Leveraging mixed criticality task budgets

Mohamed Amine Khelassi, Yasmina Abdeddaïm

Leveraging mixed criticality task budgets

Mohamed Amine Khelassi
Univ Gustave Eiffel, CNRS, LIGM
F-77454 Marne-la-Vallée, France
mohamedamine.khelassi@esiee.fr

Yasmina Abdeddaïm
Univ Gustave Eiffel, CNRS, LIGM
F-77454 Marne-la-Vallée, France
yasmina.abdeddaim@esiee.fr

***Abstract*—In mixed-criticality systems, classical models assume an estimated execution time budget for tasks in each possible criticality, and when a job presents a budget overrun, they tend to drop the lower criticality tasks to preserve the schedulability of the higher criticality tasks. However, few studies focus on how the different execution time budgets are calculated, and it has been emphasized that tasks of lower criticality must provide a minimum service for the system to function properly. In this paper, we present a greedy heuristic for calculating the execution time budget to be allocated to real-time tasks according to their criticality and the shape of their execution time distribution. This budget is calculated with the aim of minimizing potential budget overruns. We first introduce a new statistical dispersion parameter called the maximum coefficient of variation (VWCET). This parameter describes the tendency of execution times towards the worst-case execution time and can be adjusted according to the criticality of the task. We show through experiments that the proposed heuristic reduces the probability of exceeding the time budget allocated to the task, and improves the performance of low-criticality tasks.**

***Index Terms*—Mixed criticality systems, Execution time variability, Statistical dispersion parameter**

I. Introduction

With the growing complexity of processor architectures, one of the issues in real-time systems is ensuring the time predictability of a critical system. Indeed, program execution times can vary significantly from one execution to another in complex processors, this execution time variability is even greater for complex algorithms used in AI applications.

The worst-case execution time is necessary to prove that real-time systems meet their temporal constraints in their worst-case behavior. The greater the execution time variability, the more difficult it is to estimate the worst-case program execution time. And if the estimated worst-case execution time is pessimistic, the processor is not used efficiently. To overcome this issue, the mixed-criticality real-time model is one of the possible approaches.

The challenge is that low criticality tasks do not disturb the good functioning of the high criticality ones. In real-time scheduling, since the original Vestal's model [1], a classical model has emerged, see [2] for a complete survey. In this model, tasks have several execution times budgets, one budget per possible criticality. If a task does not signal its termination after the execution of its allocated budget at a certain criticality level, the system moves to the next criticality level. In every system criticality level, only tasks of criticality equal to or higher than the criticality of the system have to respect their deadlines. Therefore, low criticality tasks can be suspended to allow higher criticality tasks to meet their deadlines. Many research assumes that execution time budgets in every criticality level are provided and focus on the challenge of maximizing the execution of low criticality tasks while guaranteeing that all high criticality tasks always meet their deadlines. The execution time budgets are often defined as estimates of the worst-case execution time at different certification requirements but few studies have focused on how to determine these budgets.

In this paper, our aim is to determine execution time budgets for mixed criticality real-time tasks in order to minimize execution time budget overruns and to ensure that tasks meet their deadlines. If a job does not finish within its allocated budget, we consider that the job is suspended. The idea is to reduce monitoring as much as possible during execution, and therefore to find static execution time budgets with the least number of time overruns during execution while guaranteeing the correct functioning of the most critical tasks and a minimum quality of service for less critical tasks. Our budget calculation algorithm is not specific to a class of scheduling algorithms and has linear complexity in terms of the number of schedulability tests whatever the number of criticalities.

The computed budgets depend on the task's criticality, i.e. the higher the task's criticality, the lower the risk of budget overruns within the execution time. For example, if a task has a high criticality, we allocate it the highest possible budget with zero probability of overrun and it will never be suspended.

In order to minimize execution time budget overrun of a task, we use the variability of its execution time. For this purpose, we propose a new parameter for the quantification of execution time variability called the coefficient of variation to the maximum (VWCET). This parameter is used in a heuristic of linear time complexity to determine the task budgets.

The contributions of the paper are: (1) We propose a new statistical dispersion parameter to quantify the execution time variability of real-time tasks. (2) We propose a heuristic that uses our statistical dispersion parameter to compute the tasks execution time budgets to assign in a mixed criticality system. (3) We evaluate our approach using simulations and discuss its benefits according to different criteria. (4) We evaluate our approach on benchmarks executed on an ARM-Cortex A53 using a real-time OS.

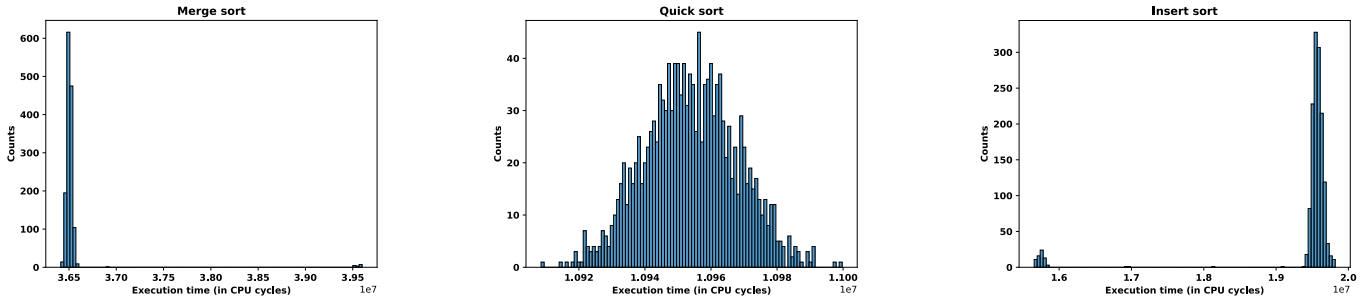


Fig. 1: Execution times (cycles) of programs executed in an ARM Cortex 53

II. Execution time variability

The quantification of time variability will help us in setting the appropriate execution time budgets for mixed criticality tasks. Fig. 1 represents histograms of execution times of three programs, merge, quick and insert sort of the Mälardalen [3] benchmark executed using the Zynq UltraScale+ MPSoC ZCU104 Evaluation Kit. For every algorithm we use the same input, and all the algorithms are executed 1000 times in the same processor configuration i.e. the caches are disabled and all other A53 cores execute the same sorting programs as contenders. We can see that the execution times of every program vary from one execution to another, we say that the programs exhibit some execution time variability.

Many definitions have been proposed for the quantification of execution time variability, in [4] and [5] the execution time variability is quantified as the ratio between run-time measurements and either the best, worst or mean execution time. In [6], [7], the execution time variability is considered to be the factor between the worst-case execution time computed in isolation and along with other workload. Let $Q_{merge} = 0.898$, $Q_{quick} = 0.989$ and $Q_{insert} = 0.787$ be the quotients between the best-case and worst-case execution times for merge sort, quick sort and insert sort programs respectively. The quotient indicates by how much execution times can vary at most, but we have no information on the distribution of execution times. Indeed the execution times of Fig. 1 are left-biased, centered or right-biased and this is the information we want to capture.

A. Dispersion parameters

Dispersion parameters are used in statistics to measure the tendency of the values of a distribution to be scattered on either side of a value. For example the coefficient of variation (CV) measures the dispersion of a set of data around the mean (similarly to the index of dispersion [8]).

The skewness parameter (skw) [9] describes which side of the distribution has a longer tail. For unimodal distributions, symmetric distributions should have a skewness value near zero, negative skewness values means that data are more shifted towards the maximum value, positive skewness values means that data are more shifted towards the minimum value of the distribution [10]. Because in real-time systems we are interested in the behavior around the worst-case value, the advantage of the skewness parameter compared to the coefficient of variation is that the parameter provides information concerning the distribution of the data towards the largest value, however it is not sensitive to location and its interpretation is valid for only unimodal distributions.

Inspired by the parameters mentioned, we suggested our own parameter that is sensitive to location (see Section II-B for an explanation of why we need this property) and can be adjusted according to the criticality of the task.

B. Coefficient of variation to the maximum

In this work, we propose our own dispersion parameter and in Section III we will show how it can be used in budget assignment for mixed criticality real-time tasks. A previous version of this parameter is presented in [11], this version does not consider the criticality of the task. Definition 1, introduces our proposed dispersion parameter, the coefficient of variation to the maximum, this parameter is inspired by the coefficient of variation and measures the dispersion of a set of data around the largest value instead of the mean value.

Definition 1 (Coefficient of variation to the maximum): The coefficient of variation to the maximum $VWCET_k^\alpha$ of X^k a random positive integer variable with n_k occurrences $\{X_1^k, \dots X_{n_k}^k\}$ with $\exists X_j^k, X_j^k > 0$ is:

$$VWCET_k^\alpha = \frac{\frac{\sum_{i=1}^{n_k}(WCET_k - X_i^k)^{\frac{1}{\alpha}}}{n_k}}{WCET_k} \times 100 \tag{1}$$

with $WCET_k = \max_{1 \le i \le n_k} X_i^k$ and α is the parameter of variability with $\alpha > 0$.

Given that the VWCET [1] is based on the sum of deviations, $(WCET_k - X_i^k)^{\frac{1}{\alpha}}$, between the WCET and the other data values, the smaller it is the more likely the data are shifted to the WCET. For example, as stated in Proposition 1, when the parameter of variability α is set to 1, if the VWCET of a random variable X^k is greater than that of X^j, it implies that the normalised mean value of X^k by its $WCET_k$ is smaller than the normalised mean value of X^j by its $WCET_j$.

Proposition 1: Let X^k be a random variable with n_k occurrences and X^j a random variable with n_j occurrences, if $VWCET_k^1 > VWCET_j^1$ then $\frac{\mu_k}{WCET_k} < \frac{\mu_j}{WCET_j}$

[1]The parameters, k and α are omitted in $VWCET_k^\alpha$ when there is no confusion

where μ_k and μ_j are the mean of the random variables X^k and X^j respectively.

Proof 1: Suppose $VWCET_k^1 > VWCET_j^1$, then

$$\frac{\frac{\sum_{i=1}^{n_k}(WCET_k - X_i^k)}{n_k}}{WCET_k} \times 100 > \frac{\frac{\sum_{i=1}^{n_j}(WCET_j - X_i^j)}{n_j}}{WCET_j} \times 100$$

$$\frac{n_k - \sum_{i=1}^{n_k}(\frac{X_i^k}{WCET_k})}{n_k} > \frac{n_j - \sum_{i=1}^{n_j}(\frac{X_i^j}{WCET_j})}{n_j}$$

$$\frac{\frac{1}{n_k}\sum_{i=1}^{n_i}(X_i^k)}{WCET_k} < \frac{\frac{1}{n_j}\sum_{i=1}^{n_j}(X_i^j)}{WCET_j}$$

The parameter α is used in the VWCET formula to control the sensitivity of the metric to variations towards the WCET, this is why we power the variation towards the WCET by $\frac{1}{\alpha}$.

First if $\alpha \geq 1$, the term $(WCET_k - X_i^k)^{\frac{1}{\alpha}}$ becomes less sensitive to variations by growing more slowly then if $\alpha < 1$ as shown if Formula 2.

$$\frac{\partial}{\partial dev}\frac{\partial}{\partial dev}(dev^{\frac{1}{\alpha}}) = -\frac{(\alpha-1)dev^{(\frac{1}{\alpha}-2)}}{\alpha^2} \quad (2)$$

with $dev = (WCET_k - X_i^k)$

In fact, as we can see in the Formula 2 the growth of the variation $(WCET_k - X_i^k)$ decreases (the second derivative is negative) if $\alpha > 1$ and increases (the second derivative is positive) if $0 < \alpha < 1$. Note that multiplying $(WCET_k - X_i^k)$ by α would have given us constant growth of $(WCET_k - X_i^k)$ equal to α thus less sensitivity to the deviation.

Secondly, Proposition 2 shows that the larger the α, the smaller the VWCET.

Proposition 2: Let X^k be a random variable with n_k occurrences. Let α_1 and α_2 be two parameters of variability, if $\alpha_1 < \alpha_2$ then $VWCET_k^{\alpha_1} > VWCET_k^{\alpha_2}$.

Proof 2: Suppose that $0 < \alpha_1 < \alpha_2$, then $\frac{\sum_{i=1}^{n_k}(WCET_k - X_i^k)^{\frac{1}{\alpha_1}}}{n_k} > \frac{\sum_{i=1}^{n_k}(WCET_k - X_i^k)^{\frac{1}{\alpha_2}}}{n_k}$

this implies

$$\frac{\frac{\sum_{i=1}^{n_k}(WCET_k - X_i^k)^{\frac{1}{\alpha_1}}}{n_k}}{WCET_k} \times 100 > \frac{\frac{\sum_{i=1}^{n_k}(WCET_k - X_i^k)^{\frac{1}{\alpha_2}}}{n_k}}{WCET_k} \times 100$$

because as X_i^k and $WCET_k \in \mathbb{N}_{\geq 0}$, $(WCET_k - X_i^k)$

This sensitivity to α means that we can adjust the α parameter in the VWCET to accentuate or diminish the weight of the fact that the data are close to the WCET, we will use this property of the VWCET to adapt the assignment of execution time budgets to the criticality of the task.

Finally, using Proposition 3, we can state that if two random variable distributions have the same shape, but the first distribution has larger values than the second one, the VWCET of the distribution with the largest value is the smallest.

Proposition 3: The VWCET coefficient is sensitive to location. If X^k and X^j are two random variables with n occurrences, with $X^k = X^j + c$ where c is a non null constant, then $VWCET_k^\alpha < VWCET_j^\alpha$

Proof 3: Let X^k and X^j be two random variables with n occurrences, with $X^k = X^j + c$ where c is a non null constant, we have

$$VWCET_k^\alpha = \frac{\frac{\sum_{i=1}^{n}((WCET_j + c) - (X_i^j + c))^{\frac{1}{\alpha}}}{n}}{WCET_j}.100.\frac{WCET_j}{WCET_j + c}$$

$$= VWCET_j^\alpha.\frac{WCET_j}{WCET_j + c}$$

We deduce that $VWCET_j^\alpha = (1 + \frac{c}{WCET_j})VWCET_k^\alpha$

Proposition 3 means that the distance with the WCET counts more for high-value distributions. The VWCET coefficient incorporates therefore the notion of quotient in addition to the dispersion of a set of data around the largest value. For example, the distance between 1 and 5 is the same as between 10001 and 10005, but the quotients between the two are different (0.2 and 0.99), which is reflected in the fact that the VWCET of 1 and 5 is larger than the one of 10001 and 10005.

C. Execution time variability parameter

Let τ_i be a real-time task with C_i a positive integer random variable that takes its values in the set of execution times of τ_i. We make no assumptions on the distribution of C_i. The way this distribution is computed is out of the scope of this work, but it can be generated for example using experiments and/or using estimates of execution times for example in the worst and/or in the best case. We define the execution time variability of a real-time task as a statistical dispersion parameter.

Definition 2 (Execution time variability): The execution time variability of τ_i is defined by TV_i a statistical dispersion parameter of C_i the execution time random variable of τ_i.

To illustrate, we compute this parameter for every program of Fig. 1 in three different cases where the statistical dispersion parameter is $VWCET^1(\alpha = 1)$ or $VWCET^{10}(\alpha = 10)$ or sKw. We see in Table I that the sKw value is close to zero for quick sort, negative for insert sort and positive for merge sort. This reflects the shape of the distributions in Fig. 1. The $VWCET$ value of merge sort is the largest, followed by insert sort and then quick sort. This order is the same for $\alpha = 1$ and $\alpha = 10$. This means that according to the VWCET parameter, the merge sort distribution has the least data around the WCET, which is the same conclusion as for sKw. However, VWCET considers that the insert sort distribution has less data around the WCET than quick sort, this is the opposite of the sKw result and of the histogram shapes of the distributions. This is due to the fact that, our VWCET parameter also takes into account the notion of quotient between BCET and WCET. The quotient for quick sort is 0.989, while that for insert sort is 0.787, which is why the VWCET is smaller for quick sort. We can also see that as the α increases, the VWCET decreases, as stated in Proposition 2. This makes the VWCET with $\alpha = 10$ of merge sort smaller than the VWCET of insert sort with $\alpha = 1$. We will see why this is relevant for mixed criticality systems in Section III.

III. Mixed criticality budget assignment

In this section, we exploit execution time variability in the budget assignment of mixed criticality real-time tasks. Our aim is to find the execution time budgets to assign to tasks

TABLE I: TV parameter of Merge, Quick and Insert sort

	Merge sort	Quick sort	Insert sort
$VWCET^1$	7.72	0.41	2.15
$VWCET^{10}$	0.77	0.04	0.21
sKw	9.493	0.029	-4.173

depending on their criticality with the goal of minimizing the budget overrun while preserving the schedulability. Jobs are stopped if they exceed the allocated time budget.

A. Problem statement

Let Γ be a task set of n mixed criticality independent real-time tasks executed using a scheduling algorithm $Sched$. Every task is assigned a criticality by the system designer among nL possible criticality $\{L_1, \ldots, L_{nL}\}$ with L_j more critical than L_{j+1}. We do not specify whether the execution platform is uniprocessor or multiprocessor, however the scheduling algorithm must be sustainable with respect to task execution times.

Each task $\tau_i \in \Gamma$ is a tuple $(TV_i, Budget_i, L_i, D_i, T_i)$ with:

- TV_i: is the execution time variability parameter of τ_i.
- $Budget_i$: is a totally ordered set of possible budgets to be assigned to τ_i with $p(b_i)$ is the probability that the budget b_i is not exceeded.
- $L_i \in \{L_1, \ldots, L_{nL}\}$: is the criticality of τ_i.
- $D_i \in \mathbb{N}^*$: is the relative deadline of τ_i.
- $T_i \in \mathbb{N}^*$: is the task's minimum inter-arrival time.

The system designer defines the possible budgets of a task and specifies their level of confidence, this latter is expressed in the task model by the probability, $p_i(b)$, that a budget b will not be exceeded. Each set of budgets of a task τ_i contains m possible budgets noted $\{b_{1,i}, \ldots b_{m,i}\}$. These budgets are arranged in a decreasing order with $b_{1,i} = WCET_i$ is the worst-case execution time of task τ_i with $p(b_{1,i}) = 1$. If for a given criticality the designer does not tolerate any budget overruns, the designer assigns to tasks of that criticality a single budget with the highest confidence. However, in this paper, for clarity of presentation, we assume without loss of generality, that all tasks have the same number of budgets.

Given a task set Γ of n tasks, a budget assignment for Γ is a vector $B = (B_1, B_2, \ldots, B_n)$ with $\forall i \in 1 \ldots n, B_i \in Budget_i$ is the execution time budget assigned to τ_i.

We define Γ^B as the task set where every task τ_i^B is defined by $(TV_i, B_i, L_i, D_i, T_i)$ with TV_i, B_i, L_i, D_i and T_i are the execution time variability parameter, the execution time, the criticality, the deadline and the period of τ_i respectively.

If the probability $p_i(B_i)$ that the budget B_i assigned to τ_i is respected is equal to 1 we say that the task will not experience a budget overrun when the budget B_i is assigned to τ_i.

The score of the task set Γ^B of a budget assignment $B = (B_1, B_2, \ldots, B_n)$ computes the mean probability of no budget overruns for the tasks in Γ_B and is defined as

$$Score(\Gamma^B) = \frac{\sum_{i \in \{1 \ldots n\}} p(B_i)}{n} \tag{3}$$

The score of criticality L of Γ^B is defined as

$$Score^L(\Gamma^B) = \frac{\sum_{i \in \{1, \ldots n\},\ L_i = L} p(B_i)}{n} \tag{4}$$

Given a task set Γ of n mixed criticality real-time tasks, the task set Γ is schedulable w.r.t. the budget assignment $B = (B_1, \ldots, B_n)$ and the scheduling algorithm $Sched$ if and only if the task set Γ^B is schedulable according to $Sched$.

Definition 3 (Budget optimal assignment): Given a task set Γ of n mixed criticality real-time tasks, the budget assignment $B = (B_1, \ldots, B_n)$ is an optimal assignment for the scheduling algorithm $Sched$ if and only if:

1) The task set Γ^B is schedulable w.r.t. the budget assignment $B = (B_1, \ldots, B_n)$
2) $\forall$ budget assignment $B' = (B'_1, \ldots, B'_n)$ with $\Gamma^{B'}$ schedulable w.r.t. B', $Score(\Gamma^{B'}) \leq Score(\Gamma^B)$

Definition 3 states that a budget assignment is optimal if it maximizes the mean probability of no budget overrun.

Definition 4 (Budget optimal per criticality assignment): Given a task set Γ of n mixed criticality real-time tasks, the budget assignment $B = (B_1, \ldots, B_n)$ is budget optimal per criticality for the scheduling algorithm $Sched$ if and only if:

1) The task set Γ^B is schedulable w.r.t. the budget assignment $B = (B_1, \ldots, B_n)$
2) $\forall$ budget assignment $B' = (B'_1, \ldots, B'_n)$ with $\Gamma^{B'}$ schedulable w.r.t. B', $\forall L \in \{L_1, \ldots, L_{nL}\}$ $Score^L(\Gamma^{B'}) \leq Score^L(\Gamma^B)$

Definition 4 states that a budget assignment is budget optimal per criticality if the assignment maximizes the mean probability of no budget overrun per criticality.

Definition 5 (Stopped tasks optimal assignment): Given a task set Γ of n mixed criticality real-time tasks, the budget assignment $B = (B_1, \ldots, B_n)$ is a stopped tasks optimal assignment for the algorithm $Sched$ if and only if:

1) The task set Γ^B is schedulable w.r.t. the budget assignment $B = (B_1, \ldots, B_n)$
2) $\forall$ budget assignment $B' = (B'_1, \ldots, B'_n)$ with $\Gamma^{B'}$ schedulable w.r.t. B', $|\{\tau_i, i \in 1, \ldots, n$ with $p(B_i) < 1\}| \leq |\{\tau_i, i \in 1, \ldots, n$ with $p(B'_i) < 1\}|$

Definition 5 states that a budget assignment is stopped tasks optimal if the assignment minimizes the total number of tasks that can be interrupted. In the same way, we can define the stopped tasks per criticality optimal assignment.

Finding a budget assignment that respects all the properties is very complex, as the problem to solve is a multi-objective problem. Even solving only the budget (or stopped) optimal assignment problem without considering the criticality of the tasks is complex as all possible assignments have to be tested and the schedulability test of algorithm Sched has to be executed $O(m^n)$ times. For this reason, we propose a greedy heuristic, Algorithm 1, where the schedulability test is executed in the worst-case $O(mn)$ times.

B. General budget assignment heuristic

The main property of the algorithm is that it has a reduced complexity while aiming to reduce the mean execution time

Algorithm 1: General budget assignment heuristic

Input : $\Gamma = \{\tau_i, i \in 1..n\}$, ∇
Output: system is not schedulable or $B = (B_1, \dots B_n)$

1 $\forall \tau_i$, $B_i = b_{m,i}$;
2 **if** Γ^B *not schedulable using algorithm Sched* **then**
3 | return system is not schedulable
4 **end**
5 $Set = \Gamma$
6 $\forall \tau_i$, $B_i = b_{1,i}$;
7 **while** Γ^B *not schedulable using algorithm Sched* **do**
8 | i is the index of τ_i with $\forall \tau_j \in Set, \nabla(\tau_i) \prec \nabla(\tau_j)$
9 | **for** $r \in 2..m$ **do**
10 | | $B_i = b_{r,i}$,
11 | | **if** Γ^B *is schedulable using algorithm Sched* **then**
12 | | | return $B = (B_1, \dots B_n)$;
13 | | **end**
14 | **end**
15 | τ_i is removed from Set;
16 **end**
17 return $B = (B_1, \dots B_n)$;

overrun probability and the number of possible stopped jobs while preserving schedulability and taking into account the criticality. For this, the algorithm reduces task budgets according to a given order. It takes a task set Γ and an order function ∇ with $\nabla(\tau_i) \prec \nabla(\tau_j)$ means that τ_i is ordered before τ_j. First, (lines 1-4) if the task set with the minimal assignment budget is not schedulable then there is no solution. Then, (line 8) a task is chosen according to its order, and its budget is reduced until a solution is found or it is no longer possible to reduce its budget (lines 9-14). If the system is still not schedulable, we proceed to the next task. The fact that a task is reduced as much as possible before moving to another task is motivated by the property of Definition 5. This avoids reducing the budget for several tasks, when it may be possible to reduce the budget for fewer tasks. In this algorithm every task is visited at most once, and all the budgets of all the tasks are at most tested in a schedulability test at most one time. For this reason, the scheduling test is executed at the worst-case $O(mn)$ times, which is a low and linear complexity compared to the optimal algorithm, which costs $O(m^n)$.

However, we still don't specify how to choose the order function ∇. An algorithm that finds the optimal order must find an assignment where there is at most one task that has not been assigned its minimum or maximum budget. This is because if two tasks are not at their maximum or minimum budget, it means that the algorithm has reduced the budget of one task without having finished reducing the budget of the previous task, which is not possible. It follows that the number of schedulability tests to find the assignment with the optimal order function is $O(mn2^{n-1})$. This is due to the fact that for each task that does not have a maximum or minimum budget, we have to test m possible budgets for 2^{n-1} (the possible combinations of minimum and maximum budgets) cases, and we have to do this for each task.

Complexity decreases relative to $O(m^n)$ but remains high. Another drawback is that task criticality is not taken into account in this assignment method, as it does not check whether high criticality tasks have higher scores than lower criticality tasks. Therefore, we need to find a method to define the order ∇ according to a constant criterion, with the objective of maximizing the mean probability of no budget overruns and taking into account the criticality of the tasks.

C. Coefficient of variation to the maximum heuristic

In this section, we present the VWCET heuristic equivalent to Algorithm 1 with the function ∇ is defined by:

$$\begin{gathered}\nabla(\tau_i) \prec \nabla(\tau_j) \text{ iff } TV_i > TV_j \text{ with} \\ TV_i = VWCET_i^{\alpha_i} \text{ and } TV_j = VWCET_j^{\alpha_j}\end{gathered} \tag{5}$$

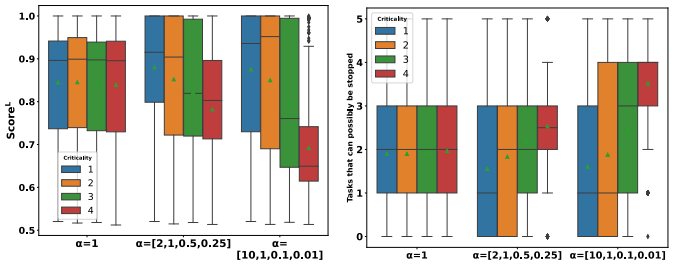
The idea is to start reducing the budget of tasks for which the distribution of execution times is the least shifted towards the WCET. By doing this, we aim to avoid as far as possible the tasks from not finishing within the budget allocated to them during execution. To do that, tasks are ordered in Algorithm 1 according to their time variability parameter where the time variability parameter is the VWCET. As explained in Section II-B, the larger the VWCET, the closer the execution times are to the WCET. For example, if α=1, we showed in Proposition 1 that the smaller the VWCET, the closer the data is to the WCET on average. However, we want to accept fewer overruns, for more critical tasks, even if a critical task is shifted less to the WCET. To account for task criticality, we assign the variability parameter α using the following rule

$$\forall \tau_i, \tau_j \in \Gamma \text{ if } L_i < L_j \text{ then } \alpha_i > \alpha_j \tag{6}$$

The idea of this rule comes from Proposition 2, which states that the larger the α, the smaller the VWCET. Thus, to be able to decrease the budget of a task before a more critical one, we assign it a smaller α. The VWCET of the less critical task can therefore be greater, and the task can be moved up in the order of Algorithm 1.

IV. Simulation-based evaluation

We present simulation-based experiments using uniprocessor Earliest Deadline First (EDF) algorithm. We generate tasks sets using unimodal and bimodal distributions of execution time. For every type of distribution, we generate 1000 task sets composed of 20 tasks. For each task set we randomly generate a maximal utilization ranging from 1 to 1.4 using [12]. The minimal random task utilizations were derived by subtracting a percentage ranging between 5% and 60% from the maximum utilizations. We generate periods using a uniform distribution in the range $[100, 502]$ and a deadline using a uniform distribution in the range $[T_i/2, T_i]$. We discard task sets with $\sum_{i=1\dots n} \frac{BCET_i}{T_i} > 1$, and task sets which do not produce a solution with at least one algorithm in the list of the algorithms



(a) Boxplot of the VWCET scores (b) Possibly stopped tasks

Fig. 2: VWCET heuristic results per criticality for different α

that we compare. For each task τ_i, the $Budget_i$ is defined as $\{WCET_i, 97th, 95th, 90th, 80th, 70th, 60th, 50th\}$ percentiles respectively, derived from the task's execution time distribution. For unimodal distributions we use a truncated normal distribution, with a mean as a uniform value ranging from BCET to the WCET of the task and a random standard deviation proportional to the span of execution times $(WCET - BCET)/(x)$ where x follows a uniform distribution in the range $[2, 40]$. For bimodal distributions we generate two unimodal distributions and concatenate them. Task criticalities ranging from 1 to 4 were randomly assigned tasks with 5 tasks per criticality in a task set.

A. Evaluation tests

We test the VWCET heuristic using the task sets with unimodal distributions of execution time in three cases: (1) VWCET0: with variability parameter $\alpha = 1$ for all tasks (2) VWCET1: with $\alpha_1 = 2, \alpha_2 = 1, \alpha_3 = 0.5, \alpha_4 = 0.25$ where α_i is the variability parameter for tasks of criticality L_i. (3) VWCET2: with $\alpha_1 = 10, \alpha_2 = 1, \alpha_3 = 0.1, \alpha_4 = 0.01$ where α_i is the variability parameter for tasks of criticality L_i.

In VWCET1 and VWCET2, we assign the parameter α s.t:

- $\alpha \geq 1$ for the most critical tasks to reduce the impact of distance from the WCET for the most critical tasks.
- The higher the criticality of the task, the higher the assigned parameter α as stated in Eq. 5.

We compare the VWCET heuristic with Algorithm 1 with different ∇ order functions: (1) Skewness: ordered by skewness in descending sequence. (2) Criticality: ordered by criticality in ascending sequence. For the tasks with the same criticality, tasks are ordered according to their VWCET with $\alpha = 1$. (3) Periods: ordered by periods in ascending sequence. (4) Deadlines: ordered by deadlines in ascending sequence. (5) Random: ordered randomly.

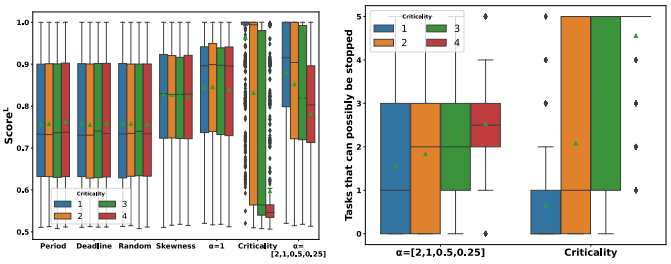
B. Results and discussion

1) Importance of α parameter: We assess how the VWCET heuristic performs under varying α parameter. The boxplot of Fig. 2a shows the distribution of the score of the 1000 task sets per criticality according to VWCET0, VWCET1 and VWCET2. We observe that the scores per criticality for $\alpha = 1$ across the 4 criticalities are fairly equal suggesting that no criticality is given preference over another. This is a predictable result, since VWCET0 starts by reducing the budget of tasks that are least shifted towards the WCET, regardless of the task's criticality.

For VWCET1 and VWCET2 we notice a pattern where the scores of the first criticality exceed those of the second criticality followed by the scores of the 3rd criticality followed at the end by the scores of the $4th$ criticality. We conclude that the higher the criticality of the task, the less it can exhibit a budget overrun. We have then established a criticality order score by adjusting the α parameter. We also observe that the mean probability of task overrun of VWCET1 per criticality level is better than the VWCET2 in terms of tightness of interquartile range suggesting less variability in scores especially for the $4th$ criticality tasks where we find that they have a 10% more probability of experiencing less budget overrun.

Fig. 2b illustrates the number of tasks that can possibly be stopped per criticality if they exhibit a budget overrun, which means their allocated budget is less than their WCET. We observe that the VWCET0 heuristic yields the same average number of possibly stopped tasks per criticality, we also notice a pattern of increasing potential number of tasks that can possibly be stopped as we decrease in criticality for VWCET1 and VWCET2, and that VWCET1 outperforms VWCET2 especially for criticality 3 and 4 with an average of one less task. When comparing VWCET2 and VWCET1 we notice that VWCET2 performed worse for low criticality tasks but better for high criticality ones. Based on the results, assigning different variability parameters to different criticalities establishes a hierarchy in task selection, thereby trying to optimize budget allocation per criticality. This approach enables us to prioritize tasks more effectively, emphasizing those with low criticality for budget reduction, thus enhancing overall budget efficiency per criticality. The system designer should assign larger $\alpha \geq 1$ to tasks of high criticality and lower $\alpha < 1$ for low criticality tasks. And depending on the scores that he wants to obtain for the highest criticality tasks w.r.t to the lowest ones, he can define the appropriate distance between the different α.

2) Relevance of the VWCET: We see in Fig. 3a that setting the order in Algorithm 1 using the period, the deadline or randomly leads to the lowest scores. Using a time variability parameter increases the scores with a slightly better score for VWCET0, similar to what we observed in the previous section, because criticalities are not taken into account. When using the criticality of the task as an order, the scores are more dispersed with a lower average score in comparison to the VWCET and skewness heuristics. We see that VWCET1 succeeds in establishing a criticality order with scores of the 1st criticality exceeding those of the 2nd followed by the 3rd and 4th criticality contrary to distribution-agnostic heuristics. Criticality order also produces the same trend between score results of criticalities, however we notice that its interquartile ranges are bigger especially for criticality 2 and 3. Criticality order starts with the 4th criticality tasks hence their 57% mean probability of no task budget overrun compared to 78% for $VWCET1$. In Fig. 3b we notice that on average the VWCET1 heuristic performs better than the criticality heuristic, the



(a) Boxplot of heuristic scores (b) Possibly stopped tasks

Fig. 3: Heuristic results per criticality with different ∇

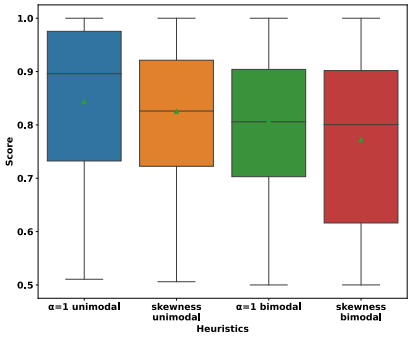


Fig. 4: Comparison w.r.t the type of distribution

results show that the VWCET1 heuristic tends to halt fewer tasks of the 2nd, 3rd and 4th criticality tasks with an average of 2 tasks for the $4th$ criticality and 1 task for the $3rd$ criticality.

On the basis of the results, we can deduce that it is relevant to use the VWCET statistical parameter. This parameter shows good results compared with other heuristics by taking into account the shape of the distribution of execution times, the quotient between the smallest and largest execution time and also the criticality of the task while using a single value.

3) Sensitivity to the type of distribution: We compare the sensitivity of variability parameters to distribution types. Thus we do not take criticality into account and compare sKw with VWCET0 to focus only on the type of distribution. We use unimodal and bimodal distributions. We see in Fig. 4 that for both distributions, VWCET0 performs better than the skewness heuristic, although there is a small decrease in the average performance for both heuristics under the bimodal distributions, we also notice that the interquartile range of the skewness heuristic under bimodal distributions increases compared to the unimodal distributions suggesting an increase in lower score results. Skewness performs worse for bimodal distributions due to the nature of the parameter whereas in unimodal distributions there is a clear mode value of the data, which is not the case for bimodal distributions with the presence of two distinct modes. We conclude that VWCET is less sensitive to the type of distribution than sKw, however further studies are needed on other types of distributions.

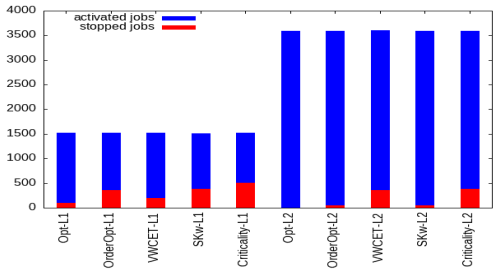


Fig. 5: Number of stopped jobs per algorithm and criticality

V. Benchmark-based evaluation

We perform experiments on the Zynq ZCU104 platform, featuring a quad-core Arm Cortex A53. We use FreeRTOS [13] with ESFree [14] as the scheduling library in order to incorporate timing features that are not proposed by FreeRTOS. We generate a set of execution times by executing 6 programs from TACLeBench [15] and Mälardalen [3] benchmarks under the following configuration: (1) We consider asymmetric multiprocessing with FreeRTOS. (2) We do not vary the inputs of the programs. (3) Caches are disabled. (4) Cores 1, 2 and 3 execute [16]. (5) Core 4 executes the 6 programs. (6) The scheduling algorithm is preemptive rate monotonic. A response time analysis [17] is used as schedulability test.

TABLE II: Program's parameters

Prog	epic	fft	iSort	ludcmp	matrix	sha
T	450	430	70	440	677	460
D	152	18	10	52	390	247
L	2	2	2	1	1	1

Table II gives for every program its deadline, period and criticality. For each program i, the budgets are $\{WCET_i, 97th, 95th, 90th, 80th, 70th, 60th, 50th\}$ percentiles respectively. Using the generated execution times we apply the VWCET heuristic with $\alpha_1 = 1, \alpha_2 = 0.5$ where α_i is the variability parameter for tasks of criticality L_i. We execute the 6 programs during 10 minutes. If a job exhibits an overrun, it is stopped. We count for every criticality L_i the number of stopped jobs of tasks. We compare this number to the number obtained using sKw and Criticality heuristics. We also compare to, Opt the optimal assignment algorithm that computes the assignment that maximizes the score and to OrderOpt the algorithm that computes the assignment with the optimal order in Algorithm 1. Opt and OrderOpt are not optimal w.r.t to the criticality, and their complexity is $O(m^n)$ and $O(mn2^{n-1})$ respectively, for these reasons we did not test them in the simulations-based evaluation where $m = 8$ and $n = 20$. We see in Fig. 5 that the VWCET heuristic stops less jobs for tasks of criticality L_1 than for criticality L_2 whereas all other algorithms do the opposite. In addition, it stops the least amount of jobs for tasks of criticality L_1 compared to the other heuristics, except for the Opt algorithm, VWCET stops 12% of the activated jobs of criticality L_1 compared to $33.6\%, 25.5\%, 23.9\%$ for Criticality-order, skewness, OrderOpt heuristics. Opt stopped 6% of the L_1 criticality tasks

but it does not respect the order of criticality, and is very complex, and suffers from a scalability problem.

VI. Related work

To our knowledge, our work is the first to propose the use of statistical parameters in real-time task models. Statistical dispersion parameters have been used in real-time systems in [18], [19] but only to compare different execution time distributions of different components of a system.

Decreasing the computation times of low criticality tasks when the system is in high criticality has been considered in [20]–[22]. In these papers, the budget of low criticality tasks is smaller when the system is in high criticality than their budget when the system is in low criticality. In our model, budgets are set offline and there are no mode changes during execution, the advantage of our approach is that the budgets have a practical significance and the only monitoring we need to perform during execution is to stop jobs if they do not terminate at their assigned budget. We also consider more than two criticalities, compared to the cited papers. Moreover, we are not concerned in this work by the scheduling strategy but by the execution time budget computation problem, but one can consider our computed budget as the execution time budget used in the normal execution mode and the overrun of the budget as a mode switch and use a classical mixed criticality scheduling algorithm. The difference between our approach and the approaches using probabilistic task models for mixed criticality systems [23]–[25] is first that our goal is not to compute the deadlines miss probabilities, but to compute the budgets to assign to tasks so that deadlines are always met. Moreover, we do not use the entire distribution, but only one parameter, this decreases the complexity. Finally, we make no hypothesis concerning the distribution of execution times.

The closest work is [26], their goal is to minimize the number of mode switches from low to high criticality while preserving EDF-VD [27] schedulability where low criticality tasks are stopped in a high criticality mode. They use the Chebyshev theorem to model the probability of mode switch and solve the problem using a genetic algorithm. In our approach, less critical tasks are stopped only if they do not respect their allocated budget as we consider static scheduling and we have more than two criticalities. Another difference, is that our approach is agnostic to the scheduling algorithm with the condition of sustainability w.r.t execution times.

VII. Conclusion

We introduce a statistical dispersion parameter to estimate the execution time variability of real-time tasks. We proposed a linear time complexity heuristic w.r.t the number of schedulability tests to compute the execution time budget to be allocated to tasks in a mixed criticality system. The goal is to minimize the number of execution time overruns. We showed that our approach provides good results in terms of number of possible overruns w.r.t the criticality of the task. We plan to apply the approach to mixed criticality algorithms where task budgets are adjusted during the execution.

References

[1] S. Vestal, "Preemptive scheduling of multi-criticality systems with varying degrees of execution time assurance," in *28th IEEE Real-Time Systems Symposium (RTSS 2007), Tucson, Arizona, USA*, pp. 239–243.

[2] A. Burns and R. I. Davis, "A survey of research into mixed criticality systems," *ACM Comput. Surv.*, vol. 50, no. 6, pp. 82:1–82:37, 2018.

[3] J. Gustafsson *et al.*, "The Mälardalen WCET benchmarks – past, present and future," B. Lisper, Ed. Belgium: OCG, Jul. 2010, pp. 137–147.

[4] C. J. Hughes *et al.*, "Variability in the execution of multimedia applications and implications for architecture," *SIGARCH Comput. Archit. News*, vol. 29, no. 2, p. 254–265, may 2001.

[5] W. Koch *et al.*, "Neuroflight: Next generation flight control firmware," *arXiv preprint arXiv:1901.06553*, 2019.

[6] V. Nélis *et al.*, "The Variability of Application Execution Times on a Multi-Core Platform," in *Workshop on Worst-Case Execution Time Analysis (WCET 2016)*, M. Schoeberl, Ed., vol. 55, Dagstuhl, Germany.

[7] J. Bin, "Controlling execution time variability using cots for safety-critical systems," Thesis, Université Paris Sud - Paris XI, Jul. 2014.

[8] D. R. Cox *et al.*, "The statistical analysis of series of events," 1966.

[9] P. v. Hippel, *Skewness*. Berlin, Heidelberg: Springer Berlin Heidelberg, 2011, pp. 1340–1342.

[10] National Institute of Standards and Technology, *Measures of Skewness and Kurtosis*.

[11] M. A. Khelassi *et al.*, "Execution time budget assignment for mixed criticality systems," in *10th International Workshop on Mixed Criticality Systems at the Real Time Systems Symposium (RTSS 2023)*, 2023.

[12] P. Emberson *et al.*, "Techniques for the synthesis of multiprocessor tasksets," in *1st International Workshop on Analysis Tools and Methodologies for Embedded and Real-time Systems (WATERS 2010)*, pp. 6–11.

[13] R. Barry *et al.*, "Freertos," *Internet, Oct*, vol. 4, 2008.

[14] R. Kase, "Efficient scheduling library for freertos," 2016. [Online]. Available: https://api.semanticscholar.org/CorpusID:64508990

[15] H. Falk *et al.*, "Taclebench: A benchmark collection to support worst-case execution time research," in *Workshop on Worst-Case Execution Time Analysis, (WCET), 2016*, M. Schoeberl, Ed., vol. 55, pp. 2:1–2:10.

[16] M. Bechtel *et al.*, "Denial-of-service attacks on shared cache in multi-core: Analysis and prevention," in *2019 IEEE Real-Time and Embedded Technology and Applications Symposium (RTAS)*, 2019.

[17] N. Audsley *et al.*, "Applying new scheduling theory to static priority pre-emptive scheduling," *Software engineering journal*, vol. 8, no. 5, pp. 284–292, 1993.

[18] M. Alcon *et al.*, "Timing of autonomous driving software: Problem analysis and prospects for future solutions," in *Real-Time and Embedded Technology and Applications Symposium (RTAS)*, 2020, pp. 267–280.

[19] F. Reghenzani *et al.*, "Timing predictability in high-performance computing with probabilistic real-time," *IEEE Access*, vol. 8, pp. 208 566–208 582, 2020.

[20] A. Burns *et al.*, "Towards a more practical model for mixed criticality systems," in *Proc. WMC, RTSS*, 2013, pp. 1–6.

[21] S. K. Baruah *et al.*, "Scheduling mixed-criticality systems to guarantee some service under all non-erroneous behaviors," in *Euromicro Conference on Real-Time Systems, (ECRTS) 2016, France*, pp. 131–138.

[22] X. Gu *et al.*, "Dynamic budget management with service guarantees for mixed-criticality systems," in *IEEE Real-Time Systems Symposium, (RTSS), Porto, Portugal, 2016*, pp. 47–56.

[23] J. Singh *et al.*, "Mixed criticality scheduling of probabilistic real-time systems," in *Dependable Software Engineering. Theories, Tools, and Applications - 5th International Symposium, (SETTA) 2019, Shanghai, China, 2019*, N. Guan *et al.*, Eds., vol. 11951, pp. 89–105.

[24] Y. Abdeddaïm *et al.*, "Probabilistic schedulability analysis for fixed priority mixed criticality real-time systems," in *Design, Automation & Test in Europe Conference & Exhibition, (DATE), Lausanne, Switzerland, 2017*, D. Atienza *et al.*, Eds., pp. 596–601.

[25] S. Draskovic *et al.*, "Schedulability of probabilistic mixed-criticality systems," *Real Time Syst.*, vol. 57, no. 4, pp. 397–442, 2021.

[26] B. Ranjbar *et al.*, "Improving the timing behaviour of mixed-criticality systems using chebyshev's theorem," in *2021 Design, Automation & Test in Europe Conference & Exhibition (DATE)*. IEEE, 2021, pp. 264–269.

[27] S. Baruah *et al.*, "The preemptive uniprocessor scheduling of mixed-criticality implicit-deadline sporadic task systems," in *2012 24th Euromicro Conference on Real-Time Systems*. IEEE, 2012, pp. 145–154.