(\tilde{h}_{+s})) & \tilde{h} \neq \mathbf{0} \end{cases} \tag{9.8}$$

Lemma 9.3. *When transitioning with non-zero carry-over workload from state s_p with accumulation vector $\tilde{h}$ into state s, and with an upper bound on the workload distribution in the previous task period $\mathcal{V}^{\uparrow}$ as $\mathcal{N}^{tail}(\mu(\tilde{h}), \sigma^2(\tilde{h}), \alpha(\tilde{h}, s_p))$, the conditional pending workload distribution is upper bounded by $\mathcal{N}^{tail}(\mu(\tilde{h}_{+s}), \sigma^2(\tilde{h}_{+s}), \alpha(\tilde{h}_{+s}, s))$.*

Proof. The strictly positive carry-over workload distribution is the normalized workload tail beyond the task period processing time, which can be written as $\mathcal{N}^{tail}(\mu(\tilde{h}) - nQ, \sigma^2(\tilde{h}), \max(0, \alpha(\tilde{h}, s_p) - nQ))$.

The execution time distribution in state s is described by $\mathcal{N}(\mu_s, \sigma_s^2)$. The resulting upper bound on the conditional workload distribution $\mathcal{V}^{\uparrow}_{\tilde{h}_{+s}}$ in state s with accumulation vector $\tilde{h}_{+s}$ is the result of the convolution, Definition 9.3.3, of the probability density functions of the execution time and the upper bound on the positive carry-over workload. This holds because execution times are independent random variables and the dependence of the Markov model is restricted to the transition probabilities.

To simplify the notation in the convolution expansion, we introduce the following:

$$\mu_R(z) = \frac{(z - \mu_s)\sigma^2(\tilde{h}) + (\mu(\tilde{h}) - nQ)\sigma_s^2}{\sigma_s^2 + \sigma^2(\tilde{h})} \tag{9.9}$$

$$\sigma_R^2 = \frac{\sigma_s^2\sigma^2(\tilde{h})}{\sigma_s^2 + \sigma^2(\tilde{h})} \tag{9.10}$$

$$\mu_{\Sigma\Delta} = \mu_s + \mu(\tilde{h}) - nQ \tag{9.11}$$

$$\sigma_\Sigma^2 = \sigma_s^2 + \sigma^2(\tilde{h}) \tag{9.12}$$

Expanding the convolution for $\mathcal{V}^{\uparrow}_{\tilde{h}_{+s}}$:

$$\begin{aligned}
&\int_{-\infty}^{\infty} f(z - x|\mu_s, \sigma_s^2) f^{tail}(x|\mu(\tilde{h}) - nQ, \sigma^2(\tilde{h}), \alpha_\Delta)\,\mathrm{d}x \\
&= K(\tilde{h}, s_p) \int_{\alpha_\Delta}^{\infty} f(z - x|\mu_s, \sigma_s^2) f(x|\mu(\tilde{h}) - nQ, \sigma^2(\tilde{h}))\,\mathrm{d}x \\
&= K(\tilde{h}, s_p) f(z|\mu_{\Sigma\Delta}, \sigma_\Sigma^2) \int_{\alpha_\Delta}^{\infty} f(x|\mu_R(z), \sigma_R^2)\,\mathrm{d}x
\end{aligned} \tag{9.13}$$

where the last step isolated the part of the expression that is independent of x. We recognize the integral in the second factor of Equation (9.13) as the survival function or 1-CDF at α_Δ of $\mathcal{N}(\mu_R(z), \sigma_R^2)$. This is monotonically increasing and goes to 0 as z goes to $-\infty$ and to 1 as z goes to ∞. Thus, we can find a value $\alpha(\tilde{h}_{+s}, s)$ where the area under the curve of the exact convolution of the pending workload distribution up to $\alpha(\tilde{h}_{+s}, s)$ equals the area between the curves of the exact pending workload distribution and the partial Gaussian distribution, $\mathcal{N}^{tail}(\mu_{\Sigma\Delta}, \sigma^2_{\Sigma}, \alpha(\tilde{h}_{+s}, s))$ from $\alpha(\tilde{h}_{+s}, s)$. An illustration is provided in Figure 9.4. Using K for normalization of the partial Gaussian distribution ensures that the tail of the upper bound approaches the tail of the full convolution asymptotically. We find the $\alpha(\tilde{h}_{+s}, s)$ which gives:

$$K(\tilde{h}, s_p) \int_{\alpha(\tilde{h}_{+s},s)}^{\infty} f(x|\mu_{\Sigma\Delta}, \sigma^2_{\Sigma})\,\mathrm{d}x = 1 \tag{9.14}$$

As we know that the result of the convolution integrates to one, this shows that the two regions described and illustrated in Figure 9.4 have the same area. Replacing the exact convolution with the partial Gaussian is equivalent to moving probability weight from lower pending workload values to higher, leading to an overestimate. We have:

$$\mu(\tilde{h}_{+s}) = \mu_{\Sigma\Delta} \tag{9.15}$$

$$\sigma^2(\tilde{h}_{+s}) = \sigma^2_{\Sigma} \tag{9.16}$$

$$\alpha(\tilde{h}_{+s}, s) = sf^{-1}\left(\frac{1}{K(\tilde{h}, s)}, \mu_{\Sigma\Delta}, \sigma^2_{\Sigma}\right) \tag{9.17}$$

This concludes our proof. □

Considering all states s_p containing the accumulation vector $\tilde{h}$, we define:

$$\alpha_{\Delta}(\tilde{h}) = \max(0, \max_{\forall s_p} \alpha(\tilde{h}, s_p) - nQ) \tag{9.18}$$

We use this instead of Equation (9.6) in Equations (9.7) and (9.8). With these lemmas we are ready to prove Theorem 9.1.

Proof. We prove this by induction.
Base case: For the task period after workload depletion, this follows by Lemma 9.2.
Inductive hypothesis: If we have such a workload distribution upper bound for all states and accumulation vectors in one task period, it also holds for the next period.
Inductive step: This follows from Lemma 9.3 and taking the maximum α in Equation (9.18). □

With similar reasoning, we can use a Gaussian distribution as a lower bound of the pending workload distribution $\mathcal{V}^{\downarrow}_{\tilde{h}} \leq \mathcal{V}_h$. This is illustrated in

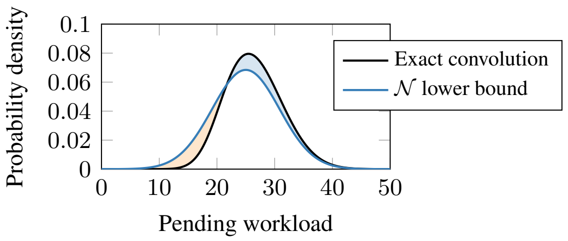


Figure 9.5: An illustration of a convolution result and the Gaussian distribution that forms a lower bound.

Figure 9.5. As $K > 1$, and the area under the curve equals one for both the Gaussian distribution with mean $\mu_{\Sigma\Delta}$ and variance σ^2_{Σ} and the result of the convolution, replacing the workload distribution with the Gaussian implies moving probability weight from higher workload values to lower, thus providing a lower bound.

9.4.4 Bounds on the Probability of Entering a State with an Accumulation Vector

Each state s in each task period is associated with one or more accumulation vectors, $\tilde{h}$. Each accumulation vector in a state is associated with lower and upper bounds on the probability of entering this state with the associated accumulation vector $p^{\downarrow}_{in}(s, \tilde{h})$ and $p^{\uparrow}_{in}(s, \tilde{h})$. Each accumulation vector in a state is also associated with lower and upper bounds on the probability of the workload contributing to carry-over into the next period, $p^{\downarrow}_{co}(s, \tilde{h})$ and $p^{\uparrow}_{co}(s, \tilde{h})$.

In the first period, with no carry-in workload, each state is associated with a single accumulation vector containing zeros except for the current state that is set to 1. The probability of entering a state in the first period after workload depletion depends on the stationary probabilities $\xi(s)$ of being in each state, the probability of workload depletion $p_{wd}(s)$ in each state, and the transition matrix. The stationary probabilities and the transition matrix are known, but the probability of workload depletion in each state is unknown at this stage. In Section 9.4.5 we will describe how to retrieve this. Assuming that we have lower and upper bounds on the probabilities of workload depletion, $p^{\downarrow}_{wd}(s)$ and $p^{\uparrow}_{wd}(s)$, we can calculate lower and upper bounds on the probability of entering the states in the first workload accumulation period as

$$p_{in}^{\downarrow}(s,\tilde{h}) = \sum_{s_p=1}^{S} \xi(s_p) p_{wd}^{\downarrow}(s_p) m_{s_p,s} \tag{9.19}$$

$$p_{in}^{\uparrow}(s,\tilde{h}) = \sum_{s_p=1}^{S} \xi(s_p) p_{wd}^{\uparrow}(s_p) m_{s_p,s} \tag{9.20}$$

Since there is only one accumulation vector in each state in the first accumulation period, there is no dependency on $\tilde{h}$.

For the following periods, accumulation vectors are created by copying each accumulation vector from the states in the previous task period and incrementing the current state element by 1. We denote this vector as $\tilde{h}_{+s}$ Note that paths from different states in the previous period can lead to the same accumulation vector. The probability of entering s with $\tilde{h}_{+s}$ depends on the probability of $\tilde{h}$ contributing to carry-over into the next period in all states, and transition probabilities.

The probability that the workload contributes to carry-over into the next period is the probability of entering the state with this accumulation vector times the probability that the conditional pending workload exceeds the available processor time in a task period. This probability is bounded by $p_{co}^{\downarrow}(s,\tilde{h})$ and $p_{co}^{\uparrow}(s,\tilde{h})$, then calculated as:

$$\begin{aligned} p_{co}^{\downarrow}(s,\tilde{h}) &= p_{in}^{\downarrow}(s,\tilde{h}) p(\mathcal{V}_{\tilde{h}}^{\downarrow} > nQ) \\ &= p_{in}^{\downarrow}(s,\tilde{h}) p(\mathcal{N}(\mu(\tilde{h}),\sigma^2(\tilde{h})) > nQ) \end{aligned} \tag{9.21}$$

$$\begin{aligned} p_{co}^{\uparrow}(s,\tilde{h}) &= p_{in}^{\uparrow}(s,\tilde{h}) p(\mathcal{V}_{\tilde{h}}^{\uparrow} > nQ) \\ &= p_{in}^{\uparrow}(s,\tilde{h}) p(\mathcal{N}^{tail}(\mu(\tilde{h}),\sigma^2(\tilde{h}),\alpha(\tilde{h})) > nQ) \end{aligned} \tag{9.22}$$

The probability of entering state s with the accumulation vector $\tilde{h}$ is lower and upper bounded by:

$$p_{in}^{\downarrow}(s,\tilde{h}_{+s}) = \sum_{s_p=1}^{S} p_{co}^{\downarrow}(s_p,\tilde{h}) m_{s_p,s} \tag{9.23}$$

$$p_{in}^{\uparrow}(s,\tilde{h}_{+s}) = \sum_{s_p=1}^{S} p_{co}^{\uparrow}(s_p,\tilde{h}) m_{s_p,s}. \tag{9.24}$$

9.4.5 Bounds on the Probability of Workload Depletion

Bounds on the probability of workload depletion for each state are used to calculate p_{in} in the first step after workload depletion in Equations (9.19)

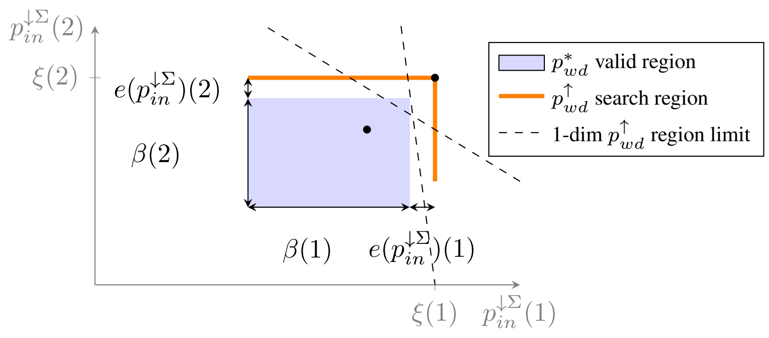

$p_{in}^{\downarrow\Sigma}(2)$
$\xi(2)$
$e(p_{in}^{\downarrow\Sigma})(2)$
$\beta(2)$
$\beta(1)$
$e(p_{in}^{\downarrow\Sigma})(1)$
$\xi(1)$
$p_{in}^{\downarrow\Sigma}(1)$
p_{wd}^{*} valid region
$p_{wd}^{\uparrow}$ search region
1-dim $p_{wd}^{\uparrow}$ region limit

Figure 9.6: An illustration of the possible valid region of $p_{in}^{\downarrow\Sigma}$ for two states, if the true probabilities of workload depletion would be used as $p_{wd}^{\downarrow}$ in Equation (9.19).

and (9.20), and are further propagated to all p_{in}. The true workload depletion probability p_{wd}^{*} is unknown, and in this section we will derive bounds for it. Had p_{wd}^{*} been known, and input as $p_{wd}^{\downarrow}$ in Equation (9.19), the sum of the lower bounds on the probabilities $p_{in}^{\downarrow\Sigma}$ associated with all accumulation vectors accounted for would be lower than the stationary probabilities for all states. Using $\tilde{h} \in (s,i)$ to denote the set of accumulation vectors associated with state s in task period i, we formulate:

$$p_{in}^{\downarrow\Sigma}(s,p_{wd}) = \sum_{i=1}^{N} \sum_{\tilde{h}\in(s,i)} p_{in}^{\downarrow}(s,\tilde{h}) \leq \xi(s), \quad \forall s \tag{9.25}$$

We define $\beta(s)_N$ as the probability of being in s with workload accumulation past N. We also define $e(p_{in}^{\downarrow\Sigma})$ as the error introduced by using the lower bounding Gaussian distribution in place of the true convolution result. Had we known the true p_{wd}^{*} and input it as $p_{wd}^{\downarrow}$ in Equation (9.19) that would give values of $p_{in}^{\downarrow\Sigma}$ in the blue area of Figure 9.6.

Had the true workload depletion probability p_{wd}^{*} been known and input as $p_{wd}^{\uparrow}$ in Equation (9.20), the sum of the upper bounds of the probabilities $p_{in}^{\uparrow\Sigma}$ associated with all accumulation vectors would be greater than the stationary probabilities minus the probability of being in the state with longer accumulation vectors $\beta(s)_N$, for all states. This is outlined in Equation (9.26):

$$p_{in}^{\uparrow\Sigma}(s,p_{wd}) = \sum_{i=1}^{N} \sum_{\tilde{h}\in(s,i)} p_{in}^{\uparrow}(s,\tilde{h}) \geq \xi(s) - \beta(s)_N, \forall s \tag{9.26}$$

We define $e(p_{in}^{\uparrow\Sigma})$ the error introduced by using the upper bounding partial Gaussian distribution in place of the true convolution result. If we input true

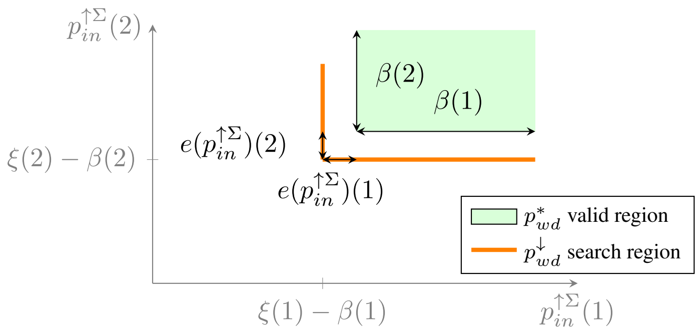

$p_{in}^{\uparrow\Sigma}(2)$
$\beta(2)$
$\beta(1)$
$e(p_{in}^{\uparrow\Sigma})(2)$
$\xi(2)-\beta(2)$
$e(p_{in}^{\uparrow\Sigma})(1)$
🟩 p_{wd}^{*} valid region
$p_{wd}^{\downarrow}$ search region
$\xi(1)-\beta(1)$
$p_{in}^{\uparrow\Sigma}(1)$

Figure 9.7: An illustration of the possible valid region of $p_{in}^{\uparrow\Sigma}$ for two states, if the true probabilities of workload depletion would be used as $p_{wd}^{\uparrow}$ in Equation (9.20).

workload depletion probabilities as $p_{wd}^{\uparrow}$ in Equation (9.20) the resulting $p_{in}^{\uparrow\Sigma}$ would be in the range depicted as green in Figure 9.7. This allows us to bound the true workload depletion probabilities to values mapping within both the blue region of Figure 9.6 and the green region of Figure 9.7.

An upper bound of the workload depletion probability p_{wd} is found for each state as the maximum of the values that lead to $p_{in}^{\downarrow\Sigma}$ along the orange lines of Figure 9.6.

Theorem 9.4. *The state-wise maximum of p_{wd} taken within the region of p_{wd} leading to $p_{in}^{\downarrow\Sigma}(s) \leq \xi(s)$ for all states, and where equality holds for all but at most one s is an upper bound of p_{wd}.*

Proof. Each $p_{in}^{\downarrow}(s,\tilde{h})$ is a linear combination of p_{wd} for all states, this follows from Equations (9.19), (9.21) and (9.23). Combined with Equation (9.25) it follows that $p_{in}^{\downarrow\Sigma}(s)$ is also a linear combination of p_{wd} for all states, which for some positive factors $A_{i,s}$ we can write:

$$p_{in}^{\downarrow\Sigma}(s,p_{wd}) = \sum_{i=1}^{S} A_{i,s}p_{wd}(i) \tag{9.27}$$

Assume that we have the true workload depletion probability p_{wd}^{*}. For an arbitrary state dimension j in p_{wd}, we can increase $p_{wd}(j)$ with an amount $\delta_{s,j}$ so that we reach a plane defined by Equation (9.28). For the lowest $\delta_{s,j}$, the first plane we encounter along the line, we have $p_{in}^{\downarrow\Sigma}(i) \leq \xi(i), \forall i \neq s$.

$$p_{in}^{\downarrow\Sigma}(s,p_{wd}) = A_{j,s}(p_{wd}^{*}(j)+\delta_{s,j}) + \sum_{i=1,i\neq j}^{S} A_{i,s}p_{wd}^{*}(i) = \xi(s) \tag{9.28}$$

Because the true p^*_{wd} gives $p^{\downarrow\Sigma}_{in}(s) \leq \xi(s)$ and due to the linear combination, it follows that at least one dimension of p_{wd} is an upper bound at every point in the planes defined by $p^{\downarrow\Sigma}_{in}(s, p_{wd}) = \xi(s)$, including the point with equality for all s - the upper right corner on the orange lines in Figure 9.6. If a particular dimension does not have an upper bound at this point, we have an upper bound on one of the planes, as the black dot in the illustration in Figure 9.6. The plane separating the region of the plane with upper bounds on this dimension from the region with underestimates will cross at least one of the orange lines, which ensures that an upper bound will be found in the region. Illustrations of possible separating planes are dashed lines in Figure 9.6. This concludes the proof. □

Similarly, a lower bound on the workload depletion probability p_{wd} is found for each state as the minimum of the values that lead to $p^{\uparrow\Sigma}_{in}$ along the orange lines of Figure 9.7. By using the lower bound from Figure 9.7 to determine the endpoints of the orange sections in Figure 9.6 and the upper bound from Figure 9.6 to determine the endpoints of the orange sections in Figure 9.7 $e(p^{\downarrow\Sigma}_{in})$ and $e(p^{\uparrow\Sigma}_{in})$ can be ignored. The endpoints are adjusted if they are outside the valid range for p_{wd}, that is if the probabilities are lower than 0 or higher than 1. As all $p^{\downarrow\Sigma}_{in}(s)$ and $p^{\uparrow\Sigma}_{in}(s)$ depend linearly on all $p_{wd}(s)$, we only need to consider the endpoints of the orange sections.

9.4.6 Upper Bounding the Deadline Miss Probability

We can then calculate an upper bound on the deadline miss probability as defined in Equation (9.3). The upper bound on the deadline miss probability $p^{\uparrow}_{dm}$ conditioned on an accumulation vector $\tilde{h}$ and a state s is:

$$\begin{aligned} p^{\uparrow}_{dm}(s, \tilde{h}) &= p(\mathcal{V}^{\uparrow}_{\tilde{h}} > kQ) \\ &= p(\mathcal{N}(\mu(\tilde{h}), \sigma^2(\tilde{h})) > kQ). \end{aligned} \tag{9.29}$$

The upper bound on the deadline miss probability in a state is

$$p^{\uparrow}_{dm}(s) = \frac{\beta(s)^{\uparrow}_N}{\xi(s)} + \frac{\sum_{i=1}^{N} \sum_{\tilde{h} \in (s,i)} p^{\uparrow}_{in}(s, \tilde{h}) p^{\uparrow}_{dm}(s, \tilde{h})}{\xi(s)}. \tag{9.30}$$

9.4.7 Bounding the Probability of Longer Workload Accumulation

The sum of p_{in} in task periods beyond N, β is still unknown, and in this section a bound is derived. β decreases monotonically with each accumulated period, as all probabilities are non-negative. For each period, $\beta(s)$ decreases with at least the lower bound on the probability of being in the state in the same period, i.e.

$$\beta(s)_N \leq \beta(s)_{N-1} - \sum_{\tilde{h} \in (s,N)} p_{in}^{\downarrow}(s,\tilde{h}) \forall s \tag{9.31}$$

We also know that β is at most the stationary probability minus the lower bound on the probabilities accounted for, i.e.

$$\beta(s)_N \leq \xi(s) - \sum_{i=1}^{N} \sum_{\tilde{h} \in (s,i)} p_{in}^{\downarrow}(s,\tilde{h}) \tag{9.32}$$

Thus, given a safe bound for the probability of accumulation vectors not accounted for, β, in one accumulation period, we can obtain safe bounds for subsequent periods as the minimum of Equations (9.31) and (9.32).

9.4.8 Estimating the Probability of Longer Workload Accumulation

As an alternative or complement to the bound of β presented in Section 9.4.7, β can be estimated. First, the probability of workload depletion is estimated as the mean of the upper and lower bounds.

$$\widehat{p_{wd}} = \frac{p_{wd}^{\downarrow} + p_{wd}^{\uparrow}}{2} \tag{9.33}$$

Then we estimate β according to

$$\hat{\beta}(s)_N = \xi(s) - \sum_{i=1}^{N-1} \sum_{\tilde{h} \in (s,i)} p_{in}^{\downarrow}(s,\tilde{h}), \tag{9.34}$$

where $\widehat{p_{wd}}$ is used instead of $p_{wd}^{\downarrow}$ in Equation (9.19) for the first accumulation period. A new estimate is retrieved for each accumulation period. β of the first period is estimated as:

1. Let the probabilities of the first task period be ξ.
2. Calculate the probability of carry over into s in the second period from all C_i and M.
3. Set $\hat{\beta}(s)$ to the probability of being in the second period relative to the sum of both periods, scaled with $\xi(s)$.

9.5 Iterative Workload Accumulation

We propose an iterative approach where workload periods are successively accumulated. The process is illustrated in Figure 9.3 and ends when one of the following stopping criteria is met:

1. The upper bounds of the workload depletion probability of all states have turned from decreasing to increasing, or the lower bounds have turned from increasing to decreasing.

2. A maximum number of task periods is reached.

The first condition is met if the workload depletion probability bounds converge, or if the region within the bounds starts to grow. With each accumulation period, a convolution is performed, potentially increasing the error introduced by using the upper and lower bounding distributions in place of the true convolution result. This is illustrated by the white space between the blue area and the orange lines in Figure 9.6, and by the white space between the green area and the orange lines in Figure 9.7. If the increase in this error is not compensated by a sufficiently low probability of the associated accumulation vectors, the bounding region of the workload depletion probability can start to increase, and we stop at the period with the tightest bound.

The second condition is needed in the case where the bounds on the workload depletion probabilities or deadline miss probabilities diverge from the beginning. This may be due to insufficient bandwidth provided to the task in the CBS, or because the errors introduced are too large. The second condition is also activated when we have a slow convergence of the workload depletion probability bounds.

As an example we take a Markov model defined by:

$$S = 2, \qquad M = \begin{pmatrix} 0.9 & 0.1 \\ 0.7 & 0.3 \end{pmatrix}, \qquad \mathcal{C} = \{\mathcal{N}(20, 9), \mathcal{N}(40, 16)\}.$$

The stationary probabilities are 0.875 for state 1 and 0.125 for state 2. In our example, the CBS is defined such that there are $n = 4$ server periods within each task period, and the budget in each server period is $Q = 8$. The deadline is defined by $k = 8$.

First, we use the bound on the probability of longer workload accumulation as described in Section 9.4.7. We initialize the accumulation with one period after workload depletion, and β to $(0.1238, 0.0397)$, the probability of being in states 1 and 2 respectively with workload carried over from at least one task period. These probabilities are obtained from the simulation. In Figure 9.8 the

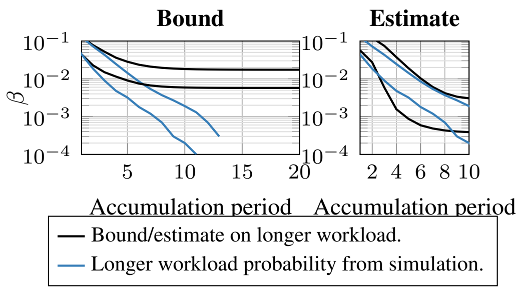


Figure 9.8: Bounds and estimates on β for the two states in black, along with probability estimates of longer accumulation histories obtained from simulation in blue. (Log scale.)

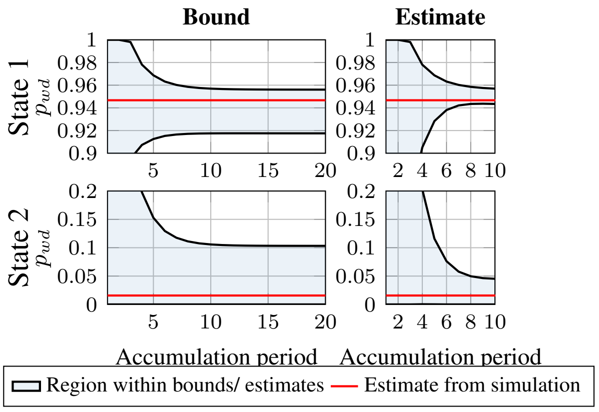


Figure 9.9: The region between the upper and lower bounds/ estimates on the per-state probability of workload depletion in the example, along with the estimates obtained from simulation in red.

obtained bounds for beta for the two states as we add accumulation periods are displayed in black. Estimated probabilities of longer accumulation histories obtained from simulation are displayed in blue.

The upper and lower bounds of the probabilities of workload depletion obtained with these values for β are shown in black in Figure 9.9, along with estimates obtained by simulation shown as red lines. The workload accumulation stops at the maximum number of task periods, 20.

In Figure 9.10 the bounds on the deadline miss probabilities during the workload accumulation process of our example are displayed. The parts of the second terms resulting from the sum over the accumulation vectors are shown as dashed. In the example this sum approaches the p_{dm} from simulation, and the pessimism comes from the pessimism in β. These bounds are compared to

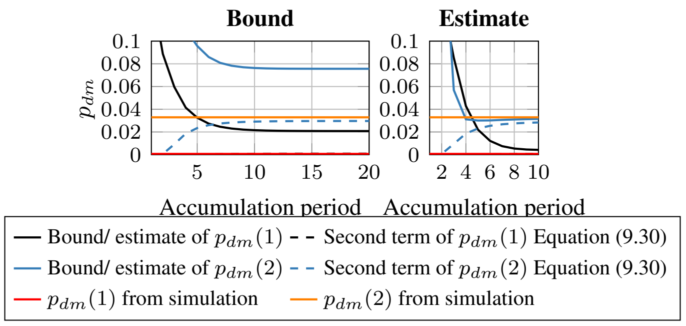


Figure 9.10: The bounds and estimates on the deadline miss probabilities during the workload accumulation process of the example, along with results from simulation.

the results from the simulation.

Using estimates on β as outlined in Section 9.4.8 in our example gives the values displayed in Figure 9.8. Here the initial values of beta are taken as an estimate of the probability of being in the second accumulation period for each state.

Using these estimates of β to find upper and lower estimates of the probabilities of workload depletion gives the results shown in Figure 9.9. The workload accumulation process stops after 10 accumulation periods.

The estimates of the deadline miss probabilities for each state during the workload accumulation process, along with the deadline miss probabilities from the simulation, are shown in Figure 9.10. The parts stemming from the second term of Equation (9.30) are dashed.

Comparing the two approaches, the bound on β is a safe overestimate, but relies on having a bound or close estimate for the first accumulation period. The estimate on β, however, is not a safe bound but can be initiated with a rough estimate in the first accumulation period. Using the estimate of β results in a lower first term in the deadline miss probability $p_{dm}^{\uparrow}$ calculation of Equation (9.30). In the example, this estimate is about 2.5 times higher than the deadline miss ratio obtained in simulation, while the bound is about 5 times higher.

9.6 Evaluation

9.6.1 Goal of the Evaluation

We aim to evaluate the proposed method of bounding and over-estimating the deadline miss probability p_{dm} for a task implementing a Markov Model, where the execution times of the jobs vary depending on the task's state. In addition, we aim to investigate the method's sensitivity to the shape of the execution time distribution.

For this purpose, we use a test program with a known Markov Chain structure. The deadline miss probability bounds and estimates calculated with the proposed method are compared to deadline miss probabilities from simulations, and to the deadline miss ratio obtained when running a task under the Linux `SCHED_DEADLINE` scheduling policy that implements CBS based on EDF. To investigate the sensitivity to the distribution shape, test programs with two shapes of execution time distributions are implemented, one with Gaussian and one with shifted/ translated exponential distributions. The Gaussian distribution is used to evaluate the method with the conditions fulfilled. We choose exponential distributions for comparison. With a lower bound on the computation times, at which the probability density is highest and a wider tail compared to the Gaussian distribution, it is chosen as a challenge to the proposed method.

A test program with a three-state Markov chain structure is implemented that activates jobs periodically. In each job, a state transition may be performed, and different computations are performed depending on the current state. Execution time traces are obtained from running the program under FIFO scheduling. These traces are used to estimate the means and standard deviations for the three states of the Markov model. They are also used to estimate the rate and translation parameter for the exponential distribution used in the simulation for comparison with the exponential test program. The transition matrix is known from the test program implementation.

The Markov models obtained in this way are used with the methods described in Section 9.4 to calculate the deadline miss probability bounds and estimates. The maximum number of accumulation periods is set to 20. Three different configurations of server budget and period ratios are used. For each of these configurations, two relative deadlines are evaluated. The configurations are listed in Table 9.2.

The test program is run under `SCHED_DEADLINE` with the different con-

Table 9.2: Server parameters.

Q (ms)	n	k_1	k_2
100	4	7	8
120	3	7	8
90	4	9	10

figurations of server budget, period ratio, and deadline. The deadline miss ratios are calculated and compared to the estimates.

A simulation is performed, where the Markov Model is used to generate execution time samples for 10^6 task periods. Gaussian distributions with the parameters from Table 9.3 are used in the simulation for comparison with the Gaussian test case. Translated exponential distributions with translation and rate parameters from Table 9.4 are used in the simulation for comparison with the exponential test program. The workload in simulation is tracked according to Equation (9.2). Deadline miss ratios for each state and the overall deadline miss ratio are recorded.

9.6.2 Test Setup

A test program with three states has been implemented. The program executes periodically, and jobs perform a state transition according to a transition matrix, followed by a state-dependent computation with a pseudo-random variation. We evaluate two versions of the task, one where the execution times of each state are distributed according to a Gaussian and one where they are distributed according to an exponential distribution. That is, a number is generated from a Gaussian or exponential distribution with parameters depending on the current state. An iteration of additions, modulo operations, and swaps are performed in a small (100 integers) memory area, and the number of iterations is proportional to the generated number. The test program versions are implemented so that the means and standard deviations of the states' distributions are similar. The exponential distributions are shifted to accommodate this. The test program contains a deadline miss counter, and at the end of each job, a check for deadline miss is performed. The program activates 500 jobs before termination. The tests are performed on a Raspberry Pi 3B+ single-board computer with Arch Linux ARM kernel 4.14.87 patched with `PREEMPT_RT` 4.14.87-49, configured with a fully preemptible kernel and timer frequency of 100Hz. The test program is pinned to a core set up as an exclusive `cpuset`, and the scaling governor is set to performance.

9.6.3 Timing Traces and Markov models

Timing information is collected with the ftrace framework, `trace-cmd` is run, recording `sched_switch` events. The execution time is calculated as the time from the process is switched in until the time when it is switched out. Executions of the program for collecting timing information are performed under FIFO scheduling with maximum user-space priority. The traces are used for estimating means and standard deviations of the states. The first 50 execution time measurements are excluded because in some cases there have been outliers in this region. The execution times from the traces are classified into states by cutoff points at 250 ms and 400 ms. The means and standard deviations need to be estimated. Although we know the parameters of the distributions to generate the random numbers, we do not know how these translate into execution times. The means and standard deviations are calculated incrementally. The first estimate uses the first execution time trace, then traces are added until the addition of a trace changes all means and standard deviations by less than 1%. This results in 7 traces being included in the estimate for the Gaussian distributions and 9 traces for the exponential distribution. Histograms of the traces can be seen in Figure 9.11.

The transition matrix of the test program is

$$M = \begin{pmatrix} 0.7 & 0.2 & 0.1 \\ 0.5 & 0.3 & 0.2 \\ 0.5 & 0.4 & 0.1 \end{pmatrix}, \tag{9.35}$$

which gives stationary probabilities of 0.625, 0.25, and 0.125 for the respective states. The means and standard deviations for the states are shown in Tables 9.3 and 9.4. In Table 9.4 the rate and translation parameters of the exponential distribution are also shown. The rate parameter is σ^{-1}, and the translation parameter is $\mu - \sigma$.

Histograms of all execution times from the traces are shown in Figure 9.11. Gaussian distributions with the means and standard deviations are overlaid and scaled with the stationary probabilities.

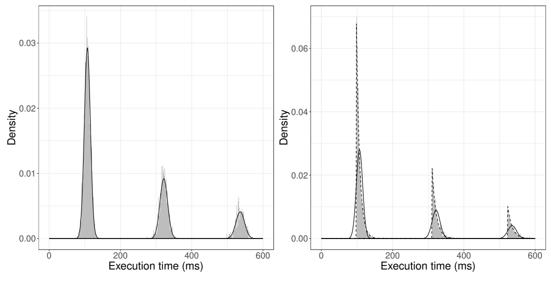
9.6.4 Evaluated Methods for Deriving a Deadline Miss Probability

In the evaluation, we compared four different methods for deriving the deadline-miss probability. Those are:

- **Linux-CBS** : A deadline-miss ratio using a Linux CBS evaluation with a `SCHED_DEADLINE`. For each evaluated combination of server budget Q,

Table 9.3: Characterization of the states of the Gaussian version traces.

State	mean (ms)	standard deviation (ms)
1	107.111	8.513
2	321.611	10.853
3	536.221	12.174



(a) **(b)**

Figure 9.11: Histograms of the execution times from the Gaussian (a) and exponential (b) version FIFO traces with the Gaussian distributions used in the method and the exponential distributions used in the simulation for (b) overlaid.

task to server period ratio n and the relative deadline to server period ratio k, 60 runs of the program under `SCHED_DEADLINE` are performed. Deadline misses after the first 50 periods of each run are recorded, as there appears to be an increased number of deadline misses in a run-in period. The bandwidth is 50%.

- **SIM**: A deadline-miss probability derived with Markov chain simulation. The obtained Markov models are used to simulate a sequence of 10^6 samples. For the Gaussian test program we use the Gaussian parameters in Table 9.3, and for the exponential test program we use the rate and translation parameters from Table 9.4. The output execution time sequence is analyzed with the different configurations of server reservation, period ratio, and deadline as listed in Table 9.2. The workload depletion ratio and the deadline miss ratio for each state are recorded.
- **Bound**: A safe bound on the deadline-miss probability, using the accumulation process defined in Section 9.5 with the upper bound on β as defined in Section 9.4.7.
- **Estimate**: An estimate of the deadline-miss probability, using the accumu-

Table 9.4: Characterization of the states of the exponential version traces.

State	mean (ms)	stddev (ms)	rate (ms^{-1})	translation (ms)
1	106.960	8.891	0.11248	98.0696
2	321.761	11.143	0.089742	310.6178
3	535.293	12.242	0.081688	523.0508

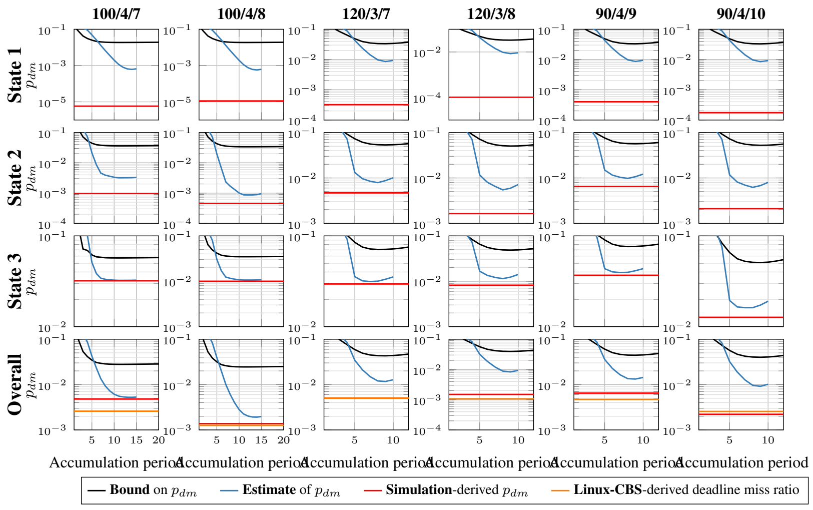


Figure 9.12: Result from test program with Gaussian emission distributions.

lation process defined in Section 9.5 with the estimate on β as defined in Section 9.4.8.

9.6.5 Results & Discussion

The deadline miss probability bounds and estimates p_{dm} obtained during the workload accumulation process are shown in Figures 9.12 and 9.13. Here, we also show deadline miss ratios of simulation with the Markov Model and the mean deadline miss ratios of the executions under `SCHED_DEADLINE`.

From the results shown in Figure 9.12, it is clear that the calculated bound adds significant pessimism compared to the estimates. The pessimism increases with 5 to 70 times when using the bounds on β compared to the estimates. Estimates and bounds are tightest for the state with the highest deadline miss probability. The pessimism in the overall case is 5 to 20 times higher compared to state 3. We also see lower pessimism for the cases with lower utilization and for shorter deadlines. In the case with β estimates and $Q/n/k$ = 100/4/7 the pessimism in state 3 is 1%, but with parameters 120/3/8 it increases to 40%.

When compared to the test with exponential execution time distributions as shown in Figure 9.13, we see that as expected the shape of the execution time distribution affects the deadline miss probability. When the assumption of Gaussian distributions does not hold, in one case (state 3 with server/ deadline parameters 100/4/7), the resulting deadline miss probability estimate, 3.11%,

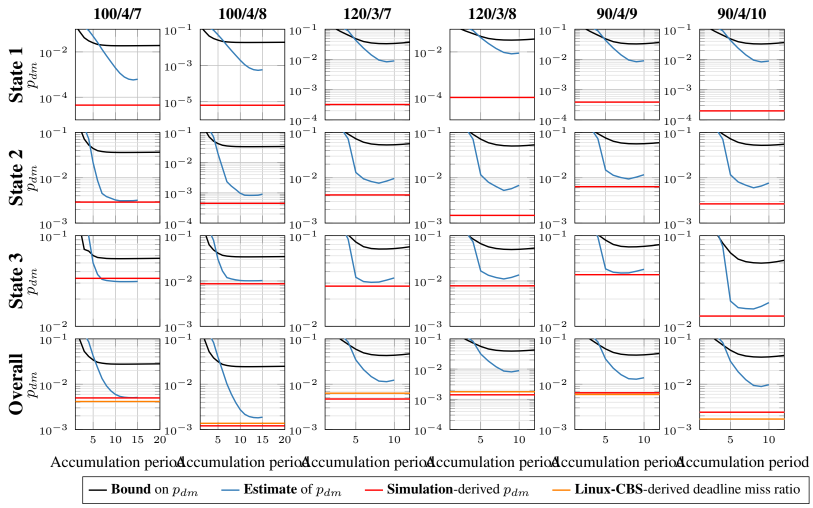


Figure 9.13: Result from test program with exponential emission distributions.

is lower than the deadline miss ratio obtained from simulation, 3.38%.

9.7 Conclusions and Future Work

In this paper, we proposed a workload accumulation scheme for upper bounding or estimating the deadline miss probability of a task executing in a Constant Bandwidth Server (CBS), having execution times modeled by an Hidden Markov Model (HMM) with Gaussian emission distributions. The deadline miss probability bounds and estimates obtained with the method are compared with deadline miss ratios of tasks running under the Linux kernel implementation of CBS. The bounds and estimates are also compared with the results from the simulation for each state separately and for the overall case. Tasks with Gaussian and exponential execution time distributions are evaluated. The comparison of the analytical and empirical results shows that the proposed methods result in a safe upper bound, except in one experiment instance. With Gaussian distributions all bounds and estimates are overestimates. The estimate for the state with the highest DMP is optimistic in one experiment instance performed on the exponential distribution. The estimate over all states is still safe in this case.

The performed evaluation has focused on assessing the pessimism introduced for a case where assumptions hold, and getting an initial estimate of the feasibility of the approach when the shape of the emission distributions differs from the Gaussian assumption. In future work, we intend to perform further evaluation. First, we will evaluate the method with a more realistic workload. Second, we will evaluate the scalability of the approach, in comparison to the method proposed by Frías et al. [16, 5], and investigate the usefulness of the proposed method in adaptive settings. The estimates could potentially be used for monitoring changes in the deadline miss probability and adapting the Quality-of-Service (QoS) level.

Bibliography

[1] Jaume Abella, Maria Padilla, Joan Del Castillo, and Francisco J Cazorla. Measurement-based worst-case execution time estimation using the coefficient of variation. *ACM Trans. Des. Aut. of Elect. Syst. (TODAES)*, 22(4):1–29, 2017.

[2] Luca Abeni and Giorgio Buttazzo. Integrating multimedia applications in hard real-time systems. In *IEEE Real-Time Systems Symp. (RTSS)*, pages 4–13, 1998.

[3] Luca Abeni and Giorgio Buttazzo. Qos guarantee using probabilistic deadlines. In *Euromicro Conf. on Real-Time Systems (ECRTS)*, pages 242–249, 1999.

[4] Luca Abeni and Giorgio Buttazzo. Stochastic analysis of a reservation based system. In *Int. Workshop on Parallel and Distributed Real-Time Systems*, volume 1, 2001.

[5] Luca Abeni, Daniele Fontanelli, Luigi Palopoli, and Bernardo Villalba Frías. A markovian model for the computation time of real-time applications. In *IEEE international instrumentation and measurement technology conference (I2MTC)*, pages 1–6, 2017.

[6] Luca Abeni, Nicola Manica, and Luigi Palopoli. Efficient and robust probabilistic guarantees for real-time tasks. *Journal of Systems and Software*, 85(5):1147–1156, 2012.

[7] Sergey Bozhko, Georg von der Brüggen, and Björn Brandenburg. Monte carlo response-time analysis. In *IEEE Real-Time Syst. Symp. (RTSS)*, pages 342–355, 2021.

[8] Giorgio C Buttazzo, Giuseppe Lipari, Luca Abeni, and Marco Caccamo. *Soft Real-Time Systems: Predictability vs Efficiency*. Springer, 2005.

[9] Xavier Civit, Joan del Castillo, and Jaume Abella. A reliable statistical analysis of the best-fit distribution for high execution times. In *Euromicro Conf. Dig. Syst. Des. (DSD)*, pages 727–734, 2018.

[10] David D. Clark, Scott Shenker, and Lixia Zhang. Supporting real-time applications in an integrated services packet network: Architecture and mechanism. *SIGCOMM Comput. Commun. Rev.*, 22(4):14–26, 1992.

[11] Liliana Cucu-Grosjean, Luca Santinelli, Michael Houston, Code Lo, Tullio Vardanega, Leonidas Kosmidis, Jaume Abella, Enrico Mezzetti, Eduardo Quiñones, and Francisco J Cazorla. Measurement-Based probabilistic timing analysis for multi-path programs. In *Euromicro Conf. on Real-Time Systems (ECRTS)*, pages 91–101, 2012.

[12] Robert Ian Davis and Liliana Cucu-Grosjean. A survey of probabilistic schedulability analysis techniques for Real-Time systems. *LITES: Leibniz Transactions on Embedded Systems*, pages 1–53, 2019.

[13] Robert Ian Davis and Liliana Cucu-Grosjean. A survey of probabilistic timing analysis techniques for Real-Time systems. *LITES: Leibniz Transactions on Embedded Systems*, 6(1):03–1–03:60, 2019.

[14] José Luis Díaz, Daniel F García, Kanghee Kim, Chang-Gun Lee, L Lo Bello, José María López, Sang Lyul Min, and Orazio Mirabella. Stochastic analysis of periodic real-time systems. In *IEEE Real-Time Systems Symp. (RTSS)*, pages 289–300, 2002.

[15] Jose Luis Diaz, Jose Maria Lopez, Manuel Garcia, Antonio M Campos, Kanghee Kim, and Lucia Lo Bello. Pessimism in the stochastic analysis of real-time systems: Concept and applications. In *IEEE Real-Time Systems Symp. (RTSS)*, pages 197–207, 2004.

[16] Bernardo Villalba Frias, Luigi Palopoli, Luca Abeni, and Daniele Fontanelli. Probabilistic real-time guarantees: There is life beyond the i.i.d. assumption. In *IEEE Real-Time and Embedded Technology and Applications Symposium (RTAS)*, pages 175–186, 2017.

[17] Anna Friebe, Filip Marković, Alessandro V. Papadopoulos, and Thomas Nolte. Adaptive runtime estimate of task execution times using bayesian modeling. In *IEEE Int. Conf. on Embedded and Real-Time Computing Systems and Applications (RTCSA)*, pages 1–10, 2021.

[18] Anna Friebe, Alessandro V. Papadopoulos, and Thomas Nolte. Identification and validation of markov models with continuous emission distributions for execution times. In *IEEE Int. Conf. on Embedded and Real-Time Computing Systems and Applications (RTCSA)*, pages 1–10, 2020.

[19] Matthias Ivers and Rolf Ernst. Probabilistic network loads with dependencies and the effect on queue sojourn times. In *International Conference on Heterogeneous Networking for Quality, Reliability, Security and Robustness*, pages 280–296. Springer, 2009.

[20] M Ross Leadbetter, Georg Lindgren, and Holger Rootzén. Conditions for the convergence in distribution of maxima of stationary normal processes. *Stochastic Processes and their Applications*, 8(2):131–139, 1978.

[21] John P Lehoczky. Real-time queueing theory. In *IEEE Real-Time Systems Symp. (RTSS)*, pages 186–195, 1996.

[22] Juri Lelli, Claudio Scordino, Luca Abeni, and Dario Faggioli. Deadline scheduling in the linux kernel. *Software: Practice and Experience*, 46(6):821–839, 2016.

[23] Rui Liu, Alex F Mills, and James H Anderson. Independence thresholds: Balancing tractability and practicality in soft real-time stochastic analysis. In *IEEE Real-Time Syst. Symp. (RTSS)*, pages 314–323, 2014.

[24] Yue Lu, Thomas Nolte, Iain Bate, and Liliana Cucu-Grosjean. A statistical response-time analysis of real-time embedded systems. In *IEEE Real-Time Systems Symp. (RTSS)*, pages 351–362, 2012.

[25] Yue Lu, Thomas Nolte, Johan Kraft, and Christer Norstrom. 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 (RTCSA)*, pages 153–160, 2010.

[26] Yue Lu, Thomas Nolte, Johan Kraft, and Christer Norstrom. Statistical-based response-time analysis of systems with execution dependencies between tasks. In *IEEE Int. Conf. on Engineering of Complex Computer Systems*, pages 169–179, 2010.

[27] Nicola Manica, Luigi Palopoli, and Luca Abeni. Numerically efficient probabilistic guarantees for resource reservations. In *IEEE Int. Conf. on Emerging Technologies & Factory Automation (ETFA)*, pages 1–8, 2012.

[28] Pau Martí, Josep M Fuertes, Gerhard Fohler, and Krithi Ramamritham. Improving quality-of-control using flexible timing constraints: metric and scheduling. In *IEEE Real-Time Systems Symp. (RTSS)*, pages 91–100, 2002.

[29] Dorin Maxim and Liliana Cucu-Grosjean. Response time analysis for fixed-priority tasks with multiple probabilistic parameters. In *IEEE Real-Time Systems Symp. (RTSS)*, pages 224–235. IEEE, 2013.

[30] Dorin Maxim, Frank Soboczenski, Iain Bate, and Eduardo Tovar. Study of the reliability of statistical timing analysis for real-time systems. In *International Conference on Real Time and Networks Systems (RTNS)*, pages 55–64, 2015.

[31] Alex F. Mills and James H. Anderson. A multiprocessor server-based scheduler for soft real-time tasks with stochastic execution demand. In *IEEE Int. Conf. on Embedded and Real-Time Computing Systems and Applications (RTCSA)*, pages 207–217, 2011.

[32] Luigi Palopoli, Daniele Fontanelli, Luca Abeni, and Bernardo Villalba Frias. An analytical solution for probabilistic guarantees of reservation based soft real-time systems. *IEEE Trans. Parallel and Distributed Systems*, 27(3):640–653, 2015.

[33] Luca Santinelli, Jérôme Morio, Guillaume Dufour, and Damien Jacquemart. On the sustainability of the extreme value theory for wcet estimation. In *Int. Workshop on Worst-Case Execution Time Analysis (WCET)*, 2014.

[34] Bernardo Villalba Frias. *Bringing Probabilistic Real-Time Guarantees to the Real World*. PhD thesis, University of Trento, 2018.

[35] Georg von der Brüggen, Nico Piatkowski, Kuan-Hsun Chen, Jian-Jia Chen, Katharina Morik, and Björn B Brandenburg. Efficiently approximating the worst-case deadline failure probability under EDF. In *IEEE Real-Time Syst. Symp. (RTSS)*, pages 214–226, 2021.