A Comparison between Fixed Priority and EDF Scheduling accounting for Cache Related Pre-emption Delays

Will Lunniss[1], Sebastian Altmeyer[2], and Robert I. Davis[1]

1 **Department of Computer Science**
University of York
York, UK
`{wl510,rob.davis}@york.ac.uk`

2 **Computer Systems Architecture Group**
University of Amsterdam
Amsterdam, The Netherlands
`altmeyer@uva.nl`

Abstract

In multitasking real-time systems, the choice of scheduling algorithm is an important factor to ensure that response time requirements are met while maximising limited system resources. Two popular scheduling algorithms include *fixed priority* (FP) and *earliest deadline first* (EDF). While they have been studied in great detail before, they have not been compared when taking into account *cache related pre-emption delays* (CRPD). Memory and cache are split into a number of blocks containing instructions and data. During a pre-emption, cache blocks from the pre-empting task can evict those of the pre-empted task. When the pre-empted task is resumed, if it then has to re-load the evicted blocks, CRPD are introduced which then affect the schedulability of the task.

In this paper we compare FP and EDF scheduling algorithms in the presence of CRPD using the state-of-the-art CRPD analysis. We find that when CRPD is accounted for, the performance gains offered by EDF over FP, while still notable, are diminished. Furthermore, we find that under scenarios that cause relatively high CRPD, task layout optimisation techniques can be applied to allow FP to schedule tasksets at a similar processor utilisation to EDF. Thus making the choice of the task layout in memory as important as the choice of scheduling algorithm. This is very relevant for industry, as it is much cheaper and simpler to adjust the task layout through the linker than it is to switch the scheduling algorithm.

2012 ACM Subject Classification Software and its engineering, Software organization and properties, Software functional properties, Correctness, Real-time schedulability

Keywords and phrases Real-Time Systems, Fixed Priority, EDF, Pre-emptive, Scheduling, Cache Related Pre-emption Delays

Digital Object Identifier 10.4230/LITES-v001-i001-a001

Received 2013-08-22 **Accepted** 2014-03-03 **Published** 2014-04-14

1 Introduction

Today's real-time applications are complex systems built up of a large number of interacting tasks running on hardware with non-deterministic performance enhancing features such as caches, pipelines and out-of-order execution. To manage the available resources efficiently, scheduling algorithms are used to determine which task should run and at which time in order to fulfil the functional and temporal requirements of the system. The scheduling algorithms are often *pre-empting*, in that they allow important tasks to interrupt less important tasks before they have finished. Two popular scheduling algorithms for real-time systems are *fixed priority* (FP) and *earliest deadline first* (EDF). FP scheduling uses statically defined priorities to run the task with the highest priority first. In comparison, EDF is a dynamic scheduling algorithm that schedules the task with the earliest absolute deadline first. EDF is an optimal scheduling algorithm without pre-emption costs, whereas FP is not, and is therefore typically able to schedule tasksets at a higher processor utilisation than FP [20]. However, despite the significant performance benefits over FP, EDF is not widely used in commercial real-time operating systems.

In real-time systems, and especially hard real-time systems, the schedulability of each task must be known in order to verify that the timing requirements will be met. The schedulability of a taskset is determined using information about the scheduling algorithm, the arrival pattern of tasks and the tasks' *worst-case execution times.* Worst-case execution times are typically obtained assuming no pre-emption. However, in pre-emptive multi-tasking systems, caches introduce additional *cache related pre-emption delays* (CRPD) caused by the need to re-fetch blocks belonging to the pre-empted task which were evicted from the cache by the pre-empting task. These CRPD effectively increase the worst-case execution time of the tasks. It is therefore important to be able to calculate, and therefore account for, CRPD when determining if a system is schedulable or not.

In 2005, Buttazzo [13] performed a detailed study of FP and EDF scheduling. This work covered both schedulability under a variety of scenarios, in addition to practical implementation considerations. Results showed that the FP scheduling algorithm introduces more pre-emptions than EDF, especially at high processor utilisation levels. This leads to FP performing worse than EDF. Yet, FP has an advantage over EDF, in that it is generally simpler to implement in commercial kernels which do not provide explicit support for timing constraints. Despite being a very detailed study, these comparisons where done under the assumption that there were no pre-emption costs due to CRPD.

In this paper we build on the work by Buttazzo [13] and use state of the art CRPD analysis for FP [3] and EDF [22] to perform a comprehensive study of these two popular scheduling algorithms when accounting for CRPD.

1.1 Related Work on CRPD

Analysis of CRPD uses the concept of *useful cache blocks* (UCBs) and *evicting cache blocks* (ECBs) based on the work by Lee et al. [18]. Any memory block that is accessed by a task while executing is classified as an ECB, as accessing that block may evict a cache block of a pre-empted task. Out of the set of ECBs, some of them may also be UCBs. A memory block m is classified as a UCB at program point $\mathcal{P}$, if (i) m may be cached at $\mathcal{P}$ and (ii) m may be reused at program point $\mathcal{Q}$ that may be reached from $\mathcal{P}$ without eviction of m on this path. In the case of a pre-emption at program point $\mathcal{P}$, only the memory blocks that are (i) in cache and (ii) will be reused, may cause additional reloads. The maximum possible pre-emption cost for a task is determined by the program point with the highest number of UCBs. For each subsequent pre-emption, the program point with the next smallest number of UCBs can be considered. Altmeyer and Burguière [1] presented a tighter definition of UCBs however, we only need the basic concept for this paper.

Depending on the approach used, the CRPD analysis combines the UCBs belonging to the pre-empted task(s) with the ECBs of the pre-empting task(s). Using this information, the total number of blocks that are evicted, which must then be reloaded after the pre-emption, can be calculated and combined with the cost of reloading a block to then give the CRPD.

A number of approaches have been developed for calculating the CRPD when using FP pre-emptive scheduling. They include Lee et al. [18] UCB-Only approach, which considers just the pre-empted task(s), and Busquets et al. [12] ECB-Only approach which considers just the pre-empting task. Approaches that consider the pre-empted and pre-empting task(s) include Tan and Mooney [26] UCB-Union approach, Altmeyer et al. [2] ECB-Union approach, and an alternative approach by Staschulat et al. [25]. Finally, there are advanced multiset based approaches that consider the pre-empted and pre-empting task(s) by Altmeyer et al. [3], ECB-Union Multiset, UCB-Union Multiset, and a combined multiset approach.

There has been less work towards developing CRPD analysis for EDF pre-emptive scheduling. In 2007, Ju et al. [17] considered the intersection of the pre-empted task's UCBs with the pre-empting task's ECBs. However, this method for handling nested pre-emptions can lead to significant pessimism as each pair of tasks is considered separately. In 2013, Lunniss et al. [22] adapted a number of approaches for calculating CRPD for FP to work with EDF. Including the ECB-Only, UCB-Only, UCB-Union, ECB-Union, ECB-Union Multiset, UCB-Union Multiset and combined multiset CRPD analysis for FP given by Busquets et al. [12], Lee et al. [18], Tan and Mooney [26], and Altmeyer et al. [2, 3].

A different methodology was used by Bastoni et al. [8]. Instead of focusing on how to calculate an upper bound on the CRPD, they used measurements on real hardware to estimate a lower bound on the CRPD and *cache related migration delays* for data caches in a multi-processor system.

CRPD can have a significant effect on schedulability, and can also vary dramatically depending on a number of factors. In particular, the CRPD is highly dependent on how tasks are placed in cache. As the layout of tasks in memory determines how they are positioned in cache, choosing a sensible layout can have a big impact on the CRPD caused due to pre-emptions. In 2012, Lunniss et al. [21] presented an approach that uses a *Simulated Annealing* algorithm to optimise the layout of tasks to increase system schedulability when using FP scheduling.

1.2 Organisation

The paper is organised as follows. Section 2 introduces the system model, terminology and notation used. Existing schedulability tests and CRPD analysis are outlined in Section 3 for FP scheduling, and in Section 4 for EDF scheduling. Section 5 briefly covers optimising task layout to reduce CRPD. Section 6 compares FP and EDF with CRPD analysis using a set of case studies. In Section 7, we investigate the effect of a variety of configuration parameters in a series of evaluations using synthetic tasksets. Finally, we conclude in Section 8.

2 System Model, Terminology and Notation

This section describes the system model, terminology, and notation used in the rest of the paper.

We assume a single processor system, running a statically defined taskset under either pre-emptive FP or pre-emptive EDF scheduling. The system comprises a taskset Γ made up of a fixed number of tasks $(\tau_1, \ldots, \tau_n)$ where n is a positive integer. In the case of FP, each task has a unique fixed priority and the priority of task τ_i, is i, where a priority of 1 is the highest and n is the lowest. Each task, τ_i may produce a potentially infinite stream of jobs that are separated by a minimum inter-arrival time or period T_i. Each task has a relative deadline D_i, and each job of a task has an absolute deadline d_i which is D_i after it is released. In the case of EDF, each task has a unique task index ordered by relative deadline from smallest to largest. In the case of a tie when assigning the unique task indices, an arbitrary choice is made. Each task also has a worst case execution time C_i (determined for non-pre-emptive execution). In this paper, we consider tasks with *constrained* deadlines. (Task deadlines may be referred to as *constrained* deadlines, i.e. $D_i \leq T_i$ or *implicit* i.e. $D_i = T_i$). We assume a discrete time model. We define T_{max} as the largest period of any task in the taskset, and similarly D_{max} as the largest relative deadline of any task in the taskset. Each task has a utilisation U_i, where $U_i = C_i/T_i$, and each taskset has a utilisation U which is equal to the sum of its tasks' utilisations.

A taskset is said to be *schedulable* with respect to a scheduling algorithm if all valid sequences of jobs generated by the taskset can be scheduled by the algorithm without any missed deadlines. A taskset is *feasible* if there exists some scheduling algorithm that can schedule all possible sequences of jobs that may be generated by the taskset without any missed deadlines. A scheduling algorithm is said to be *optimal* with respect to a task model if it can schedule all of the feasible tasksets that comply with the task model.

Each task τ_i has a set of UCBs, UCB_i and a set of ECBs, ECB_i represented by a set of integers. If for example, task τ_1 contains 4 ECBs, where the second and fourth ECBs are also UCBs, these can be represented using $\mathrm{ECB}_1 = \{1, 2, 3, 4\}$ and $\mathrm{UCB}_1 = \{2, 4\}$. The *block reload time* (BRT) is the time taken to load a block from memory into cache. This cost is incurred every time that a UCB has to be reloaded after a pre-emption. We assume that the remaining context switch costs, i.e., pipeline and scheduler related costs are subsumed in the execution time bound of each task. Furthermore, we assume that the OS resides in a different cache partition and therefore scheduler operations do not cause CRPD.

We use the term *cache utilisation* to describe the ratio of the total size of the tasks to the size of the cache. A cache utilisation of 1 means that the tasks fit exactly in the cache, whereas a cache utilisation of 5 means the total size of the tasks is 5 times the size of the cache.

We focus on instruction only caches. In the case of data caches, the analysis would either require a write-through cache or further extension in order to be applied to write-back caches. We assume that tasks do not share any code. We also assume a direct mapped cache, but the work extends to set-associative caches with the LRU replacement policy[1]. In the case of set-associative LRU caches, a single cache-set may contain several UCBs. For example, $\mathrm{UCB}_1 = \{2, 2, 4\}$ means that task τ_1 has two UCBs in cache-set 2 and one UCB in cache set 4. As one ECB suffices to evict all UCBs of the same cache-set, multiple accesses to the same set by the pre-empting task do not appear in the set of ECBs. A bound on the CRPD in the case of LRU caches due to task τ_j directly pre-empting τ_i is thus given by the intersection $\mathrm{UCB}_i \cap' \mathrm{ECB}_j = \{m | m \in \mathrm{UCB}_i : m \in \mathrm{ECB}_j\}$, where the result is a multiset that contains each element from UCB_i if it is also in ECB_j. A precise computation of CRPD in the case of LRU caches is given in Altmeyer et al. [4]. The equations provided in this paper can be applied to set-associative LRU caches with the above adaptation to the set-intersection.

3 CRPD Analysis for FP Scheduling

In this section, we give an overview of FP scheduling and schedulability analysis for it. We then cover the state of the art CRPD analysis for FP scheduling, by Altmeyer et al. described in detail in [3].

Under FP scheduling, the sets of tasks that can pre-empt each other are based on the statically assigned fixed task priorities. Using the fixed priorities, we can define the following sets of tasks for determining which tasks can pre-empt each other. $\mathrm{hp}(i)$ and $\mathrm{lp}(i)$ are the sets of tasks with higher and lower priorities than task τ_i, and $\mathrm{hep}(i)$ and $\mathrm{lep}(i)$ are the sets containing tasks with higher or equal and lower or equal priorities to task τ_i. Additionally, $\mathrm{aff}(i, j) = \mathrm{hep}(i) \cap \mathrm{lp}(j)$ is used to represent all tasks that can have CRPD caused by task τ_j pre-empting them, which affects the response time of task τ_i. In other words, it is the set of tasks that may be pre-empted by task τ_j and have at least the priority of task τ_i.

[1] The concept of UCBs and ECBs cannot be applied to the FIFO or PLRU replacement policies as shown by Burguière [11].

Schedulability tests are used to determine if a taskset is schedulable, i.e. all the tasks will meet their deadlines given the worst-case pattern of arrivals and execution. For a given taskset, the response time R_i for each task τ_i, can be calculated and compared against the tasks' deadline, D_i. If every task in the taskset meets its deadline, then the taskset is schedulable. The equation used to calculate R_i is defined as [5]:

$$R_i^{\alpha+1} = C_i + \sum_{\forall j \in hp(i)} \left\lceil \frac{R_i^\alpha}{T_j} \right\rceil (C_j) \,. \tag{1}$$

Equation (1) can be solved using fixed point iteration. Iteration continues until either $R_i^{\alpha+1} > D_i$ in which case the task is unschedulable, or until $R_i^{\alpha+1} = R_i^\alpha$ in which case the task is schedulable and has a worst-case response time of R_i^α.

Note the convergence of (1) may be speeded up using the techniques described in [14].

3.1 CRPD Analysis

To account for the CRPD, a component $\gamma_{i,j}$ is introduced into (1). There are a number of different methods that can be used to compute $\gamma_{i,j}$ described by Altmeyer et al. in [3]. Depending on the method used, $\gamma_{i,j}$ represents either a single pre-emption, or multiple pre-emptions and is calculated using the cost incurred when reloading a block, the *block reload time* (BRT), multiplied by the number of blocks which may need to be reloaded after each pre-emption.

We will now summarise the *combined multiset* approach, which has been shown to dominate all other approaches [3]. For worked examples of the analysis, see Section 4 ECB-Union and Multiset Approaches of Altmeyer et al. [3].

In the combined multiset approach, $\gamma_{i,j}$ represents the total cost of all pre-emptions due to jobs of task τ_j executing within the response time of task τ_i. Incorporating $\gamma_{i,j}$ into (1) gives a revised equation for R_i:

$$R_i^{\alpha+1} = C_i + \sum_{\forall j \in hp(i)} \left(\left\lceil \frac{R_i^\alpha}{T_j} \right\rceil C_j + \gamma_{i,j} \right) \,. \tag{2}$$

$\gamma_{i,j}$ is then calculated using two separate approaches, the *UCB-Union multiset*, and *ECB-Union multiset* which are described below. The key concept behind them is to calculate the cost of each individual pre-emption by jobs of task τ_j that could occur within the response time of task τ_i. By calculating the cost of each pre-emption, the analysis is able to account for the fact that intermediate tasks in a nested pre-emption will often be pre-empted less than the lowest priority task. Consider the following example with three tasks shown in Figure 1.

In the example, the total cost of all jobs of task τ_1 pre-empting task τ_3 within the response time of task τ_3 is equal to the cost of task τ_1 pre-empting task τ_2 and task τ_3 once (nested pre-emption), and task τ_3 on its own twice. It is therefore important to recognise that the cost of one task pre-empting another is highly dependent on any intermediate tasks that may be involved in a nested pre-emption. To calculate the number of pre-emptions, we use $E_j(R_i)$ to denote the

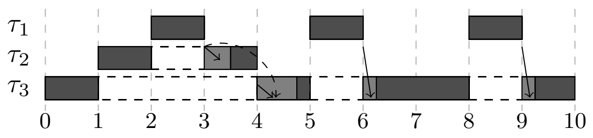

Figure 1 The pre-emption cost of all jobs of task τ_1 pre-empting task τ_2 can only contribute once to the total pre-emption cost of task τ_1 pre-empting τ_3 during the response time of τ_3.

maximum number of jobs of task τ_j that can execute during the response time, R_i, of task τ_i. For our model, $E_j(R_i) = \lceil R_i/T_j \rceil$.

3.1.1 ECB-Union Multiset

The ECB-Union multiset approach computes the union of all ECBs that may affect a pre-empted task during a pre-emption by task τ_j. Specifically, it accounts for nested pre-emptions by assuming that task τ_j has already been pre-empted by all tasks of a higher priority.

The first step is to calculate the number of UCBs that task τ_j could evict when pre-empting an intermediate task, τ_k. This is given by calculating the intersection of the UCBs of the pre-empted task, task τ_k, with the set of ECBs belonging to the pre-empting tasks $\bigcup_{h \in \mathrm{hp}(j) \cup \{j\}} \mathrm{ECB}_h$ to give:

$$\left| \mathrm{UCB}_k \cap \left(\bigcup_{h \in \mathrm{hp}(j) \cup \{j\}} \mathrm{ECB}_h \right) \right| . \tag{3}$$

Note $h \in \mathrm{hp}(j) \cup \{j\}$ is used to account for the case when tasks can share priorities.

The ECB-Union multiset approach recognises that task τ_j cannot pre-empt each intermediate task τ_k more than $E_j(R_k)E_k(R_i)$ times during the response time of task τ_i. Therefore, the next step is to form a multiset $M_{i,j}$ that contains the cost of task τ_j pre-empting task τ_k (3) repeated $E_j(R_k)E_k(R_i)$ times, for each task $\tau_k \in \mathrm{aff}(i,j)$ hence:

$$M_{i,j} = \bigcup_{k \in \mathrm{aff}(i,j)} \left(\bigcup_{E_j(R_k)E_k(R_i)} \left| \mathrm{UCB}_k \cap \left(\bigcup_{h \in \mathrm{hp}(j) \cup \{j\}} \mathrm{ECB}_h \right) \right| \right) . \tag{4}$$

As only $E_j(R_i)$ jobs of task τ_j can execute during the response time of task τ_i, the maximum CRPD is obtained by summing the $E_j(R_i)$ largest pre-emptions, i.e. the $E_j(R_i)$ largest values in $M_{i,j}$:

$$\gamma_{i,j}^{\text{ecb-m}} = \mathrm{BRT} \cdot \sum_{l=1}^{E_j(R_i)} M_{i,j}^{l} , \tag{5}$$

where $M_{i,j}^{l}$ is the l^{th} largest integer value from the multiset $M_{i,j}$.

3.1.2 UCB-Union Multiset

The UCB-Union multiset approach accounts for the effects of nested pre-emptions by assuming that the UCBs of any tasks that could be pre-empted, including nested pre-emptions, by task τ_j are evicted by the ECBs of task τ_j. The first step is to form a multiset $M_{i,j}^{\text{ucb}}$ containing $E_j(R_k)E_k(R_i)$ copies of the UCB_k of each task $\tau_k \in \mathrm{aff}(i,j)$ that could be pre-empted by task τ_j and has at least the priority of task τ_i. This multiset reflects the fact that jobs of task τ_j cannot evict the UCBs of jobs of task τ_k within the response time of task τ_i more than $E_j(R_k)E_k(R_i)$ times. Hence:

$$M_{i,j}^{\text{ucb}} = \bigcup_{k \in \mathrm{aff}(i,j)} \left(\bigcup_{E_j(R_k)E_k(R_i)} \mathrm{UCB}_k \right) . \tag{6}$$

The second step is to form a separate multiset $M_{i,j}^{\text{ecb}}$ containing $E_j(R_i)$ copies of the ECB_j of task τ_j. This multiset reflects the fact that there can be no more than $E_j(R_i)$ jobs of task τ_j

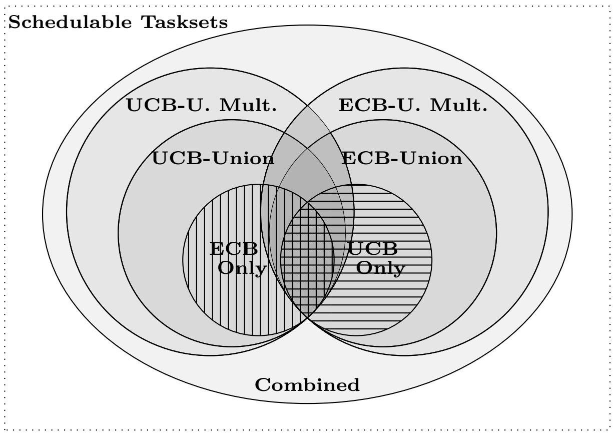

Figure 2 Venn diagram showing the relationship between different approaches for CPRD analysis under FP scheduling.

within the response time of task τ_i, each of which can evict cache blocks in the set ECB_j:

$$M_{i,j}^{\mathrm{ecb}} = \bigcup_{E_j(R_i)} (\mathrm{ECB}_j)\,. \tag{7}$$

$\gamma_{i,j}^{\text{ucb-m}}$ is then given by the size of the multiset intersection between $M_{i,j}^{\mathrm{ucb}}$ and $M_{i,j}^{\mathrm{ecb}}$:

$$\gamma_{i,j}^{\text{ucb-m}} = \mathrm{BRT} \cdot \left| M_{i,j}^{\mathrm{ucb}} \cap M_{i,j}^{\mathrm{ecb}} \right| \,. \tag{8}$$

3.1.3 Combined Multiset

The ECB-Union multiset and UCB-Union multiset approaches are incomparable, meaning that each approach can find different sets of tasksets schedulable. Because of this property, they can be combined together to form a combined approach:

$$R_i = \min(R_i^{\text{ucb-m}}, R_i^{\text{ecb-m}})\,. \tag{9}$$

The response time for every task is calculated using each approach and then the minimum is taken, because of this, the combined approach can deem some tasksets schedulable that are not schedulable by either approach on its own.

3.2 Comparison of Approaches

Figure 2 shows a Venn diagram that conveys the relationship between a number of different approaches for calculating CRPD under FP scheduling [3]. However, it does not include the method by Staschulat et al. [25] because it is incomparable to them. Specifically, while it typically deems a lower number of tasksets schedulable, it could potentially find a taskset schedulable that is not schedulable by any of the other approaches. Aside from the approach by Staschulat et al. [25], it can be seen that the combined multiset approach dominates all other approaches. See Altmeyer et al. [3] for a detailed comparison between each approach.

4 CRPD Analysis for EDF Scheduling

In this section, we give an overview of EDF scheduling and schedulability analysis for it. We then cover the state of the art CRPD analysis for EDF scheduling, by Lunniss et al. [22].

EDF is a dynamic scheduling algorithm which always schedules the job with the earliest absolute deadline first. In pre-emptive EDF, any time a job arrives with an earlier absolute deadline than the current running job, it will pre-empt the current job. When a job completes its execution, the EDF scheduler chooses the pending job with the earliest absolute deadline to execute next.

In 1973, Liu and Layland [20] gave a necessary and sufficient schedulability test that indicates whether a taskset is schedulable under EDF iff $U \leq 1$, under the assumption that all tasks have implicit deadlines ($D_i = T_i$). In the case where $D_i \leq T_i$ this test is still necessary, but is no longer sufficient.

In 1974, Dertouzos [15] proved EDF to be optimal among all scheduling algorithms on a uniprocessor, in the sense that if a taskset cannot be scheduled by pre-emptive EDF, then this taskset cannot be scheduled by any algorithm.

In 1980, Leung and Merrill [19] showed that a set of periodic tasks is schedulable under EDF iff all absolute deadlines in the interval $[0, \max\{s_i\} + 2H]$ are met, where s_i is the start time of task τ_i, $\min\{s_i\} = 0$, and H is the hyperperiod (least common multiple) of all tasks' periods.

In 1990 Baruah et al. [6, 7] extended Leung and Merrill's work [19] to sporadic tasksets. They introduced $h(t)$, the processor demand function, which denotes the maximum execution time requirement of all tasks' jobs which have both their arrival times and their deadlines in a contiguous interval of length t. Using this they showed that a taskset is schedulable iff $\forall t > 0, h(t) \leq t$ where $h(t)$ is defined as:

$$h(t) = \sum_{i=1} \max\left\{0, 1, + \left\lfloor \frac{t - D_i}{T_i} \right\rfloor \right\} C_i \, . \tag{10}$$

Examining (10), it can be seen that $h(t)$ can only change when t is equal to an absolute deadline, which restricts the number of values of t that need to be checked. In order to place an upper bound on t, and therefore the number of calculations of $h(t)$, the minimum interval in which it can be guaranteed that an unschedulable taskset will be shown to be unschedulable must be found. For a general taskset with arbitrary deadlines t can be bounded by L_a [16]:

$$L_a = \max\left\{ D_1, \ldots, D_n, \frac{\sum_{i=1}^{n} (T_i - D_i) U_i}{1 - U} \right\} . \tag{11}$$

Spuri [24] and Ripoll et al. [23] showed that an alternative bound L_b, given by the length of the synchronous busy period can be used. Where L_b is computed by solving the following equation using fixed point iteration:

$$w^{\alpha+1} = \sum_{i=1}^{n} \left\lceil \frac{w^{\alpha}}{T_i} \right\rceil C_i \, . \tag{12}$$

There is no direct relationship between L_a and L_b, which enables t to be bounded by $L = \min(L_a, L_b)$. Combined with the knowledge that $h(t)$ can only change at an absolute deadline, a taskset is therefore schedulable under EDF iff $U \leq 1$ and:

$$\forall t \in Q : h(t) \leq t \, , \tag{13}$$

where Q is defined as:

$$Q = \{ d_k \mid d_k = kT_i + D_i \wedge d_k < \min(L_a, L_b), k \in \mathbb{N} \} . \tag{14}$$

In 2009, Zhang and Burns [27] presented their Quick convergence Processor-demand Analysis (QPA) algorithm which exploits the monotonicity of $h(t)$. QPA determines schedulability by starting with a value of t that is close to L, and then iterating back towards 0 checking a significantly smaller number of values of t than would otherwise be required.

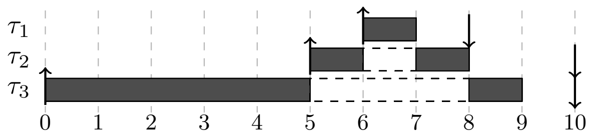

Figure 3 Example schedule showing how the scheduler chooses which task should execute. Task τ_3 is released at $t = 0$. At $t = 5$, task τ_2 is released, pre-empting τ_3 as although it has the same absolute deadline, it has a lower task index. At $t = 6$, task τ_1 is released, pre-empting task τ_2. At $t = 7$, τ_1 completes, the scheduler then chooses to resume task τ_2 as although it has the same absolute deadline as task τ_3, it has the lower task index.

4.1 CRPD Analysis

Due to the undefined behaviour of EDF when two or more jobs have the same absolute deadline, an assumption needs to be made before we can tightly calculate CRPD for EDF. In the case where two or more jobs have the same absolute deadline, Lunniss et al. [22] assume the scheduler always picks the job belonging to the task with the lowest unique task index, see Figure 3. This has the benefit of minimising the number of pre-emptions. In the case where jobs of two tasks have the same absolute and relative deadlines, it ensures that they cannot pre-empt each other. Furthermore, it ensures that after a pre-emption, the task that was pre-empted last is resumed first.

Following the analysis of Lunniss et al. [22], we now define a number of terms with respect to EDF scheduling. Some of the terms are also present in the analysis for FP, but have slightly different meanings under EDF. Assuming any task τ_j with a relative deadline $D_j < D_i$ can pre-empt task τ_i, the set of tasks that may have a higher priority, and can pre-empt task τ_i, under EDF is:

$$\mathrm{hp}(i) = \{\tau_j \in \Gamma \mid D_j < D_i\}\,. \tag{15}$$

The set of tasks that can be pre-empted by jobs of task τ_j in an interval of length t, $\mathrm{aff}(t, j)$ is based on the relative deadlines of the tasks. It captures all of the tasks whose relative deadlines are greater than the relative deadline of task τ_j excluding tasks whose deadlines are larger than t as they do not need to be included when calculating $h(t)$. This gives:

$$\mathrm{aff}(t, i) = \{\tau_j \in \Gamma \mid t \geq D_i > D_j\}\,. \tag{16}$$

To determine how many pre-emptions can occur, we use $P_j(D_i)$ to denote the maximum number of jobs of task τ_j that can pre-empt a single job of task τ_i:

$$P_j(D_i) = \max\left(0, \left\lceil \frac{D_i - D_j}{T_j} \right\rceil\right)\,. \tag{17}$$

Finally, we also use $E_j(t)$ to denote the maximum number of jobs of task τ_j that can have both their release times and their deadlines in an interval of length t:

$$E_j(t) = \max\left(0, \left\lfloor \frac{t - D_j}{T_j} \right\rfloor\right)\,. \tag{18}$$

We now summarise CRPD analysis for EDF by Lunniss et al. [22] using the *combined multiset* approach as it has been shown to dominate all other approaches for calculating CRPD for EDF. This approach is based on the combined multiset approach for FP as described in Section 3.1, and as such the equations and the intuition behind them are similar. The difference is to do with which tasks pre-empt each other and the timeframe used to determine which jobs to include in the calculation.

CRPD analysis can be integrated into the EDF schedulability test by introducing an additional parameter, $\gamma_{t,j}$ to represent the CRPD. In this case, $\gamma_{t,j}$ represents the cost of the maximum number $E_j(t)$ of pre-emptions by jobs of task τ_j that have their release times and absolute deadlines in an interval of length t. It is therefore included in (10) as follows:

$$h(t) = \sum_{j=1}^{n} \left(\max\left\{0, \left\lfloor \frac{t - D_j}{T_j} \right\rfloor \right\} C_j + \gamma_{t,j} \right) . \tag{19}$$

$\gamma_{t,j}$ can then be calculated using two different methods and the lowest value of the two used to calculate the processor demand. These methods calculate the cost of each possible individual pre-emption by task τ_j that could occur during an interval of length t.

4.1.1 ECB-Union Multiset

The ECB-Union multiset approach computes the union of all ECBs that may affect a pre-empted task during a pre-emption by task τ_j. Specifically, it accounts for nested pre-emptions by assuming that task τ_j has already been pre-empted by all other tasks that may pre-empt it. The first step is to form a multiset $M_{t,j}$ that contains the cost:

$$\left| \mathrm{UCB}_k \cap \left(\bigcup_{h \in \mathrm{hp}(j) \cup \{j\}} \mathrm{ECB}_h \right) \right| \tag{20}$$

of task τ_j pre-empting task τ_k repeated $P_j(D_k)E_k(t)$ times, for each task $\tau_k \in \mathrm{aff}(t, j)$. Hence:

$$M_{t,j} = \bigcup_{k \in \mathrm{aff}(t,j)} \left(\bigcup_{P_j(D_k)E_k(t)} \left| \mathrm{UCB}_k \cap \left(\bigcup_{h \in \mathrm{hp}(j) \cup \{j\}} \mathrm{ECB}_h \right) \right| \right) . \tag{21}$$

As there are only $E_j(t)$ jobs of task τ_j with release times and deadlines in an interval of length t, the maximum CRPD is obtained by summing the $E_j(t)$ largest values in $M_{t,j}$:

$$\gamma_{t,j}^{\text{ecb-m}} = \mathrm{BRT} \cdot \sum_{l=1}^{E_j(t)} M_{t,j}^{l} , \tag{22}$$

where $M_{t,j}^{l}$ is the l^{th} largest integer value from the multiset $M_{t,j}$.

4.1.2 UCB-Union Multiset Approach

The UCB-Union multiset approach accounts for the effects of nested pre-emptions by assuming that the UCBs of any tasks that could be pre-empted, including nested pre-emptions, by task τ_j are evicted by the ECBs of task τ_j. The first step is to form a multiset $M_{t,j}^{\text{ucb}}$ containing $P_j(D_k)E_k(t)$ copies of the UCB_k of each task $\tau_k \in \mathrm{aff}(t, j)$. This multiset reflects the fact that jobs of task τ_j cannot evict the UCBs of jobs of task τ_k that have both their release times and deadlines in an interval of length t more than $P_j(D_k)E_k(t)$ times. Hence:

$$M_{t,j}^{\text{ucb}} = \bigcup_{k \in \mathrm{aff}(t,j)} \left(\bigcup_{P_j(D_k)E_k(t)} \mathrm{UCB}_k \right) . \tag{23}$$

The second step is to form a separate multiset $M_{t,j}^{\mathrm{ecb}}$ containing $E_j(t)$ copies of the ECB_j of task τ_j. This multiset reflects the fact that there are at most $E_j(t)$ jobs of task τ_j that have their release times and deadlines in an interval of length t, each of which can evict cache blocks in the set ECB_j:

$$M_{t,j}^{\mathrm{ecb}} = \bigcup_{E_j(t)} \left(\mathrm{ECB}_j\right). \tag{24}$$

$\gamma_{t,j}^{\text{ucb-m}}$ is then given by the size of the multi-set intersection between $M_{t,j}^{\mathrm{ucb}}$ and $M_{t,j}^{\mathrm{ecb}}$:

$$\gamma_{t,j}^{\text{ucb-m}} = \mathrm{BRT} \cdot \left| M_{t,j}^{\mathrm{ucb}} \cap M_{t,j}^{\mathrm{ecb}} \right| . \tag{25}$$

4.1.3 Combined Multiset Approach

The ECB-Union Multiset and UCB-Union Multiset approaches provide upper bounds that are incomparable, therefore, $h(t)$ can be calculated at each stage of the QPA algorithm using both approaches and the minimum value taken to form a combined approach:

$$h(t) = \min\left(h(t)^{\text{ucb-m}}, h(t)^{\text{ecb-m}}\right) . \tag{26}$$

As the processor demand is calculated using each approach, for each interval t, the combined approach can deem some tasksets schedulable that are not schedulable by either approach on its own.

4.1.4 Effect on Task Utilisation and $h(t)$ Calculation

As the multiset approaches effectively inflate the execution time of task τ_j by the CRPD that it can cause in an interval of length t, the upper bound L, used for calculating the processor demand $h(t)$, must be adjusted. This is achieved by calculating an upper bound on the utilisation due to CRPD that is valid for all intervals of length greater than some value L_c. This CRPD utilisation value is then used to inflate the taskset utilisation and thus compute an upper bound L_d on the maximum length of the synchronous busy period. This upper bound is valid provided that it is greater than L_c, otherwise the actual maximum length of the busy period may lie somewhere in the interval $[L_d, L_c]$, hence we can use $\max(L_c, L_d)$ as a bound.

The first step is to assign $t = L_c = 100T_{\max}$ which limits the overestimation of the CRPD utilisation $U^\gamma = \gamma_t/t$ to at most 1%. Next, γ_t is calculated using (22) for ECB-Union Multiset and (25) for UCB-Union Multiset. However, in (21) and (23) & (24), $E_x^{\max}(t)$ is substituted for $E_x(t)$ to ensure that the computed value of U^γ is a valid upper bound for all intervals of length $t \geq L_c$:

$$E_x^{\max}(t) = \max\left(0, 1 + \left\lceil \frac{t - D_x}{T_x} \right\rceil\right) . \tag{27}$$

If $U + U^\gamma \geq 1$, then the taskset is deemed unschedulable, otherwise an upper bound on the length of the busy period can be computed via a modified version of (12):

$$w^{\alpha+1} \leq \sum_{\forall j} \left(\frac{w^\alpha}{T_j} + 1\right) C_j + w^\alpha U^\gamma \tag{28}$$

rearranged to give:

$$w \leq \frac{1}{(1 - (U + U^\gamma))} \sum_{\forall j} U_j T_j \,. \tag{29}$$

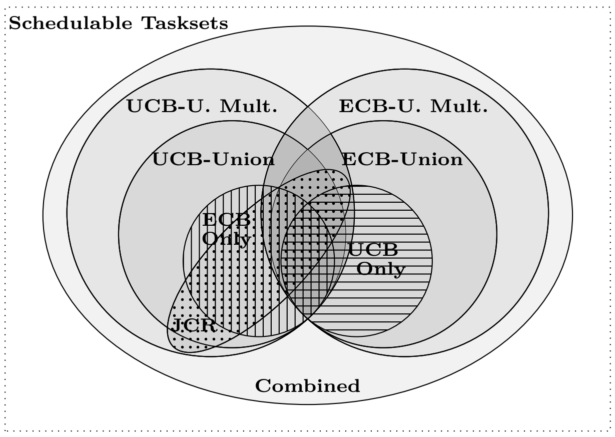

Figure 4 Venn diagram showing the relationship between different approaches for CPRD analysis under EDF scheduling.

Then, substituting in $T_{\max}$ for each value of T_j the upper bound is given by:

$$L_d = \frac{U \cdot T_{\max}}{(1 - (U + U^{\gamma}))} . \tag{30}$$

Finally, $L = \max(L_c, L_d)$ can then be used as the maximum value of t to check in the EDF schedulability test.

4.2 Comparison of Approaches

Figure 4 shows a Venn diagram that conveys the relationship between the different approaches for calculating CRPD under EDF scheduling [22]. Note that JCR represents the approach of Ju et al. [17]. It can be seen that the combined multiset approach dominates all other approaches, see Lunniss et al. [22] for detailed comparisons of each approach.

5 Task Layout

The layout of tasks in memory determines how they are positioned in cache, which then affects the CRPD that occurs during pre-emptions. Figure 5 shows an example layout of five tasks in cache. If scheduled under FP, task τ_1 has the highest priority, so its UCBs can never be evicted as it cannot be pre-empted. Task τ_2 and τ_3's UCBs are safe from eviction as they are not mapped to the same location in cache as higher priority task's ECBs. However, task τ_4's UCBs could be evicted by task τ_1, and τ_5's UCBs could be evicted by task τ_1, τ_2 or τ_4. This layout could be improved by shifting task τ_5 so that its UCBs can only be evicted by task τ_3.

In 2012, Lunniss et al. [21] presented an approach that uses *Simulated Annealing* to optimise the layout of tasks to increase system schedulability. It does so by changing the order of tasks in memory, which can be implemented in practice by presenting the tasks to the linker in the desired order. The approach is driven by the schedulability of the taskset, favouring layouts that allow the taskset to be scheduled at a higher utilisation. While this approach was originally used for FP scheduling, it can also be applied to the EDF scheduling algorithm by switching the schedulability test used. We therefore use this approach to optimise the layout of the tasksets to make each scheduling algorithm as competitive as possible.

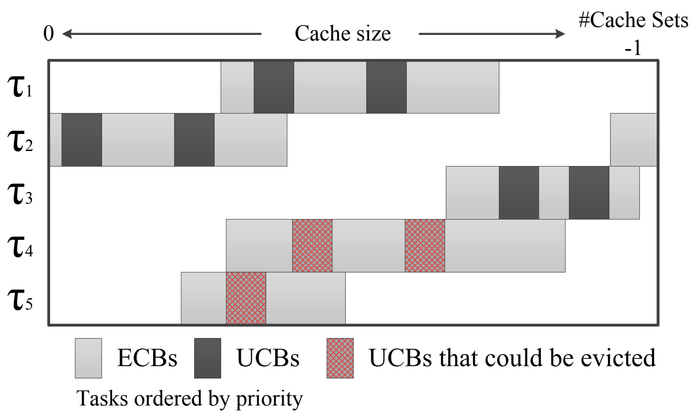

Figure 5 Example layout showing how the position of tasks in cache affects whether their UCBs could be evicted during a pre-emption.

6 Case Studies

In this section we compare the different approaches for calculating CRPD using a set of case studies based on PapaBench[2], the Mälardalen[3] benchmark suite and a set of SCADE[4] tasks (partially provided by SCADE, partially from our own SCADE models). In all cases the system was set up to model an ARMv7[5] processor clocked at 100 MHz with a 2 KB direct-mapped instruction cache and a line size of 8 Bytes, giving 256 cache sets, 4 Byte instructions, and a BRT of 8 μs.

6.1 Single Taskset Case Study

PapaBench is a real-time embedded benchmark based on the software of a GNU-license UAV, called Paparazzi. PapaBench contains two sets of tasks, *fly-by-wire* and *autopilot*. In this paper we used the *autopilot* tasks, for which the WCETs, periods, UCBs, and ECBs were collected using aiT[6] – see Table 1. We made the following assumptions in our evaluation to handle the interrupt tasks:

- Interrupts have higher priority than the normal tasks, but they cannot pre-empt each other
- Interrupts can occur at any time
- All interrupts have the same deadline which must be greater than or equal to the sum of their execution times in order for them to be schedulable
- The cache is disabled whenever an interrupt is executing and enabled again after it completes

In the case of FP scheduling, the interrupts can be modelled as normal tasks with no UCBs or ECBs. Due to the interrupts having the same deadline which is large enough to accommodate the interrupts execution times, no other changes need to be made to the analysis. For EDF scheduling, a number of adjustments must be made to correctly account for the interrupts not being able to pre-empt each other. First we modify equation (19) to exclude interrupts when calculating the *processor demand*, $h(t)$. We then calculate the execution time of each interrupt in the interval t using equation (2) of [10]. The result of which is then added onto the result of the modified version

[2] `http://www.irit.fr/recherches/ARCHI/MARCH/rubrique.php3?id_rubrique=97`
[3] `http://www.mrtc.mdh.se/projects/wcet/benchmarks.html`
[4] Esterel SCADE `http://www.esterel-technologies.com/`
[5] `http://www.arm.com/products/processors/cortex-m/cortex-m3.php`
[6] AbsInt `http://www.absint.com/ait/`

of (19), giving the processor demand for both tasks and interrupts. We then adjust the upper bound L used when checking $h(t)$. This is implemented by substituting $U = U^{tasks} + U^{interrupts}$ into equation (30). Note that we leave U^{γ} to represent the utilisation of the CRPD caused by just tasks as we assume that the cache is disabled while the interrupts are executing and as such they cannot cause any CRPD.

We assigned a deadline of 2 ms to all of the interrupt tasks, and implicit deadlines i.e. $D_i = T_i$, to the normal tasks. We then calculated the total utilisation for the system and then effectively scaled the clock speed in order to reduce the total utilisation to the target utilisation for the system. We used the number of UCBs and ECBs obtained via analysis, placing the UCBs in a group at a random location in each task.

In each evaluation, the taskset utilization not including pre-emption cost was varied from 0.025 to 1 in steps of 0.001. For each utilization value, the schedulability of the taskset was determined under both FP and EDF. Specifically, we compared each scheduling algorithm (i) assuming no pre-emption cost, (ii) using CRPD analysis using the standard task layout, and (iii) using CRPD analysis after optimising the task layout using Simulated Annealing as described in [21]. The standard task layout is obtained by ordering tasks sequentially in memory based on their unique task indices.

Table 2 shows the breakdown utilisation for the single taskset based on PapaBench. There are a few interesting points to note. Firstly the breakdown utilisation is very high for both FP and EDF, this is due to the nearly harmonic periods and small range of task periods, with EDF outperforming FP. Secondly, the CRPD is very low when scheduled using either FP or EDF due to the small number of UCBs. As the CRPD is very low, the layout optimisation makes little to no difference.

6.2 Multiple Taskset Case Studies

The single taskset case study provides one specific example based on the PapaBench tasks and their periods. The remaining case studies used tasksets generated by randomly selecting tasks from a set of benchmarks. In the case of the PapaBench tasks, we treated the interrupts as normal tasks. We obtained tasksets by randomly selecting 10 tasks from Table 1 (PapaBench benchmarks), or 10 tasks from Table 3 (Mälardalen and SCADE benchmarks) or 15 tasks from the two tables (Mixed benchmarks). Using the UUnifast algorithm [9], we calculated the utilisation, U_i of each task so that the utilisations added up to the desired utilisation level for the taskset. Based on the target utilisation and task execution times, T_i was calculated such that $C_i = U_i T_i$. We used $D_i = y + x(T_i - y)$ to generate the constrained deadlines, where x is a random number between 0 and 1, and $y = \max(T_i/2, 2C_i)$. This generates constrained deadlines that are no less than half the period of the tasks. Note, allowing deadlines to be as small as C_i would result in tasks that were unschedulable once CRPD were introduced. We used the number of UCBs and ECBs obtained using aiT, placing the UCBs in a group at a random location in each task.

We generated 1000 tasksets for the multiple taskset case studies, and evaluated them using the same method as the single taskset case study, except that we varied the utilisation excluding pre-emption costs from 0.025 to 1 in steps of 0.0125.

6.2.1 PapaBench Benchmark

The tasks in the PapaBench benchmarks are simple, short control tasks with limited computations and data accesses. Figure 6 shows the percentage of tasksets that were deemed schedulable by each approach for the 1000 tasksets of cardinality 10 that we randomly selected from Table 1. The results are similar to those obtained using the single taskset PapaBench case study. Specifically,

Table 1 Execution times, periods and number of UCBs and ECBs for the tasks from PapaBench. (ms = milisecond)

Task	Name	UCBs	ECBs	WCET	Period
I4	interrupt_modem	2	10	0.303 ms	100 ms
I5	interrupt_spi_1	1	10	0.251 ms	50 ms
I6	interrupt_spi_2	1	4	0.151 ms	50 ms
I7	interrupt_gps	3	26	0.283 ms	250 ms
T5	altitude_control	20	66	1.478 ms	250 ms
T6	climb_control	1	210	5.429 ms	250 ms
T7	link_fbw_send	1	10	0.233 ms	50 ms
T8	navigation	10	256	4.432 ms	250 ms
T9	radio_control	0	256	15.681 ms	25 ms
T10	receive_gps_data	22	194	5.987 ms	250 ms
T11	reporting	2	256	12.222 ms	100 ms
T12	stabilization	11	194	5.681 ms	50 ms

Table 2 Breakdown utilisation under the different approaches for the single PapaBench taskset.

	Breakdown Utilisation
EDF – No Pre-emption Cost	0.999
FP – No Pre-emption Cost	0.977
EDF – Optimised Layout	0.985
EDF – Standard Layout	0.985
FP – Optimised Layout	0.970
FP – Standard Layout	0.969

Table 3 Execution times and number of UCBs and ECBs for the largest benchmarks from the Mälardalen Benchmark Suite (M), and SCADE Benchmarks (S). (s = second, ms = milisecond)

Source	Description	UCBs	ECBs	WCET
M	adpcm	24	226	5.541 s
M	compress	25	114	3.664 s
M	edn	56	98	244.9 ms
M	fir	28	50	21.53 ms
M	jfdctinit	40	162	62.53 ms
M	ns	17	26	73.38 ms
M	nsichneu	53	256	149.6 ms
M	statemate	3	256	77.96 ms
S	cruise control system	25	107	1.959 s
S	flight control system	70	256	2.138 s
S	navigation system	45	82	1.409 s
S	stopwatch	58	130	3.786 s
S	elevator simulation	40	114	1.586 s
S	robotics systems	68	256	4.311 s

Table 4 Weighted schedulability measures for the mixed case study shown in Figure 8. The higher the weighted schedulability measure, the more tasksets deemed schedulable by the approach.

	Weighted Schedulability
EDF – No Pre-emption Cost	0.922
FP – No Pre-emption Cost	0.855
EDF – Optimised Layout	0.830
EDF – Standard Layout	0.771
FP – Optimised Layout	0.784
FP – Standard Layout	0.747

EDF outperformed FP as it deemed a higher number of tasksets schedulable at each utilisation level. Because the range of execution times is relatively small, so is the typical range of task periods for the generated tasksets, hence the number of pre-emption is also relatively small. Further, the number of UCBs is small, resulting in low CRPD. Therefore, the task layout optimisation was only able to make a small improvement, but did so for both FP and EDF.

6.2.2 Mälardalen and SCADE Benchmarks

The second multiple taskset case study was based on tasks from the Mälardalen and SCADE benchmarks, shown in Table 3. Compared to the tasks from PapaBench, these tasks have higher execution times, high amounts of computation, and a larger number of UCBs. Figure 7 shows the percentage of tasksets that were deemed schedulable by each approach for the 1000 tasksets of cardinality 10 that we randomly selected from Table 3. As with the PapaBench benchmarks, EDF outperformed FP scheduling as it has a higher percentage of schedulable tasksets at each utilisation level. Likewise, because the range of task periods was also relatively small, CRPD is minimised.

6.2.3 Mixed Benchmarks

The third multiple taskset case study was based on a mixture of the small and short PapaBench tasks, and the large and long Mälardalen and SCADE tasks. Here the tasksets had 15 tasks each, and represent systems with background tasks combined with short control tasks. As we mixed tasks from both tables, it also allowed us to generate tasksets with a higher number of tasks.

The results, shown in Figure 8, show that when a taskset contains tasks with a wide range of periods, CRPD can become a significant factor in the schedulability of the taskset. This is because short high priority tasks are able to pre-empt long running low priority tasks multiple times.

While EDF still outperformed FP, the gain in schedulability of using EDF over FP was diminished once CRPD was taken into account. Optimising the task layout resulted in a significant improvement for both FP and EDF, showing the task layout optimisation can be effectively applied to both EDF and FP scheduling. Furthermore, by optimising the task layout, FP was able to schedule a similar number of tasksets to EDF with the standard layout. In other words, in cases where the CRPD is relatively high, selecting an optimised task layout can be as effective as changing the scheduling algorithm. The results are summarised in Table 4 using weighted schedulability measures [8], see Section 7.2 for details. They show that for these tasksets, FP with an optimised layout achieved a weighted measure of 0.784, outperforming EDF with the standard layout as it achieved a weighted measure of 0.771.

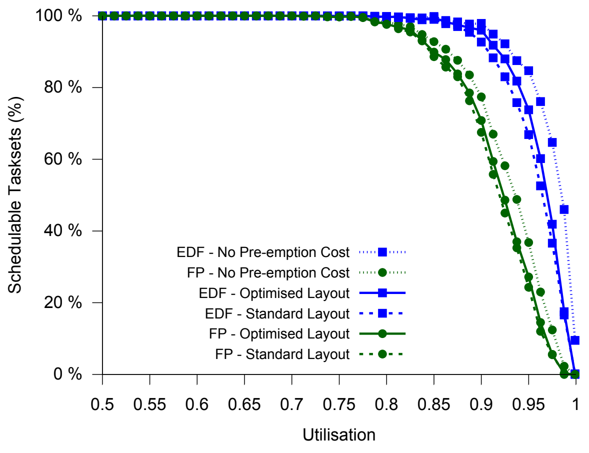

Figure 6 Percentage of schedulable tasksets at each utilisation level for the PapaBench benchmark for tasksets of cardinality 10.

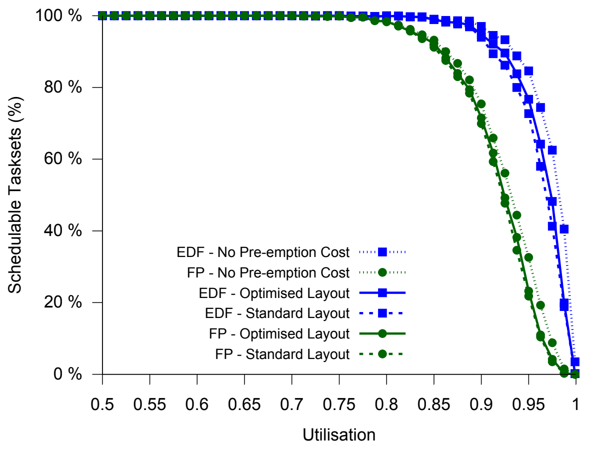

Figure 7 Percentage of schedulable tasksets at each utilisation level for the Mälardalen and SCADE benchmarks for tasksets of cardinality 10.

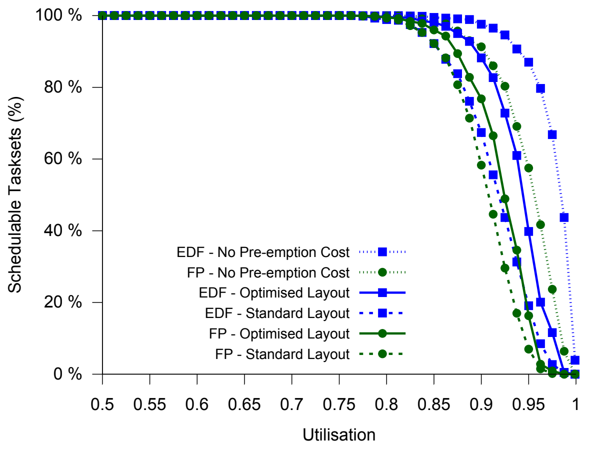

Figure 8 Percentage of schedulable tasksets at each utilisation level for the mixed case study with tasks randomly selected from both the PapaBench and Mälardalen and SCADE benchmarks (taskset cardinality 15).

7 Evaluation

In addition to the case studies based on the PapaBench, Mälardalen and SCADE benchmarks, we evaluated FP and EDF with CRPD analysis using synthetically generated tasksets. This enabled us to investigate the effect of varying key parameters under each scheduling algorithm.

The UUnifast algorithm [9] was again used to calculate the utilisation, U_i of each task so that the utilisations added up to the desired utilisation level for the taskset. Task periods T_i, were generated at random between 5 ms and 500 ms according to a log-uniform distribution. C_i was then calculated via $C_i = U_i T_i$.

We generated two sets of tasksets, one with implicit deadlines and one with constrained deadlines. In the following section, we present the results for constrained deadline tasksets. In general, the results for implicit deadline tasksets gave a higher number of schedulable tasksets for every approach compared to the constrained deadline tasksets. Additionally, the task layout had a similar or slightly larger effect on schedulability in relation to the chosen scheduling algorithm.

The UCB percentage for each task was based on a random number between 0 and a maximum UCB percentage specified for the evaluation. UCBs were split into N groups (where N was chosen at random between 1 and 5), and placed at a random starting point within the task's ECBs.

7.1 Baseline Configuration

To investigate the effect of key cache and taskset configurations we varied the following parameters:

- Cache utilisation (default of 10)
- Maximum UCB percentage (default of 30%)
- Number of tasks (default of 15)
- Block Reload Time (BRT) (default of 8 μs)
- Period range (default of [5, 500] ms)

We used $1,000$ randomly generated tasksets for the evaluation.

In addition to testing the different analyses as done for the case study, we also performed a simulation of the schedule for the tasksets[7]. For FP, the simulation tested each task τ_i in turn by releasing it at time $t = 0$. It then released all of the other tasks that have a higher priority than task τ_i, sorted by lowest to highest priority, at $t = 1$, $t = 2$, $t = 3$, etc. If all tasks were schedulable it also performed the same test but instead of staggering the other tasks, released them at random. For EDF, it is more complicated to generate the worst case arrival pattern. The first step is to determine the interval that needs to be checked, L, which can be achieved by using equation (30). Then for each task τ_i in turn, we scheduled a job of task τ_i so that it has a deadline at $t = L$. We then scheduled a job of all of the other tasks, sorted by longest to shortest deadline, so that they have their deadlines at $t = L - 1$, $t = L - 2$, $t = L - 3$ etc... Based on the final jobs' deadlines, we then calculated when the first jobs for each task need to be released. If all tasks are schedulable, we repeated the process using $t = L - 1$ for all of the other tasks' jobs, and also using a random schedule.

The results for the baseline configuration are shown in Figure 9 and are summarised in Table 5 using weighted schedulability measures. The results follow a similar pattern to the results from the case study. EDF outperformed FP finding a higher number of tasksets schedulable. The results for the simulations show that the CRPD affects both FP and EDF, with the CRPD being slightly lower for EDF. Specifically, the simulation shows that CRPD reduced the weighted measure by at least 0.129 for EDF (0.925 – 0.795) and 0.141 for FP (0.774 – 0.633) in this case. However, once

[7] Note that the simulation effectively provides a necessary, but not sufficient test of schedulability.

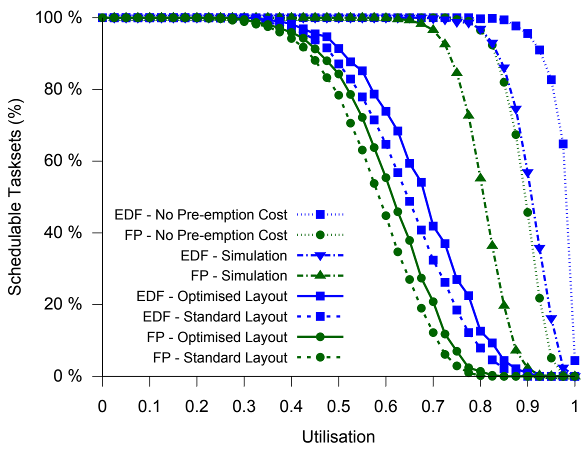

Figure 9 The percentage of schedulable tasksets at each utilisation level for the baseline configuration (taskset cardinality 15).

Table 5 Weighted schedulability measures for the baseline configuration study shown in Figure 9. The higher the weighted schedulability measure, the more tasksets deemed schedulable by the approach.

	Weighted Schedulability
EDF – No Pre-emption cost	0.925
EDF – Simulation	0.796
FP – No Pre-emption cost	0.774
FP – Simulation	0.633
EDF – Optimised layout	0.455
EDF – Standard layout	0.413
FP – Optimised layout	0.369
FP – Standard layout	0.336

the CRPD obtained via analysis is taken into account, the performance gains of using EDF over FP are diminished. This is most likely caused by increased pessimism in the CRPD analysis for EDF. The results for the layout optimisation showed that it was able to make improvements to the schedulability of tasksets scheduled under both FP and EDF.

7.2 Weighted Schedulability

Evaluating all combinations of different parameters is not possible. Therefore, the majority of our evaluations focused on varying one parameter at a time. To present the results, weighted schedulability measures [8] are used. This allows a graph to be drawn which shows the weighted schedulability, $W_l(p)$, for each method used to obtain a layout l as a function of parameter p. For each value of p, this measure combines the data for all of the generated tasksets τ for all of a set of equally spaced utilisation levels, where the utilisation is based on no pre-emption cost.

The schedulability test returns a binary result of 1 or 0 for each layout at each utilisation level. If this result is given by $S_l\ (\tau, p)$, and $u(\tau)$ is the utilisation of taskset τ, then:

$$W_l(p) = \frac{\left(\sum_{\forall_\tau} u(\tau) S_l(\tau, p)\right)}{\sum_{\forall \tau} u(\tau)}. \tag{31}$$

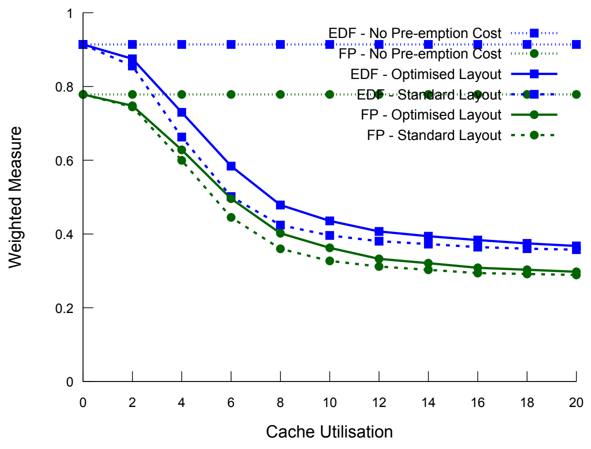

Figure 10 Weighted measure for varying the cache utilisation from 0 to 20 in steps of 2.

The benefit of using a weighted schedulability measure is that it reduces a 3-dimensional plot to 2 dimensions. Individual results are weighted by taskset utilisation to reflect the higher value placed on a being able to schedule higher utilisation tasksets.

7.2.1 Cache Utilisation

As the cache utilisation increases the likelihood of tasks evicting each other from cache increases, this causes higher CRPD reducing the number of schedulable tasksets. It can be seen in Figure 10 that task layout optimisation is effective for FP and EDF across the same range of cache utilisations. In both cases it becomes less effective once the cache utilisation becomes high. We note that because the number of tasks is fixed, that the average size of the tasks is equal to the cache utilisation divided by the number of tasks. This means that as the cache utilisation increases, so does the size of the tasks and therefore, the number of UCBs. This in turn makes it harder to find an improved layout.

7.2.2 Maximum UCB Percentage

As the maximum UCB percentage increases, the CRPD increases resulting in a reduction in the number of tasksets that are deemed schedulable, as can be seen in Figure 11. With a low percentage of UCBs, the CRPD is low which means there is little benefit from layout optimisation. When the UCB percentage is very high, there are so many conflicts that there is very little that can be done to improve the layout. When the maximum UCB percentage is around 40–60%, there is a notable amount of CRPD, but there is also room for the task layout algorithm to optimise the layout. This allows FP using an optimised task layout to schedule a similar number of tasksets as EDF using the standard layout.

7.2.3 Number of Tasks

When varying the number of tasks, Figure 12, we scaled the cache utilisation to keep the average size of tasks constant based on a cache utilisation of 10 for 15 tasks. This is because it would be unrealistic for the size of tasks to decrease as more tasks are added to the system. Hence with 8 tasks the cache utilisation is equal to 5.33, whereas for 32 tasks, it is equal to 21.33. As the number of tasks increases, it becomes harder the schedule all tasks which leads to a decrease

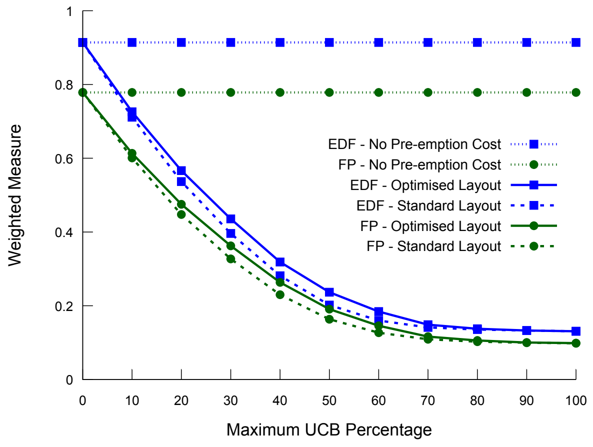

Figure 11 Weighted measure for varying the maximum UCB percentage from 0 to 100 in steps of 10.

in the overall weighted measure. The task layout optimisation performs best when there is a moderate number of tasks, as there are enough conflicts that optimising the layout can give an improvement, but not so many that there is nothing that can be done to avoid the conflicts.

7.2.4 Block Reload Time

As the block reload time is increased, it becomes more costly to reload a block, which causes an increase in CRPD. It can be seen in Figure 13 that as the block reload time is increased, the analysis that takes into account the pre-emption cost shows a decrease in the overall weighted measure. We note that as the cost of reloading a block increases, the potential gains of optimising the layout increase. Once the block reload time exceeds 14 μs, using an optimised layout under FP scheduling outperforms using a standard layout under EDF scheduling.

7.2.5 Period Range

We also investigated the effect of the scaling factor used to generate task periods to simulate tasksets with shorter to longer execution times. We varied the scaling factor, w, from 0.5 to 10 and hence the range of task periods given by $w[1, 100]$ ms. A lower scaling factor resembles tasks with shorter execution times, as seen in the PapaBench benchmark, and a higher scaling factor resembles tasks with higher execution times and commensurately longer periods, as seen in the Mälardalen and SCADE benchmarks. The results in Figure 14 show the layout optimisation performs best when task periods are relatively short, as that is when the pre-emption costs are highest. Once the period range is greater than [10, 1000] ms, the relative pre-emption costs are low enough that performing the layout optimisation only makes a very small improvement on the schedulability of the tasksets.

8 Conclusion

The EDF scheduling algorithm is an optimal scheduling algorithm for single processors however, it has been largely disregarded by industry. Whereas FP, despite offering lower theoretical schedulable processor utilisation, is relatively popular with many commercial real- time operating systems supporting it.

Previous work by Buttazzo [13] has compared the two algorithms, but it did not take into account CRPD which can have a significant effect on the schedulability of a taskset.

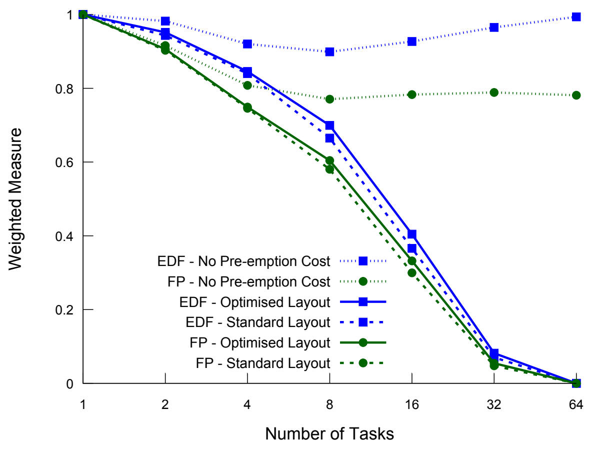

Figure 12 Weighted measure for varying the number of tasks from 2^0 to 2^6 while maintaining a constant ratio of number of tasks to cache utilisation.

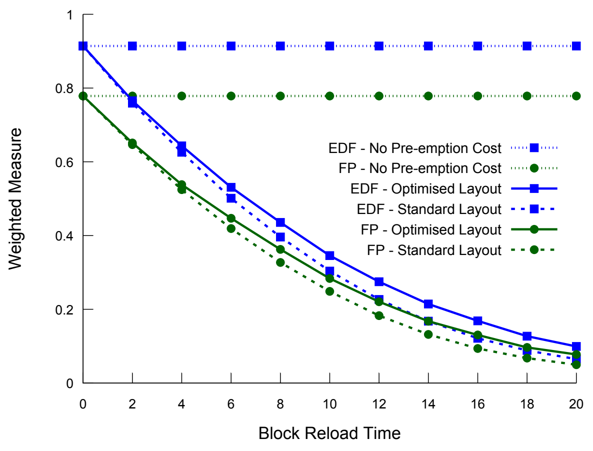

Figure 13 Weighted measure for varying the block reload time from 0 to 20 μs in steps of 2.

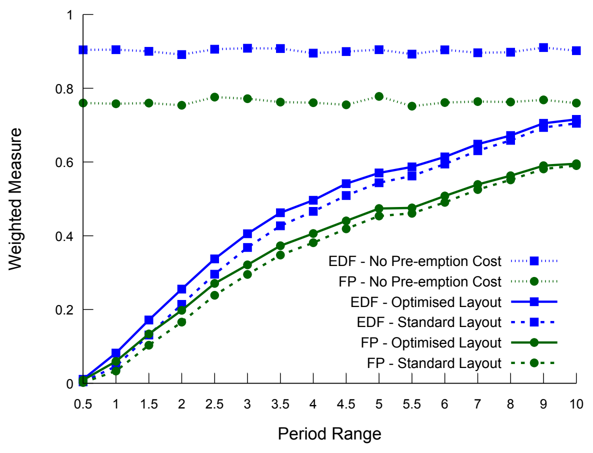

Figure 14 Weighted measure for varying the scaling factor used to generate periods, w, in w[1, 100] ms, from 0.5 to 10.

The main contributions of this paper are:

- Performing a detailed comparison of FP and EDF taking into account CRPD using state-of-the-art CRPD analysis [3, 22].
- Showing the feasibility of simple, yet effective, task layout optimisation techniques for EDF.
- Finding that when CRPD is considered, the performance gains offered by EDF over FP, while still significant, are somewhat diminished. This is most likely due to greater pessimism in the CRPD analysis for EDF than FP.
- Discovering that in configurations that cause relatively high CRPD, optimising task layout can be just as effective as changing the scheduling algorithm from FP to EDF.

We investigated the effects of performing task layout [21] optimisation based on Simulated Annealing under both FP and EDF scheduling algorithms. We found that in the scenarios that cause the pre-emption cost to be relatively high in relation to task execution times, applying task layout optimisation to a system scheduled using FP scheduling can allow it to be schedulable at a similar processor utilisation compared to using EDF scheduling with a standard layout. This is important in an industrial setting as it is considerably simpler and cheaper to control the task layout via the linker, than it is to change the scheduler. Nevertheless, our evaluations showed that changing to an EDF scheduler and optimising the task layout provides a gain over FP scheduling. Although this gain was not as pronounced as the advantage that EDF has over FP when pre-emption costs are not accounted for via analysis.

In the future we plan to further investigate techniques for CRPD analysis, and to apply them in an industrial context comparing the results of analysing a real system with those obtained via measurement.

Grant Information. This work was partially funded by the UK EPSRC through the Engineering Doctorate Centre in Large-Scale Complex IT Systems (EP/F501374/1), the UK EPSRC funded MCC (EP/K011626/1), and the European Community's ARTEMIS Programme and UK Technology Strategy Board, under ARTEMIS grant agreement 295371-2 CRAFTERS.

References

1 Sebastian Altmeyer and Claire Burguière. A new notion of useful cache block to improve the bounds of cache-related preemption delay. In *21st Euromicro Conference on Real-Time Systems, ECRTS 2009, Dublin, Ireland, July 1–3, 2009*, pages 109–118. IEEE Computer Society, 2009. doi:10.1109/ECRTS.2009.21.

2 Sebastian Altmeyer, Robert I. Davis, and Claire Maiza. Cache related pre-emption delay aware response time analysis for fixed priority pre-emptive systems. In *Proceedings of the 32nd IEEE Real-Time Systems Symposium, RTSS 2011, Vienna, Austria, November 29 – December 2, 2011*, pages 261–271. IEEE Computer Society, 2011. doi:10.1109/RTSS.2011.31.

3 Sebastian Altmeyer, Robert I. Davis, and Claire Maiza. Improved cache related pre-emption delay aware response time analysis for fixed priority pre-emptive systems. *Real-Time Systems*, 48(5):499–526, 2012. doi:10.1007/s11241-012-9152-2.

4 Sebastian Altmeyer, Claire Maiza, and Jan Reineke. Resilience analysis: tightening the CRPD bound for set-associative caches. In *Proceedings of the ACM SIGPLAN/SIGBED 2010 Conference on Languages, Compilers, and Tools for Embedded Systems, LCTES 2010, Stockholm, Sweden, April 13–15, 2010*, pages 153–162. ACM, 2010. doi:10.1145/1755888.1755911.

5 Neil C. Audsley, Alan Burns, Mike F. Richardson, Ken Tindell, and Andrew J. Wellings. Applying new scheduling theory to static priority pre-emptive scheduling. *Software Engineering Journal*, 8(5):284–292, 1993. URL: http://ieeexplore.ieee.org/stamp/stamp.jsp?tp=&arnumber=238595&isnumber=6134.

6 Sanjoy K. Baruah, Aloysius K. Mok, and Louis E. Rosier. Preemptively scheduling hard-real-time sporadic tasks on one processor. In *Proceedings of the Real-Time Systems Symposium – 1990, Lake Buena Vista, Florida, USA, December 1990*, pages 182–190. IEEE Computer Society, 1990. doi:10.1109/REAL.1990.128746.

7 Sanjoy K. Baruah, Louis E. Rosier, and Rodney R. Howell. Algorithms and complexity concerning the preemptive scheduling of periodic, real-time tasks on one processor. *Real-Time Systems*, 2(4):301–324, 1990. doi:10.1007/BF01995675.

8 Andrea Bastoni, Björn B. Brandenburg, and James H. Anderson. Cache-related preemption and migration delays: Empirical approxim-

ation and impact on schedulability. In *Proceedings of the 6th International Workshop on Operating Systems Platforms for Embedded Real-Time Applications, OSPERT 2010*, pages 33–44, 2010. URL: `http://www.mpi-sws.org/~bbb/papers/pdf/ospert10.pdf`.

9 Enrico Bini and Giorgio C. Buttazzo. Measuring the performance of schedulability tests. *Real-Time Systems*, 30(1-2):129–154, 2005. `doi:10.1007/s11241-005-0507-9`.

10 Björn B. Brandenburg, Hennadiy Leontyev, and James H. Anderson. Accounting for interrupts in multiprocessor real-time systems. In *15th IEEE International Conference on Embedded and Real-Time Computing Systems and Applications, RTCSA 2009, Beijing, China, 24–26 August 2009*, pages 273–283. IEEE Computer Society, 2009. `doi:10.1109/RTCSA.2009.37`.

11 Claire Burguière, Jan Reineke, and Sebastian Altmeyer. Cache-related preemption delay computation for set-associative caches - pitfalls and solutions. In *9th International Workshop on Worst-Case Execution Time Analysis, WCET 2009, Dublin, Ireland, July 1–3, 2009*, volume 10 of *OASICS*, pages 1–11. Schloss Dagstuhl – Leibniz-Zentrum fuer Informatik, Germany, 2009. `doi:10.4230/OASIcs.WCET.2009.2285`.

12 José V. Busquets-Mataix, Juan José Serrano, Rafael Ors, Pedro J. Gil, and Andy J. Wellings. Adding instruction cache effect to schedulability analysis of preemptive real-time systems. In *2nd IEEE Real-Time Technology and Applications Symposium, RTAS 1996, June 10–12, 1996, Boston, MA, USA*, page 204. IEEE Computer Society, 1996. `doi:10.1109/RTTAS.1996.509537`.

13 Giorgio C. Buttazzo. Rate monotonic vs. EDF: judgment day. *Real-Time Systems*, 29(1):5–26, 2005. `doi:10.1023/B:TIME.0000048932.30002.d9`.

14 Robert I. Davis, A. Zabos, and Alan Burns. Efficient exact schedulability tests for fixed priority real-time systems. *IEEE Transactions on Computers*, 57(9):1261–1276, 2008. `doi:10.1109/TC.2008.66`.

15 Michael L. Dertouzos. Control robotics: The procedural control of physical processes. In *IFIP Congress*, pages 807–813, 1974.

16 Laurent George, Nicolas Rivierre, and Marco Spuri. Preemptive and non-preemptive real-time uniprocessor scheduling. Technical report, INRIA, 1996. URL: `http://hal.inria.fr/inria-00073732`.

17 Lei Ju, Samarjit Chakraborty, and Abhik Roychoudhury. Accounting for cache-related preemption delay in dynamic priority schedulability analysis. In *2007 Design, Automation and Test in Europe Conference and Exposition, DATE 2007, April 16–20, 2007, Nice, France*, pages 1623–1628. ACM, 2007. URL: `http://dl.acm.org/citation.cfm?id=1266366.1266723`.

18 Chang-Gun Lee, Joosun Hahn, Yang-Min Seo, Sang Lyul Min, Rhan Ha, Seongsoo Hong, Chang Yun Park, Minsuk Lee, and Chong-Sang Kim. Analysis of cache-related preemption delay in fixed-priority preemtive scheduling. *IEEE Trans. Computers*, 47(6):700–713, 1998. `doi:10.1109/12.689649`.

19 Joseph Y.-T. Leung and M. L. Merrill. A note on preemptive scheduling of periodic, real-time tasks. *Information Processing Letters*, 11(3):115–118, 1980. `doi:10.1016/0020-0190(80)90123-4`.

20 C. L. Liu and James W. Layland. Scheduling algorithms for multiprogramming in a hard-real-time environment. *Journal of the ACM*, 20(1):46–61, 1973. `doi:10.1145/321738.321743`.

21 Will Lunniss, Sebastian Altmeyer, and Robert I. Davis. Optimising task layout to increase schedulability via reduced cache related preemption delays. In *20th International Conference on Real-Time and Network Systems, RTNS 2012, Pont a Mousson, France – November 08–09, 2012*, pages 161–170. ACM, 2012. `doi:10.1145/2392987.2393008`.

22 Will Lunniss, Sebastian Altmeyer, Claire Maiza, and Robert I. Davis. Integrating cache related preemption delay analysis into EDF scheduling. In *19th IEEE Real-Time and Embedded Technology and Applications Symposium, RTAS 2013, Philadelphia, PA, USA, April 9–11, 2013*, pages 75–84. IEEE Computer Society, 2013. `doi:10.1109/RTAS.2013.6531081`.

23 Ismael Ripoll, Alfons Crespo, and Aloysius K. Mok. Improvement in feasibility testing for real-time tasks. *Real-Time Systems*, 11(1):19–39, 1996. `doi:10.1007/BF00365519`.

24 Marco Spuri. Analysis of deadline scheduled real-time systems. Technical report, INRIA, 1996. URL: `http://hal.inria.fr/inria-00073920`.

25 Jan Staschulat, Simon Schliecker, and Rolf Ernst. Scheduling analysis of real-time systems with precise modeling of cache related preemption delay. In *17th Euromicro Conference on Real-Time Systems, ECRTS 2005, 6–8 July 2005, Palma de Mallorca, Spain*, pages 41–48. IEEE Computer Society, 2005. `doi:10.1109/ECRTS.2005.26`.

26 Yudong Tan and Vincent John Mooney III. Timing analysis for preemptive multitasking real-time systems with caches. *ACM Transactions on Embedded Computing Systems (TECS)*, 6(1), 2007. `doi:10.1145/1210268.1210275`.

27 Fengxiang Zhang and Alan Burns. Schedulability analysis for real-time systems with EDF scheduling. *IEEE Transactions on Computers*, 58(9):1250–1258, 2009. `doi:10.1109/TC.2009.58`.