On Assessing the Viability of Probabilistic Scheduling with Dependent Tasks

Jaume Abella, Enrico Mezzetti, Francisco J. Cazorla
Barcelona Supercomputing Center (BSC), Spain
Email: { jaume.abella, enrico.mezzetti, francisco.cazorla } @bsc.es

***Abstract*—Despite the significant interest, in the last years, in probabilistic scheduling and probabilistic timing analysis, the interrelation between them has been scarcely addressed. Probabilistic scheduling approaches typically build on a series of assumptions on the probabilistic behavior of each task – or single jobs activations – that have not been shown to be entirely fulfilled by the distributions computed with probabilistic timing analysis. This paper aims at providing a clear understanding of probabilistic Worst-Case Execution Time distributions (pWCET) as a common concept of probabilistic timing and schedulability analysis. We focus on independence of pWCET estimates as the main concern in the application of probabilistic scheduling, with particular emphasis on *measurement-based* probabilistic timing analyses, for which independence across pWCET estimates may not be guaranteed. We relate pWCET (in)dependence to the platform-induced timing dependencies that occur among tasks, and even jobs of the same task. We conclude that independent pWCET distributions can be obtained, even if dependencies exist, by either controlling the measurement protocol, or by deriving distinct pWCET estimates for particular instances of a task.**

I. Introduction

The increasing need for computing performance in critical real-time embedded systems is relentlessly pushing towards the adoption of high-performance hardware features, making current-practice timing analysis more complex, expensive and less effective [1]. The latter aspect also relates to the fact that execution time behavior for programs running on high-performance platforms typically exhibits a tremendously large variability, turning the Worst-Case Execution Time (WCET) behavior into an extreme scenario that hardly happens in practice. Relying on overly pessimistic and extremely infrequent WCET bounds leads to poor utilization and over-provisioning. Symmetrically, it also causes schedulability tests to reject task sets that would be perfectly schedulable in practice, or for which occasional deadline misses would be deemed acceptable. In this scenario, probabilistic scheduling approaches [2], [3], [4], [5] compensate for overly pessimistic and rarely triggered WCET overruns by considering the probability of occurrence of a given timing behavior when assessing the schedulability of a task set.

In the context of probabilistic scheduling, the absolute, monolithic WCET value is typically replaced by a probability distribution. Probabilistic timing analysis (PTA) identifies a family of techniques that has been proposed as a promising approach to deliver WCET estimates for platforms comprising high-performance features, e.g. cache hierarchies and multicores [6]. PTA naturally enables probabilistic scheduling by delivering a probabilistic WCET (pWCET) distribution where each execution time value is associated to an *exceedance probability*. The latter, in turn, upper-bounds the probability that one run of the program exceeds such execution time.

Existing probabilistic scheduling analyses build on the assumption that pWCET estimates are *probabilistically independent* so that subsequent activations of jobs (whether from the same task or not) do not carry time dependencies. Roughly speaking, independence allows the same pWCET distribution to stay valid across jobs. However, we observe that independent pWCET are not guaranteed in the general case for all PTA approaches [7], and in particular for Measurement-Based Probabilistic Timing Analysis (MBPTA), which is the focus of our work.

Within the MBPTA framework, the relation between the platform states (i.e. hardware and software) observed during the test campaign at analysis time and those triggered during operation, determines pWCET estimate's *scope of applicability*. In fact, the pWCET distribution typically computed by MBPTA approaches is not representing the distribution of the timing behavior of a program but it models the probability of its worst-case behavior to exceed a given threshold value. As a consequence, the pWCET does not generally model the behavior of any possible execution. In practice, this may restrict the pWCET distributions to be only valid for a specific job of a task (i.e. activation) but not for any job or sequences of jobs, as may not account for dependencies across jobs and tasks. As an illustrative example, let us assume the pWCET distribution of an actuator system that reacts to periodic inputs from the external temperature. Let us further assume the system exhibits its worst-case timing behavior when the monitored temperature exceeds a given threshold. In general, the pWCET provides us with an estimate and probability for a single exceedance event (corresponding to a temperature peak exceeding a given threshold) but, for example, it does not convey any information on whether multiple exceedance events can happen in a row. This is because the scenario after a first exceedance depends on the natural temperature progression, which would not be properly captured by the probabilistic model.

The scope of applicability of pWCET estimates has only been sporadically and superficially addressed in the literature. In this paper, we cover this gap by identifying the key properties that make pWCET distributions probabilistically independent (dependent), and thus amenable (or not) to

schedulability analysis. We abstract away from the particular approach to derive pWCET estimates and discuss the main aspects at hardware/software level that might *create and convey* dependencies between tasks and across jobs of the same task. In particular we relate the presence of dependence in the execution time (ET) sample – MBPTA input – and in the computed pWCET distribution. We identify and analyze a *taxonomy of dependence scenarios* considering ET and pWCET dependence and relate each scenario to both synthetic examples and benchmarks. We provide both negative and positive evidence on pWCET (in)dependence. On the negative side, we identify concrete examples where pWCET distributions cannot meet the independence requirement of probabilistic scheduling, and show how this can jeopardize schedulability results. On the positive side, we also show that independent pWCET can be achieved despite the existence of time dependencies among tasks both during analysis and operation.

II. Scheduling with probabilistic WCET distributions

Schedulability analyses' [8] objective is to determine whether a task set is schedulable under a given scheduling algorithm. Response Time Analysis (RTA) [9], [10] focuses on determining the worst-case response time of all tasks in the task set, under a given scheduling algorithm. Multiple RTA variants exist, typically differing in the complexity of the underlying task model: the WCET of a task is invariably an essential input parameter to RTA. The tighter the WCET the larger task set can be deemed schedulable by RTA.

The adoption of high-performance hardware, which may produce extremely variable ETs, has motivated the use of probabilistic means to model high execution times on the account that worst-case scenarios can typically happen with extremely low probabilities. *Probabilistic real-time systems* [11], [2], [4], [3] rely on task models where at least one parameter takes the form of a random variable. Typically the WCET bound of a task is not any longer an exact value but a distribution of pWCET values with associated probability of occurrence (or exceedance).

The most natural way of using probabilistic WCET bounds is by adapting the analysis framework to handle distributions instead of deterministic (scalar) values. A formulation of the probabilistic RTA equation is given in [12] to compute the worst-case response time of a job $\tau_{i,j}$, released at time instant $\lambda_{i,j}$:

$$R_{i,j} = B_i(\lambda_{i,j}) \otimes C_i \otimes I_i(\lambda_{i,j}) \quad (1)$$

where $B_i(\lambda_{i,j})$ stands for the backlog of higher priority tasks that have not completed their execution at $\lambda_{i,j}$, C_i is the pWCET distribution for τ_i and $I_i(\lambda_{i,j})$ is modeling the interference of higher priority tasks at $\lambda_{i,j}$. All terms in Eq. 1 are probability distribution functions (PDFs) whose combined effect is obtained by applying the convolution operator, $\otimes$. For a correct application of convolution, however, operands (i.e. pWCET) are required to be probabilistically independent.

Probabilistic scheduling reduces the amount of over-provisioning in real-time systems and increases the number of systems that are deemed schedulable. Alternatively, when probabilistic independence cannot be demonstrated, there are still at least two options: more complex (yet less mature) dependent-aware formulations can be used [13]; or, alternatively, a single deterministic WCET value can be selected from the pWCET at the probability of exceedance that is considered to be acceptable, for example, from the standpoint of a certification authority. The latter solution enables the adoption of standard (deterministic) schedulability analyses.

Note that probabilistic independence is a requirement on pWCET distributions and not on single job-level probabilistic ETs [7]. The distinction here is relevant as we may very well detect probabilistic dependence or stationarity [14] among job ETs, but this does not necessarily mean that those dependencies are preserved in the pWCET distribution. This consideration will be analyzed in detail in the next Sections.

A. Computing probabilistic ET bounds

PTA is the branch of timing analysis that aims at the computation of pWCET distributions. Two main families of PTA exist. Static Probabilistic Timing Analysis (SPTA), alike standard static timing analysis, is capable of deriving the pWCET distribution of a program without actually executing it. SPTA models the probabilistic execution-time distribution of all instructions in a program and then combines them according to the program static structure. At any granularity, probabilistic execution-time (discrete) distributions are represented as execution time profiles (ETPs) that associate each timing behavior to a probability of occurrence [15], [16], [17]. SPTA (not to be confused with [18]) assumes that the probabilistic behavior directly emanates from a characteristic of the execution platform (e.g., a random replacement cache). Current SPTA approaches [16], [17] build on convolutions to sum up the ETPs from different instructions and blocks thereof. The use of convolutions is permitted as ETPs are normally considered by definition independent: ETPs are assumed to be *absolute*, in the sense that they hold for any possible execution condition [7] and hence impose no constrain in the scope of applicability of the obtained pWCET estimates.

MBPTA, instead, is based on collecting ET evidence directly from the actual hardware and program (normally at the level of individual program units). That evidence is later used to fit some known probability distribution, which in turn allows to derive a pWCET estimate. In particular, most MBPTA approaches aim at modeling the WCET of a program as a rare event, and typically achieve this by resorting to the Extreme Value Theory (EVT) [19], [20] framework. EVT builds upon the fact that the asymptotic tail distribution of a sample of independent and identically distributed (*i.i.d.*) random variables converges to specific families of (extreme value) distributions, known as Generalized Extreme Value [21] and Generalized Pareto Distribution [22]. Although some differences exist among specific formulations [23], [24], [25], MBPTA instantiates the EVT framework by following well-

defined procedural steps: (1) assessment of statistical prerequisites for the application of EVT (e.g., i.i.d. and stationarity); (2) data filtering to select those values that belong to the tail of the ET distribution; (3) parameter selection and fitness, a partially guided process to select the most suitable (best likelihood) pWCET distribution to model the sample and assess its precision.

This paper builds on the critical observation that pWCET distributions computed with MBPTA and SPTA do not coincide exactly. While pWCET estimates computed with SPTA are generally considered to be absolutely valid, the validity of pWCET distributions produced by MBPTA is limited to the execution conditions observed at analysis. This is in fact a common trait of measurement-based approaches as they involve two distinct phases in the execution of the target program: analysis and operation. MBPTA is applied on the system at analysis time and is expected to produce results that are valid during operation. As an implied argument, the properties on the timing behavior at analysis are automatically transferred to the timing behavior at operation, from which two observations arise. First, the computed pWCET bounds can be considered reliable only if the observed execution conditions are *representative* of the execution conditions that may rise at operation [24], [26], [27]. Guaranteeing representativeness is not an easy task as execution conditions are determined by the combination of several factors, linked to the software and hardware execution layers. Second, the pWCET distributions obtained with MBPTA are tightly linked to the execution conditions represented in the sample fed to EVT. The particular execution conditions observed at analysis (and in which order and number) determine how the pWCET distribution relates to the ET of the single task activations during operation. It is however difficult to understand this relation: the pWCET can be a valid prediction for (i.e. its scope of applicability covers) a specific job, any single job, or any sequence of jobs. As a consequence, whenever a pWCET does not hold for any sequence of jobs, then it likely carries some sort of dependencies, as conveyed by specific execution conditions. Consequently, MBPTA distributions cannot be seamlessly used to perform probabilistic RTA without assuring they meet the probabilistic independence requirement imposed by convolution.

B. Related work

Some authors analyze the differences across pWCET distributions obtained with SPTA and MBPTA, claiming that the latter may be subject to both, aleatoric variability and epistemic uncertainty, whereas the former are only subject to aleatoric variability [28]. However, SPTA is free of epistemic uncertainty just as long as the timing model on which it builds is reliable, which has been shown unattainable in the general case [1]. Hence, in this work we do not consider epistemic uncertainty as a differentiator across pWCET estimates obtained with either SPTA or MBPTA.

Probabilistic independence is relevant for both SPTA and MBPTA [13], [16], [14], [23], [29], [17]. Under SPTA, it enables the convolution of the ETPs of any sequence of instructions along an execution path. Independence is not a given and needs to be guaranteed by either making assumptions on the underlying hardware platform [16] or flattening dependencies by implementing conservative assumptions on ETPs. Under MBPTA, independence is a main requirement in the original formulation of EVT, which operates on i.i.d. observations, although the i.i.d. requirement can be softened [14], [20]. For this reason, statistical tests are usually applied to check for dependencies within the sample fed to EVT. However, the pWCET obtained for a task may carry out dependencies across jobs or across tasks.

Probabilistic scheduling approaches typically focus on the characterization of the task model and formalization of a schedulability test. In general, they do not discuss on how the probabilistic parameters are obtained in practice. Although generally acknowledged, the independence requirement has not been directly questioned and hastily assumed to be fulfilled [3], [4], [5].

A first attempt to better understand the independence requirement has been made in [30], [12] where it is observed that independence is typically assumed by definition, thus putting no extra requirement on probabilistic scheduling than those generally accepted in deterministic scheduling. The pWCET is defined as an upper bound to all possible probabilistic ETs of a program, so that it cannot depend on any event. While we agree with the definition, we also think that it cannot be always met in practice. As shown later, independent pWCET distributions may not be had, which challenges the use of probabilistic scheduling.

The conclusions from [30] are reused in [31] and attached to the concept of pWCET distributions obtained with EVT. However, the authors do simply refer to [30] and assume that inter and intra-task dependencies are already properly captured by the pWCET. On the contrary, the concept of independence under MBPTA has been preliminary addressed in [7] by introducing a distinction between pWCET derived with SPTA and those computed with MBPTA. In this work we build on the same observations but make a step further in the identification of the sources of probabilistic dependencies in PTA. Focusing on MBPTA, we provide a deeper analysis of the relation between ET and pWCET distributions. In contrast to previous works, we consider different pWCET (in)dependence scenarios showing how independent pWCET can be obtained even in the presence of ET dependencies, hence enabling the use of probabilistic scheduling. In some cases, we show that independent pWCET distributions cannot be obtained, but we provide means to resort to deterministic scheduling.

III. Understanding dependencies across pWCET distributions

In order to analyze whether dependencies across pWCET distributions exist, we provide first a formal definition of the meaning of probabilistic independence for pWCET distributions. Two pWCET distributions are said to be probabilistically independent if and only if the realization of one pWCET distribution does not have any effect of the pWCET distribution

of the other. Formally stated, given two pWCET distributions $\mathcal{X}$ and $\mathcal{Y}$, they are independent if:

$$f_{(\mathcal{X},\mathcal{Y})}(x,y) = f_{(\mathcal{X})}(x) \cdot f_{(\mathcal{Y})}(y) \Rightarrow f_{(\mathcal{Y}|\mathcal{X}=x)}(y) = f_{(\mathcal{Y})}(y) \quad (2)$$

Note that probabilistic independence is not the same as statistical independence. In particular, probabilistic independence refers to probability distribution functions, whereas statistical independence applies to observation samples. Statistical independence is often assessed by means of statistical independence tests, such as for instance, the Ljung-Box test [32]. Those tests may suffer false positives and false negatives. Instead, probabilistic independence can only be assessed theoretically. For instance, given two probabilistically independent pWCET distributions $\mathcal{X}$ and $\mathcal{Y}$, any statistical independence test may reject the independence hypothesis when applied on random samples of those distributions.

A. Consequences of pWCET Dependencies

Different sources may create dependencies across either the ET of tasks, their pWCET distributions, or both. In the case of pWCET distributions, these dependencies relate to the fact that the particular realization of one of the pWCET distributions affects the other pWCET distribution.

This can be illustrated with a simple example: let us assume two tasks, τ_1 and τ_2 so that the pWCET distribution of τ_1 is $pWCET_{\tau_1} = \{\{1,2\},\{0.5,0.5\}\}$, thus meaning that the WCET can be either 1 or 2 time units with 50% chances each, and $pWCET_{\tau_2} = \{\{3,4\},\{0.5,0.5\}\}$. Further, let us assume that whether the outcome is one or another depends on the result of flipping a coin in both cases, where τ_1 and τ_2 execute in 1 and 3 cycles in case of heads, and in 2 and 4 respectively in case of tails. If we flip a coin for each task, τ_1 and τ_2, their pWCET distributions are independent, since chances are 50% for both of them regardless of the realization of the other task. However, if the outcome of both tasks is determined with *the same coin flip*, then both of them have 50% chances, but their realizations are dependent. In particular, if τ_1 takes 1 time unit, then τ_2 can only take 3. Analogously, if τ_1 takes 2 time units, then τ_2 can only take 4.

In the context of real-time tasks, this can occur when one of the parameters influencing the pWCET (as modelled during the analysis phase) of one task, also influences the pWCET of the other. Remarkably, the fact that dependencies occur across ETs does not necessarily causes pWCET distributions to be dependent: different scenarios are evaluated in Section IV, where dependencies across ETs and pWCET distributions are analyzed and discussed in the light of practical examples.

Whether pWCET distributions are independent or dependent has a direct impact on probabilistic scheduling as the latter may deliver unreliable results in the presence of dependencies. If they are independent, then pWCET distributions can be reliably convolved. This means that the resulting addition of the pWCET distributions can be computed multiplying each pair of probabilities and adding their latencies. For instance, recalling the previous example, the joint pWCET distribution of τ_1 and τ_2 would be $pWCET_{\tau_1+\tau_2} = \{\{4,5,6\},\{0.25,0.5,0.25\}\}$ since latency would be, for instance, 4 time units if τ_1 takes 1 time unit (0.5 probability) and τ_2 takes 3 (0.5 probability). Hence, the probability of 4 time units is $0.5 \cdot 0.5 = 0.25$.

However, if τ_1 and τ_2 pWCET distributions are dependent, as illustrated in the previous example, then $pWCET_{\tau_1+\tau_2} = \{\{4,6\},\{0.5,0.5\}\}$ since either they take 1 and 3 time units respectively (0.5 probability) or they take 2 and 4 (also 0.5 probability). Clearly the joint pWCET distribution is different if individual ones are dependent, and hence, it cannot be obtained by means of convolution. In this case, the default solution consists in resorting to deterministic WCET estimates, obtained as the pWCET value such that its exceedance probability is deemed acceptable in relation to a given functional safety standard. Note that in the context of PTA, pWCET distributions upper-bound the actual ET distribution. Hence, the real exceedance probability will be lower than estimated (and potentially null). The probability threshold selected is an upper-bound to the residual risk of exceedance rather than a failure rate and thus, it can be used even for the most critical systems [33], [34].

B. Sources of Dependencies

A wide variety of parameters may create dependencies across pWCET distributions. It is not our purpose enumerating all of them. We want to provide, instead, some examples to show that dependencies are a real concern and are not just an artifact created for the sake of discussion.

Some works obtain pWCET estimates leveraging the probabilities of different input values for tasks, which may lead to different execution paths [14]. These approaches rely on the fact that input distributions considered at analysis match those during operation or, at least, allow upper-bounding ETs during operation. Whether this is doable in the general case or for specific systems is beyond the scope of this paper. Instead, the fact that the actual ET (under worst-case assumptions) depends on input values, may easily create dependencies across tasks sharing at least one such input. For instance, if such an input is provided by a sensor (e.g. current speed of a car) that is used as input for multiple systems, their pWCET distributions may easily be dependent. The existence of dependencies across inputs may cause dependencies across the ET of jobs of a task (under worst-case conditions for other sources of ET variation) and may eventually have an impact on the dependencies of the pWCET distribution of the task with itself. We make specific considerations on this matter in Section IV.

Dependencies may also exist across *random inputs* for tasks. For instance, in the context of time-randomized processors [6], time-randomized caches build on random placement, where the placement function is determined based on a random seed. However, such placement is fully deterministic for a given random seed value and is re-randomized (and thus made independent) across random seed changes. In general, seeds cannot be changed across individual jobs and are only changed periodically at specific time intervals, as in the case of avionics systems [34]. In this case, pWCET distributions are indepen-

dent across time intervals (in different time partitions), but dependent within time partitions since random seeds remain constant within that scope. Hence, the placement function is identical for all functions executed within such partition, and pWCET distributions of tasks in the scope of the time partition are not independent.

C. Probabilistic Independence under SPTA

In principle, both SPTA and MBPTA can produce pWCET estimates that carry out dependencies. Both approaches may, theoretically, obtain pWCET estimates building on the probability distribution induced by input values or, for example, random seeds for time-randomized resources (on which SPTA relies on). Hence, the reasoning above about dependence across pWCET estimates applies to both SPTA and MBPTA, yet with some differences.

By default, existing SPTA techniques have been designed to deliver independent pWCET estimates. SPTA approaches proposed so far have considered either deterministic upper-bounding (in fact for most sources of ET variation) or resources with randomized time behavior (usually for a reduced set of resources such as caches). In this sense, we observe that independence is not intrinsic to SPTA but is determined by the way SPTA is designed to operate. To illustrate this point, we consider an example of SPTA application on time-randomized caches resulting in dependent pWCET estimates (i.e. with a limited scope of applicability). Let us assume a function $f_c()$ that can be called either from $f_a()$ or $f_b()$, each one leaving a different initial cache state for $f_c()$, and thus leading to a different ET for $f_c()$. It could be reasonable, in this scenario, to consider different call contexts, as a means to tighten pWCET estimates, by forcing SPTA to compute a pWCET estimate associated to the case in which $f_c()$ is called from $f_a()$ or $f_b()$. As a consequence, SPTA would produce a pWCET estimate that would be dependent on that context (initial cache state), e.g. from $f_a()$, and would not be valid when $f_c()$ is called from another context, e.g. from $f_b()$.

The natural way to apply SPTA is by considering the worst case scenario – either deterministically or probabilistically – so resulting pWCET estimates do not convey any dependence. One can trade off the lack of dependencies in exchange of tighter results. This contrasts with MBPTA, where specific countermeasures are required to increase (as opposed to reduce) the scope of applicability of pWCET estimates. For MBPTA, dependent pWCET estimates have been shown to be obtainable, for example, when time-randomized caches are used, as an effect of not changing random seeds for placement across individual jobs [34], as explained before.

IV. Probabilistic Independence of pWCET distributions

A key distinction we make in this work is between probabilistically independent ETs and probabilistically independent pWCET distributions. Such a distinction is fundamental to understand why dependences are triggered as well as whether they compromise the probabilistic scheduling assumptions.

In this section we analyze different scenarios where dependencies across ETs and pWCET distributions may occur. For each scenario we describe the type of existing dependencies, their impact on scheduling and how they can be managed efficiently. Finally, we provide specific quantitative assessments and practical examples for real systems. In this discussion, we focus on MBPTA as the primary example since, as shown in previous section, dependencies appear more naturally for MBPTA than for SPTA. However, the presented discussion generally holds for any PTA flavor.

A. Scenario 1: Independent ET and pWCET distributions

In this scenario neither the ET nor the pWCET distributions do exhibit any type of dependence. Hence, Equation 2 applies to both, ET and pWCET distributions for jobs of any pair of tasks[1]. In fact, if ETs are independent, then pWCET distributions are naturally also independent since the pWCET estimation process may preserve or remove existing dependencies, but cannot introduce new ones.

In our case, to obtain independent ETs, the tasks involved must not share any input whose value has an effect on ET. For instance, two tasks controlling cabin climatization and 3D path planning in an aircraft may easily operate on independent data, thus leading to probabilistically independent ETs.

Likely, the most challenging case for independent ETs corresponds to that of different jobs of a given task since, obviously, the task operates always on the same set of inputs. Hence, independent ETs will only be possible either when those input values change independently across jobs or when they do not influence ET. An example of the latter could be that of a task that applies a mask on all pixels of a car camera. On the one hand, input images are not independent since consecutive images will typically be highly similar. On the other hand, if the camera provides fixed-size images and the operation on each pixel has constant latency, the ET of the task will not carry out dependencies across jobs despite the existing dependence across inputs.

1) A practical example: We illustrate the scenario when no dependencies occur by using concocted synthetic distributions. Later, in Section V, we provide examples with real programs. We produce two synthetic benchmarks, which we refer to as *syn1* and *syn2* for short. Both benchmarks consist of a bimodal distribution, one mode occurring once every 40 observations and the other the remaining 39 times. We collect 2,000 measurements for each one, so one distribution is observed 50 times and the other 1,950 times. In particular, those benchmarks are as follows:

- **Syn1**: Once every 40 observations it follows a Gaussian distribution ($\mathcal{N}\left(\mu, \sigma^2\right)$) with parameters $\mu = 500$ and $\sigma = 20$. The other 39 out of 40 observations follow a Gaussian distribution with $\mu = 200$ and $\sigma = 20$.
- **Syn2**: Once every 40 observations it follows a Gaussian distribution ($\mathcal{N}\left(\mu, \sigma^2\right)$) with parameters $\mu = 1000$ and

[1]Note that, potentially, those two tasks could be *the same task*, meaning that the ET of any job of such task is independent of previous jobs.

$\sigma = 20$. The other 39 out of 40 observations follow a Gaussian distribution with $\mu = 700$ and $\sigma = 20$.

While this experiment can be built on single-mode benchmarks, we use bimodal distributions because we reuse those benchmarks for the experiments in following scenarios.

We have estimated the pWCET distribution for both benchmarks building upon MBPTA-CV [24], which is suitable for analysing distributions that can be upper-bounded with EVT distributions that use exponential tails, as in our case. Moreover, MBPTA-CV has been devised explicitly to account for multi-modal (a.k.a. mixture) distributions. Figure 1a shows the empirical complementary cumulative distribution function (empirical CCDF or ECCDF) for the two benchmarks with dashed lines (black for *syn1* and blue for *syn2*). The CCDF is a convenient way to plot distribution tails as exceedance probabilities are visually evident. In the plot, the x-axis indicates the value that is exceeded with a probability at most as high as indicated in the y-axis. Straight lines correspond to the pWCET distributions computed for each benchmark.

Figure 1b shows the result of convolving the pWCET estimates of both benchmarks (straight line). The figure also shows the result of adding one value of each benchmark in a random order, thus reflecting the lack of dependencies across *syn1* and *syn2*. As expected, the pWCET distribution from the convolution of the individual pWCET distributions upper-bounds the sample obtained by adding individual measurements. This provides evidence that pWCET distributions can be operated freely in the absence of dependence across ETs.

B. Scenario 2: Dependent ET and independent pWCET distributions

Dependencies across ETs of jobs (either of the same or different tasks) are quite common in embedded real-time systems. However, the existence of dependencies across ETs does not imply that those dependencies will persist across pWCET distributions. In fact, the WCET estimation process can remove those dependencies, thus leading to independent pWCET estimates.

To illustrate this scenario, tasks need to share some inputs, and/or hardware and software states must affect their ETs. For instance, a pipelined process processing input data from I/O may have different tasks with ET highly correlated with the size of the input data to process. Hence, either all tasks execute slowly or quickly depending on the size of the data to process, thus reflecting dependencies across tasks. This particular example corresponds to a real scenario in the avionics domain [33].

Analogous examples can be drawn for jobs of the same task. This would be the case, for example, of an engine cooling system that monitors engine temperature to decide how to configure the cooling solution. ETs can be presumably low if temperatures are low since no specific action is required. Conversely, ETs may increase if temperature is high enough to require re-configuring the cooling solution so that to dissipate enough heat and avoid overheating of the engine. While such a task could be executed, for instance, every 100ms or even every 1s, temperature may likely change little in such a short time period, thus causing the ET of a job to be in most cases similar to the ET of previous jobs, with large ET variations occurring only sporadically. This is illustrated with a hypothetical example in Figure 2, where engine temperature is low during the first 16 seconds, thus requiring the monitor to execute for around 10-11ms. Then, temperature increases enough to require the activation of cooling aids, which increases the ET of the monitor up to 22-25ms during 14 seconds. Finally, temperature becomes low enough so that the monitor does not need to perform any further action during the last 10 seconds.

The WCET estimation process can remove dependencies, at least, in two different ways: ① and ②.

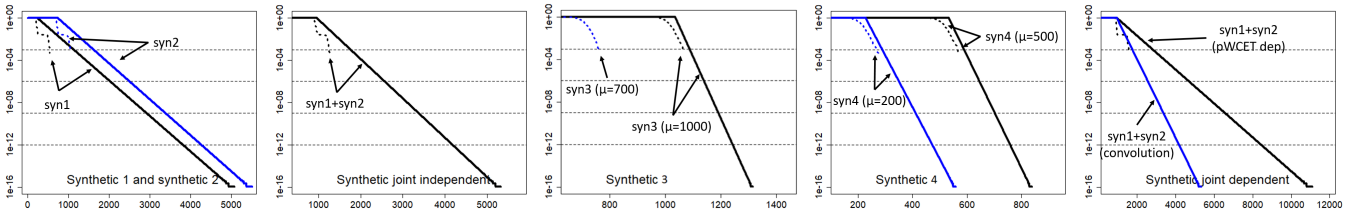
① By imposing the input causing dependencies to take its worst-case value (or set of values) at analysis time: this will lead to pWCET distributions that are pessimistic when inputs are not the worst-case ones during operation, but they will always be reliable. For instance, in the example in Figure 2, pWCET estimation should use only high temperatures as input. While ETs during operation are not independent, pWCET estimates will be since they already account for worst-case conditions, thus upper-bounding for all jobs those temperature conditions that lead to ETs in the range 22-25ms.

② By specialization: in the case of stationary processes where ETs vary with specific periods, pWCET estimates may be specialized for different jobs in each period. For instance, given a task whose input values (or hardware/software initial state) vary across the N jobs in an execution period, we could compute N pWCET distributions and use each one of them for the jobs in the i^{th} position in that period. This case is common for tasks with multiple (periodic) operation modes such as, for instance, a task monitoring the crank angle in a car, which can perform specific actions every d rotation degrees (e.g. every 45 rotation degrees), thus repeating them every $N = 360/d$ jobs. We could, therefore, either use a single pWCET distribution upper-bounding the N execution modes, or use a specific pWCET distribution for each execution mode accounting for the specific conditions in each mode, thus leading potentially to lower pWCET distributions for some jobs in the period.

1) A practical example: For this scenario we use two synthetic dependent benchmarks (*syn3* and *syn4*) similar to the previous ones. In particular, those benchmarks are as follows:

- **Syn3**: Has two operation modes. In one of them, it follows a Gaussian distribution with $\mu = 1000$ and $\sigma = 20$. In the other mode, it follows a Gaussian distribution with $\mu = 700$ and $\sigma = 20$.
- **Syn4**: Analogously, it also has two operation modes. In one of them, it follows a Gaussian distribution with $\mu = 500$ and $\sigma = 20$. In the other mode, it follows a Gaussian distribution with $\mu = 200$ and $\sigma = 20$.

Hence, both benchmarks have a "fast" and a "slow" mode. Those are meant to mimic, for instance, different inputs or processor states for tasks, where one is intended to upper-bound the other. This reflects dependencies inside a single task. We



(a) ETs and pWCET for syn1 and syn2.

(b) Joint ET and pWCET convolution for syn1+syn2.

(c) ETs and pWCET for syn3 with $\mu = 700$ and $\mu = 1000$.

(d) ETs and pWCET for syn4 with $\mu = 200$ and $\mu = 500$.

(e) ETs for syn1+syn2, their pWCET and pWCET convolution.

Fig. 1: Examples for the different ETs (dashed lines) and pWCET (straight lines) scenarios. Plots (a) and (b) correspond to independent ETs and pWCET, (c) and (d) to dependent ETs and independent pWCET, and (e) to dependent ETs and pWCET.

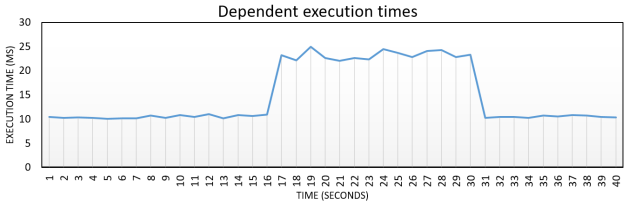


Fig. 2: Example of time-dependent ETs for a hypothetical engine cooling solution monitor.

collect samples with 2,000 observations for each benchmark and each mode, thus resulting in 8,000 measurements in total.

Figure 1c plots the ECCDF for *syn3* under each individual mode, and the pWCET distribution for the "slow" mode (black straight line). In this particular case, we cannot determine a priori when each mode will occur during operation, and a safe assumption would be that the task will always behave in the worst ("slow") mode . As shown in the figure, the pWCET distribution for the "slow" mode upper-bounds both modes. Of course, this approach is only reasonable when enough evidence exists that one mode necessarily leads to higher ETs, either deterministically or probabilistically, than the other.

Figure 1d studies *syn4*. This example is similar to the previous one, with the exception that in this case we assume that we can determine when each mode will be triggered. For instance, the task could be run twice in a scheduling window, being the first one in the "slow" mode and the second one in the "fast" mode. In this case we can resort to the solution of using the pWCET distribution for the "slow" mode, as done for *syn3*, or use specialized pWCET distributions for each set of jobs of the task. As shown in the figure, the pWCET distribution for the "fast" mode (blue straight line) would produce lower resource utilization, but at the same time it cannot be used to upper-bound the ET of "slow" jobs.

C. Scenario 3: Dependent ETs and pWCET distributions

Whether dependencies across ETs are removed or preserved depends on the pWCET estimation process. For instance, recalling the example of the pipelined functions processing a input data from I/O, we may be able to derive either independent or dependent pWCET distributions. Independent pWCET distributions would be obtained by considering those message (input data) sizes that lead to the highest ETs (typically the largest messages). Nonetheless, if we know the message size frequency during operation, we could leverage such information to tighten pWCET distributions. For instance, we could derive pWCET distributions capturing a scenario where there are two types of messages (long and short) and we know that up to 10% of the messages lead to significantly higher ETs. In this case, however, pWCET distributions of the functions would not be independent. In particular, we know that either all functions execute with short ETs (short messages) or all of them do it with long ETs (long messages). The later would occur up to 10% of the times. Notably, assuming 5 functions, if we convolve pWCET estimates as if they were independent, we would obtain that long ETs for all 5 functions simultaneously are expected up to 0.001% of the times ($0.1^5 = 0.00001 = 0.001\%$), which is clearly wrong (and optimistic). Moreover, convolution would also lead to non-null probabilities of having exactly 1, 2, 3 or 4 functions with long ETs, when the probability of those scenarios is known to be zero. In this case pWCET distributions are clearly dependent, and those dependencies cannot be ignored.

With dependent pWCET distributions, we can still select an exceedance threshold of relevance and resort to deterministic pWCET estimates individually for each task, to later feed them to deterministic schedulability tests. By applying this approach to the example above, we would account for the impact of long messages for each function in their individual WCET estimates, and hence, for the scenario of all of them operating on long messages simultaneously.

In the more general case, in order to avoid resorting to deterministic pWCET estimates and being able to operate with pWCET distributions, dependencies across pWCET distributions must be modeled probabilistically with techniques alternative to convolutions, as the latter can only be used with independent distributions. Some authors have suggested copulas as a means to combine dependent distributions [13], but precise information about dependencies would be needed, which we regard as hard to obtain in general.

1) A practical example: To illustrate the case of dependent ETs and pWCET distributions, we reuse the example of scenario 1, but we assume *syn1* and *syn2* to carry dependencies

TABLE I: Amenability of pWCET distributions to probabilistic scheduling.

	Scenario	Common wisdom	Conclusion of this paper
MBPTA	Independent ETs	(i) Lead to *independent* pWCET, (ii) Must be ascertained, (iii) Can be used for scheduling	Confirmed
	Dependent ETs	(i) Lead to *dependent* pWCET, (ii) Cannot be used for scheduling	Independent pWCET estimates can still be obtained for some types of dependencies, controlled experiments and scope of applicability
SPTA	N/A	(i) pWCET assumed *independent* by definition, (ii) Always usable for scheduling	SPTA can in principle operate on dependent inputs, possibly resulting in dependent pWCET estimates

on each other: whenever $\mu = 500$ for *syn1*, $\mu = 1000$ for *syn2*, and whenever $\mu = 200$ for *syn1*, $\mu = 700$ for *syn2*. Hence, individual pWCET distributions for each benchmark are the same as those shown in Figure 1a, but pWCET distributions are no longer independent. For instance, if they were independent, the probability of both benchmarks experiencing high values of μ would be $1/40 \cdot 1/40 = 1/1600$, and there would be a non-null probability of "exactly" (i.e. only) one of them experiencing a high value of μ. However, in our example, the probability of high μ values for both benchmarks simultaneously is $1/40$, and that of "exactly" one high μ value is zero.

Figure 1e shows the cumulative ECCDF of *syn1* and *syn2* (black dashed line), and its pWCET distribution, which accounts for dependencies. For completeness, we show the pWCET distribution obtained convolving individual pWCET distributions of each benchmark (blue straight line). As shown, assuming independence leads to a "lower" pWCET distribution than if accounting for dependencies. Even worse, in this particular example, such pWCET distribution does not even upper-bound measured data, which confirms the risk of obtaining optimistic bounds. As a consequence, either we derive a pWCET distribution accounting for dependencies, which may not always be possible, or we must resort to deterministic scheduling, as explained before. For instance, at an exceedance probability of 10^{-6} per run, the pWCET estimate for dependent data is 4,695 time units, whereas that assuming independent data is 2,566 time units. If we use instead individual pWCET estimates for each benchmark, they are 2,026 and 2,502 time units for *syn1* and *syn2* respectively, thus totalling 4,528 time units. While this value is slightly below the actual pWCET estimate of the distribution obtained adding the measurements of both benchmarks, it is close enough to tentatively attribute the discrepancy to statistical variation in the samples themselves. In fact, arguably individual pWCET bounds implicitly account for any potential dependence, so the discrepancy is likely accountable to some degree of pessimism.

D. Putting it all together

The analyzed scenarios and examples exhaustively cover all possible (in)dependence relations between ETs and pWCET distributions. From our analysis we derive valuable conclusions on whether and under what conditions PTA results, either from SPTA or MBPTA, can be used in probabilistic scheduling analysis. Table I summarizes our conclusions and assesses them against the common perception. Notably, we have shown (also by example) the following facts:

- MBPTA allows obtaining independent pWCET distributions from independent ET measurements, and from measurements that exhibit certain types of dependence. Independent pWCET distributions can be unconstrainedly used for scheduling purposes.
- In the presence of other types of dependence, pWCET distributions will be inevitably dependent and unamenable to probabilistic scheduling. Still, deterministic pWCET bounds, at the desired exceedance threshold, can be used for deterministic scheduling.
- Not only MBPTA, but even SPTA can deliver dependent pWCET distributions that cannot be used in probabilistic schedulability analysis.

V. Evaluation

In this section we collect evidence on common real-time benchmarks rather than on synthetic examples. Also, ETs are collected by executing the benchmarks on a specific simulation-based platform, rather than assuming that they follow a particular distribution. Next we present the evaluation framework and specific results for each of the three scenarios described in the previous section.

A. Experimental framework

We consider several programs from the Mälardalen benchmark [35] suite. This suite has been regarded as representative for WCET estimation and includes a wide variety of programs and kernels. Since our analysis is agnostic of the particular benchmarks – and in fact the particular benchmark suite – evaluated (and size thereof), we have performed our analysis on an arbitrary selection from Mälardalen benchmarks. The conclusions we have reached are analogous across all those used. In particular, we present our results on two specific benchmarks: `compress` and `insertsort`.

We run those benchmarks on the SoCLib [36] framework, implementing a cycle-accurate processor model supporting PowerPC binaries [37]. In particular, we model a pipelined processor with in-order execution consisting of 4 stages: fetch (F), decode (D), execute (E), and write-back (WB) (see Figure 3). Our reference processor design implements 4KB 8-way set-associative instruction (IL1) and data (DL1) caches, with 16-bytes cache lines. Hit latency is 1 cycle and miss latency 100 cycles to serve data from memory. As we aim at obtaining pWCET distributions for our analysis, we implement random placement and replacement policies [38], which remove the dependence of pWCET bounds on the actual memory placement of objects and enable a smooth application

of EVT. As stated before, we build upon MBPTA-CV [24] for the estimation of pWCET distributions out of the different data samples considered.

Measurements for the different benchmarks are collected in two different ways: on empty caches and preserving cache state across executions. These two measurement protocols allow modelling the scenarios presented in Section IV. In each scenario we provide details of the system being modelled. For each experiment we collected between 500 and 1,000 execution time measurements, as dictated by MBPTA-CV to obtain statistically-reliable pWCET estimates.

B. Scenario 1 evaluation: Independent ET and pWCET distributions

We consider a scenario analogous to that modelled in Section IV, where each benchmark exhibits a bimodal behavior. In particular, 5% of the executions are performed on empty caches and the remaining 95% on warmed up caches for both `compress` and `insertsort`. Executions with empty caches or with cache reuse are completely uncorrelated across benchmarks as they are independent, and follow no particular order in the individual samples for each benchmark.

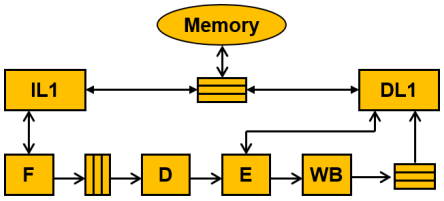


Fig. 3: Block diagram of our reference architecture.

Figure 4a shows the ECCDF for each benchmark (`compress` in black and `insertsort` in blue) with dashed lines, and their individual pWCET distributions with straight lines. As observed, pWCET distributions upper-bound the set of measurements of each benchmark. Figure 4b, instead, shows the result of convolving the pWCET distributions of each benchmark (straight line) and the ECCDF of a sample generated by adding one measurement from each benchmark in no particular order (samples of the benchmarks are randomly sorted). Since ETs are independent across benchmarks, their pWCET distributions can be safely convolved for schedulability analysis. As expected, the joint pWCET distribution upper-bounds the addition of the measurements for both programs.

C. Scenario 2 evaluation: Dependent ET and independent pWCET distributions

In this scenario we study dependencies across jobs of a given task. For that purpose we assume that benchmarks are scheduled in a way that caches are flushed every two executions, as if the boundary of a timing partition was reached and cache contents were, therefore, flushed, similarly to what is done for ARINC653 systems in the avionics domain [39]. Hence, benchmarks alternatively execute on an empty cache or reusing cache contents, with a clear dependence across ET measurements, since those in odd positions have higher values (empty caches) than those in even positions (cache reuse).

As explained before, in this case we can get rid of ET dependencies by shifting the source of dependence to its worst-case state. Hence, we assume that caches are always empty and obtain the pWCET distribution under this assumption. Results are shown in Figure 4c for `compress`. In particular, the straight line corresponds to the pWCET distribution assuming caches are always empty, and the dashed lines correspond to the ECCDF for measurements with empty cache (black) and with cache reuse (blue). As shown, the pWCET distribution is a tight upper-bound to the case with empty caches. The other operation mode, with cache reuse, falls apart (on the lower side) indicating that its execution conditions can only lead to (probabilistically) lower execution times than the case with empty caches. Hence, resorting to the pWCET distribution obtained on empty caches removes dependencies on input cache state and delivers independent pWCET bounds that can be freely operated by probabilistic scheduling.

However, enforcing worst-case conditions for pWCET estimation may result in overly-pessimistic bounds. Hence, we can estimate separated pWCET distributions for odd and even jobs, and reach a more efficient use of resources, as illustrated in Figure 4d for `insertsort` (blue lines with cache reuse and black lines on empty caches). As shown, using specialized pWCET distributions provides around 15-20% tighter bounds.

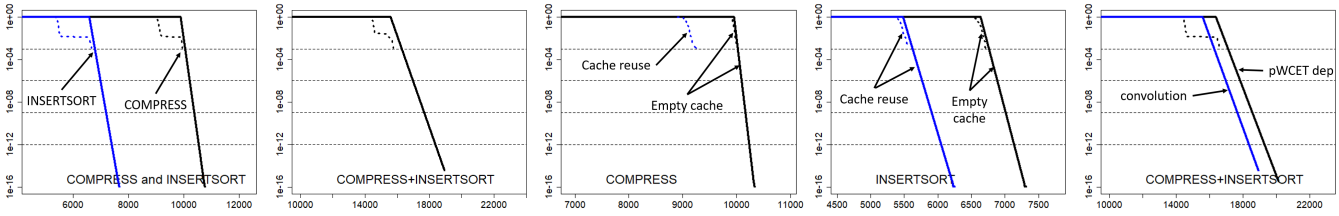
A similar analysis has been conducted on other Mälardalen benchmarks (see top part of Table II), where the improvement with specialized pWCET estimates varies with cache reuse. In general, specialized pWCET estimates may be used for different task operation modes providing arbitrarily large gains.

D. Scenario 3 evaluation: Dependent ET and pWCET distributions

Finally, we assess the case where both, ETs and pWCET distributions, are no longer independent. For that purpose, we reuse the experiment in Scenario 1, where each benchmark runs 5% of the times on an empty cache and 95% of the times on a warmed up cache. However, in this experiment, either both of them run on an empty cache or both of them run on a warmed up cache. This allows modelling scenarios where execution times of jobs belonging to different tasks follow similar patterns due to dependencies.

TABLE II: Gains with specialized pWCET estimates (top) and joint pWCET estimates with convolution or accounting for dependencies (exceedance probability of 10^{-6} per run).

Benchmarks	**pWCET empty**	**pWCET reuse**	**Gain**
cnt	8,001	7,575	5.3%
crc	56,981	56,389	1.0%
qsort	2,850	2,572	9.8%
select	10,414	9,250	11.2%
Benchmarks	**pWCET convolution**	**pWCET dependent**	**Diff**
cnt+crc	66,963	71,804	6.7%
qsort+select	17,107	20,127	15.0%



(a) ETs and pWCET. (b) Joint ET and pWCET convolution. (c) ETs and pWCET with cache reuse and empty caches for `compress`. (d) ETs and pWCET with cache reuse and empty caches for `insertsort`. (e) ETs for `compress+insertsort`, and their pWCET.

Fig. 4: Examples for the different ETs (dashed lines) and pWCET (straight lines) scenarios for `compress` and `insertsort` benchmarks. Plots (a) and (b) correspond to independent ETs and pWCET, (c) and (d) to dependent ETs and independent pWCET, and (e) to dependent ETs and pWCET.

Results for this case are shown in Figure 4e. The blue straight line corresponds to the pWCET distribution obtained assuming independent measurements across benchmarks. Notably, besides being arguably unreliable, such pWCET distribution does not even upper-bound the joint ET of both benchmarks. As shown, the ECCDF of the addition of the ETs of both benchmarks clearly crosses the pWCET distribution derived under the independence assumption.

Instead, if we account for dependencies and compute the pWCET estimate of the (informed) addition of the measurements of both tasks, then we obtain a reliable pWCET distribution, shown with a black straight line in the plot. As explained before, precise information about dependencies may not be available, and it may be convenient resorting to deterministic WCET estimates and scheduling practices. For that purpose, we have analyzed the behavior at a given exceedance probability (10^{-6} per run), although our conclusions hold for virtually any exceedance threshold. Individual pWCET estimates at this exceedance probability are 10,216 and 6,989 cycles for `compress` and `insertsort` individually, thus somewhere in between both pWCET distributions (17,205 cycles in total). Hence, since deterministic pWCET estimates can be argued to be reliable by construction by removing any influence of dependencies, we regard the pWCET estimate assuming independence as potentially unreliable. Thus, scheduling must resort to deterministic WCET estimates and probabilistic schedulability cannot be applied in this scenario.

A similar analysis has been applied on some other program pairs from the Mälardalen benchmarks (see bottom part of Table II), showing that, as expected, the discrepancies between reliable pWCET estimates (accounting for dependencies) and unreliable ones (obtained with convolutions and ignoring dependencies) can be significant, up to 15% for the pair `qsort+select`.

VI. Conclusions

Probabilistic scheduling is increasingly considered as a way to reconcile real-time schedulability requirements with increasingly pessimistic and rare (extremal) WCET bounds, as those typically derived on modern, complex systems. In this scenario, schedulability is assessed on pWCET distributions rather than on single monolithic WCET values. To combine the effect of multiple tasks, most approaches rely on the convolution operator, which in turn requires its operands to be probabilistically independent. The common perception is that pWCET distributions are independent when obtained with SPTA and dependent if obtained with MBPTA, thus suggesting that only SPTA results can be used for scheduling purposes.

In this paper we provide arguments and quantitative evidence, both on synthetic examples and on concrete benchmarks, disproving common wisdom in several directions, concluding that: (i) MBPTA can deliver independent pWCET distributions compatible with probabilistic scheduling requirements; (ii) in some cases ET dependencies can be removed in the pWCET estimation process; (iii) for some type of dependence, both MBPTA ***and*** SPTA may fail to deliver independent pWCET distributions. Yet, even in these cases, the outcomes of probabilistic timing analysis are still usable: single pWCET bounds selected at specific exceedance thresholds can be used in deterministic scheduling analysis without further constraints.

Acknowledgements

This work has been partially supported by the Spanish Ministry of Economy and Competitiveness (MINECO) under grant TIN2015-65316-P, the European Research Council (ERC) under the European Union's Horizon 2020 research and innovation programme (grant agreement No. 772773) and the HiPEAC Network of Excellence. Jaume Abella and Enrico Mezzetti have been partially supported by MINECO under Ramon y Cajal and Juan de la Cierva-Incorporación postdoctoral fellowships number RYC-2013-14717 and IJCI-2016-27396 respectively.

References

[1] J. Abella et al., "WCET analysis methods: Pitfalls and challenges on their trustworthiness," in *SIES*, 2015.
[2] J. P. Lehoczky, "Real-time queueing theory," in *RTSS*, 1996.
[3] J. Diaz et al., "Stochastic analysis of periodic real-time systems," in *RTSS*, 2002.
[4] H. Zhu et al., "Optimal partitioning for quantized EDF scheduling," in *RTSS*, 2002.
[5] A. Masrur, "A probabilistic scheduling framework for mixed-criticality systems," in *DAC*, 2016.

[6] L. Kosmidis et al., "Fitting processor architectures for measurement-based probabilistic timing analysis," *Microprocessors and Microsystems*, vol. 47, pp. 287 – 302, 2016.

[7] E. Mezzetti et al., "Work-in-progress paper: An analysis of the impact of dependencies on probabilistic timing analysis and task scheduling," in *RTSS*, 2017.

[8] C. Liu and J. Layland, "Scheduling algorithms for multiprogramming in a hard-real-time environment," *J. ACM*, vol. 20, no. 1, pp. 46–61, 1973.

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

[10] N. Audsley et al., "Applying new scheduling theory to static priority pre-emptive scheduling," *Software Engineering Journal*, vol. 8, pp. 284–292(8), September 1993.

[11] T. S. Tia et al., "Probabilistic performance guarantee for real-time tasks with varying computation times," in *RTAS*, 1995.

[12] D. Maxim and L. Cucu-Grosjean, "Response time analysis for fixed-priority tasks with multiple probabilistic parameters," in *RTSS*, 2013.

[13] G. Bernat and M. Newby, "Probabilistic WCET analysis, an approach using copulas," *Journal of Embedded Computing*, 2006.

[14] L. Santinelli et al., "On the sustainability of the extreme value theory for WCET estimation," in *WCET Workshop*, 2014.

[15] G. Bernat, A. Colin, and S. Petters, "WCET analysis of probabilistic hard real-time systems," in *RTSS*, 2002.

[16] F. Cazorla et al., "PROARTIS: probabilistically analyzable real-time systems," *ACM Trans. Embedded Comput. Syst.*, vol. 12, no. 2s, 2013.

[17] S. Altmeyer and R. I. Davis, "On the correctness, optimality and precision of static probabilistic timing analysis," in *DATE*, 2014.

[18] L. David and I. Puaut, "Static determination of probabilistic execution times," in *ECRTS*, 2004.

[19] S. Kotz and S. Nadarajah, *Extreme value distributions: theory and applications*. World Scientific, 2000.

[20] S. Coles, *An Introduction to Statistical Modeling of Extreme Values*. Springer, 2001.

[21] R. Fisher and L. Tippett, "Limiting forms of the frequency distribution of the largest or smallest member of a sample," *Mathematical Proceedings of the Cambridge Philosophical Society*, vol. 24, no. 2, 1928.

[22] J. Pickands, "Statistical inference using extreme order statistics," *The Annals of Statistics*, vol. 3, no. 1, pp. 119–131, 1975.

[23] L. Cucu-Grosjean et al., "Measurement-based probabilistic timing analysis for multi-path programs," in *ECRTS*, 2012.

[24] J. Abella et al., "Measurement-based worst-case execution time estimation using the coefficient of variation," *ACM Trans. Des. Autom. Electron. Syst.*, vol. 22, no. 4, pp. 72:1–72:29, Jun. 2017. [Online]. Available: http://doi.acm.org/10.1145/3065924

[25] L. Santinelli et al., "Revising measurement-based probabilistic timing analysis," in *RTAS*, 2017.

[26] S. Milutinovic et al., "Modelling probabilistic cache representativeness in the presence of arbitrary access patterns," in *ISORC*, 2016.

[27] C. Maxim et al., "Reproducibility and representativity - mandatory properties for the compositionality of measurement-based WCET estimation approaches," in *CRTS Workshop*, 2016.

[28] R. Davis et al., "On the meaning of pwcet distributions and their use in schedulability analysis," in *Real-Time Scheduling Open Problems Seminar*, June 2017.

[29] B. Lesage et al., "Static probabilistic timing analysis for multi-path programs," in *RTSS*, 2015.

[30] L. Cucu-Grosjean, "Independence - a misunderstood property of and for probabilistic real--time systems," in *Alan Burns, in occasion of his 60th Anniversary*, 2013. [Online]. Available: https://hal.inria.fr/hal-00920504

[31] L. Santinelli and L. George, "Probabilities and mixed-criticalities: the probabilistic C-space," in *Workshop on Mixed Criticality Systems*, 2015.

[32] G. Box and D. Pierce, "Distribution of residual autocorrelations in autoregressive-integrated moving average time series models," *Journal of the American Statistical Association*, vol. 65, no. 332, pp. 1509–1526, 1970.

[33] F. Wartel et al., "Measurement-based probabilistic timing analysis: Lessons from an integrated-modular avionics case study," in *SIES*, 2013.

[34] ——, "Timing analysis of an avionics case study on complex hardware/software platforms," in *DATE*, 2015.

[35] J. Gustafsson et al., "The Mälardalen WCET benchmarks-past, present and future," in *WCET Workshop*, 2010.

[36] SoCLib, "-," 2003-2012, http://www.soclib.fr/trac/dev.

[37] J. Wetzel, E. Silha, C. May, B. Frey, J. Furukawa, and G. Frazier, *PowerPC User Instruction Set Architecture*, IBM Corporation, 2005.

[38] L. Kosmidis et al., "A cache design for probabilistically analysable real-time systems," in *DATE*, 2013.

[39] ARINC, *Specification 653: Avionics Application Standard Software Interface*, Aeronautical Radio, Inc, 1996.