Real-Time Systems
https://doi.org/10.1007/s11241-024-09431-7

Efficiently bounding deadline miss probabilities of Markov chain real-time tasks

Anna Friebe[1] · **Filip Marković**[1,2] · **Alessandro V. Papadopoulos**[1] · **Thomas Nolte**[1]

Accepted: 16 September 2024

Abstract

In real-time systems analysis, probabilistic models, particularly Markov chains, have proven effective for tasks with dependent executions. This paper improves upon an approach utilizing Gaussian emission distributions within a Markov task execution model that analyzes bounds on deadline miss probabilities for tasks in a reservation-based server. Our method distinctly addresses the issue of runtime complexity, prevalent in existing methods, by employing a state merging technique. This not only maintains computational efficiency but also retains the accuracy of the deadline-miss probability estimations to a significant degree. The efficacy of this approach is demonstrated through the timing behavior analysis of a Kalman filter controlling a Furuta pendulum, comparing the derived deadline miss probability bounds against various benchmarks, including real-time Linux server metrics. Our results confirm that the proposed method effectively upper-bounds the actual deadline miss probabilities, showcasing a significant improvement in computational efficiency without significantly sacrificing accuracy.

Keywords Real-time systems · Hidden Markov model · Probabilistic schedulability analysis · Deadline miss probability

✉ Anna Friebe
anna.friebe@mdu.se

Filip Marković
fmarkovic@mpi-sws.org

Alessandro V. Papadopoulos
alessandro.papadopoulos@mdu.se

Thomas Nolte
thomas.nolte@mdu.se

[1] Mälardalen University, IDT, Box 883, 721 23 Västerås, Sweden

[2] Max Planck Institute for Software Systems (MPI-SWS), Paul-Ehrlich-Strasse, Building G 26, 67663 Kaiserslautern, Germany

Published online: 10 October 2024

1 Introduction

Soft real-time systems permit limited deadline misses, impacting the Quality of Service (QoS) (Clark et al. 1992) or Quality of Control (QoC) (Martí et al. 2002) to a tolerable degree. The tolerance is often modeled as a constraint on the number of deadline misses to maintain an acceptable level of QoS or QoC (Buttazzo et al. 2005). Findings from a recent survey by Åkesson et al. (2022) indicate a predominant presence of soft real-time systems in the industry, underscoring the significance of their analytical study.

Hidden Markov Models (HMMs) have been effectively utilized to model execution times in systems with dependencies that exhibit regular variations. An introduction to the HMM concept can be found in Rabiner (1989). Studies like (Abeni et al. 2017; Frías et al. 2017) have employed Markov models with discrete emission distributions, particularly in estimating the probability of missing deadlines under a Constant Bandwidth Server (CBS). Additionally, continuous-emission distributions have been explored by Friebe et al. (2020, 2021, 2023).

While HMMs with continuous emission distributions have been applied in execution time estimation (Friebe et al. 2020, 2021), the extension to workload distribution inference and deadline-miss probabilities has been a recent development (Friebe et al. 2023). As in previous work (Abeni et al. 2017; Frías et al. 2017), this analysis is done with a reservation-based scheduling approach, that allows for analysis of each server separately, due to the timing isolation property. This newer exploration has shown potential but highlighted challenges in computational efficiency when dealing with complex systems.

Building upon these recent findings, our paper specifically targets the computational efficiency issue identified by Friebe et al. (2023). We propose an enhanced method for bounding the deadline-miss probability of real-time tasks using HMMs with continuous emission distributions. A key contribution of this work is developing a state-merging technique that enhances computational efficiency in terms of time and space complexity, where traditional methods are computationally intensive or even infeasible (Sect. 6).

The evaluation, presented in Sect. 7, employs a task managing a Furuta pendulum (Vreman et al. 2021). It compares the derived bounds with real-time deadline miss ratios under Linux's CBS implementation, `SCHED_DEADLINE` (Lelli et al. 2016), alongside estimates from a discrete-emission Markov Model (Frías et al. 2017, 2018), and simulation-based estimates.

The paper's organization is as follows: Related work is reviewed in Sect. 2. Section 3 defines the notation and system model. The execution time model and methodology for deriving and analyzing the deadline miss probability bounds are presented in Sect. 4. The iterative update process for the bounds is detailed in Sect. 5. State reduction techniques and their impact on time complexity are discussed in Sect. 6. Section 7 showcases evaluations and results, and Sect. 8 concludes the paper with a discussion on future work.

2 Related work

The surveys by Davis and Cucu-Grosjean (2019a, 2019b) offer a detailed overview of the field of probabilistic schedulability and timing analysis in real-time systems. Two of the many challenges highlighted in their surveys are the probabilistic analysis of dependent tasks and the safe estimation of deadline-miss probabilities (DMP) for such tasks.

The issue of dependence in execution-time distributions and its impact on the potential unsound estimation of DMP when independence is assumed was initially identified by Tia et al. (1995). The importance of assuming independence among jobs of the same task for deriving sound response time distribution was first recognized by Díaz et al. (2002), while in the concluding remarks of their paper, they restated the fact that many systems do not adhere to the independence assumption. For this reason, a fundamental concept of *stochastic pessimism* for proper upper-bounding of the execution-time distributions was explored by Diaz et al. (2004). Over the years, several research directions have evolved to address the above-mentioned issues.

One of the most used approaches was the one based on the probabilistic Worst-Case Execution Time (pWCET), which is supposed to upper-bound the ground-truth execution-time distribution of a job such that it can be safely used with convolution and independence-assuming analytical approaches in spite of possible dependence with other jobs. In this line of research, Cucu-Grosjean (2013) established the relation between the ground-truth execution-time distributions and pWCET, while Davis et al. (2017) clarified the difference between the confidence-based pWCET and the upper-bounding one. The surveys (Davis and Cucu-Grosjean 2019a, b) also provide extensive investigation on the definition and use of pWCETs. More recently, Bozhko et al. (2023) formalized a rigorous, axiomatic definition of pWCET, using the Coq proof assistant.

Many probabilistic schedulability analyses have been proposed over the years using pWCET and similar independence-assuming distributions for fixed-priority fully-preemptive scheduling. von der Brüggen et al. (2018) used the Hoeffding and Bernstein inequalities for the estimation of DMP, while Chen et al. (2019) used Chernoff bound. Marković et al. (2021) contributed an optimal resampling strategy and an efficient circular-convolution algorithm. Bozhko et al. (2021) introduced a method based on Monte-Carlo sampling. In contrast, von der Brüggen et al. (2021) suggested a method to approximate the DMP under Earliest Deadline First (EDF) scheduling, accommodating dependencies across a limited number of consecutive jobs. More recent work by Chen et al. (2022) corrected an error in the critical-instant assumption commonly found in various independence-based methods. Zagalo et al. (2022) have developed queuing theory-based approximations for the response-time distributions, while Markovic et al. (2022) utilized the Berry–Esseen theorem to approximate response-time distributions. In the context of other scheduler assumptions, Marković et al. (2018) provided probabilistic analysis for limited-preemptive scheduling, which is a generalization over fully and non-preemptive scheduling. Most recently, Günzel et al. (2023)

proposed a probabilistic reaction time analysis for cause-effect chains based on sporadic tasks.

The issue of dependence has also been addressed within the framework of Extreme Value Theory (EVT), particularly in its application to measurement-based statistical analysis for both execution times (Cucu-Grosjean et al. 2012; Lima et al. 2016; Lima and Bate 2017) and response times (Lu et al. 2010a, b, 2012).

Despite the widespread adoption of EVT in both academic research and practical applications, it is not without certain limitations (de Barros Vasconcelos and Lima 2022). EVT relies on the premise that statistical limit laws are applicable to the sample set at hand (Coles 2001). EVT analysis necessitates certain conditions like the assumption of stationarity (Leadbetter et al. 1978) or extremal independence in the distribution under consideration (Santinelli et al. 2014).

Regarding the works that do not consider independence-assuming distributions, Bernat et al. (2005) introduced the concept of copulas in timing analysis. Copulas model dependencies between random variables, a copula transforms marginal distributions of random variables into a joint probability distribution. Ivers and Ernst (2009) developed an approach for fixed-priority preemptive scheduling systems, utilizing completely known ground-truth probability distribution for each task. Their method, incorporating copulas and Frechet bounds, facilitated the derivation of probabilistic response-time bounds. Recently, Marković et al. (2023) introduced a correlation-tolerant analysis for DMP estimation, leveraging upper bounds on both the expected values and standard deviations of job execution-time distributions. Their analysis improves upon Cantelli's inequality to derive sound probabilistic response times in the presence of possibly correlated distributions.

More in line with this work, in the context of *server-based schedulers*, Mills and Anderson (2011) derived bounds for response time and tardiness for soft real-time tasks with stochastic execution times, focusing on execution time dependence within distinct time windows. In a related development (Liu et al. 2014) proposed the concept of independence thresholds, positing that execution times above a certain threshold can be treated as independent. One major advantage of server-based scheduling is that it provides timing isolation, allowing for analysis of each server separately.

The *Constant-Bandwith Server* (CBS), was originally introduced by Abeni and Buttazzo (1998) and later used for deriving probabilistic deadlines to ensure Quality of Service (QoS) guarantees (Abeni and Buttazzo 1999). In later works, it has also been analyzed with probabilistic execution times(Abeni and Buttazzo 2001; Palopoli et al. 2016) and probabilistic interarrival times (Abeni et al. 2012; Manica et al. 2012).

In one of the seminal papers for probabilistic analysis of real-time systems, Díaz et al. (2002) conducted a response time analysis for periodic tasks characterized by independent random execution times, demonstrating that the backlog in this context can be modeled as a *Markov chain*.

Recent studies, diverging from the worst-case DMP that has been prevalent in the previously cited works, have embraced the long-run frequency interpretation of DMP. In this vein, Abeni et al. (2017) and Frías et al. (2017) utilized Markov chain models with discrete emission distributions under CBS. Their work concentrated on

analyzing the steady-state response time distribution and included comparisons with results obtained under Linux's SCHED_DEADLINE. They noted that the analysis duration is influenced by factors such as the range of execution times, the number of states, and the scaling factor used for resampling (Frías 2018), which can significantly affect the analysis time and space complexity.

Furthermore, the estimation of the execution times modeled as continuous Gaussian distributions within Markov chains have been explored by Friebe et al. (2020, 2021), while the analysis in terms of deadline miss probabilities was conducted recently (Friebe et al. 2023). Friebe et al. (2023) addressed the DMP estimation, applying Hidden Markov Models (HMMs) with Gaussian emission distributions for schedulability analysis. This approach, akin to the work of Frías et al. (2017) and Abeni et al. (2017), explicitly incorporates dependencies within the HMM framework, with the CBS providing task isolation, thereby focusing the workload analysis on the specific task rather than the entire system. Although the analysis by Friebe et al. (2023) offered improvements due to utilizing the continuous-based HMM model of execution times, they showed that the analysis still may suffer from time and space complexity.

In Sect. 6 of this paper, we introduce state-merging techniques designed to enhance the time and space efficiency of the methods presented by Friebe et al. (2023). These techniques are developed to maintain high accuracy in DMP estimations. In Table 1 the HMM approaches with continuous and discrete emission distributions are compared, and the contributions of this paper are outlined.

3 System model and notation

Table 2 contains the notation used in the paper. Superscript * indicates true values, $^{\uparrow}$, and $^{\downarrow}$ indicate upper and lower bounds.

We use the concept of upper bounding random variables according to Definition 1. This is also referred to as the usual stochastic order (Shaked 2007) or first-order statistical dominance (Diaz et al. 2004). This paper uses the term upper bound as in Davis and Cucu-Grosjean (2019b).

Definition 1 (cf. Diaz et al. 2004; Davis and Cucu-Grosjean 2019b; Shaked 2007) 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 $\mathcal{Y}$. We denote this relation by $\mathcal{X} \geq \mathcal{Y}$.

We define a partial Gaussian distribution in Definition 2, that is used to upper bound workload distributions. Consider a Gaussian $\mathcal{N}(\mu, \sigma^2)$ with probability density function $f(x|\mu, \sigma^2)$. Let $\Phi(x)$ be the cumulative density function of the standard normal distribution.

Definition 2 A *partial Gaussian distribution* $\mathcal{N}^{tail}(\mu, \sigma^2, \alpha)$, originated from a Gaussian distribution $\mathcal{N}(\mu, \sigma^2)$, is defined by the probability density function:

Table 1 Contribution overview

	Discrete emission	Continuous emission
Overview of the theoretical contributions		
Timing analysis (TA)	Frías et al. (2017), Abeni et al. (2017))	Friebe et al. (2020)
DMP analysis	Frías et al. (2017), Abeni et al. (2017)	Friebe et al. (2023) as described in this paper
Comparison of the models and their analysis properties		
State No. identification	–	Friebe et al. (2020)
Adaptive TA	–	Friebe et al. (2021)
Time and space complexity (DMP analysis)	Frías et al. (2017), Abeni et al. (2017)	Friebe et al. (2023) as described in this paper
	Dependent on:	Independent of:
	– Resampling scale	– Resampling scale
	– Execution time range	– Execution time range
	Requires full steady-state distribution (Frías et al. 2017)	Iterative procedure with an adjustable complexity
		State number reduction procedure (Sect. 6)

Table 2 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

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

The probability of values lower than α is set to zero in the partial Gaussian distribution. The probability density of the remaining values are normalized, so that the distribution integrates to one.

We use convolutions, as defined in Definition 3, in the derivation of workload distributions.

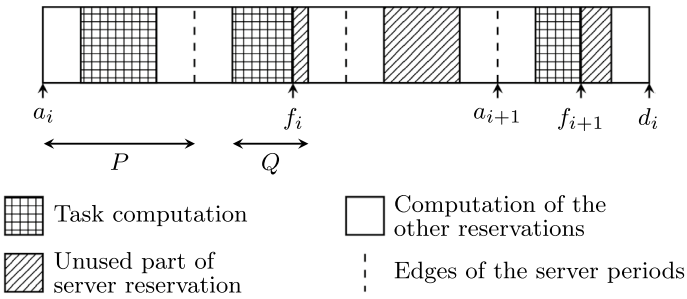

Fig. 1 An illustration of the task model and the reservation-based server

Definition 3 The convolution of f and g, denoted with the $*$ operator is:

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

3.1 Task model

Let the real-time task τ consist of a sequence of jobs J_i, $i \in \mathbb{N}$. Each job J_i has the arrival time a_i, execution time c_i and finishing time f_i. The task is periodic and jitter-free, i.e., $a_{i+1} = a_i + T$, a_0 is the arrival time of the first job. Jobs can be preempted, $f_i \geq a_i + c_i$. The execution time is modeled as a random variable. The random variable $\mathcal{R}$ models the response time, the duration from activation time to finish time of a job.

A job J_i has the deadline d_i determined by a relative deadline D such that $d_i = a_i + D$. Jobs are executed until completion, even if deadlines are missed. The relative deadline may be longer than the task period. The probability that a randomly selected job finishes after the deadline, $p_{dm} = \mathbb{P}(\mathcal{R} > D)$ is the main concern of this paper.

3.2 Scheduling algorithm

The task is served as the sole task of a reservation-based server, and guaranteed to receive Q units of processing time within each server period. The bandwidth $B = Q/P$ is the fraction of the processing time dedicated to the task. $T = n \cdot P$, that is the task period is an integer multiple of the server period. The relative deadline is also an integer multiple of the server period, $D = k \cdot P$. In the evaluation a CBS is used, a CBS with a properly selected server period fulfills the necessary requirements.

The task model and reservation-based server are illustrated in Fig. 1. Here, we have $n = 3$, $T = 3 \cdot P$. In the illustration the relative deadline is longer than the

period. With a deadline longer than the period, a longer job may steal computation time from the next job of the task. If the next job is short, they may both meet their deadlines.

4 Execution time model and analysis

4.1 Markov chain execution times

The execution times of the task we consider are described by a Markov model defined by a set of S states $\mathbb{S}$, a $S \times S$ state transition matrix M and a set $\mathbb{C}$ of S execution time distributions or *emission distributions* related to the respective state, $S \in \mathbb{N}$. We have $\mathbb{S} = \{1, 2, \ldots, S\}$. In M the element $m_{a,b}$ represents the probability of the task being in state b at task period $i + 1$, given that it is in state a at task period i. $\mathbb{C} = \{C_1, C_2, \ldots, C_S\}$ where each C_s is modeled as Gaussian distributions with mean μ_s, and variance σ_s^2, i.e., $C_s \sim \mathcal{N}(\mu_s, \sigma_s^2)$. The Markov Chain is irreducible, that is a chain where from any state you can reach any other state in a sequence of steps. For an irreducible finite-state Markov Chain, stationary probabilities of the different states (Harchol-Balter 2024) exist and are unique. The stationary probabilities represent the long-run proportion of jobs with execution times described by the different Gaussian distributions.

Example 1 When introducing the ideas and analysis, we will use an example execution time Markov Model and reservation-based server. The parameters are chosen mainly for illustration, and to arrive at simple numerical answers in some of the applications of the example. The Markov Model is defined by:

$$S = 2, \quad M = \begin{pmatrix} 0.9 & 0.1 \\ 0.7 & 0.3 \end{pmatrix}, \quad \mathcal{C} = \{\mathcal{N}(1, 0.25), \mathcal{N}(2, 1)\}.$$

In this example all transition probabilities are strictly positive, so the Markov Chain is clearly irreducible and we can calculate the stationary probabilities. These are 0.875 for state 1 and 0.125 for state 2. In our example, the CBS is defined as $n = 2$ and $Q = 1$. The deadline is defined by $k = 4$.

The representation of Gaussian emission distributions requires only a few distribution parameters, for example, the emission distribution associated with state 1 in Example 1 is fully specified by the mean 1 and the variance 0.25. With discrete distributions probabilities for each execution time value need to be stored, with respect to a chosen scaling factor. For Example 1 we might choose to represent execution times with a resolution of 0.01. For state 1 we could list execution times from 0.01 to 4.01, where each of these is associated with a probability. The probability associated with 2.01 would be the probability of execution times et, $2 < et \leq 2.01$. Gaussian emission Markov models are shown to be applicable, in Friebe et al. (2020) for a video decompression use case, and in Friebe et al. (2021) in a dynamic setting

with model adaptation. The Gaussian distribution may appear simplistic. However, general distribution shapes can be approximated by a combination of Gaussians. The assumption of independent execution times within each state implies that more states may be necessary in the model to capture dependencies in the transition matrix. A close to discrete model can be envisioned with states where the emission-distribution variance is near zero. The fact that a representation with Gaussian emission distributions may require more states is a disadvantage of this approach. With discrete representation, pessimism is introduced with the scaling factor and if downsampling is needed. With continuous representation, pessimism is introduced at other points, for example in our case when upper bounding the workload distributions. Further, there are other options for continuous representations, for example Zagalo et al. (2022) use inverse Gaussian mixture distributions for response times. In our approach we rely on the simplicity of convolution of Gaussian distributions.

4.2 Problem formulation

We bound the expected deadline miss probability of a randomly selected job of a task. Task execution times are defined as in Sect. 4.1, and the task is served by a reservation-based server as described in Sect. 3.2.

In the survey on schedulability analysis by Davis and Cucu-Grosjean (2019a) three interpretations of the probability of a deadline miss are listed:

1. "As a probability with a long-run frequency interpretation equating to the expected number of missed deadlines divided by the total number of deadlines in a long (tending to infinite) time interval.
2. As the probability that a randomly selected job will miss its deadline, which is broadly equivalent to the long-run frequency interpretation.
3. As a bound on the probability that any specific job will miss its deadline."

Chen et al. (2024) refer to the same concept as the deadline miss rate, and formulate the question: "What is the ratio of jobs missing their deadlines in the long run?" We agree with Davis and Cucu-Grosjean that interpretations 1 and 2 are broadly equivalent. Extending interpretation 2 to include the average component that is in focus in interpretation 1, we focus on the expected deadline miss probability of a randomly selected job. The intention is to remove any ambiguity with interpretation 3 or the Worst Case Deadline Failure Probability, an upper bound on the probability that any single job of a task misses its deadline (Davis and Cucu-Grosjean 2019a). We find the term deadline miss probability more natural compared to deadline miss rate in our context with states with different execution time distributions.

4.3 Overview of the proposed approach

We will obtain an upper bound on the expected deadline miss probability of a randomly selected job of the task in a reservation-based server.

The proposed method is based on a workload accumulation scheme. The main idea is outlined below, followed by the details in the remaining subsections. The starting point of the approach is that the deadline miss probability of a job depends on the execution time of the job, and on the amount of remaining work from previous jobs that have not been completed yet. We categorize jobs into different classes with different deadline miss probabilities. By calculating or bounding the deadline miss probabilities of jobs belonging to each class, and the probability of randomly selecting a job from each class, we bound the expected deadline miss probability of a randomly selected job.

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 Abeni and Buttazzo (1999):

$$v_i = \underbrace{\max(0, v_{i-1} - nQ)}_{\text{carry-in workload}} + c_i. \tag{3}$$

where the first term accounts for the previous workload, that is 0 for the first period, and for task periods where all work from previous jobs has been completed before the new job arrival. In these periods, jobs arrive at idle points with 0 carry-in workload and $v_i = c_i$, in particular $v_1 = c_1$.

Observation 1 The pending workload at a job arrival is affected by the execution time requirements of jobs arriving since the last idle point.

In our proposed method, the job classes are related to the state sequence since the last idle point. In Example 1, let the jobs arriving at an idle point when the task is in the first state of the Markov Model belong to one specific class. When selecting a job at random, there is a probability of about 0.78 that the job belongs to this class. The deadline miss probability for this class of jobs only depends on the execution time distribution for the first state and the server properties. It is the survival function or 1-CDF of $\mathcal{N}(1, 0.25)$ at 4, about $9.8 \cdot 10^{-10}$.

Observation 2 The deadline miss probability for a class of jobs is at most 1.

In the proposed method, we will derive more precise bounds for several classes. For the remaining, we will use Observation 2 to upper bound the deadline miss probability. We construct an approximate bound of the expected deadline miss probability of a randomly selected job from Example 1 to illustrate the idea:

$$DMP \lessapprox 0.78 \cdot 0.98 \cdot 10^{-10} + 0.22 \cdot 1 \tag{4}$$

To model a state sequence from the latest idle point, we introduce the concept of workload accumulation sequences. Illustrations of workload accumulation sequences for some classes in Example 1 are displayed in Fig. 2.

The class where, at a job arrival, the task is in state 1, and there is no carry-in workload, is displayed as the black node at task period 1. The workload accumulation sequence is $h = (1)$. The gray node and arrow represent the class of jobs where

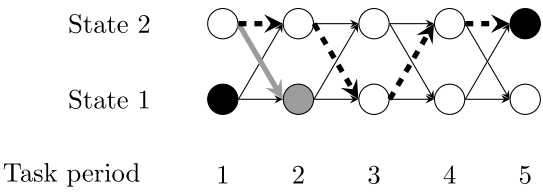

Fig. 2 Illustration of workload accumulation sequences of the example

jobs arrive in state 1 and in the second task period after an idle point, with the first period in state 2. The accumulation sequence $h = (2, 1)$. The black node and dashed arrows represent the class of jobs arriving at the fifth task period after the latest idle point, with $h = (2, 2, 1, 2, 2)$. The *accumulation sequence* is modeled as a random variable $\mathcal{H}$ that can take the values of any possible workload accumulation path. In Fig. 2, with the gray path we illustrate one possible value $h = (2, 1)$, taken by $\mathcal{H}$.

Definition 5 Each arrival of a job J_i results in an *accumulation sequence* $h(J_i)$. Let b denote the task state at the arrival of J_i. If there is an idle point directly prior to the arrival, the resulting $h(J_i) = (b)$. If there is carry-over workload from the previous job, let $h(J_{i-1}) = (\ldots, a)$ denote the accumulation sequence resulting from the prior job arrival. Then $h(J_i) = (\ldots, a, b)$.

In this way, each job that arrives is related to one specific h that describes the accumulated workload since the last idle point, and the task's state at the arrival of this job is always in the last component of the corresponding h.

The evolution of $\mathcal{H}$ is described by an infinite-state Discrete-Time Markov Chain, and each workload accumulation sequence represents one job class and one state in this chain. State transitions occur at job arrivals. The possible transitions from a state $h = (\ldots, a)$ are:

1. A transition from h to $(\ldots, a, b)$ has strictly positive probability if $m_{a,b} > 0$.
2. A transition from h to $(b)(b)$ has strictly positive probability if $m_{a,b} > 0$.
3. No other transitions from h are possible.

The probability of randomly selecting a job resulting in a certain accumulation sequence is the stationary probability of the state in the Markov Chain. This stationary probability exists for an infinite-state Discrete-Time Markov if the chain is ergodic (Harchol-Balter 2024)—that is when the chain is irreducible, aperiodic and recurrent. The accumulation sequence Markov Chain is irreducible if the execution time Markov Chain is irreducible. If all states can be reached from all states in the execution time Markov Chain, the same is true for the accumulation sequence Markov Chain. The accumulation sequence Markov Chain is also aperiodic, which means that the greatest common divisor of the set of integers n, such that you can get from one state to the same state in n steps, is 1 for all states. In an infinite-state Markov Chain, either all states are recurrent, and the chain is recurrent, or all states are transient. A state is recurrent if when we start in that state, the probability is 1 that we ever return to the same state. The workload in the server can be seen as a queue, and a queue is in steady-state if the average arrival rate is lower than the average service rate (Medhi 2003). This is equivalent

to the average utilization of the task being lower than the server's bandwidth. Under this condition the accumulation sequence Markov Chain is recurrent. Returning to Example 1, the average computation requirement over a task period is $0.875 \cdot 1 + 0.125 \cdot 2 = 1.125$, resulting in an average utilization of $\frac{1.125}{2 \cdot P} = \frac{0.5625}{P}$. Since the server's bandwidth is $\frac{1}{P}$, the task's computational requirement is met over time, and the accumulation sequence Markov Chain is ergodic.

Definition 6 The probability $p_{in}(h)$ of randomly selecting a job resulting in the accumulation sequence h is the stationary probability of this state in the accumulation sequence Markov Chain.

Definition 7 The conditional probability that a job resulting in h misses its deadline is defined as $p_{dm}(h) = \mathbb{P}(\mathcal{R} > D | \mathcal{H} = h)$.

Definition 8 The Deadline Miss Probability $DMP(j)$ for the j-th job since the last depletion point is defined as

$$DMP(j) = \frac{1}{\sum_{\forall h \in H(j)} p_{in}(h)} \sum_{\forall h \in H(j)} p_{in}(h) \cdot p_{dm}(h) \tag{5}$$

where the set $H(j)$ represents accumulation sequences resulting from job arrivals at the j-th task period from the last idle point.

Returning to Example 1, there are two accumulation sequences in $H(1)$, arriving at an idle point. We already discussed $h = (1)$. The second is $h = (2)$, and the probability of randomly selecting a job resulting in $h = (2)$ is about $p_{in}((2)) \approx 0.099$. The deadline miss probability $p_{dm}((2))$ is the survival function of $\mathcal{N}(2, 1)$ at 4, about $p_{dm}((2)) \approx 0.023$. In our example, $DMP(1) \approx (7.6 \cdot 10^{-10} + 0.099 \cdot 0.023)/(0.78 + 0.099) \approx 2.6 \cdot 10^{-3}$.

Definition 9 The Deadline Miss Probability DMP is the expected deadline miss probability of a randomly selected job from the task. Given a task with execution times described by the model in Sect. 4.1, and served by a reservation-based server with a bandwidth exceeding the average task utilization, DMP is obtained as

$$DMP = \sum_{i=1}^{\infty} DMP(i) \sum_{\forall h \in H(i)} p_{in}(h). \tag{6}$$

Since $p_{in}(h)$ are the stationary probabilities of the accumulation sequence Markov Chain (Definition 6), the sum of $p_{in}(h)$ over all h equals 1.

Problem The sum of Eq. (6) has a countably infinite number of terms. This paper investigates finding a bound for DMP with a finite number of terms.

Observation 3 Consider a job that arrives with a carry-in workload, i.e., it does not arrive directly after an idle point. Then this job arrives $j+1$ task periods after the last idle point, and it succeeds a job arriving j task periods after the last idle point. The probability of randomly selecting a job with workload accumulation from $j+1$ periods is never higher than the probability of randomly selecting a job arriving with workload accumulation from j periods.

The main idea is to find tighter bounds $DMP(i)$ for the first terms in Eq. (6), since due to Observation 3 the weighting sums over p_{in} are highest for the first terms. For larger i, when the sums over p_{in} are small, we let $DMP(i) = 1$.

Returning to Example 1, using the information from the first task period, we have:

$$\begin{aligned} DMP &= \sum_{i=1}^{\infty} DMP(i) \sum_{\forall h \in H(i)} \approx 2.3 \cdot 10^{-4} + \sum_{i=2}^{\infty} DMP(i) \sum_{\forall h \in H(i)} p_{in}(h) \\ &\leq 2.3 \cdot 10^{-4} + \sum_{i=2}^{\infty} \sum_{\forall h \in H(i)} p_{in}(h) \approx 0.12 \end{aligned} \tag{7}$$

The bound is still very pessimistic. However, going from one accumulation sequence $h = (1)$ in Eq. (4) to two $h = (1)$ and $h = (2)$ reduces the bound from 0.22 to 0.12.

4.3.1 Outline of the remainder of this section

An upper bound on DMP is obtained by deriving upper bounds on p_{in} and p_{dm}. The probability $p_{in}(h)$ of randomly selecting a job with the accumulation sequence h depends on the execution time distributions along h, the transition probabilities, and the probability of workload depletion in each state. The conditional deadline miss probability $p_{dm}(h)$ for jobs where the arrival results in h depend on the execution time distributions in h. We divide the workload accumulation process in two steps. We first compute upper bounds on p_{in} and p_{dm} up until N task periods from the latest idle point. As N grows, the sum of the products of $p_{in}(h)$ and $p_{dm}(h)$ approaches the true deadline miss probability. Second, the sum of p_{in} values in the remaining accumulation sequences of length $N+1$ to infinity, is upper bounded. We refer to this sum as β. We assume that the p_{dm} is 1 for jobs that result in these accumulation sequences, and this gives a safe upper bound on DMP in Eq. (40).

The *steps for deriving a safe bound* on DMP are outlined in the following sections:

Section 4.4: Upper bounding p_{dm} and p_{in} for the terms in Eq. (6) requires upper bounding the pending workload distributions associated with each accumulation sequence. In this section we describe how to derive the parameters of an upper bounding partial Gaussian distribution as given in Eqs. (19)–(21). The bounds on p_{in} also rely on a lower bound on the pending workload distributions. The parameters for a lower bounding Gaussian distribution are derived as Eqs. (19) and (20).

Section 4.5: For jobs arriving at an idle point, p_{in} depends on the probability of workload depletion p_{wd} for each state, the stationary state probabilities and the transition probabilities $m_{a,b}$. Upper and lower bounds are provided in Eqs. (24) and (23).

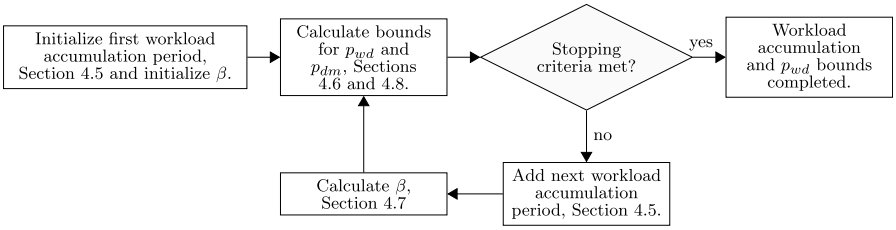

Fig. 3 The workload accumulation process

Jobs arriving with carry-in workload have p_{in} depending on transition probabilities, and the probability of jobs with shorter accumulation sequences resulting in carry-over workload. We find that all p_{in} bounds are linear combinations of p_{wd} for the different states, and given in in Eqs. (28) and (27).

Section 4.6: Bounds on p_{wd} are derived, relying on stationary probabilities and the sum of p_{in} in accumulation periods after N, denoted as β.

Section 4.7: A bound on β is derived and given in Eq. (37). This bound is utilized for computing the lower bounds on p_{in}, p_{co}, and finally p_{wd}.

Section 4.8: An upper bound on p_{dm} is presented, using the bounds on workload distributions, p_{in} and β. The p_{dm} for a state is upper bounded in Eq. (39). Each job's deadline miss probability is accounted for with the accumulation sequence resulting from the job's arrival. This is the case even with long relative deadlines, when actual deadlines are not missed until after the arrivals of subsequent jobs.

The iterative workload accumulation process connects all these different parts, and is presented in Sect. 5 with an example. An illustration of the process with references to relevant sections is provided in Fig. 3. The workload distribution bounds from Sect. 4.4 are used in all remaining sections and are not specifically referenced in the figure.

4.4 Bounding the conditional pending workload distribution associated with a workload accumulation sequence

Upper and lower bounds of pending workload distributions conditioned on the job arrival resulting in a given accumulation sequence since the last idle point are derived. To derive upper bounds on p_{dm} we require the upper bounds on the workload distributions. To derive upper and lower bounds on p_{in}, p_{co}, β and p_{wd} we require upper and lower bounds on the workload distributions. With an example from Fig. 2, we consider the pending workload distribution for jobs arriving in state 1, in the second task period from the last idle point, given that the first job after the idle point arrived in state 2, that is the path marked as gray, $h = (2, 1)$.

Definition 10 The conditional pending workload distribution $\mathcal{V}_h$ for jobs resulting in a given accumulation sequence h has the probability density function $\mathbb{P}(v|\mathcal{H} = h)$.

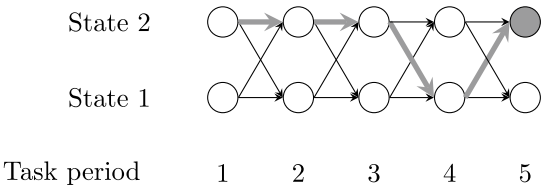

Fig. 4 Illustration of a workload accumulation sequence that contributes to the same workload accumulation vector as the dashed sequence in Fig. 2

We will derive bounds for this conditional pending workload distribution that are independent of the order of the state visits in the accumulation sequence, and only dependent on the number of visits in each state. For this purpose we define the random variable $\tilde{\mathcal{H}}$ that takes S-dimensional vector values, where each element denotes the number of visits in the corresponding state since the last idle point. As an example, the dashed line in Fig. 2 showing $h = (2, 2, 1, 2, 2)$ and another accumulation sequence $h = (2, 2, 2, 1, 2)$ illustrated in Fig. 4 contribute to the same accumulation vector. Both sequences result from jobs arriving 5 task periods after the latest idle point, and they have the same number of visits in each state, $\tilde{h} = [1, 4]$. Let $\tilde{h}[s]$ denote taking the s-th element of $\tilde{h}$, and $\tilde{h}_{+s}$ is the accumulation vector with elements: $\tilde{h}_{+s}[i] = \tilde{h}[i], i \neq s, \tilde{h}_{+s}[s] = \tilde{h}[s] + 1$. This simplifies the notation of the workload distribution of jobs arriving in state s with carry-in workload from $\tilde{h}$.

In a system with S states, the number of accumulation vectors of length N is $\binom{N+S-1}{N} = \frac{(N+S-1)!}{N!(S-1)!}$. If we take ordering into account, there are S^N accumulation sequences of length N. The number of accumulation vectors increases with the length N as $\mathcal{O}(N^{S-1})$ for a fixed number of states S.

We derive an upper bounding pending workload distribution $\mathcal{V}_{\tilde{h}}^{\uparrow} \geq \mathcal{V}_h$, recalling Definition 1.

We show that a *partial Gaussian distribution* (see Definition 2) is an upper bound to the conditional pending workload distribution. An illustration based on Example 1 is shown in Fig. 5. The dashed curve illustrates the exact convolution result of the workload of the gray accumulation sequence from Fig. 2. The carry-in workload is the partial Gaussian distribution of $\mathcal{N}^{tail}(2 - n \cdot Q, 1, 0)$, that is the normalized part of the computation time distribution in the second state that remains after the budget of $n \cdot Q = 2$ has been exhausted. The carry-in distribution is convolved with the computation time distribution of the first state, $\mathcal{N}(1, 0.25)$, resulting in the dashed curve. When we replace it with the partial Gaussian distribution shown in Fig. 5 as the black curve and line, the probabilities of lower workloads (the light gray area) are moved to higher workloads (the dark gray area), providing an upper bound.

Theorem 1 *$\mathcal{N}^{tail}\big(\mu(\tilde{h}), \sigma^2(\tilde{h}), \alpha(\tilde{h}, s)\big)$ upper bounds the conditional pending workload distribution $\mathcal{V}_{\tilde{h}}$ associated with each state s and accumulation vector $\tilde{h}$.*

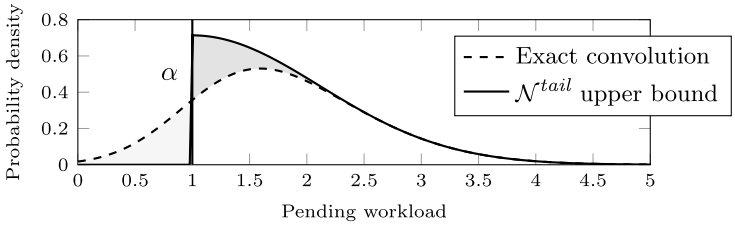

Fig. 5 Illustration of a convolution result with an upper bounding partial Gaussian distribution

The proof is by induction. We state Lemma 2 for the base case and further Lemma 3 combined with Lemma 4 for the inductive step.

Lemma 2 *The partial Gaussian distribution* $\mathcal{N}^{tail}\left(\mu_s, \sigma_s^2, 0\right) \geq \mathcal{V}_{\tilde{h}}$ *in state s at a job arrival immediately after a point of workload depletion, with* $\mathcal{V}_{\tilde{h}}$ *being the conditional pending workload distribution.*

Proof At the first job arrival after a point of workload depletion, the conditional pending workload distribution $\mathcal{V}_h$ is the execution time distribution of the entered state s. $\mathcal{N}^{tail}\left(\mu_s, \sigma_s^2, 0\right)$ excludes the negative workload values from $\mathcal{N}\left(\mu_s, \sigma_s^2\right)$. Normalization increases the probability of positive values. The probability density is moved from lower workload values to higher, providing an upper bound. □

In the following, we consider the case with non-zero carry-over workload when a job arrives in state s transitioning from state s_p. In s_p the accumulation vector is $\tilde{h}$, and $\mathcal{N}^{tail}(\mu(\tilde{h}), \sigma^2(\tilde{h}), \alpha(\tilde{h}, s_p))$ upper bounds the workload distribution. We show that the partial Gaussian distribution $\mathcal{N}^{tail}(\mu(\tilde{h}_{+s}), \sigma^2(\tilde{h}_{+s}), \alpha(\tilde{h}_{+s}, s))$ is an upper bound on the conditional pending workload distribution. In Eqs. (8) and (9) below we define $\mu(\tilde{h}_{+s})$ and $\sigma^2(\tilde{h}_{+s})$. To simplify the starting value $\alpha(\tilde{h}_{+s}, s)$ of the upper bound on the pending workload distribution defined in Eq. (12), we define Eqs. (10) and (11). $\mathrm{sf}^{-1}(q, \mu, \sigma^2)$ in Eq. (12) denotes the inverse survival function at quantile q of a Gaussian distribution with mean μ, and variance σ^2. The work value at which the survival function takes the value q. Equation (11) defines $K(\tilde{h}, s_p)$, the normalization factor needed for the conditional probability calculation. A convolution Definition 3 of the execution time distribution C_s and an upper bound of the carry-over workload gives a bound on the pending workload distribution. The part extending past the task period of the upper bounding workload distribution in s_p with $\tilde{h}$ constitutes an upper bound of the carry-over workload. $K(\tilde{h}, s_p)^{-1}$ is the integral of this part used for normalization.

$$\mu(\tilde{h}_{+s}) = \mu_s + \sum_{i=1}^{S} \tilde{h}[i] \cdot (\mu_i - n \cdot Q) \tag{8}$$

$$\sigma^2(\tilde{h}_{+s}) = \sigma_s^2 + \sum_{i=1}^{S} \tilde{h}[i] \cdot \sigma_i^2 \tag{9}$$

$$\alpha_\Delta(\tilde{h}, s_p) = \max(0, \alpha(\tilde{h}, s_p) - n \cdot Q) \tag{10}$$

$$K(\tilde{h}, s_p) = \left[\Phi\left(\frac{\mu(\tilde{h}) - n \cdot Q - \alpha_\Delta(\tilde{h}, s_p)}{\sigma(\tilde{h})} \right) \right]^{-1} \tag{11}$$

$$\alpha(\tilde{h}_{+s}, s) = \begin{cases} 0 & \tilde{h} = \mathbf{0} \\ \mathrm{sf}^{-1}\left(\frac{1}{K(\tilde{h}, s_p)}, \mu(\tilde{h}_{+s}), \sigma^2(\tilde{h}_{+s}) \right) & \tilde{h} \neq \mathbf{0} \end{cases}. \tag{12}$$

Lemma 3 *When a job arrives in state s with non-zero carry-over workload from state s_p with accumulation vector $\tilde{h}$, and the previous task period upper bound on the workload distribution $\mathcal{V}^{\uparrow}$ 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 normalized workload tail beyond the task period time is the strictly positive carry-over workload distribution. We formally express this as $\mathcal{N}^{tail}\big(\mu(\tilde{h}) - n \cdot Q \;\; \sigma^2(\tilde{h}) \;\; \max(0, \alpha(\tilde{h}, s_p) - n \cdot Q)\big)$.

$\mathcal{N}(\mu_s, \sigma_s^2)$ describes the execution time distribution in state s. By convolving Definition 3 the probability density functions of the execution time and the upper bound on the positive carry-over workload, we derive an upper bound on the conditional workload distribution $\mathcal{V}^{\uparrow}_{\tilde{h}_{+s}}$ in state s with accumulation vector $\tilde{h}_{+s}$. This holds because execution times are independent random variables, and the dependence of the Markov model is restricted to the transition probabilities.

We introduce $\mu_R(z)$, σ_R^2, $\mu_{\Sigma\Delta}$ and σ_Σ^2 below to simplify the notation in the convolution expansion:

$$\mu_R(z) = \frac{(z - \mu_s) \cdot \sigma^2(\tilde{h}) + (\mu(\tilde{h}) - n \cdot Q) \cdot \sigma_s^2}{\sigma_s^2 + \sigma^2(\tilde{h})} \tag{13}$$

$$\sigma_R^2 = \frac{\sigma_s^2 \cdot \sigma^2(\tilde{h})}{\sigma_s^2 + \sigma^2(\tilde{h})} \tag{14}$$

$$\mu_{\Sigma\Delta} = \mu_s + \mu(\tilde{h}) - n \cdot Q \tag{15}$$

$$\sigma_{\Sigma}^2 = \sigma_s^2 + \sigma^2(\tilde{h}). \tag{16}$$

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

$$\begin{aligned}
&\int_{-\infty}^{\infty} f\left(z - x|\mu_s, \sigma_s^2\right) \cdot f^{tail}\left(x|\mu(\tilde{h}) - n \cdot Q, \sigma^2(\tilde{h}), \alpha_{\Delta}\right) \mathrm{d}x \\
&= K(\tilde{h}, s_p) \int_{\alpha_{\Delta}}^{\infty} f(z - x|\mu_s, \sigma_s^2) \cdot f(x|\mu(\tilde{h}) - nQ, \sigma^2(\tilde{h})) \mathrm{d}x \\
&= K(\tilde{h}, s_p) \cdot f\left(z|\mu_{\Sigma\Delta}, \sigma_{\Sigma}^2\right) \cdot \int_{\alpha_{\Delta}}^{\infty} f\left(x|\mu_R(z), \sigma_R^2\right) \mathrm{d}x,
\end{aligned} \tag{17}$$

where we isolate the part of the expression independent of x in the last step. The integral in the last row of Eq. (17) is the survival function or 1-CDF at α_{Δ} of $\mathcal{N}(\mu_R(z), \sigma_R^2)$. The survival function is monotonically increasing with respect to z and goes to 0 as z goes to $-\infty$, and to 1 as z goes to ∞. This implies that there is a point $\alpha(\tilde{h}_{+s}, s)$ 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, $\mathcal{N}^{tail}(\mu_{\Sigma\Delta}, \sigma_{\Sigma}^2, \alpha(\tilde{h}_{+s}, s))$ from $\alpha(\tilde{h}_{+s}, s)$. This is illustrated in Fig. 5. Normalizing the partial Gaussian distribution with $K(\tilde{h}, s_p)$ as in Eq. (18) means that we derive the lowest possible $\alpha(\tilde{h}_{+s}, s)$ that upper bounds the full convolution. As the integral in the last row of Eq. (17) goes to 1 as z goes to infinity, the tail of the upper bound approaches the tail of the full convolution asymptotically.

$$K(\tilde{h}, s_p) \cdot \int_{\alpha(\tilde{h}_{+s}, s)}^{\infty} f\left(x|\mu_{\Sigma\Delta}, \sigma_{\Sigma}^2\right) \mathrm{d}x = 1. \tag{18}$$

The convolution result integrates to one, and so does the partial Gaussian distribution from Definition 2. The two regions described and illustrated in Fig. 5 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:

$$\mu(\tilde{h}_{+s}) = \mu_{\Sigma\Delta} \tag{19}$$

$$\sigma^2(\tilde{h}_{+s}) = \sigma_{\Sigma}^2 \tag{20}$$

$$\alpha(\tilde{h}_{+s}, s) = \mathrm{sf}^{-1}\left(\frac{1}{K(\tilde{h}, s_p)}, \mu_{\Sigma\Delta}, \sigma_{\Sigma}^2\right). \tag{21}$$

This concludes our proof. □

The values of α and K depend on the order in the accumulation sequence, as Eq. (10) depends on the previous state. Returning to Example 1, consider a job

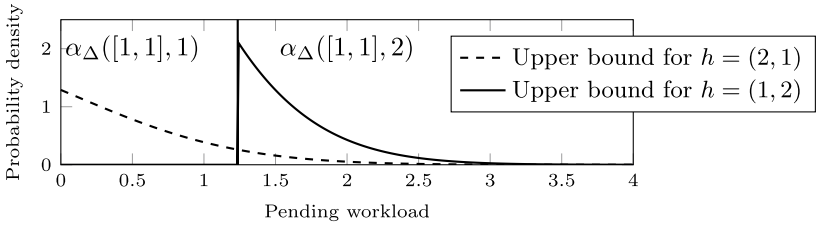

Fig. 6 Illustration of upper bounding partial Gaussian distributions for the carry-in workload of two accumulation sequences with the same vector

arriving in the third task period after an idle point. Two accumulation sequences determine the carry-in workload of visiting both state 1 and state 2 since the idle point; those are $h = (2, 1)$ and $h(1, 2)$. The first is the gray sequence in Fig. 2, and the upper bounding workload distribution is shown in Fig. 5. The tail extending past the period's budget is the carry-in to the next period, illustrated as the dashed curve in Fig. 6. Since $\alpha = 1 < n \cdot Q = 2$, $\alpha_\Delta = 0$ for the sequence (2, 1). For the order (1, 2) however, the resulting $\alpha \approx 3.236 > n \cdot Q = 2$. The upper bounding carry-in work will have $\alpha_\Delta \approx 1.236$ and is the solid curve illustrated in Fig. 6.

We state this formally in Lemma 4. We show that shifting the starting point α to a higher value while keeping the mean and variance unchanged gives an upper bounding distribution. This is illustrated in Fig. 7.

Lemma 4 *The partial Gaussian distribution* $\mathcal{N}^{tail}(\mu, \sigma^2, \alpha_1) \geq \mathcal{N}^{tail}(\mu, \sigma^2, \alpha_2)$ *if* $\alpha_1 \geq \alpha_2$.

Proof The CDF is 0, $x < \alpha_2$ for both $\mathcal{N}^{tail}(\mu, \sigma^2, \alpha_1)$ and $\mathcal{N}^{tail}(\mu, \sigma^2, \alpha_2)$. The CDF of $\mathcal{N}^{tail}(\mu, \sigma^2, \alpha_2) > 0$ for $\alpha_2 \leq x \leq \alpha_1$, but the CDF of $\mathcal{N}^{tail}(\mu, \sigma^2, \alpha_1) = 0$ in this range. For $x > \alpha_1$, we have from Definition 2 that the PDFs of the two distributions only differ in the scaling factor. This means that the CDF of $\mathcal{N}^{tail}(\mu, \sigma^2, \alpha_1)$ is the CDF of $\mathcal{N}^{tail}(\mu, \sigma^2, \alpha_2)$ past α_1 shifted to start at 0 and scaled to go to 1 at infinity. Therefore the CDF of $\mathcal{N}^{tail}(\mu, \sigma^2, \alpha_1)$ is always below the CDF of $\mathcal{N}^{tail}(\mu, \sigma^2, \alpha_2)$ □

We remove the dependency on the state order by taking the maximum $\alpha(\tilde{h}, s_p), s_p \in \tilde{h}$ to determine $\alpha_\Delta(\tilde{h})$. This is illustrated by selecting the black curve in Fig. 6 as carry-in from $\tilde{h} = [1, 1]$, and formalized as:

$$\alpha_\Delta(\tilde{h}) = \max\left(0, \max_{\forall s_p} \alpha(\tilde{h}, s_p) - n \cdot Q\right). \tag{22}$$

We use this instead of Eq. (10) in Eqs. (11) and (12). Let us proceed to the proof of Theorem 1, restated here for convenience:

Theorem 1 $\mathcal{N}^{tail}\big(\mu(\tilde{h}), \sigma^2(\tilde{h}), \alpha(\tilde{h}, s)\big)$ *upper bounds the conditional pending workload distribution* $\mathcal{V}_{\tilde{h}}$ *associated with each state* s *and accumulation vector* $\tilde{h}$.

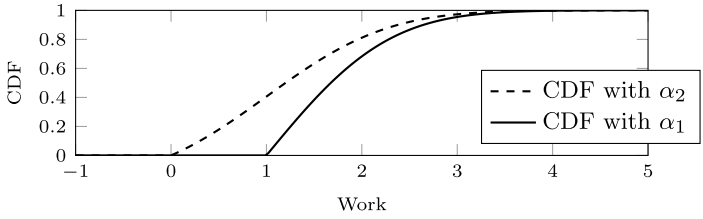

Fig. 7 CDFs of two partial Gaussian distributions as in Lemma 4. In this figure $\mu = 1$, $\sigma^2 = 1$, $\alpha_1 = 1$ and $\alpha_2 = 0$

Proof We prove this by induction.

Base case: For the first job arrival after workload depletion, this follows by Theorem 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 a job that arrives with a carry-in workload from a previous period.

Inductive step: This follows from Lemma 3 and by taking the maximum α in Eq. (22) due to Lemma 4. □

Analogously, a Gaussian distribution is a lower bound of the pending workload distribution $\mathcal{V}_{\tilde{h}}^{\downarrow} \leq \mathcal{V}_h$. This is illustrated in Fig. 8 for the accumulation sequence example drawn in gray in Fig. 2. From Eq. (17), we see that $K(\tilde{h}, s_p) > 1$, implying a heavier tail on the convolution result compared to the Gaussian distribution. The area under the Gaussian PDF curve with mean $\mu_{\Sigma\Delta}$ and variance σ_{Σ}^2 is one, and so is the area under 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.

4.5 Bounds on the joint probability of a job arriving in a state with an accumulation vector

A job arriving in state s N task periods after the last workload depletion can result in one or more accumulation vectors, $\tilde{h}$, of length N. We refer to this set of accumulation vectors as being in state s at task period N. Each accumulation vector in a state is associated with the joint probability of randomly selecting a job that arrives in state s and results in the accumulation vector $\tilde{h}$. $p_{in}^{\downarrow}(s, \tilde{h})$ denotes a lower bound on this joint probability and $p_{in}^{\uparrow}(s, \tilde{h})$ an upper bound. Each accumulation vector in a state is also associated with a probability of the workload contributing to carry-over into the next period. $p_{co}^{\downarrow}(s, \tilde{h})$ denotes a lower bound on this probability and $p_{co}^{\uparrow}(s, \tilde{h})$ an upper bound.

For jobs arriving at a point of workload depletion with no carry-in workload, each state is associated with a single accumulation vector containing zeros except for the

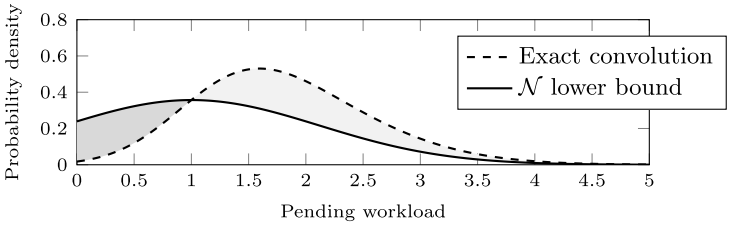

Fig. 8 An illustration of a convolution result and the Gaussian distribution that forms a lower bound

current state, which is set to 1. The probability of a job arriving in a certain state s at a point of workload depletion depends on

- the stationary probabilities $\xi(s_p)$ of all states s_p,
- the workload depletion probabilities $p_{wd}(s_p)$ of all states s_p,
- the state transition probabilities $\xi_{s_p,s}$ from all states s_p into s.

The stationary probabilities and the transition matrix are known from the execution time model described in Sect. 4.1. In Sect. 4.6, we will describe how to derive the workload depletion probabilities of all states. Let us assume that we have lower and upper bounds on the workload depletion probabilities, $p^{\downarrow}_{wd}(s)$ and $p^{\uparrow}_{wd}(s)$. Then, lower and upper bounds on the probability of randomly selecting a job arriving in each state s at a point of workload depletion are given as:

$$p^{\downarrow}_{in}(s,\tilde{h}) = \sum_{s_p=1}^{S} \xi(s_p) \cdot p^{\downarrow}_{wd}(s_p) \cdot m_{s_p,s} \tag{23}$$

$$p^{\uparrow}_{in}(s,\tilde{h}) = \sum_{s_p=1}^{S} \xi(s_p) \cdot p^{\uparrow}_{wd}(s_p) \cdot m_{s_p,s}. \tag{24}$$

There is only one accumulation vector in each state for jobs arriving at an idle point, and there is no dependency on $\tilde{h}$. We introduce it in the expression to have the common notation $p_{in}(s,\tilde{h})$ for all accumulation periods.

Relating this to Example 1, the lower bound on the probability of a job arriving in state 2 after an idle point is $p^{\downarrow}_{in}(2,[0,1]) = \xi(1) \cdot m_{1,2} \cdot p^{\downarrow}_{wd}(1) + \xi(2) \cdot m_{2,2} \cdot p^{\downarrow}_{wd}(2) = 0.875 \cdot 0.1 \cdot p^{\downarrow}_{wd}(1) + 0.125 \cdot 0.3 \cdot p^{\downarrow}_{wd}(2)$, a linear combination of the lower bounds on workload depletion probability for the states. The upper bound is the same linear combination of the upper bounds on workload depletion.

We further consider jobs arriving with a carry-in workload. Step by step, we add jobs that arrive one more task period after the last idle point, resulting in accumulation vectors containing one more state. We copy each accumulation vector from the states in the previous task period for these accumulation periods and increment the

current state element by 1. Such an accumulation vector copied from $\tilde{h}$ with a job that arrives in state s is denoted $\tilde{h}_{+s}$. When $\tilde{h}$ exists in different states in the previous accumulation period, they all lead to $\tilde{h}_{+s}$ in s. The joint probability of randomly selecting a job arriving in s and resulting in $\tilde{h}_{+s}$ depends on the probabilities $p_{co}(s_p, \tilde{h})$ of a randomly selected job arriving with unfinished workload from $\tilde{h}$ in each state s_p, and transition probabilities $m_{s_p,s}$.

The probability that a job arrives with a carry-in workload from $s_p, \tilde{h}$ is the probability of being in s_p with this $\tilde{h}$ multiplied by the probability that the conditional pending workload of $s_p, \tilde{h}$ exceeds the available processor time in a task period. Let random variables $X \sim \mathcal{V}^{\downarrow}_{\tilde{h}}$ and $Y \sim \mathcal{V}^{\uparrow}_{\tilde{h}}$. Then we have lower $p^{\downarrow}_{co}(s, \tilde{h})$ and upper $p^{\uparrow}_{co}(s, \tilde{h})$ bounds on the probability of a job arriving with the carry-in workload from $\tilde{h}$ and where the previous task period state was s as: $p^{\downarrow}_{co}(s, \tilde{h})$ and $p^{\uparrow}_{co}(s, \tilde{h})$, further calculated as:

$$p^{\downarrow}_{co}(s, \tilde{h}) = p^{\downarrow}_{in}(s, \tilde{h}) \cdot \mathbb{P}(X > n \cdot Q) \tag{25}$$

$$p^{\uparrow}_{co}(s, \tilde{h}) = p^{\uparrow}_{in}(s, \tilde{h}) \cdot \mathbb{P}(Y > n \cdot Q) \tag{26}$$

with $\mathcal{V}^{\downarrow}_{\tilde{h}}$ given as $\mathcal{N}(\mu(\tilde{h}), \sigma^2(\tilde{h}))$, and $\mathcal{V}^{\uparrow}_{\tilde{h}}$ as $\mathcal{N}^{tail}(\mu(\tilde{h}), \sigma^2(\tilde{h}), \alpha(\tilde{h}))$.

In Example 1 we find the lower bound of the probability of workload carry-over from state 2 and $\tilde{h} = [0, 1]$ as $p^{\downarrow}_{co}(2, [0, 1]) = p^{\downarrow}_{in}(2, [0, 1]) \cdot \mathbb{P}(\mathcal{N}(2, 1) > 2) = 0.5 \cdot p^{\downarrow}_{in}(2, [0, 1])$. The upper bound is $p^{\uparrow}_{co}(2, [0, 1]) = p^{\uparrow}_{in}(2, [0, 1]) \cdot \mathbb{P}(\mathcal{N}^{tail}(2, 1, 0) > 2) \approx 0.51 \cdot p^{\uparrow}_{in}(2, [0, 1])$.

The lower $p^{\downarrow}_{in}(s, \tilde{h}_{+s})$ and upper $p^{\uparrow}_{in}(s, \tilde{h}_{+s})$ bounds on the joint probability a job arriving in s resulting in $\tilde{h}_{+s}$ are:

$$p^{\downarrow}_{in}(s, \tilde{h}_{+s}) = \sum_{s_p=1}^{S} p^{\downarrow}_{co}(s_p, \tilde{h}) \cdot m_{s_p,s} \tag{27}$$

$$p^{\uparrow}_{in}(s, \tilde{h}_{+s}) = \sum_{s_p=1}^{S} p^{\uparrow}_{co}(s_p, \tilde{h}) \cdot m_{s_p,s}. \tag{28}$$

Returning to Example 1 and the gray accumulation sequence in Fig. 2, we have the probability of a job arriving in state 1 with carry-in from one task period in state 2. The lower bound on this probability is $p^{\downarrow}_{in}(1, [1, 1]) = p^{\downarrow}_{co}(2, [0, 1]) \cdot m_{2,1} = 0.7 \cdot p^{\downarrow}_{co}(2, [0, 1])$. In this case, the sum has only one term since only $\tilde{h} = [0, 1]$ in the first period can lead to $\tilde{h} = [1, 1]$ and $s = 1$ in the second period. The upper bound is derived in the same manner as $p^{\uparrow}_{in}(1, [1, 1]) = 0.7 \cdot p^{\uparrow}_{co}(2, [0, 1])$. The derivations from Eqs. (27), (25) and (23) can be combined, giving $p^{\downarrow}_{in}(1, [1, 1]) \approx 0.0306 \cdot p^{\downarrow}_{wd}(1) + 0.0131 \cdot p^{\downarrow}_{wd}(2)$. Combining Eqs. (28), (26) and (24) gives $p^{\uparrow}_{in}(2, [1, 1]) \approx 0.0313 \cdot p^{\uparrow}_{wd}(1) + 0.0134 \cdot p^{\uparrow}_{wd}(2)$.

4.6 Bounds on the probability of workload depletion

The probability of having no work remaining at the end of the task period p_{wd} for each state are used in Eqs. (23) and (24) to derive p_{in} bounds for jobs arriving at idle points. These state-wise workload depletion probabilities p_{wd} are propagated to all p_{in} in the workload accumulation process via Eqs. (25)–(28).

We do not know the true value of the probability of workload depletion p^*_{wd}. This section outlines how to derive bounds for p_{wd} by observing bounds of state-wise sums on p_{in}.

Let $\tilde{h} \in (s, i)$ denote the set of accumulation vectors in state s at task period i from the last idle point. We now define $p^{\downarrow\Sigma}_{in}(s, N, p_{wd})$, the sum of the lower bounds on p_{in} associated with all accounted accumulation vectors in s up until task period N from the last idle point. In other words, this is a lower bound on the joint probability of a randomly selected job arriving in s and at most N from the last idle point.

$$p^{\downarrow\Sigma}_{in}(s, N, p_{wd}) = \sum_{i=1}^{N} \sum_{\tilde{h}\in(s,i)} p^{\downarrow}_{in}(s, \tilde{h}) \tag{29}$$

Observation 4 Assume the exact p^*_{wd} is known and used as $p^{\downarrow}_{wd}$ in Eq. (23). Then $p^{\downarrow\Sigma}_{in}(s, N, p_{wd}) \leq \xi(s), \forall s, \forall N$.

We denote the error in $p^{\downarrow\Sigma}_{in}$ resulting from using the Gaussian $\mathcal{V}^{\downarrow}$ lower workload distribution bounds instead of the true workload distributions as $e(p^{\downarrow\Sigma}_{in})$.

We introduce $\beta(s)_N$ as the joint probability of a job arriving in s more than N task periods after the last idle point.

$$\beta(s)_N = \sum_{i=N+1}^{\infty} \sum_{\tilde{h}\in(s,i)} p_{in}(s, \tilde{h}) \tag{30}$$

Observation 4 is illustrated in Fig. 9, where the valid region of $p^{\downarrow\Sigma}_{in}$ is displayed assuming p^*_{wd} is input in Eq. (23).

We define an upper bound on the joint probability of a randomly selected job arriving in s and at most N from the last idle point as $p^{\uparrow\Sigma}_{in}(s, N, p_{wd})$ in Eq. (31).

$$p^{\uparrow\Sigma}_{in}(s, N, p_{wd}) = \sum_{i=1}^{N} \sum_{\tilde{h}\in(s,i)} p^{\uparrow}_{in}(s, \tilde{h}) \tag{31}$$

Observation 5 Assume the true probability of workload depletion p^*_{wd} is known. Using this value in Eq. (24), we have $p^{\uparrow\Sigma}_{in}(s, N, p_{wd}) \geq \xi(s) - \beta(N)$.

Let $e(p^{\uparrow\Sigma}_{in})$ denote the error introduced by replacing the true workload distribution with the upper bounding partial Gaussian distribution. The valid region of $p^{\uparrow\Sigma}_{in}$ given from Observation 5 is displayed in Fig. 10.

Observations 4 and 5 imply that the true probability of workload depletion must lead to $p^{\downarrow\Sigma}_{in}$ in the region marked in Fig. 9 and $p^{\uparrow\Sigma}_{in}$ in the region marked in Fig. 10.

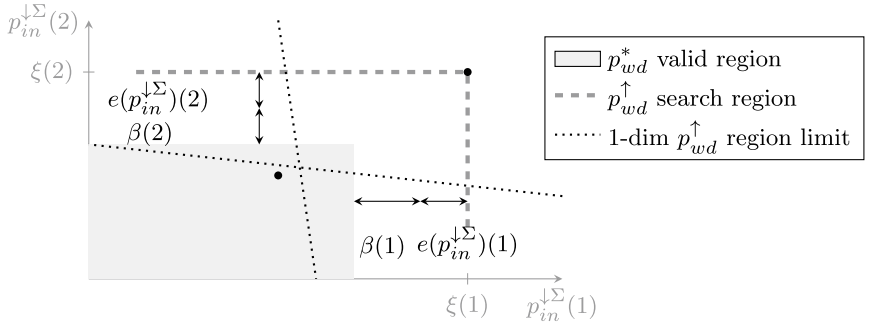

Fig. 9 An illustration of the possible valid region of $p^{\downarrow\Sigma}_{in}$ for two states, if the true probabilities of workload depletion would be used as $p^{\downarrow}_{wd}$ in Eq. (23)

The state-wise maxima of p_{wd} leading to any point along the lines illustrated as the upper and right lines in Fig. 9 upper bounds p^{*}_{wd}.

Theorem 5 *An upper bound $p^{\uparrow}_{wd}$ on the probability of workload depletion is derived by taking the state-wise maxima of p_{wd} leading to $p^{\downarrow\Sigma}_{in}(s) \leq \xi(s), \forall s$, and where there is inequality in at most one s.*

Proof From Eqs. (23), (25) and (27) it follows that $p^{\downarrow}_{in}(s, \tilde{h})$ is a linear combination of $p_{wd}(s)$. As is clear from Eq. (29), $p^{\downarrow\Sigma}_{in}(s)$ is also a linear combination of $p_{wd}(s)$, and it holds for some positive $A_{i,s}$ that:

$$p^{\downarrow\Sigma}_{in}(s, p_{wd}) = \sum_{i=1}^{S} A_{i,s} \cdot p_{wd}(i) \tag{32}$$

Starting from the true workload depletion probability p^{*}_{wd} we increase an arbitrary state dimension j of $p_{wd}(j)$ by an amount $\delta_{s,j}$ until $p^{\downarrow\Sigma}_{in}(s, p_{wd})$ reaches a hyperplane defined by ξ:

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

At the first hyperplane we encounter $\min(\delta_{s,j})\forall s$, which gives $p^{\downarrow\Sigma}_{in}(i) \leq \xi(i), \forall i \neq s$.

The true p^{*}_{wd} results in $p^{\downarrow\Sigma}_{in}(s) \leq \xi(s)$. Therefore, all p_{wd} resulting in the point with equality for all s upper bounds p^{*}_{wd} in at least one state dimension due to the linear combination. Assume that p_{wd} resulting in this point does not upper bound p^{*}_{wd} for state dimension i, $p_{wd}(i) < p^{*}_{wd}(i)$. In this case, an upper bound of $p^{*}_{wd}(i)$ results in a point on one of the hyperplanes. The hyperplane separating the region resulting from upper bounds in this dimension from the region resulting from underestimates in this dimension crosses at least one of the hyperplanes described by $p^{\downarrow\Sigma}_{in}(s) \leq \xi(s), \forall s$, with inequality in at most one s. Illustrating in Fig. 9 the result from the upper bound

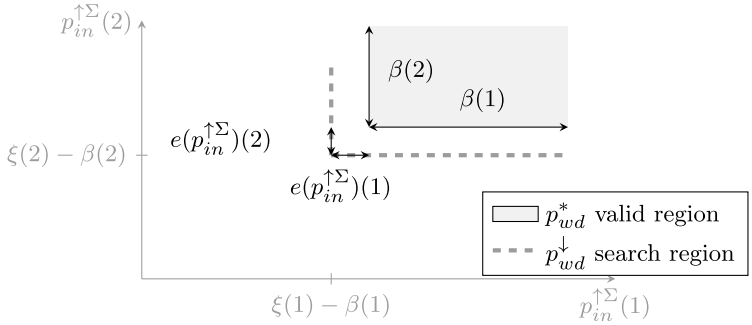

Fig. 10 An illustration of the possible valid region of $p_{in}^{\uparrow\Sigma}$ for two states, if the true probabilities of workload depletion would be used as $p_{wd}^{\uparrow}$ in Eq. (24)

in this dimension as the black dot, two possible hyperplanes that separate the regions are displayed with dotted lines. This concludes our proof. □

Analogously, we derive a lower bound on the workload depletion probability p_{wd}. The state-wise minima of p_{wd} resulting in $p_{in}^{\uparrow\Sigma}$ on the lower and left lines illustrated in Fig. 10 lower bound p_{wd}^{*}.

The endpoints are adjusted if the p_{wd} for a state is lower than 0 or higher than 1. As $p_{in}^{\downarrow\Sigma}(s)$ and $p_{in}^{\uparrow\Sigma}(s)$ are linear combinations of $p_{wd}(s)$ we only need to consider the endpoints.

Relating to Example 1, we have seen that the probability of a job arrival in state 2 after an idle point at least $p_{in}^{\downarrow}(2,[0,1]) = 0.0875 \cdot p_{wd}^{\downarrow}(1) + 0.0375 \cdot p_{wd}^{\downarrow}(2)$. The same derivation for a job arrival in state 1 after an idle point gives $p_{in}^{\downarrow}(1,[1,0]) = 0.7875 \cdot p_{wd}^{\downarrow}(1) + 0.0875 \cdot p_{wd}^{\downarrow}(2)$. From simulation we have the probability of jobs arriving with $\tilde{h}$ longer than 1 as $\beta(1)_1 \approx 0.093$ for state 1 and $\beta(2)_1 \approx 0.026$ for state 2. We solve the linear equation systems below for $(i,j) = (0,0), (1,0)$ and $(0,1)$ to get candidates for $p_{wd}^{\uparrow}$.

$$0.7875 \cdot p_{wd}(1) + 0.0875 \cdot p_{wd}(2) = 0.875 - i \cdot 0.093 \tag{33}$$

$$0.0875 \cdot p_{wd}(1) + 0.0375 \cdot p_{wd}(2) = 0.125 - j \cdot 0.026 \tag{34}$$

In this case, with only the jobs arriving at idle points, the equation system for $(i,j) = (0,0)$, the upper right corner in Fig. 9, gives the solution $p_{wd}^{\uparrow}(1) = p_{wd}^{\uparrow}(2) = 1$ that is the highest possible bound. For the lower bound of p_{wd}, we have the same linear equation system in the special case when we only consider the accumulation vectors after an idle point. This is because Eqs. (23) and (24) only differ in the workload depletion probability bounds. We now solve the system for $(i,j) = (1,1), (1,0)$ and $(0,1)$. For $(i,j) = (1,1)$, the lower left corner in Fig. 10, we get the candidates $p_{wd}(1) \approx 0.94$ and $p_{wd}(2) \approx 0.44$. For $(i,j) = (1,0)$ we get $p_{wd}(1) \approx 0.84$ and $p_{wd}(2) \approx 1.4$. This point is invalid since $p_{wd}(2) > 1$. We find the $j, 0 < j < 1$

where $(i,j) = (1,j)$ gives $p_{wd}(2) = 1$, and at this point we have $p_{wd}(1) \approx 0.88$. For $(i,j) = (0,1)$ we get $p_{wd}(1) \approx 1.1$ and $p_{wd}(2) \approx 0.064$. This point is also invalid, and we search along the line $i, 0 < i < 1, j = 1$ for the point where $p_{wd}(1) = 1$. We have $p_{wd}(2) \approx 0.31$. We can now assign the state-wise minima for the lower bound: $p^{\downarrow}_{wd}(1) \approx 0.88$ and $p^{\downarrow}_{wd}(2) \approx 0.31$.

4.7 Bounds on the probability of longer workload accumulation

In Eq. (30), we defined $\beta(s)_N$ as the joint probability of a job arriving in s with at least N elapsed since the last idle point. The values of $\beta(s)_N$ were used in obtaining the bounds of workload depletion for the different states. In this section we outline how to find bounds for $\beta(s)_N$, given safe bounds $\beta(s)_{N-1}$. As all p_{in} are non-negative, $\beta(s)_N$ decreases monotonically with N. For each period, $\beta(s)_N$ is at most $\beta(s)_{N-1}$ minus the lower bound on the probability of a job arriving in s N task periods after an idle point, i.e.

$$\beta(s)_N \leq \beta(s)_{N-1} - \sum_{\tilde{h} \in (s,N)} p^{\downarrow}_{in}(s,\tilde{h}) = \beta(s)^{\uparrow a}_N \tag{35}$$

Further, $\beta(s)_N$ is at most the stationary probability $\xi(s)$ minus the lower bound on the probability of a job arriving within N task periods after an idle point, i.e.

$$\beta(s)_N \leq \xi(s) - \sum_{i=1}^{N} \sum_{\tilde{h} \in (s,i)} p^{\downarrow}_{in}(s,\tilde{h}) = \beta(s)^{\uparrow b}_N \tag{36}$$

Safe bounds $\beta(s)_N$ are obtained by taking the minimum of right-hand sides of Inequalities (35), and (36).

$$\beta(s)^{\uparrow}_N = \min(\beta(s)^{\uparrow a}_N, \beta(s)^{\uparrow b}_N) \tag{37}$$

We return to Example 1 and consider the probability of jobs arriving in state 1 with accumulation vectors past 2 task periods, that is $\beta(1)_2$. We have $\beta(1)_1 \approx 0.093$ from simulation, the lower bound on the probability of a job arriving in state 1 with accumulation vector [1, 1] as $p^{\downarrow}_{in}(1,[1,1]) \approx 0.031 \cdot p^{\downarrow}_{wd}(1) + 0.013 \cdot p^{\downarrow}_{wd}(2)$ and with accumulation vector [2, 0] $p^{\downarrow}_{in}(1,[2,0]) \approx 0.016 \cdot p^{\downarrow}_{wd}(1) + 0.0018 \cdot p^{\downarrow}_{wd}(2)$. Entering these values in Eq. (35) we get $\beta(1)_2 \lessapprox 0.047$. Using Eq. (36) gives $\beta(1)_2 \lessapprox 0.11$, so we use $\beta(1)^{\uparrow}_2 \approx 0.047$ in the search for bounds on p_{wd} in the next accumulation period.

4.8 Upper bounding the deadline miss probability

Finally, we derive an upper bound on a randomly selected job's expected deadline miss probability as defined in Eq. (6). We derive an upper bound $p^{\uparrow}_{dm}(s,\tilde{h})$ on the deadline miss probability $p_{dm}(s,\tilde{h})$ of a job arriving in state s with the job arrival

resulting in the accumulation vector $\tilde{h}$. This bounds the deadline miss probability of all jobs, resulting in accumulation sequences h corresponding to $\tilde{h}$ where the sequence ends in s. The random variable $Y \sim \mathcal{V}_{\tilde{h}}^{\uparrow}$ upper bounds the pending work distribution of these jobs. $p_{dm}^{\uparrow}(s, \tilde{h})$ is the probability that this work exceeds the available computation time for the job until the deadline $k \cdot Q$, i.e.:

$$p_{dm}^{\uparrow}(s, \tilde{h}) = \mathbb{P}(Y > k \cdot Q) \tag{38}$$

The distribution $\mathcal{V}_{\tilde{h}}^{\uparrow}$ is the upper bounding distribution $\mathcal{N}^{tail}(\mu(\tilde{h}), \sigma^2(\tilde{h}), \alpha(\tilde{h}, s))$, as shown in Theorem 1.

The probability of randomly selecting a job arriving in state s with workload accumulation captured by $\tilde{h}$ is the joint probability $p_{in}(s, \tilde{h})$. The upper bound of this probability, $p_{in}^{\uparrow}(s, \tilde{h})$ was derived in Sect. 4.5. We derive a bound of the expected deadline miss probability conditioned on being in a state s by considering all job arrivals in s within N task periods from the last idle point, that is with $\tilde{h}$ of length up to N. The deadline miss probability of jobs arriving more than N task periods from the last idle point is upper bounded by 1. The probability of randomly selecting a job arriving more than N task periods from the last idle point is upper bounded by $\beta(s)_N^{\uparrow}$. The probability of randomly selecting a job arriving in s is the stationary probability $\xi(s)$. We upper bound the expected deadline miss probability in s by:

$$p_{dm}^{\uparrow}(s) = \frac{\beta(s)_N^{\uparrow}}{\xi(s)} + \frac{\sum_{i=1}^{N} \sum_{\tilde{h} \in (s,i)} p_{in}^{\uparrow}(s, \tilde{h}) p_{dm}^{\uparrow}(s, \tilde{h})}{\xi(s)}. \tag{39}$$

In our example, we derive for state 1 the first term $\frac{\beta(1)_2^{\uparrow}}{\xi(1)} \approx \frac{0.047}{0.875} \approx 0.054$ and the second term $\frac{\sum_{\tilde{h} \in ([1,0],[1,1],[2,0])} p_{in}^{\uparrow}(1,h) \cdot p_{dm}^{\uparrow}(1,h)}{\xi(1)} \approx \frac{0.875 \cdot 10^{-9} + 0.018 \cdot 3.4 \cdot 10^{-7} + 0.045 \cdot 0.0073}{0.875} \approx 3.7 \cdot 10^{-4}$.

Theorem 6 *The expected deadline miss probability* DMP *of a randomly selected job is upper-bounded by* $p_{dm}^{\uparrow}$, *i.e.*, DMP $\leq p_{dm}^{\uparrow}$, *where*

$$p_{dm}^{\uparrow} = \sum_{\forall s} \left(\beta(s)_N^{\uparrow} + \sum_{i=1}^{N} \sum_{\tilde{h} \in (s,i)} p_{in}^{\uparrow}(s, \tilde{h}) p_{dm}^{\uparrow}(s, \tilde{h}) \right). \tag{40}$$

Proof The deadline miss probability Eq. (38) is an upper bound on the deadline miss probability of a job arriving in s and resulting in $\tilde{h}$, because $\mathcal{N}^{tail}(\mu(\tilde{h}), \sigma^2(\tilde{h}), \alpha(\tilde{h}, s))$ upper bound on the workload distribution as shown in per Theorem 1.

The expected deadline miss probability of a randomly selected job arriving in s is upper bounded by $p_{dm}^{\uparrow}(s)$ as in Eq. (39). For jobs arriving in s within N since the last idle point, Eq. (38) upper bounds $p_{dm}(s, \tilde{h})$, and $p_{in}(s, \tilde{h})$ is an upper bound on the probability of randomly selecting a job arriving in state s and resulting in $\tilde{h}$. The probability of randomly selecting a job arriving in s more than N from the last idle

point is upper bounded by $\beta(s)_N^{\uparrow}$, and 1 upper bounds $p_{dm}(s, \tilde{h})$ for these jobs. We divide by $\xi(s)$ as per the definition of conditional probability.

We apply the law of total probability on Eq. (39) over all the states s to obtain Eq. (40). □

In our example we have $\beta(1)_2^{\uparrow} \approx 0.047$ and $\beta(2)_2^{\uparrow} \approx 0.011$, resulting in the first term of Eq. (40) as 0.058. In the second term we have for state 2 $\sum_{\tilde{h} \in ([0,1],[1,1],[0,2])} p_{in}^{\uparrow}(2,h) \cdot p_{dm}^{\uparrow}(2,h) \approx 0.125 \cdot 0.024 + 0.045 \cdot 0.16 + 0.019 \cdot 0.16$. Summing with the terms of state one, the resulting sum is approximately 0.0066. With only two accumulation periods accounted for, the largest part of the bound stems from the first term, where the deadline miss probability is set to 1 for longer accumulation vectors.

5 Iterative workload accumulation

As illustrated in Fig. 3, the steps described in Sect. 4 are applied iteratively, successively including jobs arriving with a longer time from the last idle point in the analysis. The process ends when one of the following conditions is met:

1. For each state both of the following hold:
   (a) The upper bound on the probability of workload depletion has stopped decreasing and started increasing.
   (b) The lower bound on the probability of workload depletion has stopped increasing and started decreasing.
2. The process has reached a maximum number of accumulated periods.

If the bounds on the workload depletion probability converge for each state, or if the region within the bounds starts to grow, the first condition is met. Instead of performing the convolution in each accumulation period, the upper and lower bounds on the workload distribution are used, introducing an error. The white space between the valid region and the lines to use in searching for bounds in Figs. 9 and 10 illustrate these errors. When these errors increase, the distance between the search region for our bounds and the region resulting from the true workload depletion probabilities grows. For the upper bounds, illustrated in Fig. 9, the distance between the search region and the valid region may still decrease if the increase due to this error is compensated by a decrease in β. In the case of the lower bounds, illustrated in Fig. 10, a larger error leads to a smaller value for the lower bound. The lower bounds are used in the calculations of $\beta^{\uparrow}$ in the next accumulation period, Eq. (37). Smaller lower bounds lead to a larger $\beta^{\uparrow}$. This further increasing the distance between the valid region and the bound search region, as $\beta^{\uparrow}$ is used to determine the search region. This may cause $p_{wd}^{\downarrow}$ and $\beta^{\uparrow}$ to diverge.

It may be the case that the workload depletion probability bounds diverge from the beginning. This may be caused by insufficient computational resources allocated to the task or too large errors introduced in the bound calculations. It may also be the case that the workload depletion probability bounds converge slowly or converge for one or more states while they diverge for others. In these cases, the process stops when the second condition is fulfilled.

We apply the iterative process to the following example:

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

Example 2

The stationary probability for state 1 is 0.875, and for state 2 it is 0.125. The transition matrix and stationary probabilities are identical to Example 1. In the CBS of this example, there are $n = 4$ server periods in one task period, and the task is guaranteed $Q = 8$ time units of computation time in each server period. The deadline is defined by $k = 8$ server periods.

State 1 in Example 2 could imply normal operation, and state 2 an exceptional mode. While in normal operation the task remains there with probability 0.9, but when the task is in the exceptional mode, there is a probability of 0.3 that it remains there. The initial values of probability of a randomly selected job arriving with carry-in workload from at least one task period, β_1 are obtained from simulation. Execution times are generated from the Markov Model and fed into a CBS simulator with the specified server reservation and period ratio. This results in $\beta_1 = (0.1278, 0.0442)$ for states 1 and 2, respectively. Figure 11 illustrated the evolution of β_N during the workload accumulation process compared to probabilities of jobs arriving in states 1 and 2 at least N from the last idle point resulting from simulation.

The bound regions for the probabilities of workload depletion of the two states along the accumulation process are shown in Fig. 12. Estimates of the probabilities of workload depletion obtained from simulation are also displayed. The workload accumulation continues until the maximum number of task periods, set to 20 for this example.

The bounds on the deadline miss probabilities for the two states along the accumulation process are shown in Fig. 13. The second terms of Eq. (39), the parts of the bounds resulting from the weighted sum of the accumulation vectors we have accounted for, are shown as dotted. In this example, the second terms approach the p_{dm} from simulation. The major part of the introduced pessimism originates in β, the first terms of Eq. (39). Estimates of the deadline miss probabilities obtained from simulation are also displayed in Fig. 13.

5.1 Time complexity of the iterative process

In Sect. 4.4, we have seen that the number of accumulation vectors with length N in a S-state model grows as $\mathcal{O}(N^{S-1})$. In the iterative procedure, all accumulation

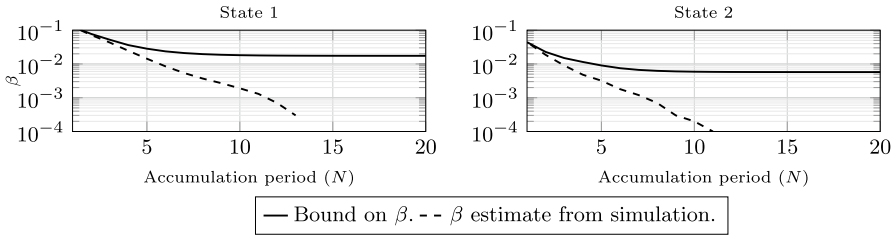

Fig. 11 Bounds on β for the two states as solid lines, along with probability estimates of longer accumulation histories obtained from simulation as dashed lines. (Log scale)

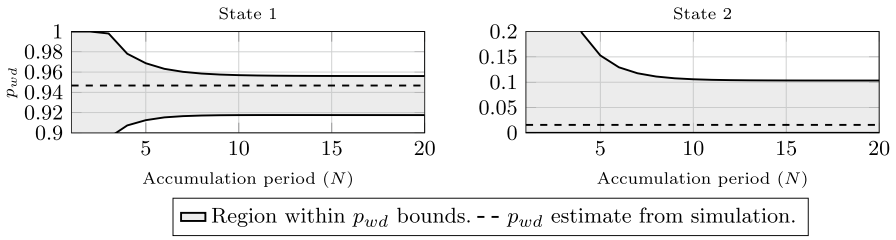

Fig. 12 The region between the upper and lower bounds on the per-state probability of workload depletion in the example, along with the estimates obtained from simulation as a dashed line

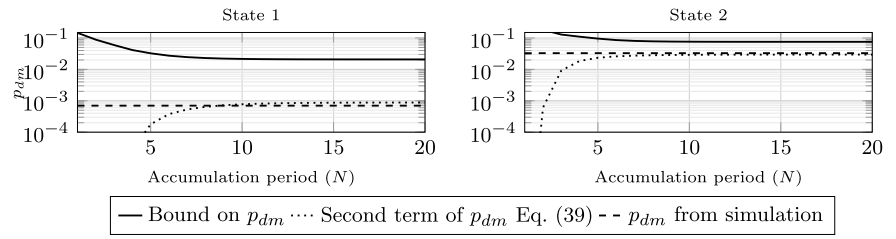

Fig. 13 The bounds on the deadline miss probabilities during the workload accumulation process of the example, along with results from simulation. (Log scale.)

vectors up until length N have been considered at iteration step N, so the time complexity of the entire iterative process is $\mathcal{O}(N^S)$. In the current implementation, all vectors up until length N are considered in the iteration step N, giving a time complexity of $\mathcal{O}(N^{S+1})$. By storing intermediate results from previous iterations, this can be reduced to $\mathcal{O}(N^S)$. The number of states is application dependent. In the robotic vision task of Frías et al. (2017), 4 discrete-emission states are identified, and in the control task in our evaluation 6 discrete-emission or 8

Gaussian-emission states are found. In the video decompression task evaluated in Friebe et al. (2020) almost 50 Gaussian-emission states are identified.

6 Reducing the number of states by merging

As the time complexity of the iterative process up until the accumulation period N with a S-state model is $\mathcal{O}(N^S)$, it is clear that if the number of states can be reduced, this would have a great effect on the bound computation time. This section outlines how to reduce the number of states by merging while ensuring a safe bound on the deadline miss probability.

6.1 Modified Markov chain execution times

In this section we define a modified execution time model, where an upper bound on the execution time distribution is defined by $\langle \mathbb{S}, M, \mathbb{C} \rangle$. As in the model defined in Sect. 4.1, $\mathbb{S} = \{1, 2, \dots, S\}$ is the set of S states, $S \in \mathbb{N}$, and M is the $S \times S$ state transition matrix. $\mathbb{C} = \{C_1, C_2, \dots, C_S\}$ is the set of upper bounding execution time distributions, or *emission distributions*, related to the respective state. These are modeled as partial Gaussian distributions with mean μ_s and variance σ_s^2 of the Gaussian distribution, and α_s as the starting point of the distribution, i.e. $C_s \sim \mathcal{N}^{tail}(\mu_s, \sigma_s^2, \alpha_s)$. Setting $\alpha_s = -\infty, \forall s$, gives the model as defined in Sect. 4.1.

6.2 Merging distributions

Definition 11 We define a *merged partial Gaussian distribution* $\mathcal{N}_m^{tail}(\mu_1, \mu_2, \sigma_1^2, \sigma_2^2, \alpha_1, \alpha_2)$, of two partial Gaussian distributions $\mathcal{N}^{tail}(\mu_1, \sigma_1^2, \alpha_1)$ and $\mathcal{N}^{tail}(\mu_2, \sigma_2^2, \alpha_2)$, as:

$$\begin{aligned}\mathcal{N}_m^{tail}(\mu_1, \mu_2, \sigma_1^2, \sigma_2^2, \alpha_1, \alpha_2) = \mathcal{N}^{tail}(&\max(\mu_1, \mu_2), \max(\sigma_1^2, \sigma_2^2), \max(\mu_1, \mu_2) \\ &+ \max(0, \alpha_1 - \mu_1, \alpha_2 - \mu_2))\end{aligned}$$

In the following, we show that the merged partial Gaussian distribution is greater than each of the distributions used in the construction, as outlined in Theorem 7. We show this step-by-step, upper bounding each of the two partial Gaussian distributions until both reach the merged distribution. We provide a lemma and illustration for each step below.

Theorem 7 *The merged partial Gaussian distribution defined by* $\mathcal{N}_m^{tail}(\mu_1, \mu_2, \sigma_1^2, \sigma_2^2, \alpha_1, \alpha_1)$ *is an upper bound of each of the two distributions* $\mathcal{N}^{tail}(\mu_1, \sigma_1^2, \alpha_1)$ *and* $\mathcal{N}^{tail}(\mu_2, \sigma_2^2, \alpha_2)$.

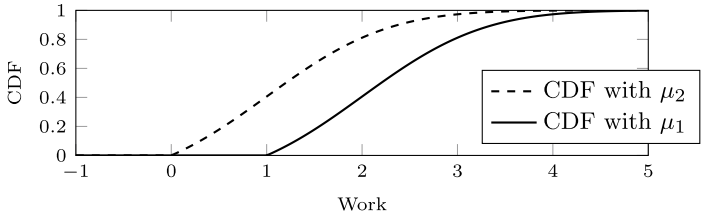

Fig. 14 CDFs of two partial Gaussian distributions as in Lemma 8. In this figure $\mu_1 = 2$, $\mu_2 = 1$, $\sigma^2 = 1$ and $\alpha_\Delta = -1$

In Lemma 8, we show that shifting the mean of a partial Gaussian distribution to a higher value while keeping the distance between the mean and the starting point α unchanged gives an upper bounding distribution. This is illustrated in Fig. 14.

Lemma 8 *The partial Gaussian distribution* $\mathcal{N}^{tail}(\mu_1, \sigma^2, \mu_1 + \alpha_\Delta) \geq \mathcal{N}^{tail}(\mu_2, \sigma^2, \mu_2 + \alpha_\Delta)$ *if* $\mu_1 \geq \mu_2$.

Proof From Definition 2, we know that the scaling factor of the partial Gaussian distribution depends only on α_Δ and σ^2 that are equal for the two distributions. From this we conclude that the CDF of $\mathcal{N}^{tail}(\mu_1, \sigma^2, \mu_1 + \alpha_\Delta)$ is the CDF of $\mathcal{N}^{tail}(\mu_2, \sigma^2, \mu_2 + \alpha_\Delta)$ translated $\mu_1 - \mu_2$ to the right. Therefore the CDF of $\mathcal{N}^{tail}(\mu_1, \sigma^2, \mu_1 + \alpha_\Delta)$ is always below that of $\mathcal{N}^{tail}(\mu_2, \sigma^2, \mu_2 + \alpha_\Delta)$. □

In Lemma 9, we show that increasing the variance to a higher value while keeping the mean and starting point unchanged gives an upper bounding distribution if the starting point α is at the mean or higher. This is illustrated in Fig. 15.

Lemma 9 *The partial Gaussian distribution* $\mathcal{N}^{tail}(\mu, \sigma_1^2, \alpha) \geq \mathcal{N}^{tail}(\mu, \sigma_2^2, \alpha)$ *if* $\sigma_1^2 \geq \sigma_2^2$ *and* $\alpha \geq \mu$.

Proof Let $\sigma_1^2 = k \cdot \sigma_2^2, k \geq 1$. Since the partial Gaussian functions are normalized to integrate to 1, the PDF of $\mathcal{N}^{tail}(\mu, \sigma_2^2, \alpha)$ at $x \geq \alpha$ can be written as $C_2 \cdot e^{-\frac{(x-\mu)^2}{2\sigma_2^2}}$, with C_2 as the normalization factor. Analogously we have the PDF of $\mathcal{N}^{tail}(\mu, \sigma_1^2, \alpha)$ as $C_1 \cdot e^{-\frac{(x-\mu)^2}{2k^2 \cdot \sigma_2^2}}$, with C_1 as the normalization factor. Let us evaluate the rate of decline in the PDFs between x and $x + \Delta x$, $\Delta x > 0$. Since $\alpha \geq \mu$ the PDF is declining. Dividing the PDF at x with the PDF at $x + \Delta x$ results in exponential functions with the coefficients $\frac{(\Delta x \cdot (\Delta x + 2(x-\mu))}{2\sigma_2^2}$ and $\frac{(\Delta x \cdot (\Delta x + 2(x-\mu))}{2k^2 \cdot \sigma_2^2}$ respectively. The PDF associated with $\sigma_1^2 = k \cdot \sigma_2^2$ has a lower rate of decrease than σ_2^2. This implies that the CDF associated with σ_2^2 has a more rapid growth from 0 and remains above the CDF associated with σ_1^2. □

With these lemmas in place, we can prove Theorem 7, restated here for convenience.

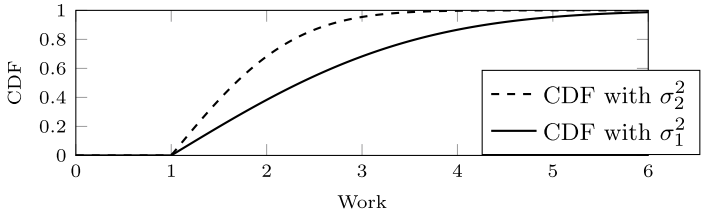

Fig. 15 CDFs of two partial Gaussian distributions as in Lemma 9. In this figure $\mu = 1$, $\sigma_1^2 = 4$, $\sigma_2^2 = 1$ and $\alpha = 1$

Theorem 7 *The merged partial Gaussian distribution defined by* $\mathcal{N}_m^{tail}(\mu_1, \mu_2, \sigma_1^2, \sigma_2^2, \alpha_1, \alpha_1)$ *is an upper bound of each of the two distributions* $\mathcal{N}^{tail}(\mu_1, \sigma_1^2, \alpha_1)$ *and* $\mathcal{N}^{tail}(\mu_2, \sigma_2^2, \alpha_2)$.

Proof In the first step, we apply Lemma 8 and upper bound the execution times of the two distributions as:

$$\mathcal{N}^{tail}(\mu_1, \sigma_1^2, \alpha_1) \leq \mathcal{N}^{tail}(max(\mu_1, \mu_2), \sigma_1^2, max(\mu_1, \mu_2) + \alpha_1 - \mu_1)$$

and:

$$\mathcal{N}^{tail}(\mu_2, \sigma_2^2, \alpha_2) \leq \mathcal{N}^{tail}(max(\mu_1, \mu_2), \sigma_2^2, max(\mu_1, \mu_2) + \alpha_2 - \mu_2)$$

In a second step we apply Lemma 4 and derive upper bounds on the distributions as:

$$\begin{aligned}&\mathcal{N}^{tail}(max(\mu_1, \mu_2), \sigma_1^2, max(\mu_1, \mu_2) + \alpha_1 - \mu_1)\\ &\quad\leq \mathcal{N}^{tail}(max(\mu_1, \mu_2), \sigma_1^2, max(\mu_1, \mu_2) + max(0, \alpha_1 - \mu_1, \alpha_2 - \mu_2))\end{aligned}$$

and:

$$\begin{aligned}&\mathcal{N}^{tail}(max(\mu_1, \mu_2), \sigma_2^2, max(\mu_1, \mu_2) + \alpha_2 - \mu_2)\\ &\quad\leq \mathcal{N}^{tail}(max(\mu_1, \mu_2), \sigma_2^2, max(\mu_1, \mu_2) + max(0, \alpha_1 - \mu_1, \alpha_2 - \mu_2)\end{aligned}$$

In a third step, we apply Lemma 9 to upper bound:

$$\begin{aligned}&\mathcal{N}^{tail}(max(\mu_1, \mu_2), \sigma_1^2, max(\mu_1, \mu_2) + max(0, \alpha_1 - \mu_1, \alpha_2 - \mu_2))\\ &\quad\leq \mathcal{N}^{tail}(max(\mu_1, \mu_2), max(\sigma_1^2, \sigma_2^2), max(\mu_1, \mu_2) + max(0, \alpha_1 - \mu_1, \alpha_2 - \mu_2))\end{aligned}$$

and:

$$\begin{aligned}&\mathcal{N}^{tail}(max(\mu_1, \mu_2), \sigma_2^2, max(\mu_1, \mu_2) + max(0, \alpha_1 - \mu_1, \alpha_2 - \mu_2))\\ &\quad\leq \mathcal{N}^{tail}(max(\mu_1, \mu_2), max(\sigma_1^2, \sigma_2^2), max(\mu_1, \mu_2) + max(0, \alpha_1 - \mu_1, \alpha_2 - \mu_2))\end{aligned}$$

This concludes our proof. □

6.3 Merging states in the Markov model

Here, we describe how to merge two states in the modified execution time model. Without loss of generality, we describe how to merge the last two states, $S-1$ and S, to reduce the number of states from S to $S-1$. States can be reordered to merge any two states, and the process can be repeated to merge any number of states.

Recall that the M 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. Let $m_{a,b}$ represent an element in the transition matrix prior to merging and $m^m_{a,b}$ an element in the transition matrix after merging. In the new $(S-1)\times(S-1)$, the element values are calculated according to Eq. (41). All $m^m_{a,b}, a < S-1, b < S-1$ remain the same as $m_{a,b}$ because these are the transition probabilities of states unaffected by the merge. For $m^m_{a,S-1}, a < S-1$, that is the probability of moving from an unchanged state into the merged state, the transition probabilities into the merged states are summed. $m^m_{S-1,b}, b < S-1$ is the probability of moving from the merged state into an unchanged state. The transition probabilities from the merged state are weighted means of the transition probabilities for the original states, weighted with the stationary probabilities. Finally, $m^m_{S-1,S-1}$ is the probability of staying in the merged state. For each of the merged states, we sum the probability of staying in the state or moving to the other of the merged states. A weighted mean is calculated for these sums with the stationary probabilities of the states.

$$m^m_{a,b} = \begin{cases} m_{a,b} & a < S-1, b < S-1 \\ m_{a,S-1} + m_{a,S} & a < S-1, b = S-1 \\ \frac{\xi(S-1)\cdot m_{S-1,b} + \xi(S)\cdot m_{S,b}}{\xi(S-1)+\xi(S)} & a = S-1, b < S-1 \\ \frac{\xi(S-1)\cdot(m_{S-1,S-1}+m_{S-1,S}) + \xi(S)\cdot(m_{S,S-1}+m_{S,S})}{\xi(S-1)+\xi(S)} & a = S-1, b = S-1 \end{cases} \tag{41}$$

In the merged Markov model, we have emission distributions $C_s \sim \mathcal{N}^{tail}(\mu_s, \sigma^2_s, \alpha_s), s < S-1$. For state $S-1$ the emission distribution is the merged partial Gaussian distribution $C_{S-1} \sim \mathcal{N}^{tail}_m(\mu_{S-1}, \mu_S, \sigma^2_{S-1}, \sigma^2_S, \alpha_{S-1}, \alpha_S)$.

In the merged Markov Model, transition probabilities remain unchanged, and emission distributions are unchanged or upper-bounded. The merged model is more pessimistic, and a DMP bound derived with the proposed method is safe. Probabilities of workload depletion are lower compared to the model prior to the merge. For jobs associated with a certain accumulation vector, the derived probability of deadline miss and the proportion of those jobs contributing carry-over into the next task period are the same or higher. The probability of job arrivals resulting in longer accumulation vectors, β is higher for the merged model compared to the original for the same number of accumulation periods N.

7 Evaluation

7.1 Goal of the evaluation

The goal of the evaluation is to compare the obtained bounds with empirical deadline miss rates to verify that the method is applicable for a realistic use case, to see how the bound evolves with the workload accumulation iterations, and to see how different server parameters and deadlines affect the pessimism. We compare to state of the art deadline miss probability estimates (Frías et al. 2018). We also compare with simulation of the fitted Gaussian-emission Markov model, to evaluate the validity of this model for the use case, and to see the pessimism for a particular execution time state. Further, we show an estimate of the deadline miss probability assuming independence to see the effect of dependence in the use case.

7.2 Use case and test setup

We evaluated the proposed deadline miss probability bound with a control task for a Furuta pendulum, a rotary inverted pendulum (Vreman et al. 2021). The control task implements a square root Kalman filter (Ljung 1999) estimating angles and angular velocities near the pendulum upright position and a PD controller for stabilizing the pendulum upright at angle 0 of the arm. A separate task simulates the pendulum dynamics and provides an asynchronous TCP server. The control task connects to the server to retrieve arm and pendulum angles and send the control signal. The control task runs periodically with a frequency of 500 Hz. Tests were performed on a Raspberry Pi 3B+ with a `PREEMPT_RT`-patched version of Raspberry Pi OS. The control task was pinned to a core set up as an exclusive `cpuset` and scheduled with the Linux CBS implementation `SCHED_DEADLINE`. The simulator task was pinned to another core using `cpuset`. It runs periodically with the same frequency and was FIFO scheduled with the highest priority. The TCP server of the simulator runs in a separate thread. All cores were run with scaling governor `performance`. USB Ethernet and WiFi were disabled during the tests.

The `ftrace` framework was used to record `sched` events and collect nanosecond-precision timestamps. The control task was scheduled with `SCHED_DEADLINE` setup with high bandwidth and a long server period, resulting in each job finishing within the server period. The time from the `sched_switch` event where the task is switched in to the event where it is switched out was taken as the execution time of a job. In some rare occasions there are several `sched_wakeup` events recorded close to each other in the same period. There are 50,011 `sched_wakeup` events in the log from 50,000 periods. One of these is due to an extra wake up when finishing the task after all periods, but 10 are due to preemptions by kernel space tasks. In these cases, the execution time is taken as the sum of the time frames from switch in to switch out.

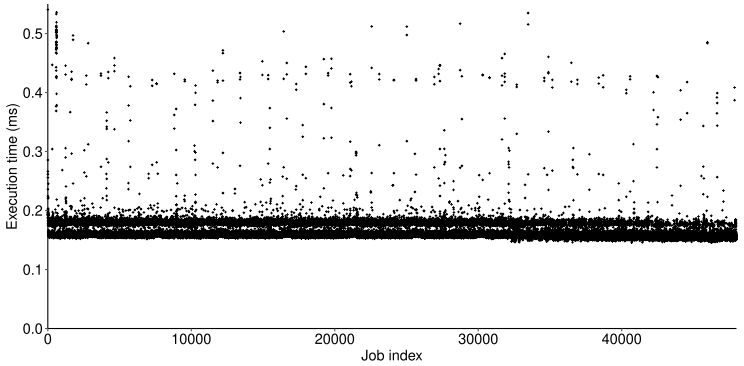

Fig. 16 The recorded execution time trace of the control task

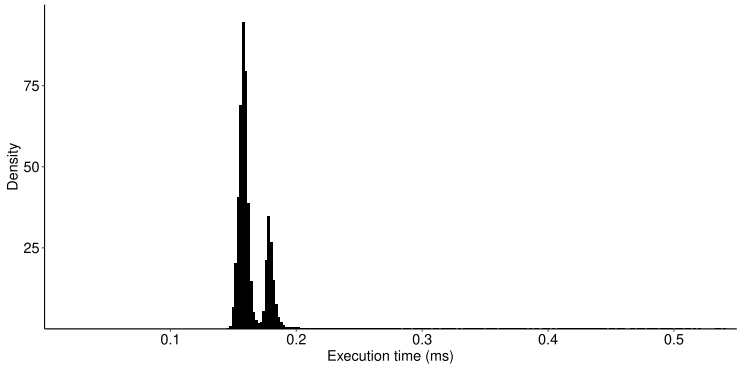

Fig. 17 The density distribution of the execution times starting at job 2000

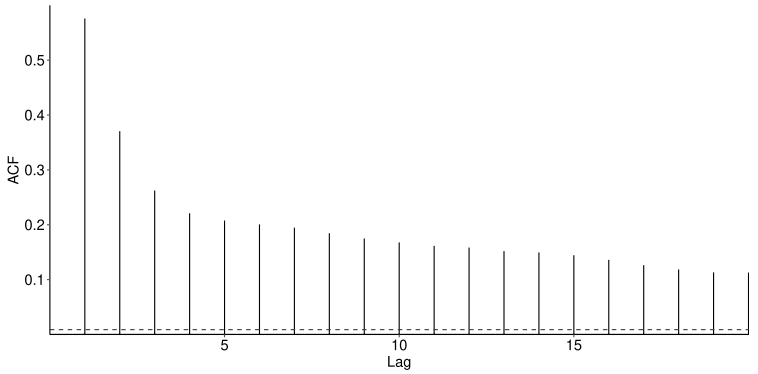

Fig. 18 The autocorrelation of the execution times sequence starting at job 2000

Table 3 Means, standard deviation, and stationary probabilities of the fitted HMM states

State number	1	2	3	4	5	6	7	8
Mean (ms)	0.178	0.178	0.323	0.158	0.159	0.169	0.181	0.153
Standard deviation (ms)	0.002	0.012	0.091	0.003	0.002	0.007	0.003	0.002
Stationary probability	0.128	0.045	0.007	0.086	0.509	0.014	0.078	0.133

7.3 Test setup

Recorded execution times from the control task running 50,000 periods are shown in Fig. 16. There was a run-in period with a higher proportion of execution times at 0.5 ms at the beginning of the trace. The execution times of the first 2000 jobs were discarded before fitting the HMM to the trace. The reason for this is that we want to perform the evaluation under the given assumptions. One assumption is stationarity, and therefore we exclude this part that appears to be a transient period. In Fig. 17, we display the density distribution of the execution time trace starting at job 2000. The autocorrelation of the trace from job 2000 is shown in Fig. 18.

The evaluation was performed with three different configurations of server budget and period ratios:

1. $Q = 0.06$ ms, $n = 5$, $k_1 = 8$, $k_2 = 10$,
2. $Q = 0.07$ ms, $n = 4$, $k_1 = 6$, $k_2 = 8$, and
3. $Q = 0.08$ ms, $n = 4$, $k_1 = 6$, $k_2 = 8$.

Two relative deadlines were evaluated for each of these configurations.

The control task was scheduled with `SCHED_DEADLINE` configured with the different server budgets and period ratios. The task was configured with the relative deadline and logged the number of deadline misses in each 500-job-interval. During these tests, `sched` events are also recorded with the `ftrace` framework, to avoid that any introduced overhead by the tracing causes higher bounds and estimates compared to the empirical results. These recorded traces are not used further.

This small but realistic use case illustrates the method's applicability. https://github.com/annafriebe/ContMM_RT_BoundDMP.[1]

7.4 Markov model

The method outlined in Friebe et al. (2020) was started with 10 initial states and identified an HMM with 8 states. The transition matrix is shown in Eq. (42), and from this we conclude that the Markov Chain is irreducible. The resulting state means, standard deviations, and stationary probabilities are displayed in Table 3. The average computational requirement over a task period is about 0.164 ms, obtained

[1] Omitted for anonymous review.

from multiplying stationary probabilities with means and summing the products. The servers providing the lowest computational resource guarantee 0.28 ms computation time per task period, so all accumulation sequence Markov Chains are ergodic and we will have idle points in the server. The highest mean and standard deviation are observed in state 3. This state has a low stationary probability, only 0.7%, but the transition probability $m_{3,3}$ of staying in state 3 from one round to the next is as high as 63%. This dependence increases the DMP in state 3 and overall.

$$\begin{pmatrix} .739 & .051 & .002 & .003 & .162 & .001 & .041 & .001 \\ .056 & .350 & .012 & .000 & .523 & .008 & .051 & .000 \\ .000 & .310 & .633 & .003 & .000 & .044 & .010 & .000 \\ .006 & .000 & .002 & .408 & .004 & .054 & .000 & .526 \\ .000 & .038 & .002 & .003 & .834 & .001 & .121 & .000 \\ .000 & .000 & .004 & .681 & .063 & .225 & .000 & .028 \\ .377 & .011 & .001 & .000 & .500 & .000 & .107 & .003 \\ .009 & .001 & .002 & .296 & .000 & .038 & .001 & .654 \end{pmatrix} \tag{42}$$

7.5 Evaluated Methods

Six different methods were compared:

- *Linux-CBS*: Empirical deadline-miss ratio. The control task was scheduled with Linux `SCHED_DEADLINE` configured with each setting of server budget Q, task to server period ratio n and evaluated with the different relative deadline to server period ratios k. The task period was 2 ms for all configurations, resulting in different bandwidths. 10 runs of the 50,000-job task were performed for each configuration. The empirical deadline miss ratio was calculated from deadline misses after the 2000-job run-in period.
- *Sim-Cont*: A deadline-miss probability derived by generating execution times from the fitted HMM and feeding them into a CBS simulator with the different server reservations, period ratios, and deadline configurations. A sequence of 10^6 samples was generated from the continuous-emission Markov model described by Table 3 and Eq. (42).
- *Ind*: A deadline-miss probability derived by assuming independent execution times, i.e., generating execution times by randomly sampling from the recorded trace and feeding them into a CBS simulator with the different server reservations, period ratios and deadline configurations. A sequence of 10^6 samples was generated.
- *PROSIT*: A deadline-miss probability derived with PROSITool (Frías et al. 2018). A 6-state discrete-emission HMM is fitted to the execution time trace, using a 10 µs scaling factor for resampling. This HMM is evaluated with PROSIT's solver for steady-state deadline-miss probabilities with the different CBS configurations.
- *Bound-8*: A deadline-miss probability bound derived from the fitted 8-state continuous-emission Markov model and the methods in Sect. 4. The HMM is char-

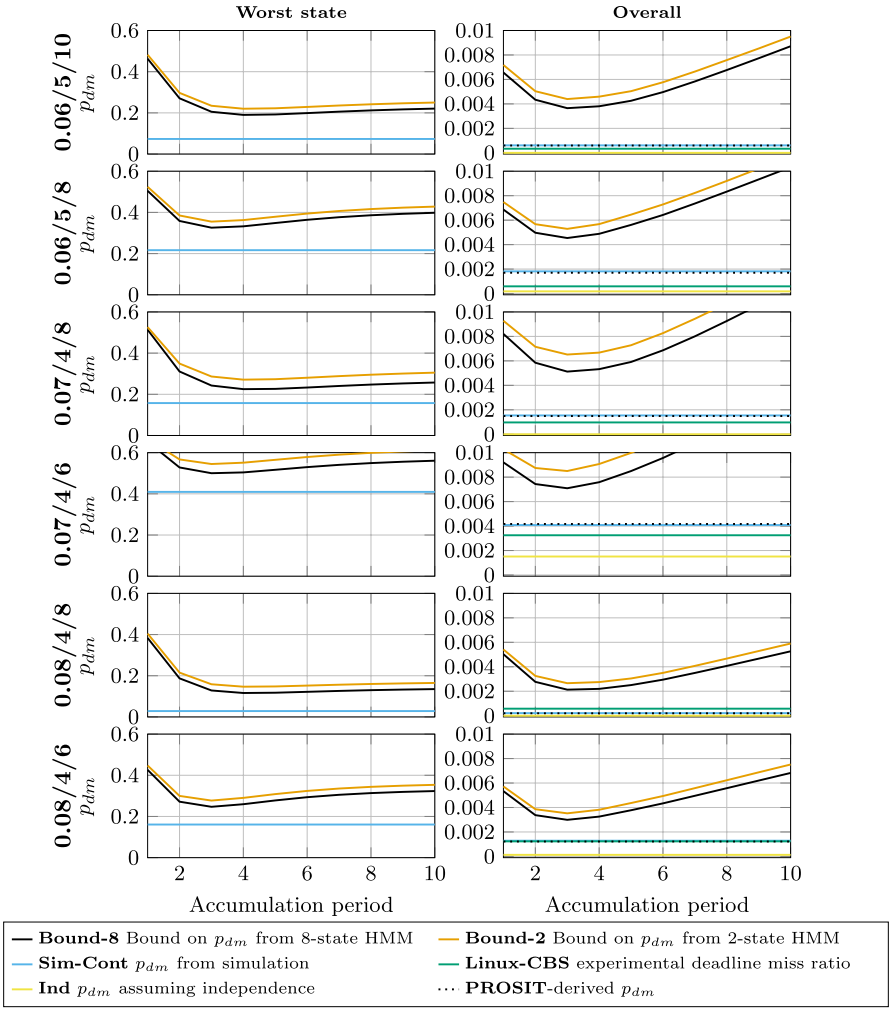

Fig. 19 Evolution over 10 accumulation periods of the *Bound-8* and *Bound-2* according to Eq. (39) for the worst state, and according to Eq. (40) for the task overall, compared to the other methods listed in Sect. 7.5

acterized by Table 3 and Eq. (42). The maximum number of accumulation periods was set to 10. The $\beta(s)_1$ for the first accumulation period were retrieved from the HMM simulation.

- *Bound-2*: A deadline-miss probability bound calculated according to Sect. 4 with a 2-state continuous-emission Markov model. The 2-state model was obtained from merging all states except State 3 from the 8-state model used for *Bound-8* as described in Sect. 6. The initial β values for the first accumulation period were retrieved from simulation with the merged 2-state model, and the maximum number of accumulation periods was set to 10.

Table 4 Time of the bound calculations with the 2-state and 8-state models for the 6 configurations over 5 and 10 accumulation periods

Bound calculation time	Mean (s)	Standard deviation (s)
Bound-2, 5 Accum. periods	0.0153	$4.88 \cdot 10^{-4}$
Bound-2, 10 Accum. periods	0.0552	$4.10 \cdot 10^{-3}$
Bound-8, 5 Accum. periods	1.00	0.0129
Bound-8, 10 Accum. periods	60.66	4.35

Bound-8 and *Bound-2* are calculated as p_{dm} for each state according to Eq. (39), and the overall bound according to Eq. (40). The bounds for the state with the highest p_{dm} of *Bound-8* and *Bound-2* are compared to the deadline miss ratio of this state from *Sim-Cont*, as the empirical DMR *Linux-CBS* and *PROSIT* do not provide per-state estimates.

7.6 Results and discussion

The bounds on the deadline miss probability p_{dm} derived along the workload accumulation for the 8-state model and the 2-state model are shown in Fig. 19, together with the average DMRs of the executions under `SCHED_DEADLINE`, deadline miss probability estimates from HMM simulation, assuming independence, and derived with PROSITool.

Deadline miss probabilities derived with HMM simulation and PROSITool are higher than empirical DMRs, except for the CBS configuration 0.08/4/8. In this case, we observe p_{dm} of 0.021% derived with *Sim-Cont* and 0.020% with *PROSIT*, compared to 0.058% for the empirical *Linux-CBS*. This may be caused by a low probability state that is not captured in the fitting of the Markov Models. It may also be due to chance. This configuration has the lowest number of deadline misses, and a larger number of runs with *Linux-CBS* may have been needed for a reliable DMR estimate.

We observe that HMM simulation *Sim-Cont* estimates are consistently close to the *PROSIT* results, which indicates that the continuous emission distribution HMM is a valid approximation in the evaluated use case.

The resulting *Bound-8* bounds for the overall deadline miss probabilities are 1.76–10 times higher compared to HMM simulation *Sim-Cont*. The bounds for the state with the highest deadline miss probability are 1.3–4.1 times higher. Higher utilization and shorter relative deadlines give tighter bounds.

For the overall bounds of the merged model, *Bound-2*, they are 2.08–12.5 times higher than HMM simulation results *Sim-Cont*. Bounds for the state with the highest p_{dm} are 1.3–5.2 times higher compared to the simulation results.

The number of states and the scaling factor need to be provided when fitting an HMM in PROSIT, and the number of states and the scaling factor need to be provided. For this evaluation, several combinations of these parameters were tested. For 6 states and scaling factor 10 µs, 4 out of 6 states passed the PROSIT independence tests; this was the largest proportion found in the limited exploration. Some pessimism is

introduced with PROSITool's resampling. Tighter or optimistic results were obtained with some of the fitted PROSIT models explored. The calculation time for PROSIT is greatly affected by the range of execution time values in the input trace and the scaling factor. For example, taking the 6-state model and decreasing the scaling factor from 10 to 1 μs causes the computation time to increase from less than 0.5 s to about 20 min on our platform, a factor of 3000. The continuous approach has no resampling concept, and the calculation time is independent of the range of execution time values. A direct comparison between the proposed bound and PROSIT has not been performed. The proposed bound is implemented in Python and PROSIT in C++. PROSITool's computation time also varies a lot with the choice of scaling factor, and therefore, we assess that a direct comparison would not add much value to the evaluation.

In the different configurations, the time for the *Bound-2* and *Bound-8* calculations are logged for 5 and 10 accumulation periods, respectively. The means and standard deviations are shown in Table 4. The Python implementation of the *Bound-8* calculation for the 8-state model runs the first 5 accumulation periods in about one second and 10 accumulation periods in around one minute. With the 2-state bound *Bound-2*, the time required for an optimized implementation grows with the number of accumulation periods as $\mathcal{O}(N^2)$ instead of $\mathcal{O}(N^8)$ for 8 states. The non-optimized Python implementation of the bound calculation for the merged model runs in about 55 ms for 10 accumulation periods.

In the evaluated use case, the tightest bound is already reached at 3–4 accumulation periods. Already at accumulation over 5 periods, the 2-state model is about 65 times faster than the 8-state model. At 10 accumulation periods, the 2-state model is more than 1000 times faster. Combining a low number of states with the use of accumulation vectors instead of accumulation sequences with ordering information provides a strong computational advantage.

Simulations and bounds of the state with the highest p_{dm} show results 50–100 times higher than the overall p_{dm}. While this should not be conflated with the Worst-Case Deadline Failure Probability, we believe that the concept of workload distribution per state is useful. In future work, we aim to develop the accumulation sequence approach relating to the probability of consecutive deadline misses.

In the evaluated use case, one state is identified with a much higher mean and variance than the others. It may be the case that this use case is especially well suited for state reduction into two states. Keeping the state with the highest mean and merging the others may add pessimism in cases where states are more similar to the state with the highest mean.

8 Conclusions and future work

We have proposed a workload accumulation scheme starting from idle points to upper bound the deadline miss probability of a task. The task's computation times are described by a Markov Model with Gaussian emission distributions, and it is running on a reservation-based server.

A Markov model with Gaussian emission distributions allows for higher fitting process automation than discrete emission distributions, where the number

of states and a scaling factor must be provided. Contrary to the discrete case, the time required to obtain the bound is independent of the range of execution times in the analyzed sequence, and no scaling factor is needed.

Further, we proposed a method for state merging. The bound computation time is reduced by reducing the number of states by merging. The time complexity for obtaining a bound for a model with S states considering N accumulation periods after an idle point is $\mathcal{O}(N^S)$. A bound is obtained early in the process and is updated successively.

The evaluation use case is a control task of a Furuta pendulum. The task is run with the Linux kernel implementation of CBS. The ratio of the number of missed deadlines to the total number of jobs is compared to the obtained bounds on the deadline miss probability. Bounds are derived from the fitted 8-state model and from a merged 2-state model obtained from the 8-state model. Furthermore, deadline miss probabilities for comparison are derived with a discrete emission-HMM (Frías et al. 2018, 2017; Abeni et al. 2017), by simulation with the fitted HMM, and simulation assuming independence.

All bounds in the evaluation are higher than the simulation results. The overall bounds for the 8-state model are 1.76–10 times higher, and in the state with the highest deadline miss probability, the bounds are 1.3–4.1 times higher. The overall bounds obtained with the merged 2-state model are 2.08–12.5 times higher, and the bounds for the state with the highest deadline miss probability are 1.3–5.2 higher. All bounds are also safe compared to experimental deadline miss ratios. In the evaluation, the bound over 10 accumulation periods takes about 0.06 s to calculate for the 2-state model, but a minute for the 8-state model, an improvement of a factor 1000. Combining the workload accumulation method with state number reduction by merging gives a strong computational benefit.

In future work, it would be interesting to develop the workload accumulation approach to evaluate the probability of consecutive deadline misses, or extend the approach to support DAG-based tasks. The bounds could potentially be evaluated for use in an adaptive setting to monitor changes in the deadline miss probability to adapt the Quality-of-Service (QoS) level.

Funding Open access funding provided by Mälardalen University. This study was supported by Vetenskapsrådet (Grants No 2016-03660 and 2020-05094) and by Stiftelsen för Kunskaps- och Kompetensutveckling (Grants No 20190034 and 20240011).

References

Abeni L, Buttazzo G (1998) Integrating multimedia applications in hard real-time systems. In: IEEE Real-Time Syst. Symp. (RTSS), pp 4–13. https://doi.org/10.1109/REAL.1998.739726

Abeni L, Buttazzo G (1999) QoS guarantee using probabilistic deadlines. In: Euromicro Conf. Real-Time Syst. (ECRTS), pp 242–249. https://doi.org/10.1109/EMRTS.1999.777471

Abeni L, Buttazzo G (2001) Stochastic analysis of a reservation based system. In: Int. Workshop on Par. and Distr. Real-Time Syst., vol 1

Abeni L, Manica N, Palopoli L (2012) Efficient and robust probabilistic guarantees for real-time tasks. J Syst Soft 85(5):1147–1156. https://doi.org/10.1016/j.jss.2011.12.042

Abeni L, Fontanelli D, Palopoli L, Frías BV (2017) A Markovian model for the computation time of real-time applications. In: IEEE Instrum. & Meas. Tech. Conf. (I2MTC), pp 1–6. https://doi.org/10.1109/I2MTC.2017.7969878

Åkesson B, Nasri M, Nelissen G, Altmeyer S, Davis RI (2022) A comprehensive survey of industry practice in real-time systems. Real-Time Syst 58(3):358–398. https://doi.org/10.1007/s11241-021-09376-1

Bernat G, Burns A, Newby M (2005) Probabilistic timing analysis: an approach using copulas. J Embed Comput 1(2):179–194

Bozhko S, Brüggen G, Brandenburg B (2021) Monte Carlo response-time analysis. In: IEEE Real-Time Syst. Symp. (RTSS), pp 342–355. https://doi.org/10.1109/RTSS52674.2021.00039

Bozhko S, Marković F, Brüggen G, Brandenburg BB (2023) What really is pWCET? A rigorous axiomatic proposal. In: IEEE Real-time systems symposium (RTSS). https://doi.org/10.1109/RTSS59052.2023.00012

Buttazzo GC, Lipari G, Abeni L, Caccamo M (2005) Soft real-time systems: predictability vs efficiency. Springer, Berlin. https://doi.org/10.1007/0-387-28147-9

Chen K-H, Ueter N, Brüggen G, Chen J-J (2019) Efficient computation of deadline-miss probability and potential pitfalls. In: Design, automation & test in Europe conference & exhibition (DATE'19), March 25–29, Florence, Italy. IEEE, pp 896–901. https://doi.org/10.23919/DATE.2019.8714908

Chen K-H, Günzel MBrüggen G, Chen J-J (2022) Critical instant for probabilistic timing guarantees: refuted and revisited. In: 2022 IEEE real-time systems symposium (RTSS). IEEE, pp 145–157. https://doi.org/10.1109/RTSS55097.2022.00022

Chen J-J, Günzel M, Bella P, Brüggen G, Chen K-H (2024) Dawn of the dead (line misses): impact of job dismiss on the deadline miss rate. arXiv preprint arXiv:2401.15503

Clark DD, Shenker S, Zhang L (1992) Supporting real-time applications in an integrated services packet network: architecture and mechanism. SIGCOMM Comput Commun Rev 22(4):14–26. https://doi.org/10.1145/144191.144199

Coles S (2001) An introduction to statistical modeling of extreme values, vol 208. Springer, London. https://doi.org/10.1007/978-1-4471-3675-0

Cucu-Grosjean L (2013) Independence-a misunderstood property of and for probabilistic real-time systems. In: Real-time systems: the past, the present and the future, pp 29–37

Cucu-Grosjean L, Santinelli L, Houston M, Lo C, Vardanega T, Kosmidis L, Abella J, Mezzetti E, Quiñones E, Cazorla FJ (2012) Measurement-based probabilistic timing analysis for multi-path programs. In: Euromicro Conf. on real-time systems (ECRTS), pp 91–101. https://doi.org/10.1109/ECRTS.2012.31

Davis RI, Cucu-Grosjean L (2019) A survey of probabilistic schedulability analysis techniques for real-time systems. In: LITES Leibniz Trans Embed Syst 1–53. https://doi.org/10.4230/LITES-V006-I001-A004

Davis RI, Cucu-Grosjean L (2019b) A survey of probabilistic timing analysis techniques for real-time systems. Leibniz Trans Embed Syst 6(1):03–10360. https://doi.org/10.4230/LITES-V006-I001-A003

Davis RI, Burns A, Griffin D (2017) On the meaning of pWCET distributions and their use in schedulability analysis. In: In Proceedings real-time scheduling open problems seminar at (ECRTS'17)

de Barros Vasconcelos J, Lima G (2022) Possible risks with EVT-based timing analysis: an experimental study on a multi-core platform. In: 2022 XII Brazilian symposium on computing systems engineering (SBESC). IEEE, pp 1–8. https://doi.org/10.1109/SBESC56799.2022.9964853

Díaz JL, García DF, Kim K, Lee C-G, Bello LL, López JM, Min SL, Mirabella O (2002) Stochastic analysis of periodic real-time systems. In: IEEE Real-Time Syst. Symp. (RTSS), pp 289–300. https://doi.org/10.1109/REAL.2002.1181583

Diaz JL, Lopez JM, Garcia M, Campos AM, Kim K, Bello LL (2004) Pessimism in the stochastic analysis of real-time systems: concept and applications. In: IEEE Int. Real-Time Syst. Symp. (RTSS), pp 197–207. https://doi.org/10.1109/REAL.2004.41

Frías BV (2018) Bringing probabilistic real-time guarantees to the real world. PhD thesis, University of Trento

Frías BV, Palopoli L, Abeni L, Fontanelli D (2017) Probabilistic real-time guarantees: there is life beyond the IID assumption. In: IEEE real-time and embedded Tech. and Appl. Symp. (RTAS), pp 175–186. https://doi.org/10.1109/RTAS.2017.18

Frías BV, Palopoli L, Abeni L, Fontanelli D (2018) The PROSIT tool: toward the optimal design of probabilistic soft real-time systems. Softw Pract Exp 48(11):1940–1967. https://doi.org/10.1002/spe.2604

Friebe A, Papadopoulos AV, Nolte T (2020) Identification and validation of Markov models with continuous emission distributions for execution times. In: IEEE Int. Conf. on Emb. and real-time Comp. Syst. and Appl. (RTCSA), pp 1–10. https://doi.org/10.1109/RTCSA50079.2020.9203594

Friebe A, Marković F, Papadopoulos AV, Nolte T (2021) Adaptive runtime estimate of task execution times using Bayesian modeling. In: IEEE Int. Conf. Emb. and real-time Comp. Syst. and Appl. (RTCSA), pp 1–10. https://doi.org/10.1109/RTCSA52859.2021.00008

Friebe A, Markovic F, Papadopoulos AV, Nolte T (2023) Continuous-emission Markov models for real-time applications: bounding deadline miss probabilities. In: 2023 IEEE 29th Real-time and embedded technology and applications symposium (RTAS), pp 14–26. https://doi.org/10.1109/RTAS58335.2023.00009

Günzel M, Ueter N, Chen K-H, Brüggen G, Chen J-J (2023) Probabilistic reaction time analysis. ACM Trans Embed Comput Syst 22(5s):1–22. https://doi.org/10.1145/3609390

Harchol-Balter M (2024) Introduction to probability for computing, 1st edn. Cambridge University Press, Cambridge

Ivers M, Ernst R (2009) Probabilistic network loads with dependencies and the effect on queue sojourn times. In: Int. Conf. Heterogeneous Netw. for Qual., Reliab., Sec. and Robust. (QShine), pp 280–296. https://doi.org/10.1007/978-3-642-10625-5_18

Leadbetter MR, Lindgren G, Rootzén H (1978) Conditions for the convergence in distribution of maxima of stationary normal processes. Stoch Proc Appl 8(2):131–139. https://doi.org/10.1016/0304-4149(78)90002-9

Lelli J, Scordino C, Abeni L, Faggioli D (2016) Deadline scheduling in the Linux kernel. Softw Pract Exp 46(6):821–839. https://doi.org/10.1002/spe.2335

Lima G, Bate I (2017) Valid application of EVT in timing analysis by randomising execution time measurements. In: 23rd IEEE real-time and embedded technology and applications symposium (RTAS'17), April 18–21, Pittsburg, PA, USA. IEEE, pp 187–198. https://doi.org/10.1109/RTAS.2017.17

Lima G, Dias D, Barros E (2016) Extreme value theory for estimating task execution time bounds: a careful look. In: 2016 28th Euromicro conference on real-time systems (ECRTS). IEEE, pp 200–211. https://doi.org/10.1109/ECRTS.2016.20

Liu R, Mills AF, Anderson JH (2014) Independence thresholds: balancing tractability and practicality in soft real-time stochastic analysis. In: IEEE Real-Time Syst. Symp. (RTSS), pp 314–323. https://doi.org/10.1109/RTSS.2014.38

Ljung L (1999) System identification: theory for the user, 2nd edn. Prentice-Hall information and system sciences series. Prentice Hall, Upper Saddle River. https://doi.org/10.1109/MRA.2012.2192817

Lu Y, Nolte T, Kraft J, Norström C (2010a) A statistical approach to response-time analysis of complex embedded real-time systems. In: IEEE Int. Conf. Emb. and Real-Time Comp. Syst. and Appl. (RTCSA), pp 153–160. https://doi.org/10.1109/RTCSA.2010.13

Lu Y, Nolte T, Kraft J, Norström C (2010b) Statistical-based response-time analysis of systems with execution dependencies between tasks. In: IEEE Int. Conf. Eng. of Compl. Comp. Syst. (ICECCS), pp 169–179. https://doi.org/10.1109/ICECCS.2010.55

Lu Y, Nolte T, Bate I, Cucu-Grosjean L (2012) A statistical response-time analysis of real-time embedded systems. In: IEEE real-time Syst. Symp. (RTSS), pp 351–362. https://doi.org/10.1109/RTSS.2012.85

Manica N, Palopoli L, Abeni L (2012) Numerically efficient probabilistic guarantees for resource reservations. In: IEEE Int. Conf. Emerg. Tech. & Factory Autom. (ETFA), pp 1–8. https://doi.org/10.1109/ETFA.2012.6489566

Marković F, Carlson J, Dobrin R, Lisper B, Thekkilakattil A (2018) Probabilistic response time analysis for fixed preemption point selection. In: 13th IEEE International symposium on industrial embedded systems (SIES'18), June 6–8, Graz, Austria. IEEE, pp 1–10. https://doi.org/10.1109/SIES.2018.8442099

Marković F, Papadopoulos AV, Nolte T (2021) On the convolution efficiency for probabilistic analysis of real-time systems. In: 33rd Euromicro conference on real-time systems (ECRTS 2021). Schloss Dagstuhl-Leibniz-Zentrum für Informatik. https://doi.org/10.4230/LIPIcs.ECRTS.2021.16

Marković F, Roux P, Bozhko S, Papadopoulos AV, Brandenburg BB (2023) CTA: A correlation-tolerant analysis of the deadline-failure probability of dependent tasks. In: Proceedings of the 44th IEEE real-time systems symposium (RTSS). https://doi.org/10.1109/RTSS59052.2023.00035

Markovic F, Nolte T, Papadopoulos AV (2022) Analytical approximations in probabilistic analysis of real-time systems. In: 2022 IEEE Real-time systems symposium (RTSS), pp 158–171. https://doi.org/10.1109/RTSS55097.2022.00023

Martí P, Fuertes JM, Fohler G, Ramamritham K (2002) Improving quality-of-control using flexible timing constraints: metric and scheduling. In: IEEE Real-Time Syst. Symp. (RTSS), pp 91–100. https://doi.org/10.1109/REAL.2002.1181565

Medhi JJ (2003) Stochastic models in queueing theory, 2nd edn. Mathematics in science and engineering. Academic Press, Amsterdam

Mills AF, Anderson JH (2011) A multiprocessor server-based scheduler for soft real-time tasks with stochastic execution demand. In: IEEE Int. Conf. Emb. and real-time Comp. Syst. and Appl. (RTCSA), pp 207–217. https://doi.org/10.1109/RTCSA.2011.30

Palopoli L, Fontanelli D, Abeni L, Frías BV (2016) An analytical solution for probabilistic guarantees of reservation based soft real-time systems. IEEE Trans Parallel Distrib Syst 27(3):640–653. https://doi.org/10.1109/TPDS.2015.2416732

Rabiner LR (1989) A tutorial on hidden Markov models and selected applications in speech recognition. Proc IEEE 77(2):257–286

Santinelli L, Morio J, Dufour G, Jacquemart D (2014) On the sustainability of the extreme value theory for WCET estimation. In: Int. W. on Worst-Case Exec. Time Anal. https://doi.org/10.4230/OASIcs.WCET.2014.21

Shaked M (2007) Stochastic orders. Springer series in statistics. Springer, New York. https://doi.org/10.1007/978-0-387-34675-5_1

Tia T-S, Deng Z, Shankar M, Storch M, Sun J, Wu L-C, Liu JW-S (1995) Probabilistic performance guarantee for real-time tasks with varying computation times. In: Proceedings real-time technology and applications symposium, pp 164–173. https://doi.org/10.1109/RTTAS.1995.516213

von der Brüggen G, Piatkowski N, Chen K-H, Chen J-J, Morik K (2018) Efficiently approximating the probability of deadline misses in real-time systems. In: 30th Euromicro conference on real-time systems (ECRTS'18), July 3–6, Barcelona, Spain. Schloss Dagstuhl-Leibniz-Zentrum fuer Informatik. https://doi.org/10.4230/LIPIcs.ECRTS.2018.6

von der Brüggen G, Piatkowski N, Chen K-H, Chen J-J, Morik K, Brandenburg BB (2021) Efficiently approximating the worst-case deadline failure probability under EDF. In: IEEE Real-Time Syst. Symp. (RTSS), pp 214–226. https://doi.org/10.1109/RTSS52674.2021.00029

Vreman N, Cervin A, Maggio M (2021) Stability and performance analysis of control systems subject to bursts of deadline misses. In: 33rd Euromicro Conf. Real-time systems (ECRTS 2021). https://doi.org/10.4230/LIPIcs.ECRTS.2021.15

Zagalo K, Abdeddaim Y, Bar-Hen A, Cucu-Grosjean L (2022) Response time stochastic analysis for fixed-priority stable real-time systems. IEEE Trans Comput 72(1):3–14. https://doi.org/10.1109/TC.2022.3211421

Anna Friebe is a PhD student at Mälardalen University, Västerås, Sweden since 2019. The PhD project relates to probabilistic tools for analysis and scheduling of real-time systems. She received her MSc in Applied Physics and Electrical Engineering from Linköping University, Sweden in 1998. Anna has a background as a software engineer in the fields of medical image processing, treatment planning systems, and 3D graphics/ haptics simulation, and as a project manager for an autonomous sailboat project at Åland University of Applied Sciences, Mariehamn, Finland 2015-2019.

Filip Markovic received his B.Sc. degree in Software Engineering from Mediterranean University, Podgorica, Montenegro, in 2014, and his M.Sc. degree in Computer Science from Mälardalen University, Västerås, Sweden, in 2015. He earned his Ph.D. in Computer Science from Mälardalen University in 2020, while from 2020 to 2022, he was a Postdoctoral Researcher in the Division of Networked and Embedded Systems at the same university. From 2022 to 2024, he is a Postdoctoral Researcher at the Max Planck Institute for Software Systems in Kaiserslautern, Germany.

Alessandro V. Papadopoulos is a Full Professor of Electrical and Computer Engineering at Mälardalen University, Västerås, Sweden, and a QUALIFICA Fellow at the University of Málaga, Spain. Since 2024, he has been the scientific leader of Applied AI at Mälardalen University. He received his B.Sc. and M.Sc. (summa cum laude) degrees in Computer Engineering from the Politecnico di Milano, Milan, Italy, and his Ph.D. (Hons.) degree in Information Technology from the Politecnico di Milano, in 2013. He was a Postdoctoral researcher at the Department of Automatic Control, Lund, Sweden (2014-2016) and Politecnico di Milano, Milan, Italy (2016). Since the end of 2016, he has been with Mälardalen University. He was the Program Chair for the Mediterranean Control Conference (MED) 2022, the Euromicro Conference on Real-Time Systems (ECRTS) 2023, and the ACM/SPEC International Conference on Performance Engineering (ICPE) 2025. He is an associate editor for the ACM Transactions on Autonomous and Adaptive Systems, Control Engineering Practice, and Leibniz Transactions on Embedded Systems. Since 2024, he is also part of the IEEE Technical Community on Real-Time Systems (TCRTS) Executive Committee. His research interests include robotics, control theory, real-time systems, and autonomic computing.

Thomas Nolte is a Full Professor of Computer Science at Mälardalen University (MDU), Västerås, Sweden, since 2012. Thomas was awarded a Ph.D. degree in Computer Engineering from MDU in 2006. He has been a Visiting Researcher at University of California, Irvine (UCI), Los Angeles, USA, in 2002, and a Visiting Researcher at University of Catania, Italy, in 2005. He has been a Postdoctoral Researcher at the University of Catania in 2006, and at MDU in 2006-2007. Thomas is Head of Research in Electrical and Computer Engineering (ECE) at MDU since 2022, Director of the industrial PhD school Automation Region Research Academy (ARRAY) since 2017, and Director of the Mälardalen University Automation Research Center (MARC) since 2024. In industry, Thomas is a Scientific Advisor at ABB since 2012 (2012-2016 @ ABB Corporate Research, and since 2017 @ ABB Robotics). Thomas has co-authored more than 350 scientific papers related to real-time systems, embedded systems and industrial automation.