technische universität
dortmund

Bachelorthesis

# Towards Exact Analysis of EDF-Like Scheduling

Komron Abdulloev

16.08.2024

Supervisors:
Prof. Dr. Jian-Jia Chen
Mario Günzel, M.Sc.

Technische Universität Dortmund
Fakultät für Informatik
Lehrstuhl Informatik 12 (Eingebettete Systeme)
Design Automation for Embedded Systems Group
https://daes.cs.tu-dortmund.de

# Contents

# 1 Introduction

This first chapter introduces the problems we are inspecting in this thesis and motivates our work, outlines its structure and states the research questions we set out to answer.

## 1.1 Motivation

In the realm of real-time systems, ensuring that tasks are executed within strict timing constraints is crucial for maintaining system reliability and performance. This plays a significant role specifically for hard real-time tasks, where missing a task's deadline can cause massive consequences on the system. Therefore, scheduling algorithms are designed to manage these tasks effectively, ensuring that deadlines are met and system resources are utilized efficiently.

Hard real-time scheduling algorithms can be broadly categorized into static and dynamic approaches. The static approach, otherwise known as offline scheduling, creates a scheduler before runtime, needing complete prior knowledge about the task set characteristics. The dynamic approach, otherwise known as online scheduling, makes scheduling decisions during execution, choosing from the currently available tasks. Dynamic schedulers are adaptable and can adjust to changing task conditions, but they are not deterministic[KS22; But24].

Dynamic schedulers execute tasks based on a priority system. Furthermore, there exists static-priority and dynamic-priority. Static-priority scheduling algorithms, such as Deadline-Monotonic Scheduling [LW82], assign priorities to tasks based on fixed criteria determined before the actual system execution. Dynamic-priority scheduling algorithms, on the other hand, like Earliest-Deadline-First[LL73], adjust priorities at runtime based on changing parameters, meaning that over time, the priorities of the tasks change.

In this thesis, we focus on the exactness of a scheduling algorithm, namely EDF-Like. EDF-Like scheduling is a class of scheduling algorithms. More specifically, a set of so-called relative priority points $\Pi_i$ determines the behavior of the EDF-Like scheduling. With a proper configuration of the priority points, the scheduler can behave like many established scheduling algorithms, including Fixed-Priority (FP) scheduling [LL73], First-In-First-Out (FIFO), and EDF as well.

From the previous work of Günzel et al. [Gün+22], a unifying suspension-aware schedulability test for uniprocessor EDF-Like has been provided, guaranteeing that no task misses its deadline using the EDF-Like scheduling algorithm.

However, it is unclear how close that test is to achieving exactness. Therefore, the primary objective of this thesis is to evaluate the performance of this schedulability test by conducting a comprehensive analysis through simulations.

## 1.2 Structure

This Section 1 briefly introduces our work and states the research questions we seek to answer.

The Section 2 addresses the background knowledge required to understand our work. This includes explaining *static-priority* and *dynamic-priority* scheduling and their differences. Then, afterward, we will explain how EDF-Like scheduling itself works.

In Section 3, we review our research approach. We will briefly explain how we will evaluate the performance of the schedulability test and the methods we use to synthetically generate task sets.

In Section 4, we cover our first research question by comparing the schedulability test of EDF-Like with exact tests. That would be the utilization-based test for dynamic-priority scheduling and Time-Demand Analysis (TDA)[JP86][LSD89] for static-priority analysis.

In Section 5, we dive further into evaluating the schedulability test of EDF-Like by experimenting with varying quantities of tasks given to the test to determine their impact.

In Section 6, we determine whether changing a task's period significantly impacts its schedulability.

We wrap up our work in Section 7, where we draw conclusions and briefly discuss possible future work to improve the performance of the schedulability test.

## 1.3 Research Questions

The main focus of this thesis is to evaluate the exactness of the EDF-Like schedulability test. However, to this moment, only sufficient but not necessary tests exist for self-suspending tasks. Since such a benchmark does not exist for us to compare the schedulability test of EDF-Like, we will consider tasks without self-suspension.

Bearing this in mind, we will first and foremost question the exactness of EDF-Like's schedulability compared to other exact tests but with tasks that lack self-suspension, which is covered in Section 4. Afterward, we look closer into the impact of varying the parameters on the schedulability. Namely, in Section 5, the task quantity of a task set, and in Section 6, the task's period.

# 2 Background

In this chapter, we provide background knowledge to understand the work of this thesis. We start by reviewing task models and scheduling algorithms essential to the field, as well as the schedulability test of EDF-Like. This foundation will help contextualize the following discussions and analyses presented in later chapters.

## 2.1 Task Models

Many real-time systems work around periodic or sporadic activities, such as sensory data acquisition or system monitoring. For this reason, these activities' tasks have their own models. The tasks of periodic activities recur in pre-defined fixed intervals, so they are periodic. The main difference between sporadic and periodic tasks is that sporadic tasks occur irregularly. However, they do have a minimum inter-arrival time. This means there is a guaranteed minimum time between consecutive instances of the task. For the following, we go over the classical task model for real-time systems.

A set of tasks can be denoted by $\mathbb{T} = \{\tau_1, ..., \tau_n\}$ for $n \in \mathbb{N}$. A single instance of a task $\tau_i$ is called a job and can be denoted as $\tau_{i,j}$. The first job of a task $\tau_i$ would then be $\tau_{i,1}$.

A periodic[LL73] or sporadic[Mok83] task $\tau_i$ itself can be characterised by a 3-tuple $(C_i, T_i, D_i)$, where

- the **execution time** $C_i$, denotes the worst-case execution time of a single job
- the **period** $T_i$, denotes the period or the minimum inter-arrival time, which is essentially the time between consecutive releases of a job
- the **deadline** $D_i$, denotes the relative deadline for each job from task $\tau_i$

For this thesis, we assume that the first job of every task will be released simultaneously at the start. So overall, a task $\tau_i$ can be viewed as releasing an endless series of jobs that get released at $T_i$ intervals with an execution time of $C_i$ and a deadline $D_i$.

The **utilization** $U_i$ of a periodic task $\tau_i$ is defined to be the ratio of its execution requirement to its period: $U_i := \frac{C_i}{T_i}$. The total utilization for the task system $\mathbb{T}$ is the sum of the utilization of all tasks in $\mathbb{T}$: $U(\mathbb{T}) := \sum_{\tau_i \in \mathbb{T}} U_i$.

Another important definition is the **response time** $R_{i,j}$ of a job $\tau_{i,j}$, which is the amount of time that has passed between the release of the job and the moment it has finished its execution.

Since we have reviewed the classical task model, we can introduce self-suspension task models. Two main models are present when considering self-suspension: *dynamic* and *segmented*. The dynamic self-suspension model is used in the paper from Günzel et al. [Gün+22].

Self-suspending tasks are those that voluntarily suspend their execution during runtime, typically to wait for external events or resources before resuming and completing their execution. A dynamic self-suspending task is similar to a sporadic task but with an additional parameter. Therefore a task $\tau_i$ can be described as a 4-tuple $(C_i, S_i, T_i, D_i)$, where $S_i$ would be the maximum suspension time. Each job $\tau_{i,j}$ can then suspend itself whenever it needs to and, however, as often, as long it does not exceed $S_i$.

## 2.2 Scheduling Algorithms

Scheduling algorithms themselves can be either non-preemptive or preemptive. Non-preemptive scheduling algorithms do not allow interruption of a currently executing task. Once a task starts, it runs to completion before the scheduler considers the next task. With preemptive scheduling algorithms, the system can interrupt the currently executing task to allocate the CPU to a higher-priority task that arrives or becomes ready. This allows the system to adapt to changing conditions and prioritize urgent tasks. We will only consider preemptive scheduling since the EDF-Like schedulability test is for preemptive scheduling.

### 2.2.1 Static-Priority Scheduling

In static-priority or fixed-priority scheduling, all the jobs generated by a single task must be assigned to a single priority. This priority should differ from the priorities assigned to other jobs generated by other tasks. Priorities are assigned as numbers to tasks. So therefore, a task $\tau_i$ has priority $i$. The lower the value of $i$, the higher the priority of that specific task. This allows us to represent the priorities of tasks in a clear and simple way. So, for instance, the highest priority task would be $\tau_1$.

So now the question remains: Given a set of tasks, how should we assign the priorities? One of the most well-known priority assignment algorithms for fixed-priority scheduling is rate-monotonic (RM)[LL73]. It is based on assigning priorities to tasks based on their request rates, so in this case, those would be their periods: shorter-period tasks receive higher priorities.

For task sets configured with implicit-deadline, meaning every task has its relative deadline parameter $D_i$ equal to its period $T_i$, RM was shown to be optimal. This means that given a set of tasks, if any static-priority scheduling method can successfully meet all deadlines, then the RM algorithm will likewise achieve this.

RM has a utilization bound, otherwise known as the Liu and Layland Bound [LL73], which is defined as follows:

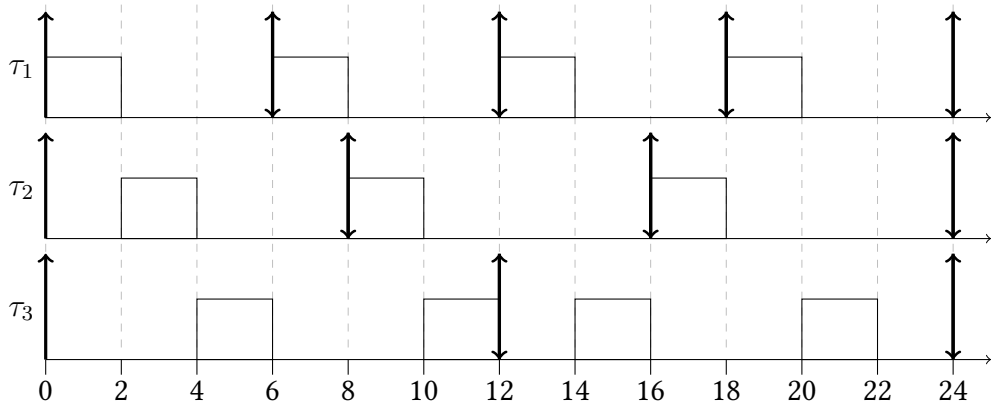
$$U_B = n(2^{1/n} - 1)$$

where $n$ is the number of tasks. This bound is derived from the mathematical properties of RM and represents the maximum CPU utilization at which the set of $n$ periodic tasks is guaranteed to be schedulable. When $n$ converges to infinity, then $U_B = \ln 2 \approx 0.693$. Therefore, if the total utilization of all the tasks is less than or equal to $U_B$, all tasks are guaranteed to meet their deadlines. Otherwise, it is unclear if the tasks can be scheduled if $U(\mathbb{T}) > U_B$.

For the task set $\mathbb{T}$ with tasks $\tau_i = (C_i, T_i, D_i)$:

- $\tau_1 = (2, 6, 6)$
- $\tau_2 = (2, 8, 8)$
- $\tau_3 = (4, 12, 12)$

for the time interval $[0, 24]$, we would obtain the following schedule produced by RM:

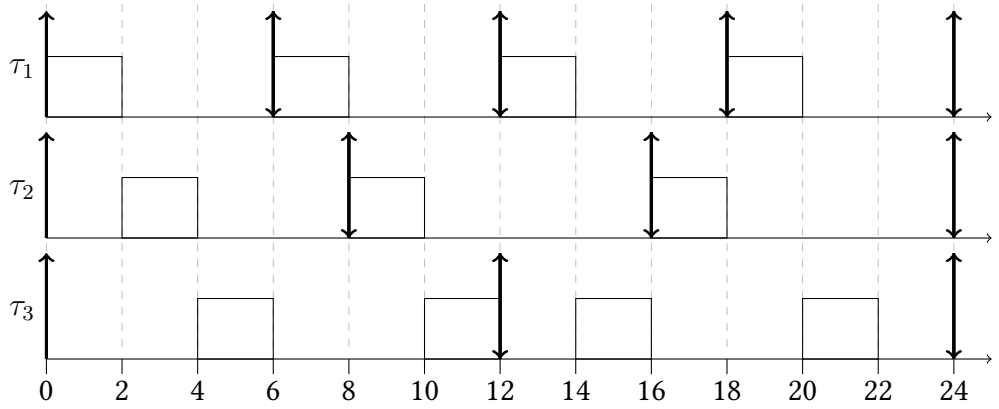



We can observe from the above schedule that the tasks are prioritized by their periods, i.e., $\tau_1$ always preempts other lower-priority tasks in order to be executed since it has the smallest period. Here, $U(\mathbb{T}) = U_1 + U_2 + U_3 \approx 0.917$ and the utilization bound is $U_B = 3 \cdot (2^{1/3} - 1) \approx 0.78$. As we can observe, although $U(\mathbb{T}) > U_B$, the task set is still schedulable. That is because the utilization bound is only a sufficient test.

Although Rate-Monotonic is a simple way of finding out if a set of tasks is schedulable or not, most of the time, real-time systems can encounter complex scenarios where task deadlines do not align perfectly with their periods. To deal with such constrained-deadline ($D_i \leq T_i$) task sets, Deadline-Monotonic (DM), an extension of Rate-Monotonic, can be applied[LW82].

DM sets task priorities based on their deadlines. Therefore, the task with the shortest relative deadline $D_i$ is executed at any point in time. Similarly to Rate-Monotonic, Deadline-Monotonic is optimal for fixed-priority scheduling.

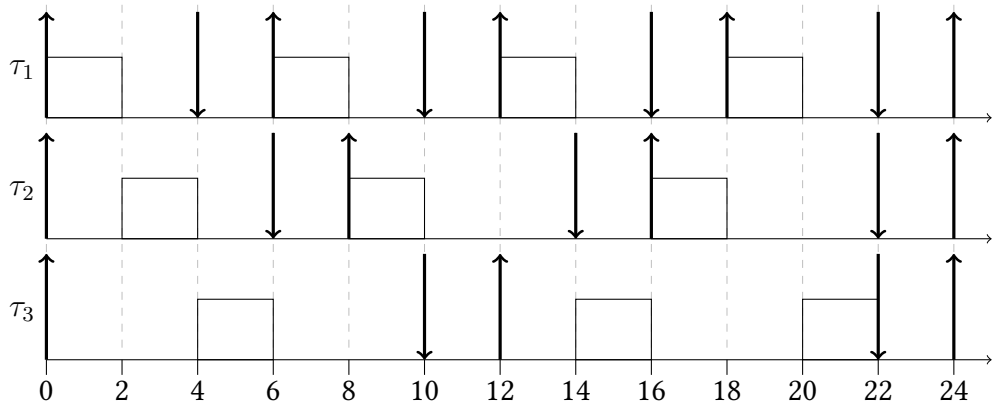
From the same example task set $\mathbb{T}$ that we used for RM, we get the following schedule produced by DM:




Since, in the example, the relative deadline of the tasks is equal to their period, DM schedules the tasks similarly. Here, the deadline is prioritized instead of their periods. However, if we would consider the same task set $\mathbb{T}$, but lower the deadline of every task by 2:

- $\tau_1 = (2, 6, 4)$
- $\tau_2 = (2, 8, 6)$
- $\tau_3 = (4, 12, 10)$

we get the following schedule:




We notice that for $\tau_3$, after starting to execute at $t = 4$, $\tau_1$ preempts $\tau_3$ and afterward $\tau_2$ executes since it becomes ready, therefore $\tau_3$ misses its deadline at $t = 10$. Another thing to notice is that both for RM and DM, the priorities of the tasks do not change once they are assigned to them.

### 2.2.2 Dynamic-Priority Scheduling

When it comes to dynamic-priority scheduling, Earliest Deadline First (EDF) takes the spotlight. EDF is quite similar to DM when it comes to assigning priorities to tasks. However, these priorities may change throughout the scheduling process. EDF chooses the currently

active job with the smallest deadline at each instant in time. To put it differently, it assigns priorities to the task according to their absolute deadline:

$$d_{i,j} = (j-1)T_i + D_i$$

For periodic implicit deadline task sets, the schedulability test of EDF consists of using the processor utilization factor[LL73]:

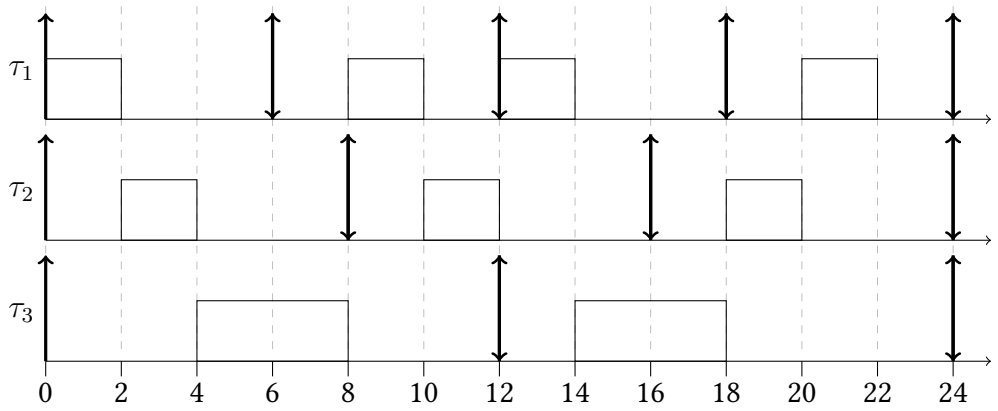
$$\sum_{i=1}^{n} \frac{C_i}{T_i} \leq 1$$

Therefore, EDF can guarantee a task set $\tau_i$ is schedulable if and only if the system total utilization is not more than 100%, making EDF optimal.

For the same task set $\mathbb{T}$ with tasks $\tau_i = (C_i, T_i, D_i)$ used for RM:

- $\tau_1 = (2, 6, 6)$
- $\tau_2 = (2, 8, 8)$
- $\tau_3 = (4, 12, 12)$

for the time interval $[0, 24]$, we would obtain the following schedule produced by EDF:




Since we know that the total utilization of the task set is 0.917, which is smaller than 1, with EDF, we know that it is schedulable, and as we can see from above, that is the case. Whenever a scheduling event occurs, i.e., a task finishes or a new task gets released, EDF searches for the task that is closest to its deadline and executes it. At $t = 6$, we can observe that both $\tau_1$ and $\tau_3$ have the same absolute deadline. Therefore, EDF may schedule either one of them by choosing arbitrarily.

## 2.3 EDF-Like Scheduling

Now, let us introduce the EDF-Like (EL) schedulability test, which will be the main focus of this thesis. The work by Günzel et al. [Gün+22] provides the first suspension-aware schedulability test for uniprocessor EDF-Like scheduling that generalizes both Fixed Priority (FP) and Earliest Deadline First (EDF)[LL73] scheduling algorithms. Therefore, by analyzing EL, we could potentially identify scenarios where it outperforms both FP and EDF, leading to a deeper understanding of the trade-offs and benefits of various scheduling strategies.

We can look at EDF-Like scheduling as a class of scheduling algorithms. More specifically, a set of so-called relative priority points $\Pi_i$ determines the behavior of the EDF-Like scheduling. Each job $\tau_{i,j}$ is assigned its own priority point $\pi_{i,j}$ which is created as follows:

$$\pi_{i,j} = r_{i,j} + \Pi_i$$

where $r_{i,j}$ is the release time of the job and $\Pi_i$ is the relative priority point of task $\tau_i$. Therefore a job $\tau_{i,j}$ has higher priority than $\tau_{i',j'}$ if $\pi_{i,j} < \pi_{i',j'}$.

With a proper configuration of the priority points, the scheduler can behave like many established scheduling algorithms including Fixed-Priority (FP) scheduling, First-In-First-Out (FIFO), and EDF as well. Hence, by assigning $\Pi_i = D_i$, we would get an EDF scheduler, and by having $\Pi_i = \sum_{j=1}^{i} D_j$, the schedulability test behaves as DM.

### 2.3.1 Issues with Self-Suspension

The sufficient schedulability test provided works on self-suspending tasks with arbitrary-deadline, meaning their relative deadline $D_i$ could be larger than the period $T_i$ for some task $\tau_i$. However, adding self-suspension to tasks introduces complexities that can lead to problems when using scheduling algorithms like EDF and RM [Che+19].

It has already been shown that both EDF and RM are optimal. However, this property does not hold anymore when self-suspending tasks are applied. Furthermore, it was shown by Ridouard, Richard, and Cottet [RRC04] that even when the suspending behavior of the task set is known, finding an optimal schedule is NP-Hard.

The duration of the suspension can be unpredictable, leading to difficulties in accurately predicting the remaining time for task completion. Therefore, this affects the accuracy of the worst-case response time (WCRT) analysis, which becomes more complicated because it must now consider the suspension periods, which can vary widely.

Since there are no exact tests under self-suspension, it is difficult to evaluate the performance and exactness of EDF-Like's schedulability test. However, to somehow evaluate the schedulability test, we will be considering task sets without self-suspension. That way, we can use the already well-known exact tests that exist for fixed-priority and dynamic-priority scheduling.

### 2.3.2 Applied EDF-Like Configuration

The test works around the idea of bounding the worst-case response time of a task $\tau_i$ by analyzing a job $\tau_{i,l}$. This is achieved by bounding the time it takes the job to run, the time it can be suspended, and also taking into account the possible interference from tasks with higher priority and as well as jobs of the same task $\tau_i$, all during a specific interval $[c, d_{i,l})$ with $d_{i,l}$ being the absolute deadline of the task $\tau_i$.

We will mostly focus on applying the sufficient schedulability test of EDF-Like to implicit-deadline task sets ($D_i = T_i$), and since we will not cover self-suspending tasks, the maximum suspension time $S_i$ of each task will be set to 0.

The paper [Gün+22] presents two different versions of the suspension-aware schedulability test that can be applied to EL scheduling algorithms. The first one is the *fixed analysis window*, where $c$ is restricted to the release time of the job $\tau_{i,l}$, giving us the following interval, which will be analyzed: $[r_{i,l}, d_{i,l})$. The other version is the *variable analysis window*, where active intervals are used. Here, the release times of earlier jobs are applied as $c$, therefore increasing the analysis window.

In addition, the paper has shown that if constrained-deadline task sets are used, then the variable analysis window matches the fixed analysis window method. Regardless, for our work, we will center our attention on the fixed interval algorithm:

**Algorithm 1** Schedulability test with fixed analysis window [Gün+22, Algorithm 1]]

**Input:** $\mathbb{T} = \{\tau_1, ..., \tau_n\}$, $(\Pi_1, ..., \Pi_n)$, $\eta$, $depth$
**Output:** True: schedulable, False: no decision

1: Order $\tau_1, ..., \tau_n$, such that $D_1 \geq \cdots \geq D_n$.
2: Set $\tilde{R}_i := D_i$ for all $i$.
3: **for** $i = 1, 2, ..., depth$ **do**
4: $\quad solved := True$
5: $\quad$ **for** $k = 1, 2, ..., n$ **do**
6: $\quad\quad cand := [\,]$; $step := \eta \cdot D_k$
7: $\quad\quad$ **for** $b = 0, step, 2 \cdot step, \cdots < D_k$ **do**
8: $\quad\quad\quad cand.append(\tilde{R}_k(b))$ using Equation (2.1).
9: $\quad\quad \tilde{R}_k := min(cand)$
10: $\quad\quad$ **if** $\tilde{R}_k > D_k$ **then**
11: $\quad\quad\quad solved := False$; $\tilde{R}_k := D_k$; **break**
12: **return** $solved$

As input, the algorithm takes in a task set $\mathbb{T}$, their relative priority points $(\Pi_1, ..., \Pi_n)$, a step parameter $\eta \in (0, 1]$ and the parameter $depth$, that specifies the number of improving runs. By iterating $depth$-times, the algorithm refines the response times by considering different values of $b_k$ in the range $[0, D_k)$. The $\eta$ parameter influences the step size in the loop. Then, the algorithm evaluates the possible response times for each task using Equation (2.1) and selects the minimum to define the new $\tilde{R}_k$. If for any task $\tau_k$, the minimum response time exceeds its deadline $D_k$, the algorithm returns $False$, meaning that it is undecidable if the task set is schedulable or not. Otherwise, it returns $True$.

The given algorithm utilizes the following theorem of the schedulability test:

**Theorem 1** (Sufficient Schedulability Test [Gün+22, Theorem 12]). *Let $\mathbb{T} = \{\tau_1, \ldots, \tau_n\}$ be an arbitrary-deadline task set with relative priority points $\{\Pi_1, \ldots, \Pi_n\}$. If for all $k = 1, \ldots, n$ there exists some $b_k \in [0, D_k)$ such that*

$$\tilde{R}_k(b_k) \leq D_k, \tag{2.1}$$

*where*

$$\tilde{R}_k(b_k) := \sum_{i \neq k} \max\left(\left\lceil \frac{G_k^i + \tilde{R}_i - b_k}{T_i} \right\rceil, 0\right) C_i + \left\lceil \frac{D_k - b_k}{T_k} \right\rceil (C_k + S_k) + b_k$$

*and $G_k^i = \min(D_k - C_i, \Pi_k - \Pi_i)$, then the task set is schedulable by EL scheduling with the given relative priority points and the worst-case response time of $\tau_k$ is upper bounded by $\tilde{R}_k := \tilde{R}_k(b_k)$.*

## 2.4 Exact Schedulability Tests

So far, we covered that there are utilization bounds both for FP and EDF. For the case that we need an exact test for dynamic-priority scheduling, we use as a comparison for EDF-Like, the utilization bound of EDF[LL73], since we are dealing with only (constrained/implicit)-deadline task sets.

For fixed-priority scheduling, however, we have covered that rate-monotonic has the utilization bound, which can be used to determine if a task set is schedulable. However, as shown in the example for scheduling with RM, this utilization bound is only a sufficient test, meaning there can be false negatives. Therefore, we need an exact test, such as Time-Demand Analysis (TDA) [LSD89].

### 2.4.1 Time-Demand Analysis

Before we explain TDA, one crucial part of the schedulability is the *Critical Instant Theorem*, which was introduced by Liu and Layland [LL73]. The theorem states that the worst-case response time for a task in a preemptive, fixed-priority scheduling system occurs when the task is released simultaneously with all higher-priority tasks. This scenario is referred to as the *critical instant*.

TDA uses this concept to compute the worst-case response time by evaluating the demand on the processor during this critical instant. It calculates the cumulative processor time required by a job released at a critical instant of a task, along with the processor time demanded by all other higher-priority tasks, as a function of time from the critical instant.

The time-demand function itself is defined as follows:

$$W_i(t) = C_i + \sum_{j=1}^{i-1} \left\lceil \frac{t}{T_j} \right\rceil C_j.$$

If the tasks are sorted by priority, then the schedulability test can be expressed by the following theorem:

**Theorem 2** (Lehoczky, Sha, Ding, 1989 [LSD89]). *A set of fully preemptive periodic tasks can be scheduled by a fixed priority algorithm if and only if:*

$$\forall i = 1, ..., n \qquad \exists t \in (0, D_i] : W_i(t) \leq t$$

# 3 Methodology

In this chapter, we will briefly explain the configurations of EDF-Like being used and how we will conduct our experiments.

## 3.1 Execution of Research

We use empirical methods to compare the schedulability tests. This involves assessing them by analyzing a wide range of task sets with varying levels of utilization. By generating properly designed task sets to examine the performance of schedulability tests, we can provide a fair and unbiased comparison. Then, by presenting graphs, it makes it possible to interpret the results that we receive from applying the schedulability tests on the task sets[Dav16].

To illustrate the performance of the schedulability tests, we plot the acceptance/success ratio. This is the proportion of task sets that are deemed to be schedulable by each test against utilization. This means that for each specific value of the utilization from 0% to 100%, the proportion of schedulable task sets is plotted. This is the primary method we use to conduct the tests. However, for some tests, we have different approaches, which we explain in detail in those chapters.

We also provide the code that we have written for all the following tests, which is available on GitHub[1].

## 3.2 Task Generation

Let us go through the process of how the task sets that will be used for the experiments are created.

We generate a n-dimensional vector using the Dirichlet-Rescale (DRS) algorithm [GBD]. This vector will then be used as the utilization for a task set $\mathbb{T}$. As input to the algorithm, we give the number of tasks per task set as $n$ and as well as the utilization target sum $U(\mathbb{T})$. In return, the algorithm generates a uniformly distributed n-dimensional vector, where the values of it sum up to $U(\mathbb{T})$[GBD20]. Thus, a task $\tau_i$ would have the utilization $U_i$.

Now, it is left to create an array of task periods of size $n$. Therefore, a log-uniform distribution with the range of [1, 100] will be applied to generate them[ESD].

[1] https://github.com/komronm8/BachelorThesis

Once we have the generated utilization array and the task period array, we use these two arrays to create the worst-case execution time $C_i$ of task $\tau_i$. Since $U_i = \frac{C_i}{T_i}$, we simply multiply the value at index $i$ from our utilization array with the value at index $i$ of our generated array of periods to get $C_i$.

Since this thesis will work primarily with implicit-deadline task sets, the deadlines will be equal to their periods. Therefore, we now have a suitable set of tasks to conduct our experiments and obtain reliable data to analyze.

# 4 Comparison to Exact Tests

In this chapter, we compare the schedulability test of EDF-Like with other known schedulability tests, and we assess the accuracy of EL.

## 4.1 Objective of the Experiment

The experiment aims to evaluate the effectiveness of the EDF-Like algorithm, configured in different ways (as EDF[LL73] and as a fixed-priority scheduler like Deadline-Monotonic [LW82]), by comparing its schedulability outcomes with those of exact tests like TDA[JP86; LSD89]. Also understand how over-approximation in the EDF-Like algorithm affects its ability to correctly determine whether a set of tasks is schedulable, especially under varying utilization levels. The experiment also slightly explores how the performance of both the EL schedulability test and exact tests changes when different parameters are applied.

## 4.2 Anticipated Outcomes

We look into configuring the relative priority points of EDF-Like such that it will behave like EDF ($\Pi_i = D_i$) and FP ($\Pi_i = \sum_{j=1}^{i} D_j$), then evaluate its performance. As mentioned, we will use the utilization-based test for the comparison with EL-EDF (EDF-Like configured as EDF), and for EL-DM (EDF-Like configured as Deadline-Monotonic), TDA will be the opposing exact test.

We know from the paper that the EL schedulability test uses over-approximation to compute the following two values:

- $\sum_{j<l} WS_{k,j}$ — the total amount of interference from previous jobs of $\tau_k$
- $\sum_{i \neq k} B^i_{k,l}$ — the total amount of interference from all other jobs of $\tau_i$ that have higher priority than $\tau_k$

during the interval $[c, d_{k,l})$. This is done since computing their exact values leads to high complexity.

With this in mind for both EL-EDF and EL-DM, we can forecast that for low to moderate utilization, the outcome of the schedulability test should be most of the time *schedulable* since the system is underutilized and has sufficient capacity to handle the tasks. For moderate to high utilization, we can expect that the success ratio will start to fall because of the over-approximation. Hence, it becomes problematic for the algorithm to find a valid $b_k$ such that the $\tilde{R}_k \leq D_k$.

## 4.3 Resulting Data

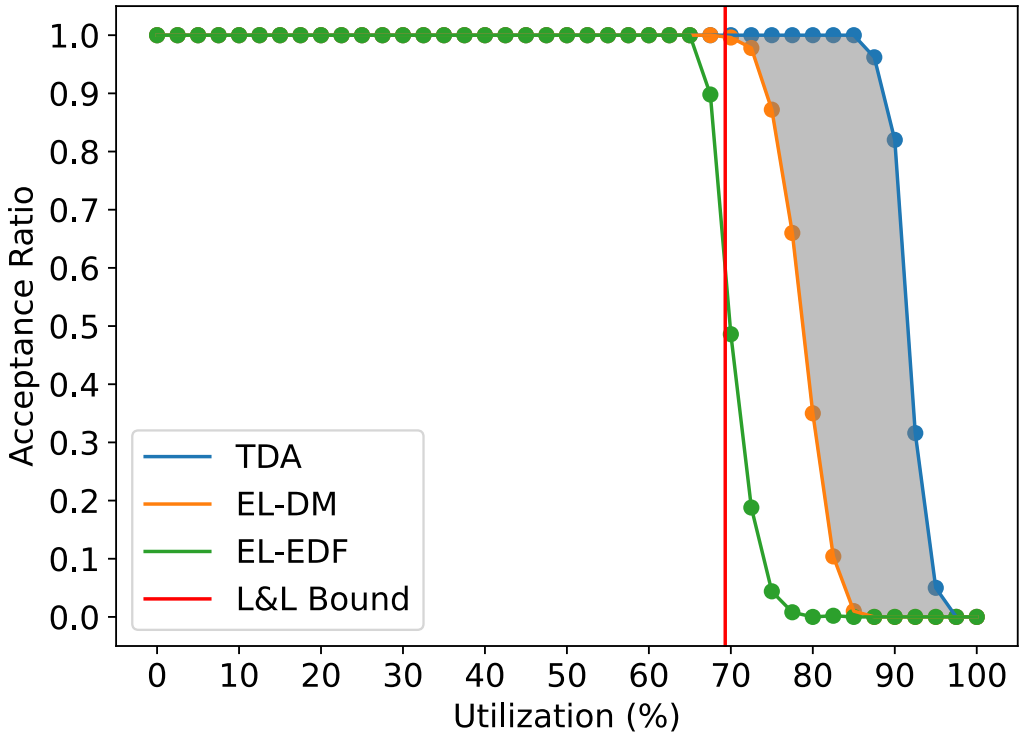



Figure 4.1: Overall Exact Test Comparison

For Figure 4.1, a total of 500 task sets, with each set containing 50 tasks, was generated to obtain the acceptance ratio per utilization, starting from 0% to 100% with a utilization step of 2.5. As we can observe, the acceptance ratios at the beginning are at 1.0, meaning that all of the 500 task sets were deemed *schedulable*, then at around 65-75% utilization, the success ratio for both tests starts to drop, which was expected.

We can see that TDA outperforms both EL-DM and EL-EDF. As the paper has already pointed out that Theorem 2.1 has similarity with TDA, we can likewise notice that EL-DM has resembling patterns with TDA. However, because of the over-approximation by EDF-Like, the schedulability ratio differs in the area with high utilization. The shaded area between EL-DM and TDA highlights the differences between the two.

Additionally, we've included the utilization bound from Liu and Layland [LL73], serving as a benchmark. Considering the utilization bound, we notice that the success rate of EL-DM starts to drop right after it exceeds the utilization bound. TDA, on the other hand, goes beyond the L&L bound, indicating that it can handle high utilization compared to the theoretical limit set by L&L. For EL-EDF, however, the schedulability begins to decrease even before the bound.

### 4.3.1 Follow-up Experiments

Next, we look into the results obtained from the tests with different parameters.

First, we only varied the number of tasks. We carried out three tests for quantities 5, 10, and 20. From the subfigures in Figure 4.2, we notice that with only five tasks per task set, both tests perform better. However, with the increase in the number of tasks in a task set, the schedulability of both EL-EDF and EL-DM decreases, and the difference (shaded area) between TDA and EL-DM becomes larger.

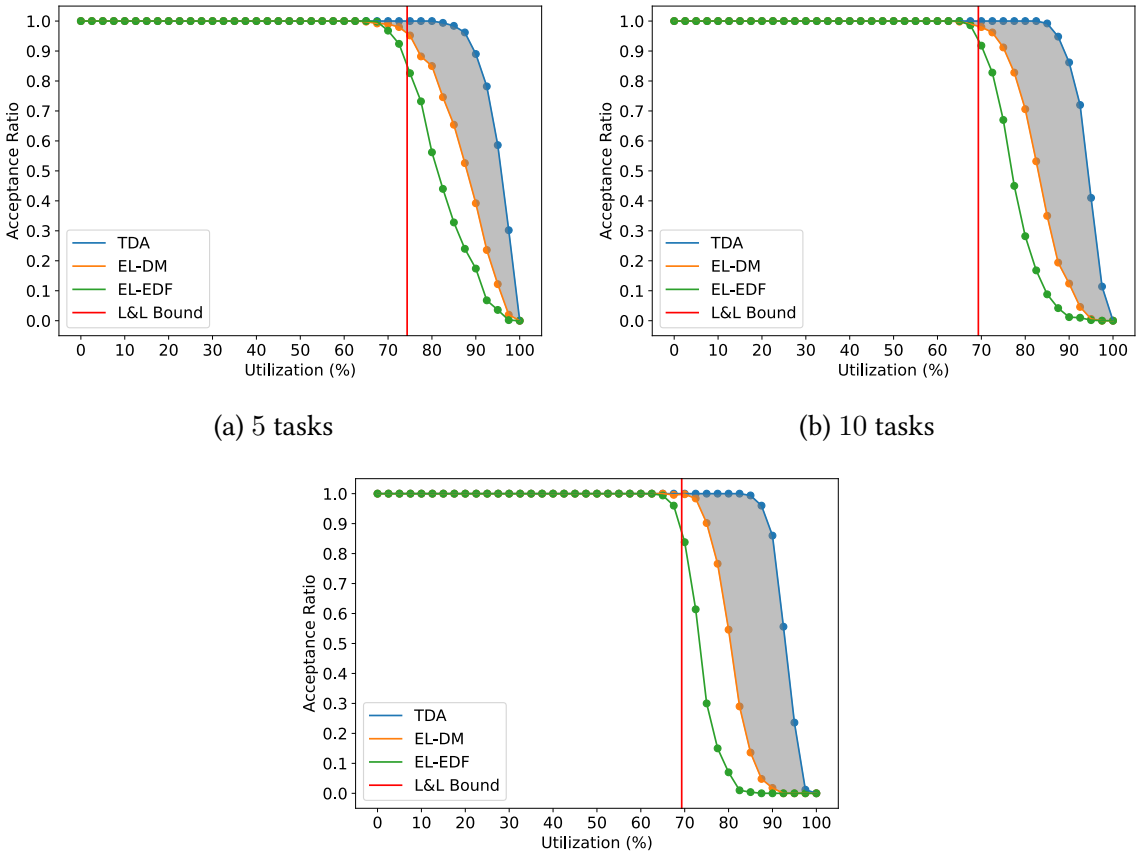
Next, we varied the period range when generating the period array of a task set with the log-uniform distribution. In Figure 4.3, there are two subfigures, one where the period range was configured to [1,10] and another with [1,1000]. By increasing the upper bound of the period range from 10 to 1000, we observe that the schedulability of EL-EDF, EL-DM, and TDA increases substantially. Additionally, we can notice that EL-DM performs slightly worse for the period range [1,10] than [1,1000] when compared to TDA.

## 4.4 Summary

The primary focus of this chapter was to assess how accurate EDF-Like's schedulability test was to exact tests. The experiment's results reveal that while the EDF-Like algorithm performs similarly to TDA under low to moderate utilization, its accuracy diminishes as utilization increases due to the inherent over-approximation in its calculations. However, it performs pretty well considering that EDF-Like's schedulability test is sufficient only. For instance, it was shown that EL-DM outperforms the utilization bound of fixed-priority scheduling. All in all, EL-DM is not far off from the exact test TDA, whereas EL-EDF has a much larger difference in accuracy when compared to the utilization-based test.

Moreover, we explored the effects of varying task set parameters, such as the number of tasks and the period range. In contrast to TDA, increasing the number of tasks per set decreased schedulability for both EL-EDF and EL-DM configurations. On the other hand, significantly expanding the period range improved the schedulability outcomes across all algorithms. Therefore, to analyze these effects in depth, we dedicate a chapter to each one: Sections 5 and 6.

One unresolved question that we did not address is why EL-DM appears to outperform EL-EDF, which was not expected from our side. Therefore, we noticed this only after the data was presented. Due to time constraints, we could not investigate this unresolved question and find the necessary answer thoroughly. However, given the required time, this would certainly be an essential area for further research.

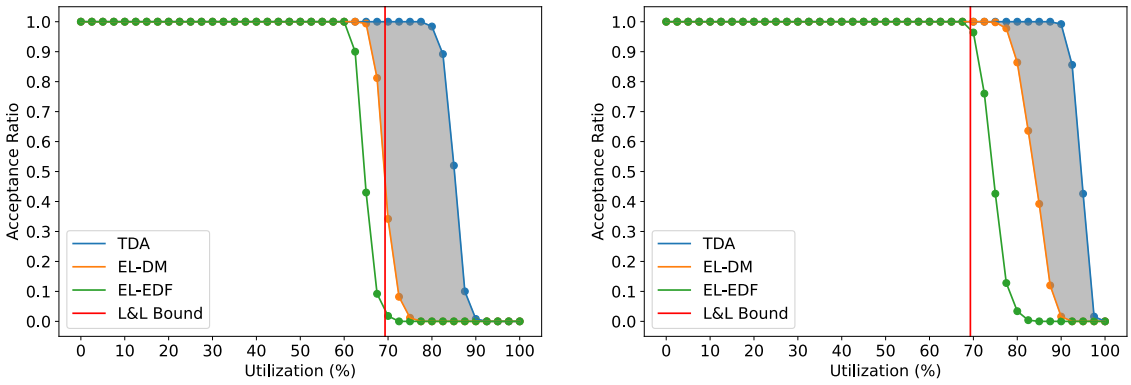



(a) 5 tasks

(b) 10 tasks

(c) 20 tasks

Figure 4.2: Impact of task quantity on schedulability




(a) Period range $[1, 10]$

(b) Period range $[1, 1000]$

Figure 4.3: Impact of period range on schedulability

# 5 Impact of Task Quantity

In this chapter, we analyze the impact of varying the task quantity per task set on the performance and behavior of EDF-Like's schedulability.

## 5.1 Objective of the Experiment

As already discussed in Section 4, we further explore the schedulability impact by looking into these parameters. The procedure for conducting this experiment will be similar to that of the previous chapter. We generate task sets and run the schedulability test on them to determine if they are schedulable. However, we pay attention to the number of tasks per task set in this case. The quantity of the tasks themselves will vary, but the number of task sets will not change. For all tests, we will be generating a total of 100 task sets for each run.

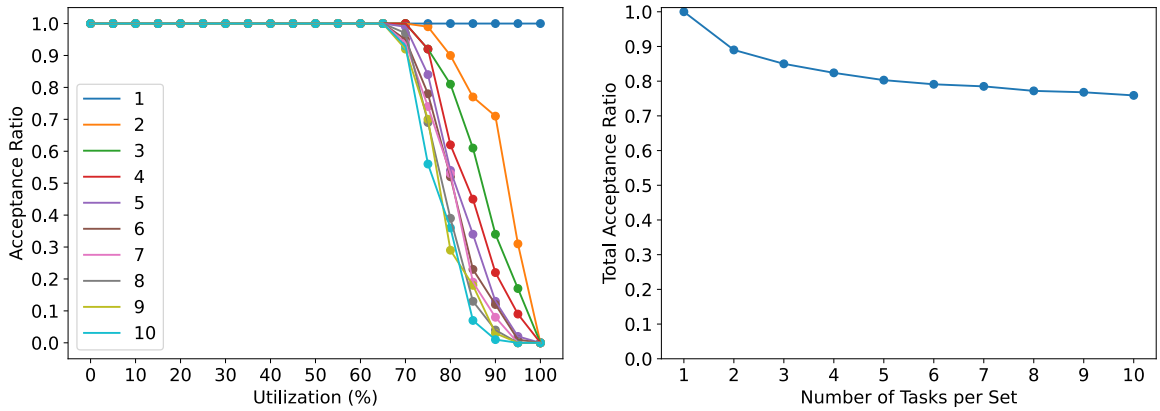
## 5.2 Anticipated Outcomes

As previously mentioned, EDF-Like works based on computing the worst-case response time of a job $\tau_k$ by estimating the total interference from previous jobs of $\tau_k$ and higher priority tasks. Therefore, with more tasks, each task may experience more interference from other tasks, increasing the likelihood of blocking and leading to delays and potential missed deadlines. With this in mind, we can anticipate that with the rise in task quantity, the schedulability of the task set will decrease.

## 5.3 Resulting Data

First, we consider starting with small quantities of tasks per set, specifically the range 1 to 10. We will begin with 1 task and gradually increase by one until we have 10 tasks.

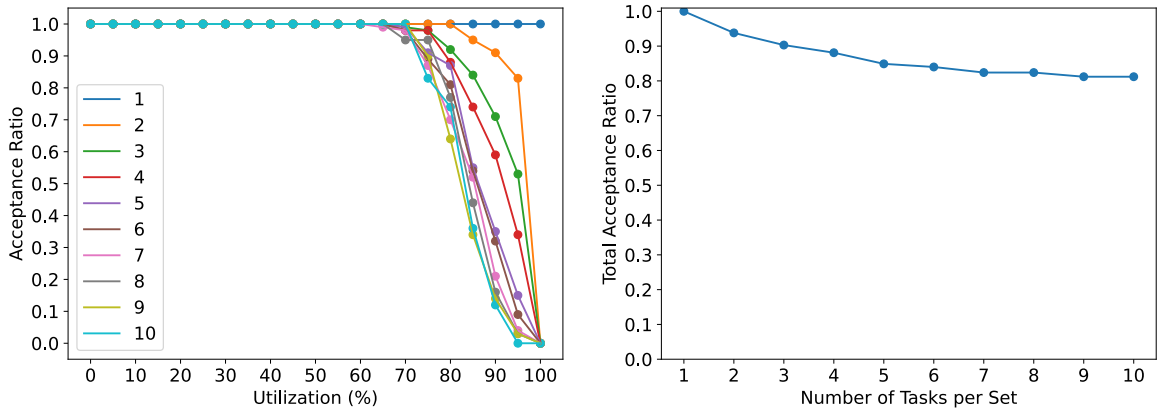
As expected, we immediately notice from Figure 5.1a that the acceptance ratio decreases step by step for each run as the task quantity per set increases. It represents the results in a satisfactory manner. However, the plotted data seems messy since we have many runs. Therefore, to visualize the graph better, we applied a different approach. Figure 5.1b shows the *total success ratio* of each run against the *task quantity*, meaning starting from the utilization level 0 to level 100, their acceptance ratios have been aggregated.




(a) Initial approach (b) Alternative approach

Figure 5.1: Quantity Test EL-EDF with [1-10] tasks

As we can observe, the decline of the total schedulability ratio is clearly depicted in Figure 5.1b in comparison to Figure 5.1a. Now we examine EDF-Like configured to fixed-priority (FP) scheduling:

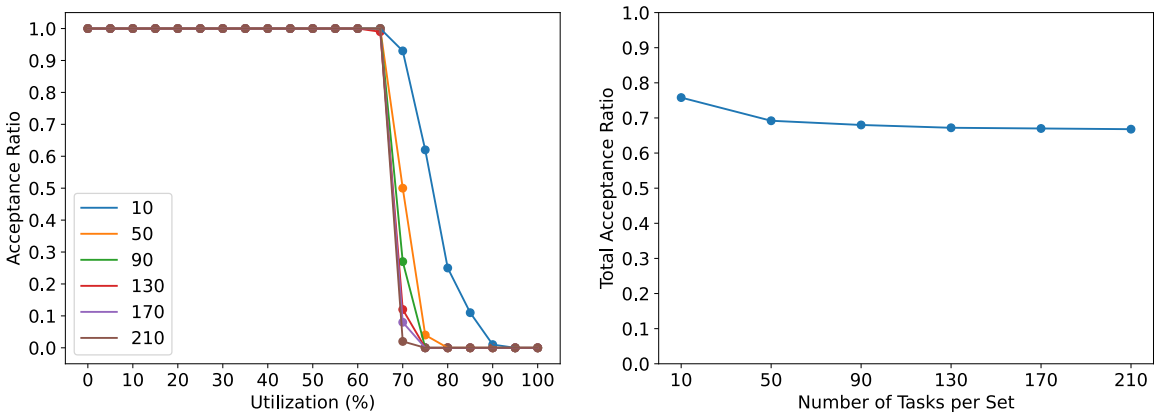



(a) Initial approach (b) Alternative approach

Figure 5.2: Quantity Test EL-DM with [1-10] tasks

Here, we observe the same pattern in Figures 5.2a and 5.2b as EDF-Like configured to EDF. The only difference is fixed-priority outperforming EDF, which we noted in the previous chapter.

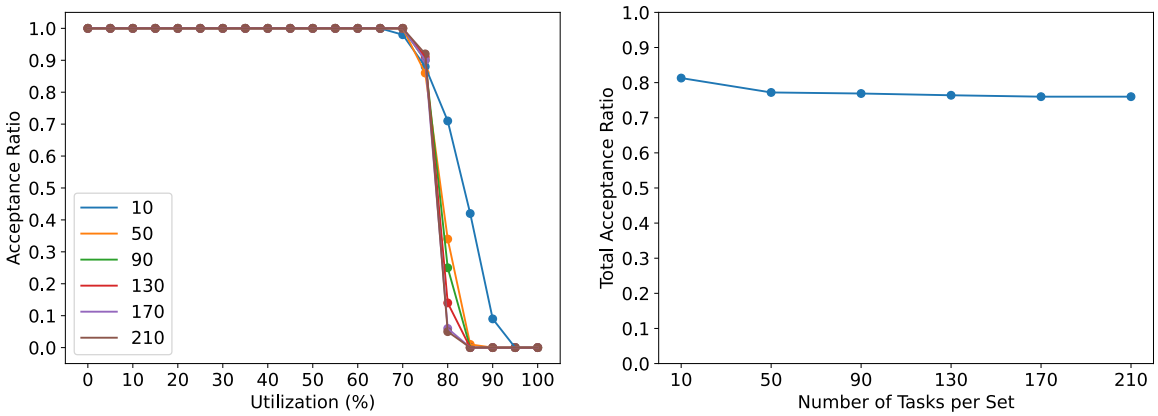
Next, we consider applying more significant amounts of tasks in Figure 5.3. We start with 10 tasks per set and gradually increase by 40 tasks until we reach 210 tasks per set. At first glance, the run with only 10 tasks per task set has the highest schedulability ratio compared to the others. As expected, with increasing task quantity, the schedulability gradually drops. However, in Figure 5.3a, we notice that the last five runs with the highest task quantities (50 to 210) seem to have clustered together. It appears that the success ratio doesn't decrease anymore, and some lower bound seems to exist. Furthermore, in Figure 5.3b, we see that the total acceptance ratio converges to approximately 0.67.




(a) Initial approach (b) Alternative approach

Figure 5.3: Quantity Test EL-EDF with large amounts of tasks

Next, we examine the outcome of EDF-Like configured to fixed-priority in Figure 5.4:




(a) Initial approach (b) Alternative approach

Figure 5.4: Quantity Test EL-DM with large amounts of tasks

As usual, fixed-priority is outperforming EDF. Here, we can similarly point out that the total acceptance ratio decreases with the increase in task quantity. However, again, only in the beginning do we observe a more or less substantial drop with the rise in the number of tasks, but then the decline immensely slows down.

Nevertheless, the same behavior from Figure 5.3a repeats itself in Figure 5.4a, with the higher quantity tasks grouping up together and having a similar success ratio pattern. Similarly to EDF, it appears that the decrease of the total success ratio with respect to the increase in task quantity has diminished and converged to around 0.76.

## 5.4 Summary

In this chapter, we covered the impact of varying task quantities per task set on the schedulability of the EDF-Like algorithm. The results from the quantity test between the range [1,10] revealed a clear trend: as the number of tasks per set increases, the overall schedulability decreases, which we anticipated.

This decline was evident in smaller task sets. However, larger sets showed a convergence in schedulability, indicating a possible lower bound. This suggests that as task quantity increases beyond a certain point, additional tasks have a diminishing impact on schedulability, leading to a stabilized success ratio, which converged to around 0.67 for EDF and 0.76 for fixed-priority scheduling. One possible reason could be that the system reached a point where the interference caused by new tasks no longer substantially increased since the maximum possible delays bound the worst-case interference. However, due to the lack of time in this thesis, this could be further explored and should be considered for future work.

# 6 Impact of Period Variation

In this chapter, we further study the impact of the periods on the schedulability.

## 6.1 Objective of the Experiment

Similarly to Section 5, we explore in depth the phenomenon that we noticed in Section 4 while increasing the period range. Therefore, to reduce the scope to fewer parts and have a clear setup, we will consider only two tasks and vary the periods between them. This will allow us to have a better view of what is happening.

## 6.2 Anticipated Outcomes

As we have already explained, we know that the schedulability test takes into account the interference from higher priority jobs $\tau_i$ when upper bounding the worst-case response time of $\tau_k$. This can be further visualized by the following figure:

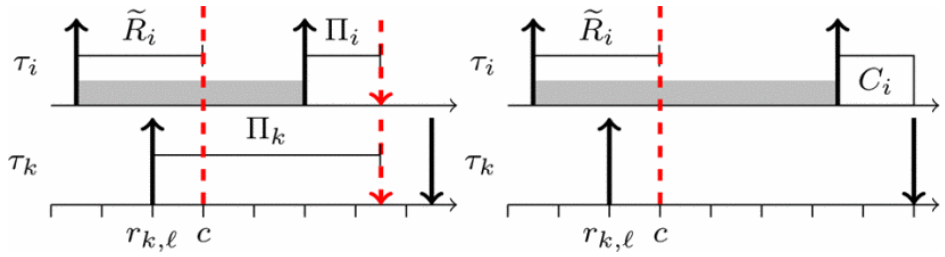



Figure 6.1: Visualization of interference from higher priority jobs from [Gün+22]
©2022 IEEE

The gray boxes represent the permitted region where jobs from $\tau_i$ that have a higher priority than $\tau_{k,l}$ can be executed within the analysis interval $[c, d_{k,l})$. Bearing this in mind, it suggests that the analysis heavily depends on the period of the tasks, as the key question is how many jobs can fit in the gray region, including the carry-in job at the start right before $r_{k,l}$. This carry-in region is upper-bounded by the period of the higher-priority job and depends on how well the periods are aligned. Therefore, if $\tau_k$'s period becomes larger and larger, the carry-in job becomes insignificant in comparison to the job under analysis $\tau_{k,l}$.

As already mentioned, we will be examining only two tasks per task set. In this case, the first task $\tau_1$ will always be assigned the period $T_1 = 100$, and the period of the second task $\tau_2$, starting with $T_2 = 100$, will be gradually increased. The utilization levels, on the other hand, will stay fixed.

## 6.3 Resulting Data

To start out, we plotted the results from the tests as usual, with each run increasing the period of $\tau_2$ by a certain degree. We noticed that for both EL-EDF and EL-DM, increasing the period of $\tau_2$ led to the acceptance ratio oscillating back and forth. Just by looking at the graph, the oscillation looked random at first, but with further inspection, it showed a specific pattern. Namely, whenever the period $T_2$ is furthest from being a multiple of $T_1$, the total schedulability ratio is at its lowest. Conversely, the overall success ratio increases as $T_2$ gets closer to being a multiple of $T_1$.

Therefore, plotting the acceptance ratio as a function of the utilization doesn't illustrate the oscillation effectively. For this reason, similar to the last chapter, we plotted the results differently. Specifically, we considered the *total acceptance ratio* as a function of the parameter $a$, which comes from the mathematical expression $T_2 = a \cdot T_1$. For the following we first consider only the interval $a \in [1, 3]$:

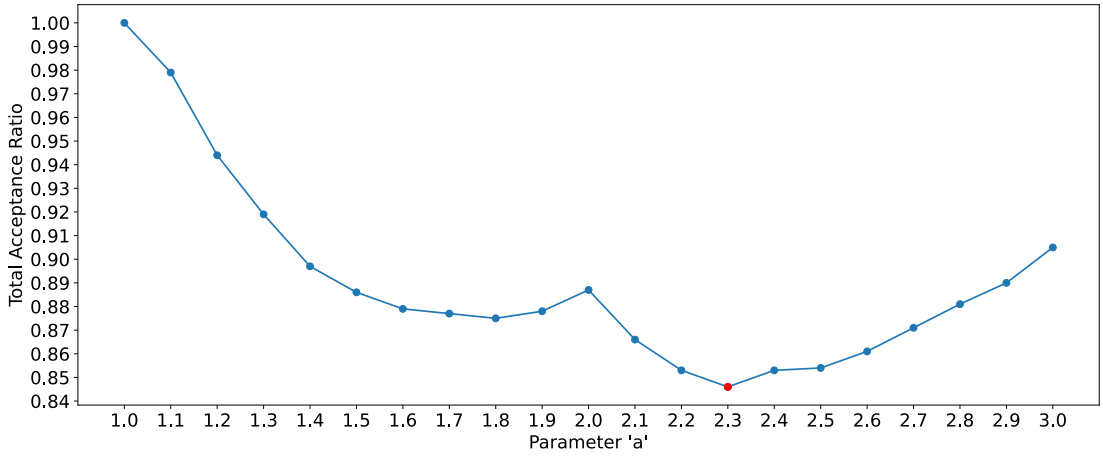



Figure 6.2: Period Variation Test EL-EDF with $a \in [1, 3]$

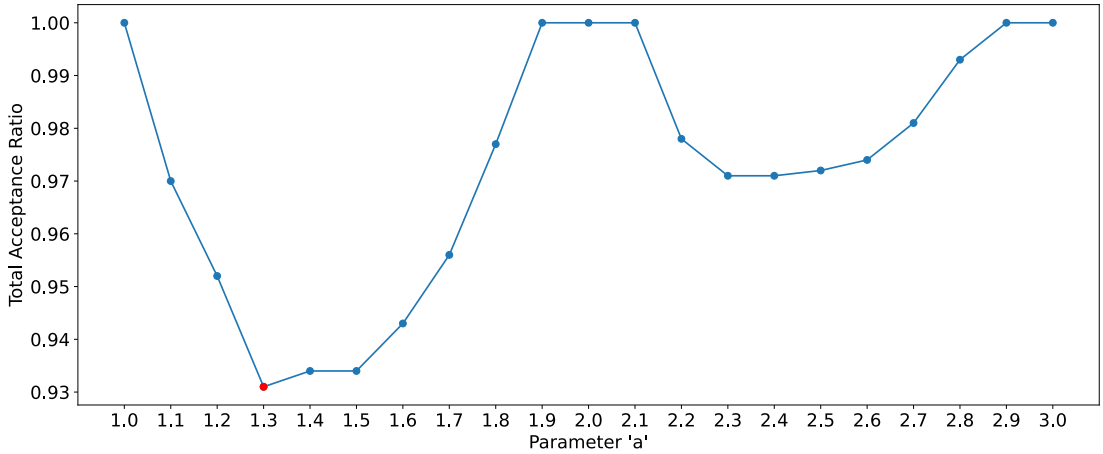



Figure 6.3: Period Variation Test EL-DM with $a \in [1, 3]$

We notice straight away in Figure 6.2 that in the beginning, with an increase in the parameter $a$, the success ratio keeps decreasing until a certain point, after which the schedulability ratio starts to rise as soon as $a$ gets closer to becoming a factor of $T_2$. Additionally, we can see that the lowest point for EL-EDF is when $a = 2.3$. Therefore, intuitively speaking, this parameter $a$ represents the worst-case interference from the higher-priority job. Since it varies the period of $T_2$ and therefore affects how well the carry-in region can be aligned.

Figure 6.3 shows a similar pattern for EL-DM. However, in this case, the total acceptance ratio rises immensely compared to EL-EDF and reaches $100\%$. Here, the lowest point for EL-DM is at $a = 1.4$, which is much smaller than EL-EDF.

We observe that for EL-EDF, the second time that $a$ becomes a factor of $T_2$ ($a = 3$), the total acceptance ratio is higher than when $a = 2$. For EL-DM, we can notice that the dip after $a = 2$ becomes much smaller. This suggests that by increasing $a$, the total schedulability ratio for both tests may keep increasing. Therefore, we turn our attention to the case where the parameter $a$ continues to grow:

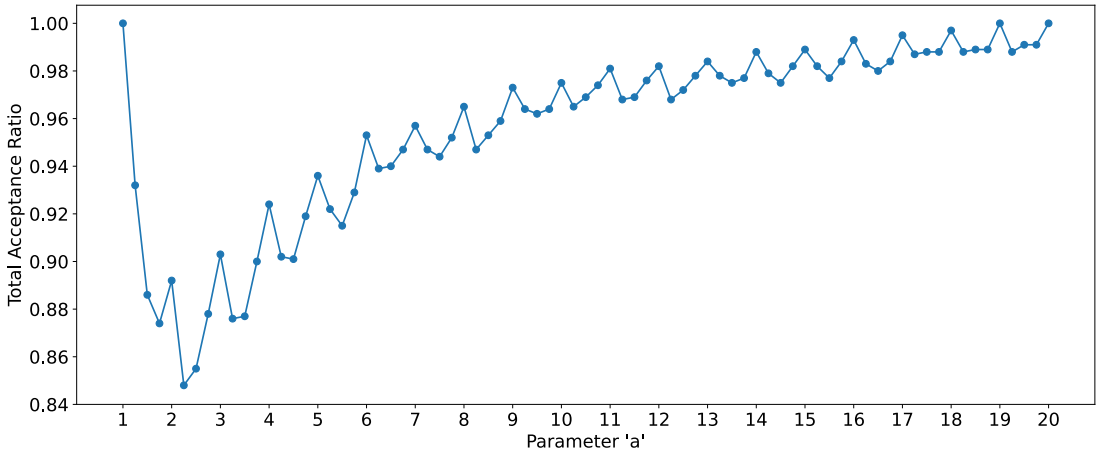



Figure 6.4: Period Variation Test EL-EDF with $a \in [1, 20]$

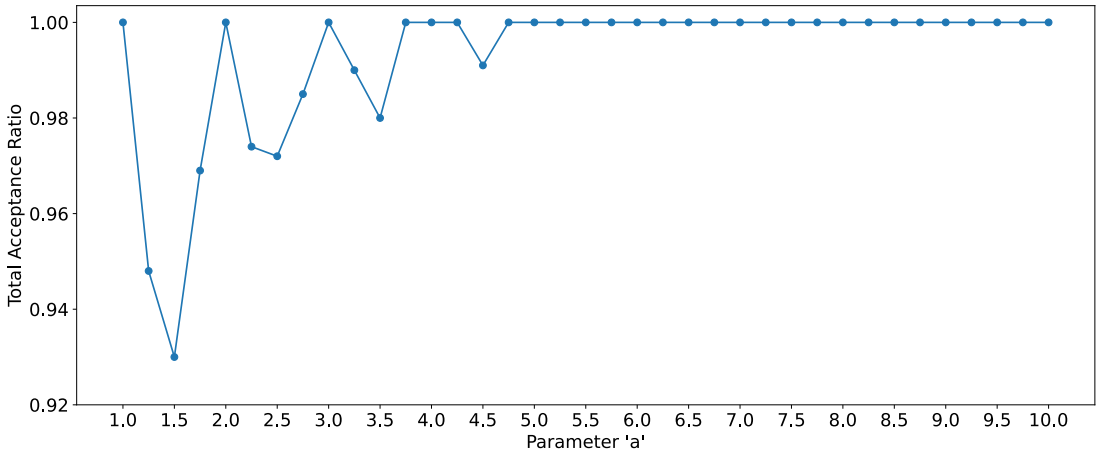



Figure 6.5: Period Variation Test EL-DM with $a \in [1, 10]$

As anticipated, in Figures 6.4 and 6.5, we immediately notice that for both tests, the total schedulability keeps increasing for larger values of $a$. For EL-EDF, we can see that with shallow values for $a$, the success ratio is at its lowest, and when $a = 19$ or $a = 20$, we already observe that the total schedulability reaches $100\%$. For EL-DM, with low values such as $a = 2.25$ or $a = 1.5$, the total schedulability ratio is at its minimum. And whenever $a$ is a factor of $T_2$, the schedulability ratio is at its fullest. Furthermore, the total acceptance ratio no longer drops after $a$ becomes larger than $4.5$.

## 6.4 Formal Proof

Figures 6.4 and 6.5 indicate that by continuously increasing the parameter $a$, the total schedulability ratio reaches $100\%$ for both tests. With the graphical representation in place, we turn to the formal proof to verify these properties.

We consider the given sufficient schedulability test (Theorem 1) from the paper [Gün+22]. We will focus on a special case by selecting $b_k = 0$. Since we are dealing with two non-self-suspending implicit-deadline tasks $\tau_1$ and $\tau_2$, the theorem simplifies as follows.

**Corollary 2.1.** *If $\tilde{R}_1 \leq T_1$ and $\tilde{R}_2 \leq T_2$ with*

$$\tilde{R}_1 = \max\left(\left\lceil \frac{G_1^2 + \tilde{R}_2}{T_2} \right\rceil, 0\right) C_2 + C_1 \tag{6.1}$$

$$\tilde{R}_2 = \max\left(\left\lceil \frac{G_2^1 + \tilde{R}_1}{T_1} \right\rceil, 0\right) C_1 + C_2 \tag{6.2}$$

*then the task set is schedulable using EDF-Like scheduling.*

In the following, we want to prove that if $\frac{T_2}{T_1} \to \infty$, the task set becomes schedulable under EDF and fixed-priority (FP) scheduling. For the proof, we provide the following sufficient condition to find a reasonable solution for $\tilde{R}_1$ and $\tilde{R}_2$ of the previous Corollary 2.1.

**Lemma 3.** *If there exist constants $K_1 \leq T_1$ and $K_2 \leq T_2$ such that*

$$\tilde{R}_1 \leq K_1 \quad \Longleftrightarrow \quad \tilde{R}_2 \leq K_2, \tag{6.3}$$

*then there exists a solution for $\tilde{R}_1$ and $\tilde{R}_2$ such that $\tilde{R}_1 \leq T_1$ and $\tilde{R}_2 \leq T_2$.*

*Proof.* We already know that $\tilde{R}_1$ depends on $\tilde{R}_2$ and vice versa. For convenience, we denote the corresponding functions from Corollary 2.1 as:

$$\tilde{R}_1 = f(\tilde{R}_2) \tag{6.4}$$

$$\tilde{R}_2 = g(\tilde{R}_1) \tag{6.5}$$

By considering the function $f \circ g$, we know that $f \circ g : [0, K_1] \to [0, K_1]$, with Equation (6.3). Furthermore, $f \circ g$ is monotonically increasing since both $f$ and $g$ are monotonically increasing. We even know that $f$ either stays constant or increases by at least $C_2$ whenever we increase the input for the function $f$. If the fixed-point iteration starting at $\tilde{R_1} = 0$ does not converge after $i$ iterations, then $(f \circ g)^i(0) \geq i \cdot C_2 \in [0, K_1]$ after $i$ iterations. Hence, if fixed-point iteration never converges, then $K_1 \geq \lim_{i \to \infty} i \cdot C_2 = \infty$, which contradicts $K_1 \leq T_1$. Therefore, the solution gives us $\tilde{R_1}$ and $\tilde{R_2} = g(\tilde{R_1})$ with $\tilde{R_1} \leq K_1$ and $\tilde{R_2} \leq K_2$. □

Before we delve into the formal proof, we must introduce two analytical properties that will be utilized.

$$\lim_{a \to \infty} \frac{G_1^2}{T_2} \leq -1 \tag{P1}$$

$$\lim_{a \to \infty} \frac{G_2^1}{T_2} \leq 1 \tag{P2}$$

These two properties will be instrumental in establishing the overall theorem:

**Theorem 4.** *Let $\mathbb{T} = \{\tau_1, \tau_2\}$ be a task set with the relative priority points $\Pi_1$ and $\Pi_2$, such that the Properties P1 and P2 holds. Assume that $U_1 = \frac{C_1}{T_1}$ and $U_2 = \frac{C_2}{T_2}$ are fixed. If $U_1 + U_2 < 1$, then there exists an $a \in \mathbb{R}_{>0}$ such that $\mathbb{T}$ is schedulable if $\frac{T_2}{T_1} \geq a$.*

*Proof.* Since $U_1 + U_2 < 1$, we know that

$$U_1 + U_2 < 1 - \epsilon \tag{6.6}$$

for some $\epsilon > 0$. We choose $K_1 := T_1$ and $K_2 := (1 - \epsilon)T_2$ and prove Equation (6.3).

First, we prove $\Rightarrow$: Assume that $\tilde{R_1} \leq T_1$. Then

$$\tilde{R_2} \leq \max\left(\left\lceil \frac{G_2^1 + T_1}{T_1} \right\rceil, 0\right) C_1 + C_2. \tag{6.7}$$

In the following we show that there exists an $a_1 \in \mathbb{R}$ such that

$$\max\left(\left\lceil \frac{G_2^1 + T_1}{T_1} \right\rceil, 0\right) C_1 + C_2 \leq (1 - \epsilon)T_2 \tag{6.8}$$

$$\Leftrightarrow \quad \max\left(\left\lceil \frac{G_2^1 + T_1}{T_1} \right\rceil, 0\right) \frac{C_1}{T_2} + \frac{C_2}{T_2} \leq 1 - \epsilon \tag{6.9}$$

for all $a \geq a_1$.
We only have to show that the over-approximation for the left-hand side falls below the

value on the right-hand side. Therefore, we will use the limiting behavior of the over-approximation, namely $\lceil x \rceil \leq x + 1$. We consider $\max\left(\left\lceil \frac{G_2^1+T_1}{T_1} \right\rceil, 0\right)\frac{C_1}{T_2}$:

$$\max\left(\left\lceil \frac{G_2^1+T_1}{T_1} \right\rceil, 0\right)\frac{C_1}{T_2} \leq \max\left(\left(\frac{G_2^1+T_1}{T_1}+1\right), 0\right)\frac{C_1}{T_2} \tag{6.10}$$

$$= \max\left(\left(\frac{G_2^1}{T_1}+2\right)\frac{C_1}{T_2}, 0\right) \tag{6.11}$$

$$= \max\left(\frac{G_2^1}{T_1}\frac{C_1}{T_2}+2\frac{C_1}{T_2}, 0\right) \tag{6.12}$$

$$= \max\left(\frac{G_2^1}{T_2}\frac{C_1}{T_1}+2\frac{C_1}{aT_1}, 0\right) \tag{6.13}$$

$$\overset{a\to\infty}{\leq} \frac{C_1}{T_1} \quad \text{(with P2)} \tag{6.14}$$

We then get the following for the left-hand side of Equation (6.9):

$$\max\left(\left\lceil \frac{G_2^1+T_1}{T_1} \right\rceil, 0\right)\frac{C_1}{T_2}+\frac{C_2}{T_2} \overset{a\to\infty}{\leq} \frac{C_1}{T_1}+\frac{C_2}{T_2} \tag{6.15}$$

$$= U_1 + U_2 \tag{6.16}$$

And since we know that $U_1 + U_2 < 1 - \epsilon$, we have shown that Equation (6.9) is correct and therefore $\tilde{R}_2 \leq K_2$ for all $a \geq a_1$.

Second, we prove $\Leftarrow$: We assume that $\tilde{R}_2 \leq (1-\epsilon)T_2$. Then we have

$$\tilde{R_1} \leq \max\left(\left\lceil \frac{G_1^2+(1-\epsilon)T_2}{T_2} \right\rceil, 0\right)C_2 + C_1 \tag{6.17}$$

We need to show that there exists an $a_2 \in \mathbb{R}$ such that

$$\max\left(\left\lceil \frac{G_1^2+(1-\epsilon)T_2}{T_2} \right\rceil, 0\right)C_2 + C_1 \leq T_1 \tag{6.18}$$

$$\Leftrightarrow \quad \max\left(\left\lceil \frac{G_1^2+(1-\epsilon)T_2}{T_2} \right\rceil, 0\right)\frac{C_2}{T_1} + \frac{C_1}{T_1} \leq 1 \tag{6.19}$$

for all $a \geq a_2$.

We consider $\left\lceil \frac{G_1^2+(1-\epsilon)T_2}{T_2} \right\rceil$:

$$\left\lceil \frac{G_1^2+(1-\epsilon)T_2}{T_2} \right\rceil = \left\lceil \frac{G_1^2}{T_2} + \frac{(1-\epsilon)T_2}{T_2} \right\rceil \tag{6.20}$$

$$\overset{a\to\infty}{\leq} \lceil -1 + 1 - \epsilon \rceil \quad \text{(with P1)} \tag{6.21}$$

$$= \lceil -\epsilon \rceil \tag{6.22}$$

$$= 0 \tag{6.23}$$

Therefore, for Equation (6.19) we get the following:

$$0 \cdot \frac{C_2}{T_1} + \frac{C_1}{T_1} \leq 1 \tag{6.24}$$

$$\Leftrightarrow \quad U_1 \leq 1 \tag{6.25}$$

Since we know that $U_1 + U_2 < 1$, we have shown that Equation (6.19) is correct, and, therefore, $\tilde{R}_1 \leq K_1$ for all $a \geq a_2$.

We conclude that $\tilde{R}_1 \leq K_1 \iff \tilde{R}_2 \leq K_2$ holds for all $a \geq \max(a_1, a_2)$. Hence, the task set is schedulable for all $a \geq \max(a_1, a_2)$ by Lemma 3. □

**Corollary 4.1.** *For a given task set $\mathbb{T} = \{\tau_1, \tau_2\}$ and relative priority points $(\Pi_1, \Pi_2)$. If EDF-Like is configured as EDF $(\Pi_i = D_i)$ and $U_1 + U_2 < 1$, then there exists an $a \in \mathbb{R}_{>0}$ such that $\mathbb{T}$ is schedulable if $\frac{T_2}{T_1} \geq a$.*

*Proof.* Since we know that $G_k^i = \min(D_k - C_i, \Pi_k - \Pi_i)$ from Theorem 1. With $\Pi_i = D_i$ and because we have implicit-deadline $(D_i = T_i)$ we get $G_k^i = \min(T_k - C_i, T_k - T_i) = (T_k - T_i)$. We show that Property P1 and P2 hold.

First for $P1$:

$$\frac{G_1^2}{T_2} = \frac{T_1 - T_2}{T_2} \tag{6.26}$$

$$= \frac{T_1}{T_2} - 1 \tag{6.27}$$

$$= \frac{T_1}{a \cdot T_1} - 1 \tag{6.28}$$

$$\xrightarrow[a \to \infty]{} -1 \tag{6.29}$$

Now for $P2$:

$$\frac{G_2^1}{T_2} = \frac{T_2 - T_1}{T_2} \tag{6.30}$$

$$= 1 - \frac{T_1}{a \cdot T_1} \tag{6.31}$$

$$\xrightarrow[a \to \infty]{} 1 \tag{6.32}$$

We have shown that P1 and P2 hold for EDF-Like configured as EDF. Therefore, by applying Theorem 1, we have proven that there exists an $a \in \mathbb{R}_{>0}$ such that $\mathbb{T}$ is schedulable. □

**Corollary 4.2.** *For a given task set $\mathbb{T} = \{\tau_1, \tau_2\}$ and relative priority points $(\Pi_1, \Pi_2)$. If EDF-Like is configured as DM $(\Pi_i = \sum_{j=1}^{i} D_j)$ and $U_1 + U_2 < 1$, then there exists an $a \in \mathbb{R}_{>0}$ such that $\mathbb{T}$ is schedulable if $\frac{T_2}{T_1} \geq a$.*

*Proof.* We know that $G_k^i = \min(D_k - C_i, \Pi_k - \Pi_i)$ from Theorem 1. With $\Pi_i = \sum_{j=1}^{i} D_j$ and since we have implicit-deadline ($D_i = T_i$) we get $G_k^i = \min(T_k - C_i, \sum_{j=1}^{k} T_j - \sum_{j=1}^{i} T_j)$. We show that the Property P1 and P2 hold.

First for $P1$:
We get $G_1^2 = \min(T_1 - C_2, T_1 - (T_1 + T_2)) = \min(T_1 - C_2, -T_2) = -T_2$.

$$\frac{G_1^2}{T_2} = -\frac{T_2}{T_2} = -1 \tag{6.33}$$

Now for $P2$:
We get $G_2^1 = \min(T_2 - C_1, T_1 + T_2 - T_1) = \min(T_2 - C_1, T_2) = T_2 - C_2$.

$$\frac{G_2^1}{T_2} = \frac{T_2 - C_2}{T_2} \tag{6.34}$$

$$= 1 - \frac{C_2}{a \cdot T_1} \tag{6.35}$$

$$\xrightarrow[a \to \infty]{} 1 \tag{6.36}$$

We have shown that P1 and P2 hold for EDF-Like configured as DM. Therefore, by applying Theorem 1, we have proven that there exists an $a \in \mathbb{R}_{>0}$ such that $\mathbb{T}$ is schedulable. □

We successfully proved that for a given task set $\mathbb{T} = \{\tau_1, \tau_2\}$ with relative priority points $(\Pi_1, \Pi_2)$ if $\frac{T_2}{T_1} \to \infty$, the task set becomes schedulable under EDF and FP scheduling.

## 6.5 Summary

In this chapter, we focused on a simplified setup with two tasks, where the period of the first task was fixed, and the period of the second task was varied. The results showed that the schedulability ratio oscillates as the period of the second task changes, with the lowest schedulability observed when the periods are not multiples of each other, suggesting the worst-case alignment of the carry-in job.

Further analysis extended this observation, showing that for larger period ratios, the system tends to reach full schedulability, particularly when the period of the second task becomes much larger than the first. This pattern was observed for both EDF and DM configurations of the EDF-Like algorithm.

We also included formal proof verifying this observation and showing that the task set becomes fully schedulable as the period ratio increases. Therefore, we were successful in proving our expectations that we had. Additionally, there is potential to extend the findings to consider multiple tasks. However, this extension would be more complex due to the increased interference among tasks and the more intricate interactions between their periods.

# 7 Conclusion

This is the last chapter of this thesis. Here, we summarize the work that we have accomplished and the questions that were answered.

The primary purpose of this thesis is to evaluate the performance of EDF-Like[Gün+22] scheduling under EDF[LL73] and fixed-priority[LW82] for preemptable sporadic/periodic task systems with implicit deadlines on a uniprocessor. Since exact tests currently do not exist for self-suspension task sets, we focused solely on non-self-suspending task sets.

We first examined the performance of EDF-Like in comparison to other exact tests, such as the utilization-based test for EDF and TDA for fixed-priority. Our findings indicate that EDF-Like performs similarly to exact tests at low to moderate utilization levels. However, its accuracy diminishes as utilization increases, primarily due to the over-approximation used in its analysis. One unexpected observation from our side was that EL-DM outperformed EL-EDF. Additionally, we carried out mini-experiments to inspect the impact of modifying different parameters, such as the task quantity and period range of the tasks. The insights from these experiments pushed us to analyze them further in depth.

The investigation into the impact of task quantity on schedulability further highlighted that contrary to initial assumptions, the schedulability did not decrease linearly with the increase in the number of tasks. Instead, the reduction plateaued after a certain point.

Last but not least, our further analysis of period variations, focusing only on two tasks, demonstrated that the alignment of task periods is critical in determining schedulability. Our results showed that when task periods are not multiples of each other, schedulability is at its lowest, while significantly increasing the period of one task relative to the other can lead to full schedulability. This observation was further validated by a formal proof, confirming that the task sets converge toward complete schedulability as the period ratio approaches infinity.

Another field that we seek to explore is task sets with constrained-deadlines since we only focused on implicit-deadlines. Nevertheless, we implemented a test for them as well. To evaluate the performance of EDF-Like with constrained-deadlines, we planned to compare the outcome with Processor Demand Analysis (PDA)[But24], which we also implemented. However, we didn't include it in this thesis due to the lack of time.

Based on the findings presented in the thesis, the EL (EDF-Like) schedulability test appears to be fairly close to being exact under certain conditions, particularly for low to moderate utilization levels. The performance of EL-DM was considerably good since the difference between its exact test TDA was relatively small. However, there is a more significant gap for EL-EDF when compared to the utilization-based test, which indicates that there are still improvements to be made.

While this thesis has provided valuable insights into the behavior of EDF-Like scheduling, several questions remain unanswered. Notably, the unexpectedly better performance of fixed-priority configuration and the plateauing of schedulability with increased task quantity suggest areas for future research. Additionally, extending this work to include self-suspending tasks or tasks with non-implicit deadlines could further enhance the understanding of EDF-Like scheduling, leading to further improved algorithms or analyses that could increase the accuracy of EDF-Like's schedulability test, bringing it closer to being an exact tool for the field of hard real-time scheduling.

# Bibliography


[But24] Giorgio Buttazzo. *Hard Real-Time Computing Systems - Predictable Scheduling Algorithms and Applications, 4th Edition*. Springer, 2024. ISBN: 978-3-031-45409-7. DOI: 10.1007/978-3-031-45410-3. URL: https://doi.org/10.1007/978-3-031-45410-3.

[Che+19] Jian-Jia Chen, Geoffrey Nelissen, Wen-Hung Huang, Maolin Yang, Björn B. Brandenburg, Konstantinos Bletsas, Cong Liu, Pascal Richard, Frédéric Ridouard, Neil C. Audsley, Raj Rajkumar, Dionisio de Niz, and Georg von der Brüggen. "Many suspensions, many problems: a review of self-suspending tasks in real-time systems". In: *Real Time Syst.* 55.1 (2019), pp. 144–207. DOI: 10.1007/S11241-018-9316-9. URL: https://doi.org/10.1007/s11241-018-9316-9.

[Dav16] Robert I. Davis. "On the Evaluation of Schedulability Tests for Real-Time Scheduling Algorithms". In: 2016. URL: https://api.semanticscholar.org/CorpusID:14995939.

[ESD] Paul Emberson, Roger Stafford, and Robert I Davis. "Techniques for the synthesis of multiprocessor tasksets". In: *proceedings 1st International Workshop on Analysis Tools and Methodologies for Embedded and Real-time Systems (WATERS 2010)*, pp. 6–11.

[GBD] David Griffin, Iain Bate, and Robert I. Davis. *dgdguk/drs*. Version latest. DOI: 10.5281/zenodo.4264857. URL: https://doi.org/10.5281/zenodo.4118058.

[GBD20] David Griffin, Iain Bate, and Robert I. Davis. "Generating Utilization Vectors for the Systematic Evaluation of Schedulability Tests". In: *41st IEEE Real-Time Systems Symposium, RTSS 2020, Houston, TX, USA, December 1-4, 2020*. IEEE, 2020, pp. 76–88. DOI: 10.1109/RTSS49844.2020.00018. URL: https://doi.org/10.1109/RTSS49844.2020.00018.

[Gün+22] Mario Günzel, Georg von der Brüggen, Kuan-Hsun Chen, and Jian-Jia Chen. "EDF-Like Scheduling for Self-Suspending Real-Time Tasks". In: *IEEE Real-Time Systems Symposium, RTSS 2022, Houston, TX, USA, December 5-8, 2022*. IEEE, 2022, pp. 172–184. DOI: 10.1109/RTSS55097.2022.00024. URL: https://doi.org/10.1109/RTSS55097.2022.00024.

[JP86] Mathai Joseph and Paritosh K. Pandya. "Finding Response Times in a Real-Time System". In: *Comput. J.* 29.5 (1986), pp. 390–395. DOI: 10.1093/COMJNL/29.5.390. URL: https://doi.org/10.1093/comjnl/29.5.390.

[KS22] Hermann Kopetz and Wilfried Steiner. *Real-Time Systems - Design Principles for Distributed Embedded Applications, Third Edition*. Springer, 2022. ISBN: 978-3-031-11991-0. DOI: 10.1007/978-3-031-11992-7. URL: https://doi.org/10.1007/978-3-031-11992-7.

[LL73] C. L. Liu and James W. Layland. “Scheduling Algorithms for Multiprogramming in a Hard-Real-Time Environment”. In: *J. ACM* 20.1 (1973), pp. 46–61. DOI: 10.1145/321738.321743. URL: https://doi.org/10.1145/321738.321743.

[LSD89] John P. Lehoczky, Lui Sha, and Y. Ding. “The Rate Monotonic Scheduling Algorithm: Exact Characterization and Average Case Behavior”. In: *Proceedings of the Real-Time Systems Symposium - 1989, Santa Monica, California, USA, December 1989*. IEEE Computer Society, 1989, pp. 166–171. DOI: 10.1109/REAL.1989.63567. URL: https://doi.org/10.1109/REAL.1989.63567.

[LW82] Joseph Y.-T. Leung and Jennifer Whitehead. “On the complexity of fixed-priority scheduling of periodic, real-time tasks”. In: *Perform. Evaluation* 2.4 (1982), pp. 237–250. DOI: 10.1016/0166-5316(82)90024-4. URL: https://doi.org/10.1016/0166-5316(82)90024-4.

[Mok83] Aloysius Ka-Lau Mok. “Fundamental design problems of distributed systems for the hard-real-time environment”. PhD thesis. Massachusetts Institute of Technology, Cambridge, MA, USA, 1983. URL: https://hdl.handle.net/1721.1/15670.

[RRC04] Frédéric Ridouard, Pascal Richard, and Francis Cottet. “Negative Results for Scheduling Independent Hard Real-Time Tasks with Self-Suspensions”. In: *Proceedings of the 25th IEEE Real-Time Systems Symposium (RTSS 2004), 5-8 December 2004, Lisbon, Portugal*. IEEE Computer Society, 2004, pp. 47–56. DOI: 10.1109/REAL.2004.35. URL: https://doi.org/10.1109/REAL.2004.35.