Real-Time Fixed and Dynamic Priority Driven Scheduling Algorithms: Theory and Experience

Jean-François Hermant, Laurent Leboucher, Nicolas Rivierre

INSTITUT NATIONAL DE RECHERCHE EN INFORMATIQUE ET EN AUTOMATIQUE

Real-time fixed and dynamic priority driven scheduling algorithms: theory and experience

Jean-François HERMANT, Laurent LEBOUCHER, Nicolas RIVIERRE

N° 3081

Décembre 1996

THEME 1

Réseaux et systèmes

ISSN 0249-6399

Real-time fixed and dynamic priority driven scheduling algorithms: theory and experience

Jean-François HERMANT[1], Laurent LEBOUCHER[2],
Nicolas RIVIERRE[1]

Thème 1 :
Réseaux et systèmes

Projet REFLECS

Rapport de recherche n˚3081 - Décembre 1996

139 pages

Abstract: There are two main positions regarding real-time scheduling algorithms. The first is based on fixed priorities and the second makes use of dynamic priorities such as deadlines. These two approaches have never really been compared because the emphasis has always been on the ease of implementation rather than the **efficiency** of the algorithms and the **complexity** of the associated feasibility conditions. In addition to traditional real-time applications, we believe that starting to look at these two criteria will be very important from the point of view of providing admission control mechanisms and real-time guarantees on large distributed systems like the Internet network.
To that end, our purpose is first to provide a **general framework** based, on the one hand, a representation of preemptive, real-time scheduling in an algebraic structure that enables us to evaluate the distance of the optimality of any scheduling algorithm ; and on the other hand, a consistent representation of the associated feasibility conditions that enables us to evaluate the number of basic operations. As a second step, considering several kinds of traffics, we initiate the comparison by a straight, but limited, application of our general framework. Our preliminary results will notably highlight, in the cases where deadlines are all greater than periods, that fixed priority schedulers (like deadline monotonic) behave as well as EDF while the worst-case response time analysis is less complex. The same observation is valid when the task sets are almost homogeneous but is in favor of EDF in the general case or when a simple feasibility analysis is needed.
Therefore, it might be of interest, given a real-time scheduling context (spanning from small embedded systems to large distributed systems), to take into account these two extra criteria in order to find a right trade-off among several possible solutions.

Key-words: busy period, comparison, complexity, efficiency, dynamic priority, fixed priority, preemptive, real-time, scheduling.

---

1. INRIA, Projet REFLECS, B.P. 105, 78153 LE CHESNAY Cedex (France)
Email: {Jean-Francois.Hermant, Nicolas.Rivierre}@inria.fr

2. CNET, France Télécom, PAA/TSA/TLR,
38-40, rue du Général Leclerc, 92794 ISSY-LES-MOULINEAUX Cedex 9 (France)
Email: lebouche@issy.cnet.fr

Unité de recherche INRIA Rocquencourt
Domaine de Voluceau - Rocquencourt - B.P. 105 - 78153 LE CHESNAY Cedex (France)
Téléphone : +33 1 39 63 55 11 - Télécopie : +33 1 39 63 53 30

Ordonnancement temps réel statique et dynamique: théorie et expérience

Résumé : Il existe deux principales familles d'algorithmes d'ordonnancement temps réel, la première s'appuyant sur des priorités fixes et la seconde sur des priorités dynamiques de type échéances. Celles ci n'ont jamais été vraiment comparées l'une à l'autre, si ce n'est en termes de mise en oeuvre. Notre propos est d'initier une telle comparaison en termes d'**efficacité** des algorithmes ainsi que de **complexité** des conditions de faisabilité associées. Au delà des applications temps réel traditionnelles, commencer à considérer ces deux critères nous semble devenir critique dans la perspective de la fourniture de mécanismes de contrôles d'admission et de garanties temps réel sur de grands systèmes répartis tels qu'Internet.
Dans un premier temps, nous introduisons un **cadre général** basé, d'une part, sur une représentation de l'ordonnancement temps réel préemptif sous forme de structure algébrique permettant d'évaluer la distance à l'optimalité de tout algorithme et, d'autre part, sur une représentation homogène des conditions de faisabilité associées permettant d'évaluer le nombre d'opérations élémentaires induites. Dans un deuxième temps, considérant différents types de trafics, nous initialisons la comparaison par une application directe, mais limitée, de notre cadre général. Nos résultats préliminaires font notamment apparaître, lorsque les échéances des tâches sont toutes supérieures aux périodes que les algorithmiques d'ordonnancement à priorité fixes (comme Deadline Monotonic) se comportent aussi bien qu'EDF alors que l'analyse des pires temps de réponses des tâches est moins complexe. La même remarque s'applique en présence de tâches homogènes mais reste en faveur de EDF dans le cas le plus général ou lorsqu'une simple analyse de faisabilité est nécessaire.
Il semble donc être intéressant, en fonction du contexte d'ordonnancement (allant du petit système embarqué jusqu'au grand système distribué) de faire intervenir les critères d'efficacité et de complexité pour choisir l'algorithme temps réel le plus adapté.

Mots-clé : comparaison, complexité, efficacité, période occupée, priorité fixe, priorité dynamique, ordonnancement, préemptif, temps réel.

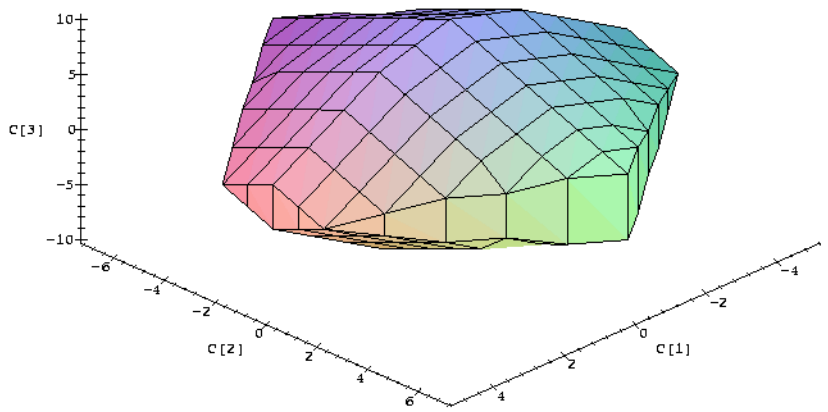
Contents

Glossary

C_i	Worst-case computation time for task τ_i (cf. Section 1.1.1, page 8).
D_i	Relative deadline for task τ_i (cf. Section 1.1.1, page 8).
EDF	Earliest Deadline First scheduling algorithm (cf. Section 1.2.1.1, page 12).
HPF/DM	Highest Priority First / Deadline Monotonic scheduling algorithm (cf. Section 1.2.2.1, page 18).
HPF/RM	Highest Priority First / Rate Monotonic scheduling algorithm (cf. Section 1.2.2.1, page 18).
$h(t)$	Processor demand function (cf. Section 1.1.2, page 10).
$L_i(a)$	Length of a deadline-$(a + D_i)$ busy period.
NSC	Necessary and Sufficient Condition.
N_{EDF}	EDF valid norm (cf. Theorem 13, page 26).
N_U	Processor utilization valid norm (cf. Theorem 13, page 26).
P	Base period (cf. Section 1.1.2, page 10).
PNTS	Periodic Non-Concrete Task set τ (cf. Definition 8, page 25).
PNTSC	Class of PNTS (cf. Definition 8, page 25).
r_i	Worst-case response time for task τ_i (cf. Section 1.1.2, page 10).
T_i	Period for task τ_i (cf. Section 1.1.1, page 8).
U	Processor utilization (cf. Section 1.1.1, page 8) later called N_U.
$w_{i,q}$	Length of the level-i busy period starting qT_i before the current activation of task τ_i (cf. Section 1.2.2.2, page 18).
$W(t)$	Cumulative workload function (cf. Section 1.1.1, page 8).
$\alpha_f(P)$	f-efficiency of a scheduling algorithm P with respect to Σ (cf. Definition 12, page 28).
$\varepsilon(P)$	Efficiency of a scheduling algorithm P with respect to Σ (cf. Definition 12, page 28).
λ	Length of the synchronous processor busy period (cf. Section 1.1.2, page 10).
λ_i	Length of a level-i busy period (cf. Section 1.1.2, page 10).
Π	Class of scheduling algorithms (cf. Definition 3, page 24).
$\Sigma = (\Upsilon,\Pi)$	Scheduling referential (cf. Definition 3, page 24).
Υ	Class of task sets τ (cf. Definition 3, page 24).
τ	Non-concrete task set τ (cf. Section 1.1.1, page 8).
τ_i	Non-concrete task τ_i of τ (cf. Section 1.1.1, page 8).
$\chi(\Sigma)$	Σ-region of a scheduling referential (cf. Section 1.1.1, page 8).

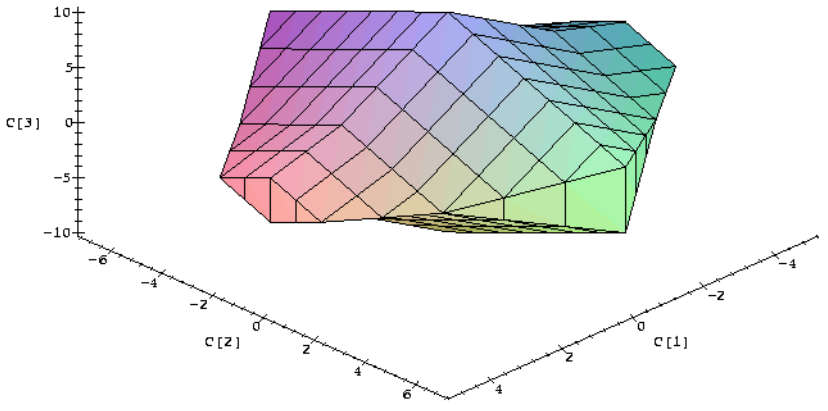
Σ – region EDF

$$C_1 \cdot e_{7,5} + C_2 \cdot e_{11,7} + C_3 \cdot e_{13,10}$$



Σ – region HPF/DM

$$C_1 \cdot e_{7,5} + C_2 \cdot e_{11,7} + C_3 \cdot e_{13,10}$$



1 Introduction

Scheduling theory as it applies to hard real-time environment has been widely studied in the last twenty years. The community of real-time researchers is currently split into two camps, those who support fixed priority driven scheduling algorithms, that were devised for easy implementation, and those who support dynamic priority driven scheduling algorithms that were considered better theoretically. Moreover since [LL73], a milestone in the field of hard real-time scheduling, these two classes of algorithms have been studied separately and a lot of results such as optimality, feasibility condition, response time, admission control have been established, relaxing some initial assumptions and considering several scheduling contexts. Consequently, these results are quite dispersed in the literature and the proofs have often been established in an ad-hoc manner.

In our opinion, the implementation and the hardware of today no longer justify a simple refusal of dynamic priority algorithms. On the other hand, the theoretical dominance of dynamic priority driven scheduling algorithms is not a determining factor in every context. Moreover, the concepts used by all the existing results are quite similar. Therefore, it is the first goal of this paper to build a consistent theoretical **framework** which could encompass the collection of existing results and that enables us to evaluate the preemptive, non-idling fixed/dynamic priority driven scheduling algorithms in terms of **efficiency** (the distance of the optimality) and the associated feasibility conditions in terms of **complexity** (the number of basic operations). As a second step, this paper will initiate such a comparison by a straight, but limited, application of our framework considering several kinds of traffics.

More precisely, the paper is organized as follows. Section 1, page 8, outlines the computational model, recalls the basic scheduling concepts and proposes a state-of-the-art regarding preemptive, fixed/dynamic priority driven scheduling algorithms. The goals of this paper are detailed at the end of this section.
Section 2, page 24, presents our framework to compare scheduling algorithms. To that end, we first propose an algebraic structure for preemptive, real-time scheduling that enables us to evaluate the gap separating a particular scheduling algorithm from an optimal one. Secondly a consistent representation of the feasibility conditions is proposed that enables us to compare the number of basic operations involved by these conditions.
Section 3, page 60 (resp. Section 4, page 68), proposes a straight, but limited (in the sense that many refinements or other results might be derived from Section 2, page 24), application of our abstract framework to initiate the efficiency comparison (resp. complexity comparison) of fixed versus dynamic priority driven scheduling algorithms in presence of general task sets, as well as particular task sets. The obtained preliminary results are illustrated in Section 5, page 101, with some simple but representative numerical examples (e.g. voice and image traffics versus embedded system traffics).
Finally, Section 6, page 116, is a synthesis of our fixed/dynamic priority driven scheduling algorithm comparison leading to new arguments in that controversial discussion.

1.1 Model, concepts and notations

1.1.1 Model

In this paper, we shall consider the problem of scheduling a set $\tau = \{\tau_1, \ldots, \tau_n\}$ of n non-concrete periodic or sporadic tasks on a single processor. This will be done in presence of hard real-time constraints and with relative deadlines not necessarily related to the respective periods of the tasks. A task τ_i is a sequential job that is invoked with some maximum frequency and result in a single execution of the job at a time, handled by a given scheduling algorithm. From the scheduling point of view, a task can then be seen as an infinite number of **activations**. By definition, we consider that:

- A **non-concrete periodic task** τ_i recurs and is represented by the t-uple (C_i, D_i, T_i), where C_i, D_i and T_i respectively represent its worst-case computation time, relative deadline and

period (note that the absolute deadline of a given activation is equal to the release time plus the relative deadline). A **concrete periodic** task ω_i is defined by $\{\tau_i, s_i\}$ where s_i, the start time, is defined as the duration between time zero and the first activation of the task.

- A **non-concrete periodic task set** $\tau = \{\tau_1, \ldots, \tau_n\}$ is a set of n non-concrete tasks. A **concrete periodic task set** $\omega = \{\omega_1, \ldots, \omega_n\}$ is a set of n concrete tasks. Therefore, an infinite number of concrete task sets can be generated from a non-concrete task set (without loss of generality we assume $\min_{1 \le i \le n}\{s_i\} = 0$).
  Note that a concrete periodic task set ω is called a **synchronous** task set iff $s_i = s_j$ for all $1 \le i, j \le n$ (s_i is then called a **critical instant** in the literature) otherwise, ω is called an **asynchronous** task set ([LM80] showed that the problem to know if an asynchronous task set can lead to a critical instant is NP-complete).

- **Sporadic** tasks were formally introduced in [MOK83] (although already used in some papers, e.g., [KN80]) and differ only from periodic tasks in the activation time: the $(k+1)^{th}$ activation of a periodic task occurs at time $t_{k+1} = t_k + T_i$, while it occurs at $t_{k+1} \ge t_k + T_i$ if the task is sporadic. Hence T_i represents the minimum interarrival time between two successive activations.

- A **general** task set $\tau = \{\tau_1, \ldots, \tau_n\}$ is a non-concrete periodic or sporadic task set such that $\forall i \in [1, n]$, T_i and D_i are not related. On the other hand, a **particular** task set $\tau = \{\tau_1, \ldots, \tau_n\}$ is a non-concrete task set such that $\forall i \in [1, n]$, T_i and D_i are related.

Throughout this paper, we assume the following:

- all the studied schedulers make use of the HPF (Highest Priority First) on-line algorithm but differ by their priority assignment scheme (in the sequel, we will no more refer to HPF). They all make use of a fixed tie breaking rule between tasks that show the same priority. They are **non idling** (i.e., the processor cannot be inactive in presence of pending activations) and **preemptive** (i.e., the processing of any task can be interrupted by a higher priority task).

- $\forall i \in [1, n]$, $C_i \le T_i$, $C_i \le D_i$ and general, as well as particular, task sets will be considered. More precisely, the following task sets will be considered:
  - general case, $\forall i \in [1, n]$, T_i and D_i are not related,
  - homogeneous case, $\forall i \in [1, n]$, $T_i = T$ and $D_i = D$,
  - $\forall i \in [1, n]$, $D_i = T_i$,
  - $\forall i \in [1, n]$, $D_i \le T_i$,
  - $\{D_i\} \ll \{T_i\}$ (all the deadlines are supposed to be dominated by all the periods),
  - $\forall i \in [1, n]$, $D_i \ge T_i$,
  - $\{D_i\} \gg \{T_i\}$ (all the periods are supposed to be dominated by all the deadlines).

- all tasks in the system are independent of one another and are critical (hard real-time), i.e., all of them have to meet their absolute deadline. Unless otherwise stated, tasks do not have resource constraints and the overhead due to context switching, scheduling... is considered to be included in the execution time of the tasks.

- time is discrete (tasks activations occur and task executions begin and terminate at clock ticks; the parameters used are expressed as a multiples of clock ticks); In [BHR90], it is shown that there is no loss of generality with respect to feasibility results by restricting the schedules to be discrete, once the task parameters are assumed to be integers.

1.1.2 Classic concepts

Let us now recall some classic concepts used in hard real-time scheduling:

- the scheduling of a concrete task set ω is said to be **valid** if and only if no task activation misses its absolute deadline.

- a concrete task set ω is said to be **feasible** with respect to a given class of scheduling algorithms if and only if there is at least one valid schedule that can be obtained by a scheduling algorithm of this class (in this paper, we consider only two classes of scheduling algorithms combining non-idling, preemptive, fixed/dynamic priority driven scheduling algorithms). Similarly, a non-concrete task set τ is said to be feasible with respect to a given class of scheduling algorithms if and only if every concrete task set ω that can be generated from τ is feasible in this class.
  Note that from the real-time specification point of view, a non-concrete task set is more realistic than a concrete one, since not one but all the patterns of arrival are indeed considered. For the same reasons, but from the scheduling point of view, a feasibility condition for a non-concrete task set is generally more selective and less complex than one for a concrete task set. In particular [BHR90], [BMR90] showed that a concrete task set cannot, in general, be tested in a polynomial time unless P=NP. Anyhow, they also showed that a concrete synchronous task set, or a non-concrete task set, can be tested in pseudo polynomial time whenever the processor utilization $U \leq c < 1$ (i.e. with c a constant smaller than 1).

- a scheduling algorithm is said to be **optimal** with respect to a given class of scheduling algorithms if and only if it generates a valid schedule for any feasible task set in this class. Note that this definition is ambiguous since it is class of scheduling algorithm dependent. In Section 2.1, page 24, we will precise this concept with respect to a given scheduling referential.

- $\mathbf{U} = \sum_{j=1}^{n} \frac{C_j}{T_j}$ is the **processor utilization** factor, i.e., the fraction of processor time spent in the execution of the task set [LL73]. An obvious Necessary Condition for the feasibility of any task set is that $U \leq 1$ (unless otherwise stated, this is assumed in the sequel).

- $\mathbf{P} = \mathrm{lcm}_{1 \leq i \leq n}\{T_i\}$ is the **base period**, i.e., a cycle such as the pattern of arrival of a periodic task set recurs similarly [LM80]. Note that, even in the case of limited task sets, P can be large when the periods are prime.

- given a critical instant at time 0 ($\forall i \in [1, n]$, $s_i = 0$), the **workload** $\mathbf{W}(t)$ is the amount of processing time requested by all activations whose release times are in the interval $[0, t)$ [BMR90]:

$$W(t) = \sum_{j=1}^{n} \left\lceil \frac{t}{T_j} \right\rceil C_j$$

- given a critical instant at time 0 ($\forall i \in [1, n]$, $s_i = 0$), the **processor demand** $\mathbf{h}(t)$ is the amount of computation time requested by all activations whose release times and absolute deadlines are in the interval $[0, t]$ [BMR90],[SPU96]:

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

- λ: Given a non-concrete task set, the **synchronous processor busy period** is defined as the time interval $[0, \lambda)$ delimited by two distinct processor idle periods[1] in the schedule of the corresponding synchronous concrete task set (where 0 is a critical instant, i.e., $\forall i \in [1, n]$, $s_i = 0$). It turns out that the value of λ does not depend on the scheduling algorithm, as far as it is non-idling, but only on the task arrival pattern. Note that (cf. Appendix A, page 120, and Appendix B, page 127, for a more formal discussion):
  - λ can be computed recursively using the workload [KLS93][2].
  - in the sequel, we will see how it is possible to generalize the concept of Busy period with respect to a fixed or dynamic priority (let λ_i denotes, for task τ_i, the maximum interval of such a priority busy period). It turns out that the value of λ_i depends on the scheduling algorithm.
- Given a non-concrete general task τ_i, $\forall i \in [1, n]$, r_i is the **worst-case response time** of τ_i, i.e., the longest time ever taken by any activation of the task from its release time until the time it completes its required computation [JP86]. Note that:
  - as shown in the sequel, the value of r_i is scheduling algorithm dependent and can be computed using the concept of priority busy period.
  - if a task has a worst-case response time greater than its period then there is the possibility for a task to re-arrive before the previous activations have completed. In our model, we consider that the new arrival is delayed from being executing after the previous activation terminate. In other words we keep the order of the events of the same task. Other ways to deal with such a situation exist (e.g. to deliver the most recent activation of the same task first), leading to an adaptation of the proposed results.

1.2 Previous work

Given a scheduling context, the main goal of the existing results is to couple an optimal scheduling algorithm with a polynomial-time or pseudo-polynomial-time (in order to be practical) Feasibility Condition (**FC**) that could be either Sufficient Condition (**SC**) or Necessary and Sufficient Condition (**NSC**). An FC can lead either to establish the feasibility only of the given real-time problem, or to extra informations as the worst-case response time of any task (in both cases, the proofs require to identify the worst pattern(s) of arrival). As said previously, the community of real-time researchers is currently split in two camps, those who support fixed priority driven scheduling algorithms (whose main results are presented in Section 1.2.2, page 18) and those who support dynamic priority driven scheduling algorithms (whose main results are presented in Section 1.2.1, page 12). Note that:

- the proposed state-of-the-art are not an exhaustive list of the papers related to real-time scheduling but more a summary of the main steps and new approaches proposed in this field (see [GRS96] for a more detailled state-of-the-art).
- the given theorems are adapted to take into account our notations.

---

1. A processor idle period can have a zero duration when there is only one outstanding computation and the end of this computation coincides with a new request of a new one.

2. The recursion ends when $\lambda^{k+1} = W(\lambda^k) = \lambda^k = \lambda$ and can be solved by successive iterations starting from $\lambda^0 = W(0^+)$. Indeed, it is easy to shown that λ^k is non decreasing. Consequently, the series converges or exceeds P if $U > 1$, in the latter case, the task set is not schedulable. Finally, this kind of recursive expression is pseudo-polynomial if $U \leq c < 1$.

1.2.1 Dynamic priority driven scheduling algorithms

In this section, we briefly summarize the principles of preemptive, non-idling dynamic priority scheduling. These principles were derived in particular from [LL73], [LM80], [BMR90], [BHR93], [KLS93], [ZS94], [SPU96], [RCM96] and [GRS96].

1.2.1.1 Optimality

We will focus on the **EDF** (Earliest Deadline First) scheduling algorithm. At any time, EDF executes among those tasks that have been released and not yet fully serviced (pending tasks), one whose absolute deadline is earliest. If no task is pending, the processor is idle.

Theorem 1 - ([DER74]) *EDF is optimal.*

The proof shows that it is always possible to transform a valid schedule to one which follows EDF. More precisely, if at any time the processor executes some task other than the one which has the earliest absolute deadline, then it is possible to interchange the order of execution of these two tasks (the resulting schedule is still valid).

Note that the optimality property of EDF is more general than the optimality property of any fixed priority driven scheduling algorithms. Indeed, EDF is said to be optimal in the sense that no other dynamic, as well as fixed, priority driven scheduling algorithm can lead to a valid schedule which cannot be obtained by EDF. This ambiguity on the concept of optimality will be overcome in Section 2.1, page 24, with respect to the concepts of scheduling referential and Σ-optimality.
Note also that, exept EDF and LLF[1], there is no other optimal scheduling algorithm, in our knowledge for the dynamic case. Consequently, in the sequel, all the results (in particular the complexity analysis of the feasibility conditions) described for that case are referred to EDF. We are fully aware that a formal justification of this choice has to be done but we let that point as an open question for the interested reader.

1.2.1.2 Feasibility

■ **Case** $\forall i \in [1, n]$, $T_i = D_i$

Theorem 2 - *([LL73]) For a given synchronous periodic task set (with* $\forall i \in [1, n]$, $T_i = D_i$*), the EDF schedule is feasible if and only if* $U \leq 1$.

This result gives us a simple O(n) procedure based on the processor utilization to check the feasibility. The proof shows that if a given processor busy period leads to an overflow at time t, the resulting synchronous processor busy period cannot lead to an idle time prior to time t. This leads to a contradiction if we assume $U \leq 1$ and an overflow at time t.

Theorem 3 - *([LL73], [KLS93]) Any non-concrete periodic task set (with* $\forall i \in [1, n]$, $T_i = D_i$*) scheduled by EDF, is feasible if and only if no absolute deadline is missed during the synchronous busy period.*

This result is related to the previous theorem. That is, if an absolute deadline is missed in a given busy period, then one is missed in the synchronous processor busy period.

---

1. The LLF (Least Laxity First) scheduling algorithm, which at any scheduling decision chooses the task activation with the smallest laxity (absolute deadline minus current time minus remaining execution time) has also been shown to be optimal in the same context [MOK83] but leads to more preemptions than EDF and worst response times. Consequently, it has received less attention in the literature.

■ **Case** $\forall i \in [1, n]$, $D_i \leq T_i$.

Historically, a NSC for this model has been known since the publication of [LM80] (improved latter by [BHR90]), in which Leung and Merril established that the feasibility of an asynchronous task set can be checked by examining the feasibility of the EDF schedule in the time interval $[0, 2P + \max\{s_i\}]$ (with P and s_i as defined in Section 1.1, page 8).

Note that this NSC operates in exponential time in the worst-case (which is usually unacceptable) since P is in the worst case a function of the product of the task periods. It turns out that for a synchronous periodic task set we can restrict our attention to the interval $[0, P]$, with 0 a critical instant. That is, the synchronous arrival pattern is the most demanding for non-concrete task set.

In addition to these results, the approach we are going to show now is based on the evaluation of the processor demand $h(t)$ on limited intervals such as the synchronous processor busy period (cf. Section 1.1.2, page 10). Indeed, being the processor demand, the amount of computation requested by all activations in the interval $[0, t]$, it follows that for any $t \geq 0$, $h(t)$ must not be greater than t in order to have a valid schedule. Note that this approach leads to a pseudo-polynomial NSC if $U \leq c < 1$ (i.e. with c a constant smaller than one).

Theorem 4 - *([BRH90], [BMR90], [ZS94],[RCM96]) A non-concrete periodic, or sporadic, task set (with* $D_i \leq T_i \ \forall i \in [1, n]$ *and* $U \leq 1$*) is feasible, using EDF, if and only if:*

$$\forall t \in S, h(t) \leq t \tag{1}$$

where $S = \left(\bigcup_{j=1}^{n} \{kT_j + D_j, k \in \mathbb{N}\} \right) \cap \left[0, \min\left\{ \lambda, \frac{\sum_{j=1}^{n} (1 - D_j/T_j) C_j}{1 - U} \right\} \right)$,

and λ *is the length of the synchronous processor busy period (cf. Section 1.1.2, page 10).*

This theorem unifies several results. A first improvement on the result of [LM80] for synchronous periodic task sets is found in [BHR90] and [BMR90], where Baruah et al. show that if $U < 1$ an NSC for the feasibility of a task set is that $\forall t$, $h(t) \leq t$ in the limited interval:

$$\left[0, \frac{U}{1 - U} \max_{i=1\ldots n}(T_i - D_i) \right).$$

The proof is based on the fact that if the task set is not feasible, then there is an instant of time t such that $h(t) > t$. By algebraic manipulations of this condition, an upper bound on the value of t can be determined. Recently, by using a similar manipulation on the same condition, [RCM96] obtained a tighter upper bound:

If there is $t < h(t)$ (with $h(t) = \sum_{j=1}^{n} \left(1 + \left\lfloor \frac{t - D_j}{T_j} \right\rfloor \right) C_j$ since $\forall i \in [1, n]$, $D_i \leq T_i$), then we have:

$$t < h(t) \leq \sum_{j=1}^{n} \left(\frac{t + T_j - D_j}{T_j} \right) C_j = tU + \sum_{j=1}^{n} \left(1 - \frac{D_j}{T_j} \right) C_j, \text{ from which:}$$

$$t < \frac{\sum_{j=1}^{n} (1 - D_j/T_j) C_j}{1 - U}.$$

Note that if $U \leq c$, with c a constant smaller than 1, this result leads to a pseudo-polynomial-time NSC [BRM90]. Note also that the interval to be checked can be large if c is close to 1.

A second upper bound on the length of the interval to be checked is given in [SPU95], [RCM96]. In particular, Ripoll et al. extend to $\forall i \in [1, n]$, $D_i \leq T_i$, the result of Theorem 3, page 12, by showing that the synchronous processor busy period is the most demanding one (using the same overflow argument as in [LL73]). That is, if an absolute deadline is missed in the schedule, then one is missed in the synchronous processor busy period.

Finally, the evaluation of the NSC is further improved as proposed by [ZS94], who propose to evaluate the processor demand only on the set S of points corresponding to absolute deadlines of task requests (i.e. the set of points where the value of $h(t)$ changes).

■ **Case** $\forall i \in [1, n]$, $D_i \geq T_i$

Theorem 5 - *([BMR90]) A non-concrete periodic, or sporadic, task set (with* $\forall i \in [1, n]$, $D_i \geq T_i$*) is feasible, using EDF, if and only if* $U \leq 1$.

The proof of [BMR90] simply shows that if $D_i \geq T_i$, $\forall i \in [1, n]$ and $U \leq 1$:

$$\forall t \geq 0 \qquad h(t) = \sum_{j=1}^{n} \max\left(0, 1 + \left\lfloor \frac{t - D_j}{T_j} \right\rfloor\right) C_j = \sum_{D_j \leq t} \left(1 + \left\lfloor \frac{t - D_j}{T_j} \right\rfloor\right) C_j ,$$

from which: $\forall t \geq 0 \qquad h(t) \leq \sum_{D_j \leq t} \left(\frac{t + T_j - D_j}{T_j}\right) C_j \leq t \sum_{j=1}^{n} \frac{C_j}{T_j} \leq t .$

■ **General task sets** ($\forall i \in [1, n]$, T_i and D_i are not related)

The results seen for task sets with relative deadlines smaller than or equal to the respective periods, are also valid for general task sets, in which there is no a priori relation between relative deadlines and periods. In particular, Theorem 3, page 12, originally conceived for task sets with relative deadlines equal to periods, was independently shown (by [RCM96] and [SPU95, SPU96], respectively) to hold also for the less restrictive models. That is, in order to check the feasibility of a general task set, we still can limit our attention to the synchronous busy period. [GRS96] defined a more specialized notion of busy period, namely the *deadline busy period*, that enables us to refine the analysis by further restricting the interval of interest.

Definition 1 - *([GRS96]) A deadline-*d *busy period is a processor busy period in which only task activations with absolute deadline smaller than or equal to* d *execute.*

It turns out that only synchronous deadline busy periods are the busy periods interesting in order to check the feasibility of task set since they derived the following property:

Lemma 1 - *([GRS96]) Given a general task set, if there is an overflow for a certain arrival pattern, then there is an overflow in a synchronous deadline busy period.*

The proof is still an extension of Theorem 3, page 12, showing that if a given processor busy period leads to an overflow at time d, the resulting synchronous *deadline-*d busy period leads also to an over-

flow. Note that [GRS96] proposed an algorithm to compute λ_i^s, the longest synchronous deadline busy period for any task τ_i and that the synchronous processor busy period being the largest busy period, it is possible to maximise λ_i^s by λ.

On the other hand [BHR93] and [ZS94] showed by algebraic manipulationst of $h(t)$, as for Theorem 4, page 13, the possibility of checking the feasibility of a general task set on the following set S of points in a limited interval. The only difference with this theorem comes from the lower bound $\max(D_i)$ that is needed to take into account due to relative deadlines greater than periods. Finally:

$$S = \left(\bigcup_{j=1}^{n} \{kT_j + D_j, k \in N\} \right) \cap \left[0, \max\left\{ \max_{j=1\ldots n}(D_j), \frac{\sum_{j=1}^{n}(1 - D_j/T_j)C_j}{1-U} \right\} \right).$$

[ZS94] derived from this result an admission control procedure meaning that if a general task set of n-1 tasks is feasible, it is then possible to determine the minimum value of the relative deadline for an n^{th} task such that all the general task set of n tasks is still feasible.

Finally, as [RCM96], in the case where $D_i \leq T_i \ \forall i \in [1, n]$, [GRS96] integrated these upper bound in one NSC for general task sets.

Theorem 6 - *([BHR93], [ZS94], [RCM96], [SPU96], [GRS96]) Any general task set with* $U \leq 1$ *is feasible, using EDF, if and only if:*

$$\forall t \in S, h(t) \leq t \tag{2}$$

where $$S = \left(\bigcup_{j=1}^{n} \left\{ kT_j + D_j, k \in \left[0, \left\lfloor \frac{\lambda_j^s - D_j}{T_j} \right\rfloor \right] \right\} \right) \cap [0, \min\{\lambda, B_1, B_2\}],$$

$$B_1 = \frac{\sum_{D_j \leq T_j}(1 - D_j/T_j)C_j}{1-U} \quad \text{and} \quad B_2 = \max\left\{ \max_{j=1\ldots n}(D_j), \frac{\sum_{j=1}^{n}(1 - D_j/T_j)C_j}{1-U} \right\}.$$

λ_j^s is the length, for task τ_i, of its synchronous deadline busy period and λ the length of the synchronous processor busy period (cf. Section 1.1.2, page 10).

1.2.1.3 Worst-case response time

Contrary to what happens in fixed priority systems, the worst-case response times of a general task set scheduled by EDF are not necessarily obtained with a synchronous pattern of arrival. [GRS96] showed the following lemma.

Lemma 2 - *([GRS96]) the worst-case response time of a task τ_i is found in a deadline busy period for τ_i in which all tasks but τ_i are released synchronously from the beginning of the deadline busy period and at their maximum rate.*

In practice (see [SPU96]), in order to find r_i, the worst-case response time of τ_i, we need to examine several scenarios in which τ_i has an activation released at time a, while all other tasks are released synchronously ($s_j = 0, \forall j \neq i$). Given a value of the parameter a, the response time of the τ_i's activation released at time a is $r_i(a) = \max\{C_i, L_i(a) - a\}$ where $L_i(a)$ is the length of the busy period that includes this τ_i's activation. $L_i(a)$ is computed by means of the following iterative computation:

$$L_i^{(0)}(a) = 0, \; L_i^{(k+1)}(a) = W_i(a, L_i^{(k)}(a)) + (1 + \lfloor a/T_i \rfloor)C_i ,$$

where $W_i(a, t)$ takes into account, for each task but τ_i, the number of instances released within [0,t) and having an absolute deadline smaller or equal to $a + D_i$, i.e.:

$$W_i(a, t) = \sum_{\substack{j \neq i \\ D_j \leq a + D_i}} \min\left\{ \left\lceil \frac{t}{T_j} \right\rceil, 1 + \left\lfloor \frac{a + D_i - D_j}{T_j} \right\rfloor \right\} C_j .$$

[SPU96] first established this result showing that Lemma 2, page 15, holds if the busy period mentioned in the lemma is a processor busy period. Similarly to what was done in the previous section for the establishment of a task set feasibility, [GRS96] improved the result of [SPU96] by using the notion of deadline busy period. Indeed, the worst-case response time of τ_i is always found in a deadline busy period for τ_i (in the contrary, we have $L_i(a) - a < 0$ that cannot lead to a worst case response time for τ_i). Therefore, considering Lemma 2, page 15's patterns of arrival, [GRS96] gave an algorithm that compute λ_i, the maximum length of a synchronous deadline busy period for any task τ_i and showed that $D_i \leq D_j \Rightarrow \lambda_i \leq \lambda_j$. Note that the synchronous processor busy period being the largest busy period, it is possible to maximise λ_i by λ.

According to Lemma 2, page 15, and to the successive considerations, the significant values of the parameter a are in the interval $[0, \lambda_i]$. Furthermore, also within this interval we can restrict our attention to the points where either $s_i = 0$ or $a + D_i$ coincides with the absolute deadline of another activation, which correspond to local maxima of $L_i(a)$. That is, $a \in A \cap [0, \lambda_i]$, where:

$$A = \bigcup_{j=1}^{n} \{kT_j + D_j - D_i, k \in N\} .$$

Finally, a general task set is feasible, using EDF, if and only if:

$$\forall i \in [1, n], \; r_i = \max_{a \geq 0} \{r_i(a)\} \leq D_i . \qquad (3)$$

The complexity of this worst-case response time analysis will be detailled in the sequel and compared with the fixed priority case.

1.2.1.4 Shared resources and release jitters

For reasons such as tick scheduling or distributed contexts (e.g. the holistic approach introduced by [TC95] for fixed priority scheduling that was extended for dynamic priority scheduling in [SPU96-2], [HS96]), tasks may be allowed to have a release jitter (i.e., a task τ_i may be delayed for a maximum time J_i before being actually released). In such case, if a task τ_i is delayed for a maximum time J_i before being actually released, then two consecutive instances of τ_i may be separated by the interval $T_i - J_i$.

Furthermore, if the tasks are allowed to share resources, the analysis must take into account additional terms, namely blocking factors, because of inevitable priority inversions. Note that:

- the maximum duration of such inversions can be bounded if shared resources are accessed by locking and unlocking semaphores according to protocols like the priority ceiling [CL90], [RSL90] or the stack resource algorithm [BA91]. In particular, for each task τ_i it is possible to compute the worst-case blocking time B_i. For example, the priority ceiling of a semaphore is defined as the priority of the highest priority task that may lock that semaphore. A task τ_i is then allowed to enter a critical section if its priority is higher than the ceiling priorities of all the semaphores locked by task other than τ_i. The task will run at its assigned priority unless it is in a critical section and blocks higher priority tasks. In the latter case, it must inherit the highest priority of the tasks it blocks but will resume at its assigned priority on leaving the critical section.

- the length λ of the processor busy periods is unaffected by the presence of blocking instead. Priority inversions may only deviate the schedule from its ordinary EDF characteristic. The required modifications on the analysis are only few. The occurrence being checked, or another one which precedes it in the schedule, may experience a blocking that has to be included as an additional term.

[SPU96] examined the feasibility of a general task sets in presence of shared resources and release jitter.

Theorem 7 - *([SPU96]) a general task set in presence of shared resources and release jitters is feasible (assuming that tasks are ordered by increasing value of D_i-J_i), using EDF, if:*

$$\forall t \le \lambda, \qquad \sum_{D_i \le t + J_i} \left(1 + \left\lfloor \frac{t + J_i - D_i}{T_i} \right\rfloor \right) C_i + B_{k(t)} \le t \qquad . \tag{4}$$

where λ is the size of the synchronous processor busy period and $k(t) = \max\{k | (D_k - J_k \le t)\}$.

The proof generalizes Theorem 6, page 15, showing that the worst processor busy period is still the synchronous processor busy period and that the worst pattern of arrival, considering release jitter, is when tasks experience their shortest inter-release times at the beginning of the schedule. Considering shared resources, it is shown that the worst pattern of arrival arises with the blocking factors of the task with the largest $D_k - J_k$ value among those included in the sum. Similarly, [SPU96] develops the same arguments for the computation of the worst-case response time.

1.2.2 Fixed priority driven scheduling algorithms

In this section, we briefly summarize the principles of preemptive, non-idling, fixed priority scheduling. These principles were derived in particular from [LL73], [LW82], [JP86], [LEH90], [AUD91] and [TBW94]. Note that, contrary to what is happening in the dynamic priority case, feasibility and response time computation are closely related in the state-of-the-art.

1.2.2.1 Optimality

Optimality property is limited in this section to the fixed priority driven class of scheduling algorithms. In other words, a scheduling algorithm x is said to be optimal in the sense that no other fixed priority assignment can lead to a valid schedule which cannot be obtained by x. This limitation reflects the theoretical dominance of the class of dynamic priority driven scheduling algorithms (e.g., a NSC for fixed priority driven scheduling algorithms is a SC considering EDF). Anyhow, this ambiguity on the concept of optimality will be overcome in Section 2.1, page 24, with respect to the concepts of scheduling referential and Σ-optimality.

In the case $\forall i \in [1, n]$, $T_i = D_i$, original work of [LL73] establishes the optimality of the Rate Monotonic (**RM**) priority ordering. The priority assigned to tasks by RM is inversely proportional to their period. Thus the task with the shortest period has the highest priority. For task sets with relative deadlines less or equal to periods, an optimal priority ordering has been shown by [LW82] to be the deadline monotonic (**DM**) ordering. The priority assigned to tasks by DM is inversely proportional to their relative deadline (note that DM is strictly equivalent to RM in the case $\forall i \in [1, n]$, $T_i = D_i$). [LEH90] points out that neither RM nor DM priority ordering policies are optimal for general tasks set (i.e. when relative deadlines are not related to the periods). Finally, [AUD91] solves this problem giving an optimal priority assignment procedure (**Audsley**) in $O(n^2)$ that must be combined with the feasibility analysis presented in the sequel.

Theorem 8 - *([AUD91]) The Audsley priority assignment procedure is optimal for general task sets.*

The procedure first tries to find out if any task with an assigned priority of level n is feasible. Audley proves that if more than one task is feasible with priority level n, one can be chosen arbitrarily amoung the matching tasks. Then the first feasible task of level n is removed from the task set and priority level is decreased by one. The procedure proceeds with the new priority level until either all remaining tasks have been assigned a priority (the task set is then feasible) or for a certain priority level (no task is feasible, then there is no priority ordering for the given task set).

1.2.2.2 Feasibility condition and worst-case response time

■ Case $\forall \mathbf{i} \in [\mathbf{1}, \mathbf{n}]$, $\mathbf{T_i = D_i}$

Theorem 9 - *([LL73]) For a given synchronous periodic task set (with* $\forall i \in [1, n]$, $T_i = D_i$*), the RM schedule is feasible if* $U \leq n\left(2^{\frac{1}{n}} - 1\right)$.

[LL73] proved the sufficiency of this processor utilization test, for tasks assigned priorities according to RM, showing that it is possible to find a least upper bound on the processor utilization. This result gives us a simple O(n) procedure to check the feasibility.

Note that [LL73] also proposed a mixed scheduling algorithm showing that when there is a task set τ scheduled by a fixed priority scheduling algorithm and another task set τ' scheduled by EDF in background (i.e., when the processor is not occupied by the first task set), then the availability function f(t), defined as the accumulated processor time from 0 to T available to τ', can be shown to be sublinear, i.e., a function for which: $f(T) \leq f(T + t) - f(t)$.
It follows that:

- if a given processor busy period obtained by these two task sets leads to an overflow at time t, the resulting synchronous processor busy period cannot lead to an idle time prior to time t.
- a NSC for the second task set is:

  $$\forall t \leq \lambda,\ f(t) \geq \sum_{\tau'_i \in \tau'} \left\lfloor \frac{t}{T_i} \right\rfloor C_i .$$

  where λ is the length of the synchronous processor busy period (cf. Section 1.1.2, page 10).

Note that the application of this result involves the resolution of a large set of inequalities and that [LL73] did not establish a closed form expression for the least upper bound on U (the global processor utilization allowed for τ and τ'). They just proposed hints for simple, but pessimist, sufficient conditions and showed in an example that the bound on U is considerably less restrictive when the number of task scheduled by EDF (rather than a fixed priority scheduling algorithm) grows.

Note also that in [LEH90], Lehoczky shows that in the particular case where periods are harmonic, a 100% processor utilization can be obtained. Other particular cases have been studied in [LEH90] using a processor utilization approach.

In addition to this first approach (based on U) another interesting approach focused on deriving the worst-case response times r_i of each task τ_i of a given non-concrete task set. This led to the obvious following NSC that unifies the feasibility of a task set with the worst-case response times:

$$\forall i \in [1, n] \qquad r_i \leq D_i .$$

Let us summarize now the main results on the worst-case response time computation in several contexts for fixed priority driven scheduling algorithms.

■ Case $\forall i \in [1, n]$, $D_i \leq T_i$

Theorem 10 - *([JP86]) The worst-case response time r_i of a task τ_i of a non-concrete periodic, or sporadic, task set (with $D_i \leq T_i$, $\forall i \in [1, n]$) is found in a scenario in which all tasks are at their maximum rate and released synchronously at a critical instant t=0. r_i is computed by the following recursive equation (where hp(i) denotes the set of tasks of higher priority than task τ_i):*

$$r_i^{k+1} = C_i + \sum_{j \in hp(i)} \left\lceil \frac{r_i^k}{T_j} \right\rceil C_j .$$

Furthermore, the task set is feasible if and only if: $\forall i \in [1, n]$, $r_i \leq D_i$.

In [LL73] tasks are periodic and relative deadlines are equal to periods. [JP86] analysis is more general because it enables relative deadlines to be smaller than periods. The proof shows that this equation is correlated to the synchronous pattern and cannot be worse in the presence of another pattern. The recursion ends when $r_i^{k+1} = r_i^k = r_i$ and can be solved by successive iterations starting from $r_i^0 = C_i$. Indeed, it is easy to show that r_i^k is non decreasing. Consequently, the series converges or exceeds D_i. In the latter case, task τ_i is not schedulable.

■ **General task sets** ($\forall i \in [1, n]$, T_i and D_i are not related)

Theorem 11 - *([LEH90], [TBW94]): The worst-case response time r_i of a task τ_i of a general task set is found in a scenario in which all tasks are at their maximum rate and released synchronously at a critical instant t=0. r_i is computed by the following recursive equation (where hp(i) denotes the set of tasks of higher priority than task τ_i):*

$$r_i = \mathrm{Max}_{0 \le q \le Q}\{w_{i,q} - qT_i\}, \tag{5}$$

where Q is the minimum value such that $w_{i,Q} \le (Q+1)T_i$ and

$$w_{i,q}^{k+1} = (q+1)C_i + \sum_{j \in hp(i)} \left\lceil \frac{w_{i,q}^k}{T_j} \right\rceil C_j \tag{6}$$

Furthermore, the general task set is feasible if and only if: $\forall i \in [1, n]$, $r_i \le D_i$.

The [JP86] analysis is still not sufficient, as it does not consider periods which are smaller than relative deadlines. This limitation is, however, overcome by [LEH90] using the notion of level-i busy period.

Definition 2 - *([LEH90]) A level-i busy period is defined as the maximum interval of time during which a processor runs tasks of higher or equal priorities than task τ_i.*

[LEH90] and [TBW94] showed that the worst response time r_i of a given task τ_i occurs during its synchronous level-i busy period. In that purpose, they showed that it is possible to look successively at several windows, each one starting at a particular arrival of task τ_i. If $w_{i,q}$ denotes the width of the busy period starting at time qT_i before the current activation of task τ_i, the analysis can be performed by the recursive eq. (6), page 20, that ends when $w_{i,q}^{k+1} = w_{i,q}^k = w_{i,q}$ (cf. Appendix A, page 120, for a more formal discussion).

Finally, the worst response time of task τ_i is then given by eq. (5), page 20, where Q is the minimum value such that $w_{i,Q} \le (Q+1)T_i$ (meaning that the maximum length of the level-i busy period has been examined).

Note that the length of the busy periods that need to be examined is bounded by the lowest common multiple of the tasks periods. It is also bounded by: $\frac{1}{T_i} \cdot \sum_{j \le i} C_j \Big/ \left(1 - \sum_{j \le i} \frac{C_j}{T_j}\right)$ [LS95].

1.2.2.3 Shared resources and release jitter

Taking into account shared resources and release jitter in presence of fixed priority driven scheduling leads to the same reasoning than in presence of dynamic priority driven scheduling (cf. Section 1.2.1.4, page 17).

First, when $\forall i \in [1, n]$, $T_i = D_i$, Sha, Rajkumar and Lehoczky [RSL90] have extended the sufficient condition of [LL73] (cf. Theorem 9, page 18) for the Priority Ceiling Protocol:

$$\sum_{j=1}^{n} \frac{C_j}{T_j} + \frac{B_i}{T_i} \leq i(2^{1/i} - 1), 1 \leq i \leq n$$

where B_i denotes the longest blocking time of τ_i a lower priority task.

The same enhancement can also be applied to the Lehoczky's busy period analysis. Moreover [TBW94], in presence of general task sets, extends this analysis further by considering the notion of release jitter (J_i for τ_i). Their analysis is an extension of Theorem 11, page 20, that results in the following final condition:

$$\forall i \in [1, n] , r_i \leq D_i$$

$$r_i = \max_q(w_{i,q} + J_i - qT_i)$$

where $w_{i,q} = (q+1)C_i + B_i + \sum_{j \in hp(i)} \left\lceil \frac{w_{i,q} + J_j}{T_j} \right\rceil C_j$

1.3 Goals

So far, the community of real-time researchers is currently split in two camps, those who support fixed priority driven scheduling algorithms, and those who support dynamic priority driven scheduling algorithms. According to the state-of-the-art (cf. Section 1.2, page 11), a lot of results concerning optimality, feasibility conditions and response time are now available in these two camps (cf. Table 1 for a summary of these results for general task sets). However, to our knowledge, there are only few things concerning a comparison of these fixed/dynamic results (except what was initiated by [LL73] in a limited context, cf. Section 1.2.2.2, page 18).

TABLE 1. existing results for general task sets

Results	fixed priorities	Dynamic priorities
optimality	(Audsley) Theorem 8, page 18	(EDF) Theorem 1, page 12
(Q1) feasibility only		Theorem 6, page 15
(Q2) response time	Theorem 11, page 20	Section 1.2.1.3, page 15

In our opinion, the implementation and the hardware of today no longer justify a simple refusal of dynamic priority algorithms. As an example, unlike the commonly-used FIFO scheduler, [ZE93] focused on the design of hardware deadline scheduler which require that either the transmitter searches

into a waiting queue to find a packet with the earliest deadline to transmit, or a newly arrived packet is inserted at a proper position in an ordered waiting queue. In particular, [ZE93] presented a design for hardware deadline scheduler which can schedule one packet in at most 12 clock cycles and showed the importance of implementing such a device for the performances of real-time channels over shared-medium Local Area Network. To that end, there has been extensive research on fast switch design [AD89], [TO90] where “fast” means that the switching/processing time of a packet can be controled under the transmission time (e.g., with an output-buffer architecture, switches supporting Gbps transmission links have been shown to be feasible with the available VLSI technology).

On the other hand, it seems to us that the theoretical dominance of the optimality property of dynamic priority driven scheduling algorithms (cf. Section 1.2.1.1, page 12, and Section 1.2.2.1, page 18) is not a clear factor to justify a simple refusal of fixed priority algorithms. In fact, given a real-time problem as specified in Section 1.1, page 8, we are interested to measure precisely the dominance of a scheduling algorithms, from another, in terms of:

- (Q1) feasibility analysis. For example, the choice of a particular scheduling algorithm looses (resp. improves) its interest if the given real-time problem is far (resp. close) to an overload situation (where overload means that a given task activation misses its absolute deadline).
- (Q2) individual worst-case response times computation. For example, LLF (cf. Section 1.2.1.1, page 12) might be optimal for a given real-time problem but leads to bad response times. To our knowledge, such a comparison has not been not considered in the literature from a real-time (i.e., non probabilistic) point of view.
  Note that, in a general manner, (Q2) is more informative than (Q1) and that, from the state of the art, (Q2) is used to solve (Q1) for fixed priority scheduling algorithms.

Therefore, that is the first goal of this paper to build a consistant theoretical framework which could encompass the collection of the existing results and that enables us to initiate the definition of new arguments to answer (Q1) and (Q2). More precisely, this will be done evaluating:

- the preemptive, non-idling fixed/dynamic priority driven scheduling algorithms in terms of efficiency for (Q1) (we let for further studies the definition of an efficiency measure to answer (Q2)). To that end, we will introduce in Section 2.1, page 24, the general concept of scheduling referential defined as a couple (class of authorized scheduling algorithms; class of authorized traffics). We will show that the particular class of periodic, non-concrete task sets can be defined as a vector space. It will then be possible to introduce norms in such a vector space and to measure the gap separating a particular scheduling algorithm to an optimal one in a periodic scheduling referential. Let us call such a measure, the efficiency.
- the three associated feasibility conditions (cf. Table 1) in terms of complexity for (Q1) and (Q2). More precisely, our purpose is to establish some upper and lower bounds on the number of basic operations (addition, multiplication and function) involved by these feasibility conditions in order to compare them. To that end, Section 2.2, page 40, will make these NSCs consistent, i.e., tractable and optimized in the same way.
  - Considering the feasibility result (only in dynamic context), the simple expression of $h(t)$ (cf. eq. (2), page 15) will be seen as a predicate that is checked in a bounded interval and on a limited number of points.
  - Considering the feasibility and response time results (in fixed and dynamic context), the NSCs makes use of a recursive expression, such as $w_{i,q}$ in the static case (cf. eq. (6), page 20) and $L_i(a)$ in the dynamic case (cf. Section 1.2.1.3, page 15), in order to check that $r_{i,q} \leq D_i$ and $r_i(a) \leq D_i$. Therefore, the bounds of the number of iterations to converge will be evaluated in details.

In a second step, this paper will initiate such a comparison by a straight, but limited (in the sense that many refinements or other results might be derived from Section 2, page 24), application of our framework. More precisely, Section 3, page 60 (resp. Section 4, page 68), will apply our abstract framework to initiate the efficiency comparison (resp. complexity comparison) of preemptive, fixed/dynamic priority driven scheduling algorithms. This will be done in presence of general task sets, as well as particular task sets, i.e, with specifics relations between relative deadline and period. The obtained preliminary results will be illustrated in Section 5, page 101, with some simple but representative numerical examples (e.g. voice and image traffics versus embedded system traffics) and the indications as well as the limitations of our results will be discussed.

Finally, Section 6, page 116, will focus on a synthesis of our fixed/dynamic priority driven scheduling algorithm comparison leading to find the right trade-off among several solutions for (Q1) and (Q2). Note that in this paper (Q2) will just be compared in terms of complexity.

2 Framework

The purpose of this part is to introduce a general framework for the comparison of scheduling algorithms in terms of efficiency (cf. Section 2.1, page 24) and complexity (cf. Section 2.2, page 40). In order to illustrate our approach, some direct applications of these concepts in our real-time context are given.

2.1 Efficiency Framework

In order to take into account the relativity of such concepts as efficiency and optimality, we will first introduce the notions of scheduling referential, algebraic structure and valid norms for periodic non-concrete task set classes (cf. Section 2.1.1, page 24, and Section 2.1.2, page 25). Then we will give a fairly general definition of the efficiency of any scheduling algorithm P (cf. Section 2.1.3, page 28) and derive an effective procedure to compute exactly the efficiency of P with respect to a periodic scheduling referential that contains EDF (cf. Section 2.1.5, page 32).

2.1.1 Scheduling referential

Definition 3 - ***Scheduling referential***

A scheduling referential is a couple (Υ,Π), where Υ is a class of task sets and Π is a class of scheduling algorithms.
Υ (resp. Π) represents then the class of authorized task sets (resp. scheduling algorithms) on which we are interested to base a comparison analysis.

This concept will enable us to introduce such notions as optimality or efficiency with respect to a particular scheduling referential.

Definition 4 - Σ***-optimality***

Let $\Sigma=(\Upsilon,\Pi)$ be a scheduling referential. $P \in \Pi$ is said to be Σ-optimal if and only if:

$$(\forall \tau \in \Upsilon), (\exists Q \in \Pi, Q(\tau)) \Rightarrow P(\tau)$$

Where τ is a task set and $Q(\tau)$ (resp. $P(\tau)$) means that τ is schedulable by Q (resp. P).

This gives a definition of the optimality of a scheduling algorithm with respect to a particular scheduling referential.

Definition 5 - Σ***-region***

Let $\Sigma=(\Upsilon,\Pi)$ be a scheduling referential. The set of schedulable task sets $\chi(\Sigma)$ is called the Σ-region.

$$\chi(\Sigma) = \{\tau \in \Upsilon, (\exists P \in \Pi, P(\tau))\}$$

Definition 6 - ***Addition***

Let τ_1 and τ_2 be two non-concrete task sets. The non-concrete task set composed of the union of τ_1 and τ_2 is called the addition of τ_1 and τ_2 and written $\tau_1 + \tau_2$.
+ is an associative and commutative operator.

Definition 7 - ***Multiplication***

Let τ be a non-concrete task set and λ be a positive real. The non-concrete task set containing concrete tasks of τ with a computation time multiplied by λ is called the multiplication of τ by λ and written $\lambda \cdot \tau$ or $\lambda\tau$.
$\cdot$ is an associative operator.

2.1.2 Periodic scheduling referential, algebraic structure and valid norms

We consider here the specific case of periodic tasks. Theorem 12, page 25, gives an algebraic structure for periodic scheduling referentials. This will be the base of a very general definition of efficiency provided further.

Definition 8 - ***Periodic scheduling referential, PNTS, PNTSC***

A periodic non-concrete task is a particular case of a non-concrete task characterized by a couple (T,D) (where T is the period and D the relative deadline of the task).

If τ is an addition of n periodic non-concrete tasks, τ is called a periodic non-concrete task set (PNTS).

A periodic scheduling referential[1] is a scheduling referential $\Sigma=(\Upsilon,\Pi)$ where Υ is a PNTSC, i.e., a class of PNTS.

Definition 9 - ***Deadline periodic equivalence***

Two PNTS are said to be deadline periodic equivalent if for every existing couple (T,D), the sum of the computation times of the periodic non-concrete tasks characterized by (T,D) is the same for both PNTS.

From the literature, it seems that all the classic scheduling algorithms use only information about periods and deadlines. Therefore, at least for preemptive algorithms, nothing distinguishes two deadline periodic equivalent PNTS in terms of schedulability and hereafter, in this paper, deadline periodic equivalent PNTS will be identified.

Let Υ be a particular PNTSC. It is easy to extend it so Υ is stable for the operators + and $\cdot$. Moreover, it is also possible to give a meaning to the difference between two PNTS and to PNTS with a negative computation time. By construction, the following theorem can easily be derived.

Theorem 12 - ***An algebraic structure for PNTSC***

Let Υ be a PNTSC, + be the addition operator and $\cdot$ be the multiplication operator.

$(\Upsilon,+,\cdot)$ can be extended to capture PNTS with a negative computation time and differences between PNTS. Then $(\Upsilon,+,\cdot)$ is a vector space and $\{e_{T,D}\}_{(T,D)\in T\times D}$ is a base where $\{e_{T,D}\}_{(T,D)\in T\times D}$ denotes the set of PNTS in Υ with an execution time equal to 1 and characterized by a period T and a deadline D.

1. Since the worst possible density of arrival for general task sets is the periodic one, the scheduling context of this paper can now be represented by a periodic scheduling referential where Υ is a PNTSC and Π is a class of non-idling, preemptive scheduling algorithms.

Definition 10 - ***Valid norms (Criteria)***

Let $\Sigma=(\Upsilon,\Pi)$ be a periodic scheduling referential, $Q \in \Pi$ a given scheduling algorithm and f a norm for the vector space $(\Upsilon,+,\cdot)$.

f is a valid norm, or criterion, for Σ if and only if:

$$(\forall \tau \in \Upsilon), ((\exists Q \in \Pi, Q(\tau)) \Rightarrow f(\tau) \leq 1)$$

In order to illustrate these concepts, let us now introduce trivial valid norms for periodic scheduling referentials.

Theorem 13 - ***N_U and N_{EDF}***

Let $\Sigma = (\Upsilon, \Pi)$ be a periodic scheduling referential where Υ is a PNTSC and Π is a set of non-idling preemptive scheduling algorithms. $\tau \in \Upsilon$ can be written:

$$\tau = \sum_{(T,D) \in T \times D} C_{T,D} e_{T,D}.$$

Let introduce the following notations:

- $N_U(\tau) = \sum_{(T,D) \in T \times D} \frac{|C_{T,D}|}{T}$ where N_U is called the «processor utilization norm».
- $N_{EDF}(\tau) = Max_{t \in \mathbb{R}^+} \left\{ \frac{1}{t} \sum_{(T,D) \in T \times D} Max\left\{0, 1 + \left\lfloor \frac{t-D}{T} \right\rfloor\right\} |C_{T,D}| \right\}$ where N_{EDF} is called the «EDF norm».

Then N_U and N_{EDF} are valid norms, $N_U \leq N_{EDF}$.

Proof

N_U and N_{EDF} are trivial norms. Let $\tau \in \Upsilon$ and $P \in \Pi$, $P(\tau)$ (i.e., τ is P feasible). Since Π contains only non-idling scheduling algorithms, P is non-idling. Since EDF is optimal among non-idling scheduling algorithms, τ is EDF feasible. Thus $N_{EDF}(\tau) \leq 1$ from Theorem 6, page 15. By making t grow towards infinity, it is easy to show that $N_U(\tau) \leq N_{EDF}(\tau)$. Finally, $N_U(\tau) \leq 1$. ❑

The fact that N_{EDF} is a valid norm is very important and can lead to non-trivial properties with respect to admission control mechanisms. Let us consider for instance the admission of multimedia channels (or multimedia bindings) on a system (centralized or distributed). The admission of such channels or bindings is explained in [LS95]. Let us suppose the system is able to admit 10 audio channels of one type (type Audio) on one hand and 2 video channels (type Video) on the other hand. Then if EDF is the scheduling algorithm, it follows that the system is able to admit:

$$\frac{\text{10 Audio Channels + 2 Video Channels}}{2} = \text{5 Audio + 1Video}$$

Theorem 14 - ***Priority workload norm***

Let $\Sigma = (\Upsilon, \Pi)$ be a periodic scheduling referential where Υ is a PNTSC and Π is a set of non-idling preemptive scheduling algorithms.

$\tau \in \Upsilon$. Let us introduce the following notation:

$$N_{WP}(\tau) = Max_i Max_{q \in \mathbb{N}} \left\{ \frac{(q+1)|C_i| + \sum_{j<i} \left\lceil \frac{qT_i + D_i}{T_j} \right\rceil |C_j|}{qT_i + D_i} \right\}$$

where N_W is called the «priority workload norm» and where the (T_i, D_i) are sorted according to a priority order specified by a particular fixed-priority based scheduling algorithm P.

N_W is a norm. In general, it is not a valid norm even if Π contains only fixed-priority based scheduling algorithms.

However, we have the following remarkable properties:

$$N_U(\tau) \leq N_W(\tau) \tag{7}$$

$$N_{WP}(\tau) \leq 1 \Rightarrow P(\tau) \tag{8}$$

Proof

The fact that N_{WP} is a norm comes easily from the following properties:
$Max_{i,q}(A + B) \leq Max_{i,q}A + Max_{i,q}B$
and $Max_{i,q}(\lambda A) = \lambda Max_{i,q}A$ where $\lambda \in \mathbb{R}^+$.

Property (8) comes easily from Theorem 11, page 20. Using eq. (6), page 20, this theorem can be written: $(\forall i)(\forall q)(\exists t \leq qT_i + D_i)\left(t \geq (q+1)C_i + \sum_{j<i} \left\lceil \frac{t}{T_j} \right\rceil C_j\right)$. This form $(\forall i)(\forall q)(\exists t)$ cannot be transformed into a form: $(\forall i)(\forall q)(\forall t)$. This result will be used further and means that there is no closed form in $(\forall t)$ for fixed priorities.

Therefore, a sufficient condition for feasibility is:

$$(\forall i)(\forall q)\left(\frac{(q+1)C_i + \sum_{j<i} \left\lceil \frac{qT_i + D_i}{T_j} \right\rceil C_j}{qT_i + D_i} \leq 1\right)$$ and formula (8) is proved.

By making q grow towards infinity, it is easy to show that $N_U(\tau) \leq N_W(\tau)$ and formula (7) is proved. ❑

We finally introduce a notion which will be used in the comparison of scheduling algorithms. The period dispersion and the deadline dispersion will characterize the PNTSC.

Definition 11 - ***Dispersion***

Let Υ be a PNTSC.

The period dispersion of Υ is equal to $\delta T = \frac{Max(T_i)}{Min(T_i)}$.

The deadline dispersion of Υ is equal to $\delta D = \frac{Max(D_i)}{Min(D_i)}$.

2.1.3 Efficiency and finest criterion

The following definitions give us a fairly general definition of the efficiency as a measure of the distance of the Σ-optimality of a particular scheduling algorithm with respect to a particular periodic scheduling referential. These definitions are very important because it gives a way to compare two different algorithms irrespective of the choice of any valid norm.

Definition 12 - ***Efficiency***

Let $\Sigma=(\Upsilon,\Pi)$ be a periodic scheduling referential, $P \in \Pi$ a given scheduling algorithm and f a norm for the vector space $(\Upsilon,+,\cdot)$. We define:

- $\alpha_f(P)$, the **f-efficiency** of P with respect to Σ as:
$$\alpha_f(P) = Max\{\alpha / (\forall \tau \in \Upsilon, f(\tau) \leq \alpha \Rightarrow P(\tau))\}$$
- $\varepsilon(P)$, the **efficiency** of P with respect to Σ as:
$$\varepsilon(P) = Max_{f \in V[\Upsilon]}\{\alpha_f(P)\}$$
where $V[\Upsilon]$ denotes the set of valid norms for $(\Upsilon,+,\cdot)$.

In other words, $\varepsilon(P)$, the efficiency of a scheduling algorithm P, is the maximum of the f-efficiencies, where f is in the set of valid norms.

Theorem 15 - Σ***-optimality***

Let $\Sigma=(\Upsilon,\Pi)$ be a periodic scheduling referential and $P \in \Pi$.

We have the following implications:

$$(\varepsilon(P) = 1) \Rightarrow (P \text{ is } \Sigma\text{- optimal}) \Rightarrow (\varepsilon(P) \text{is maximal})$$

When there is $P \in \Pi, \varepsilon(P) = 1$, Σ is said to be convex because the Σ-region $\chi(\Sigma)$ is convex.

This last theorem justifies Definition 12, page 28. The efficiency can really be considered as a measure of the optimality of a particular scheduling algorithm and Definition 12, page 28, gives us a quite abstract definition of the efficiency irrespective of any valid norm. However, we are interested by an effective meaning to classify scheduling algorithms in a given periodic scheduling referential Σ. Therefore, our goal is to characterize the «finest» effective criteria, i.e., the best valid norm which enable us to identify Σ-optimal scheduling algorithm and to classify scheduling algorithms according to their efficiency irrespective of the choice of one of these valid norms.

When $P \in \Pi, \varepsilon(P) = 1$, the Σ-region $\chi(\Sigma)$ is convex because there is a valid norm N characterizing such a region: $\chi(\Sigma) = \{\tau \in \Upsilon, N(\tau) \leq 1\}$.

Definition 13 - ***Finest criterion***[1]

Let $\Sigma=(\Upsilon,\Pi)$ be a periodic scheduling referential, a valid norm N_{fc} is said to be a finest criterion of Σ if and only if $\alpha_{N_{fc}} = \varepsilon$.

Theorem 16 - ***Existence of finest criteria***

Let $\Sigma=(\Upsilon,\Pi)$ be a periodic scheduling referential. There is at least one finest criterion of Σ.

Proof

$Max_{f \in V[\Upsilon]} f$ is obviously a valid norm (where $V[\Upsilon]$ denotes the set of valid norms for $(\Upsilon,+,\cdot)$). By construction, it is also a finest criterion. ❑

Theorem 17 - ***Optimality criterion***

With the same assumptions as Definition 13, page 29, if there exists a scheduling algorithm $P_{N_o} \in \Pi$ which verifies: $\alpha_{N_o}(P_{N_o}) = \varepsilon(P_{N_o}) = 1$, then N_o is a finest criterion and is said to be an optimality criterion.

From Theorem 15, page 28, P_{N_o} is optimal.

Proof

Let suppose N_o is an optimality criterion and let us show that it is a finest criterion. Let N be a valid norm.

$\alpha_N \leq \alpha_{N_o}$ is equivalent by definition to: $(\forall P \in \Pi)(\forall \tau \in \Upsilon)(N_o(\tau) \leq \alpha_N(P) \Rightarrow P(\tau))$.

---

1. For instance, let $\Sigma=(\Upsilon,\Pi)$ be a periodic scheduling referential where Υ is a PNTSC and Π a set of non-idling preemptive scheduling algorithms containing EDF. If $\tau \in \Upsilon$ is:

i) EDF feasible, then $N_{EDF}(\tau)$ is in the interval [0, 1],

ii) not EDF feasible, then $N_{EDF}(\tau)>1$ (i.e., $\exists$ t such that h(t)>t).

Then N_{EDF} is an optimality criterion ($\alpha_{N_{EDF}}(EDF) = \varepsilon(EDF) = 1$) and since Π contains only non-idling preemptive scheduling algorithms and EDF is optimal among non-idling preemptive scheduling algorithms then EDF is Σ-optimal.

Let $P \in \Pi$ and $\tau \in \Upsilon$. Let us assume that $N_o(\tau) \leq \alpha_N(P)$. It can be rewritten: $N_o\left(\frac{\tau}{\alpha_N(P)}\right) \leq 1$.

Since N_o is a finest criterion, then $P_{N_o}\left(\frac{\tau}{\alpha_N(P)}\right)$.

Since N is a valid norm, $N\left(\frac{\tau}{\alpha_N(P)}\right) \leq 1$.

Therefore, $N(\tau) \leq \alpha_N(P)$ and by definition, $P(\tau)$.

α_{N_o} appears to be higher than any other α_N. From Definition 12, page 28, it is equal to the efficiency ε. Finally,

$$\alpha_N \leq \alpha_{N_o} = \varepsilon \tag{9}$$

❑

The previous theorem says that an optimality criterion is able to recognize that a particular algorithm is optimal. In fact, it means that an optimality criterion is able to recognize every optimal algorithm. An example of optimality criterion is N_{EDF} if $EDF \in \Pi$.

The following theorem will show that all optimality criteria are equal in a given periodic scheduling referential.

Theorem 18 - ***There is at most one optimality scheduling criterion for a given scheduling referential.***

Proof

Let us show that all optimality criteria are equal[1]. Let N_o and N'_o be two finest criteria. We have already proved that: $\alpha_{N_o} = \alpha_{N'_o} = \varepsilon$ (cf. formula (9)). Let us suppose now that: $(\exists \tau \in \Upsilon)(N_o(\tau) \neq N'_o(\tau))$. Let $P_{N_o} \in \Pi$, such that P_{N_o} is optimal.

For instance: $N_o(\tau) < N'_o(\tau)$.

There is $\lambda \in \mathbb{R}^{+}_{*}$, $N_o(\lambda\tau) < \alpha_{N_o}(P_{N_o}) = \alpha_{N'_o}(P_{N_o}) = 1 < N'_o(\lambda\tau)$, which means that $\lambda\tau$ is schedulable by P according to N_o and not according to N'_o. Impossible. ❑

1. For example, if Υ is the class of non-concrete periodic task sets (such that for each task set, $\forall i \in [1, n]$, $D_i = T_i$) and Π is the class of non-idling, preemptive scheduling algorithms. We have:

$$(\forall t \in \mathbb{R}^{*}_{+})\left(\frac{h(t)}{t} = \frac{1}{t}\sum_{i=1}^{n} \mathrm{Max}\left\{0, 1 + \left\lfloor \frac{t - D_i}{T_i} \right\rfloor\right\} C_i \leq N_U\right) \text{ since } D_i = T_i.$$

By making t grow towards infinity: $N_{EDF} = \sum_{i=1}^{n} \frac{C_i}{T_i} = N_U$.

Theorem 19 - ***If*** Π ***contains only preemptive and non-idling algorithms and EDF, the efficiency of any scheduling algorithm*** $P \in \Pi$***, is equal to*** $\alpha_{N_{EDF}}(P)$.

Proof

Let $P \in \Pi$ and $f_c \in V[\Upsilon]$ a finest criterion.

$\alpha_{f_c}(P) = Max\{\alpha / (\forall \tau \in \Upsilon, f_c(\tau) \leq \alpha \Rightarrow P(\tau))\}$ and $\alpha_{f_c}(P) = \varepsilon(P)$.

The fact that $\alpha_{f_c}(P) \geq \alpha_{N_{EDF}}(P)$ comes from the definition of $\varepsilon(P)$.

To show that $\alpha_{f_c}(P) \leq \alpha_{N_{EDF}}(P)$, it is sufficient to show that $\forall \tau, (N_{EDF}(\tau) \leq \alpha_{f_c}(P) \Rightarrow P(\tau))$.

Since f_c is valid and EDF is in Π, it follows that: $\forall \tau, (N_{EDF}(\tau) \leq 1 \Rightarrow f_c(\tau) \leq 1)$.

And thus: $\forall \tau, (N_{EDF}(\tau) \leq \alpha_{f_c}(P) \Rightarrow f_c(\tau) \leq \alpha_{f_c}(P))$ and $\forall \tau, (N_{EDF}(\tau) \leq \alpha_{f_c}(P) \Rightarrow P(\tau))$. ❑

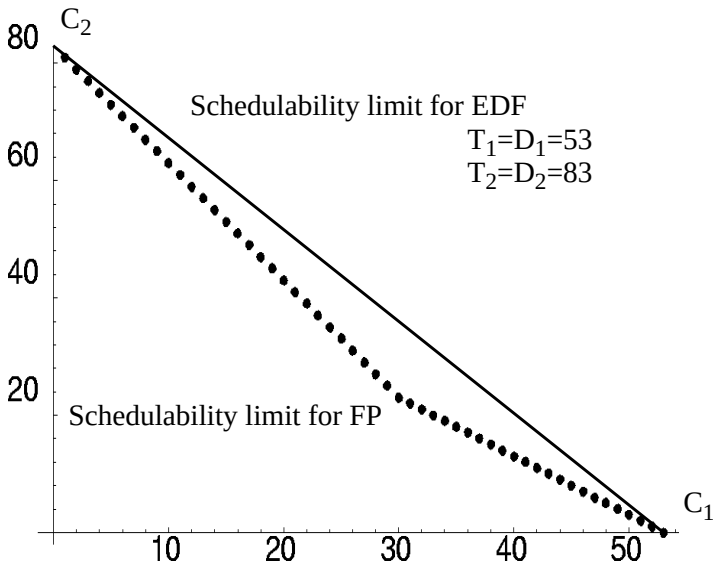
These results are quite important since it gives an effective way to compute the efficiency of a particular scheduling algorithm in a given scheduling referential $\Sigma = (\Pi, \Upsilon)$, at least in certain cases.
For instance, if Π contains only preemptive and non-idling algorithms and EDF, $\varepsilon(P)$, the efficiency of any scheduling algorithm $P \in \Pi$, is equal to $\alpha_{N_{EDF}}(P)$.
In the particular case where $\forall i \in [1, n]$, $D_i = T_i$, we saw from Note 1, page 30, that $\varepsilon(P) = \alpha_{N_{EDF}}(P) = \alpha_{N_U}(P)$ and Liu and Layland proved that if P=RM (Rate Monotonic, cf. Section 1.2.2, page 18), $\varepsilon(RM)$ is underestimated by $n\left(2^{\frac{1}{n}} - 1\right)$.
In the general case (i.e., $\forall i \in [1, n]$, T_i and D_i are not related), Section 2.1.5, page 32, will establish an exact (but costly) efficiency computation procedure. On the other hand, Section 3, page 60, will establish a fast (but rather pessimistic) procedure that underestimates $\varepsilon(P)$. As we will see in the sequel, this last procedure is based on the valid norm N_U that is not a finest criterion and the proposed method will be generalized in Section 3.4, page 66, for further refinements.

In the case where Π has optimal scheduling algorithms, there might be no optimality criterion (for instance in the case where Π contains only fixed priority algorithms, the Audsley algorithm is optimal but there is no optimality criterion) or this one might be very hard to compute. In that latter case, we still can approximate the optimality criterion by any valid norm in order to compute a lower bound on the efficiency. Finally, it might happen that the set Π has no optimal scheduling algorithm. In that case, the general definition of the efficiency is still valid but it is not obviously computable.

2.1.4 Σ-regions of Fixed-Priority based scheduling referentials

If Π contains only Fixed-Priority based scheduling algorithms, then $\chi(\Sigma)$ is in general not convex.

The following example shows the frontier of $\chi(\Sigma)$ in a «Liu and Layland» case both for Fixed-Priority based algorithms and for EDF.



From this simple property, we can easily derive the following theorem which will be used further in Section 2.2.1.1.1, page 41.

Theorem 20 - ***The feasibility test for Fixed-Priority based algorithms cannot be written:***

$((\forall \tau \in \Upsilon) \wedge (\forall t \in S(\tau)))(f(t, \tau) \leq t)$ where f is a positive function which is convex for τ and $S(\tau)$ is a set of real numbers.

Proof

Let suppose that $((\forall \tau \in \Upsilon) \wedge (\forall t \in S(\tau)))(f(t, \tau) \leq t)$ is a feasibility test for a Fixed-Priority based algorithm. Then $Max_{t \in S} \frac{f(t, \tau)}{t}$ where $S = \bigcup_{\tau \in \Upsilon} S(\tau)$ is convex for τ since $Max(A + B) \leq Max(A) + Max(B)$. Therefore $\chi(\Sigma)$ is convex which is false in general. Impossible. ❑

2.1.5 Efficiency computation procedure

From Section 2.1.3, page 28, we will now derive an effective procedure to compute exactly $\varepsilon(P)$, the efficiency of a particular scheduling algorithm P with respect to a periodic scheduling referential $\Sigma = (\Pi, \Upsilon)$ that contains EDF. By definition, Υ is a particular PNTSC characterized by a set of (T_i, D_i) and Π contains non-idling, preemptive algorithms, in particular EDF and fixed priority scheduling algorithms. As N_{EDF} is a finest criterion, $\varepsilon(P)$, the efficiency of P with respect to Σ, is equal to $\alpha_{N_{EDF}}(P)$.

In Section 2.1.5.1, page 33, we will first show that $\forall \tau \in \Upsilon$, $N_{EDF}(\tau)$ can be found in a bounded interval. Then Section 2.1.5.2, page 35, will propose an efficiency procedure to compute $\varepsilon(P)$ w.r.t. Σ.

2.1.5.1 Preliminary results

Let us introduce the following notations:

Wherever $t \geq 0$, let $h(t) = \sum_{j=1}^{n} \text{Max}\left\{0, 1 + \left\lfloor \frac{t - D_j}{T_j} \right\rfloor\right\} C_j$, and $\Psi(t) = N_U(\tau)t - h(t)$.

The following lemma shows that if $\frac{h(t)}{t} \leq N_U(\tau)$ on the interval $[0, \text{LCM}_j\{T_j\}[$ then $N_{EDF}(\tau) \leq N_U(\tau)$.

Furthermore, in that case, we have: $N_{EDF}(\tau) = N_U(\tau)$ since $N_U(\tau) \leq N_{EDF}(\tau)$.

Lemma 3 - $\forall t, 0 \leq t < \text{LCM}_j\{T_j\}, \Psi(t) \geq 0 \Rightarrow \forall t, t \geq 0, \Psi(t) \geq 0$.

Proof

Let ν be a positive real and $\chi_\upsilon(t)$ be the difference between $\Psi(t + \upsilon)$ and $\Psi(t)$.

We have: $\chi_\upsilon(t) = N_U(\tau)\upsilon - \sum_{j=1}^{n} \left(\text{Max}\left\{0, 1 + \left\lfloor \frac{t + \upsilon - D_j}{T_j} \right\rfloor\right\} - \text{Max}\left\{0, 1 + \left\lfloor \frac{t - D_j}{T_j} \right\rfloor\right\} \right) C_j$.

Given that $\forall x \in \mathbb{R}$, $\forall y \in \mathbb{R}$, $\text{Max}\{0, x\} - \text{Max}\{0, y\} \leq \text{Max}\{0, x - y\}$, we have:

$$\chi_\upsilon(t) \geq N_U(\tau)\upsilon - \sum_{j=1}^{n} \text{Max}\left\{0, \left\lfloor \frac{t + \upsilon - D_j}{T_j} \right\rfloor - \left\lfloor \frac{t - D_j}{T_j} \right\rfloor\right\} C_j .$$

Let $\delta_\upsilon^j(t) = \left\lfloor \frac{t + \upsilon - D_j}{T_j} \right\rfloor - \left\lfloor \frac{t - D_j}{T_j} \right\rfloor = \left\lfloor \frac{\upsilon}{T_j} + \frac{t - D_j}{T_j} - \left\lfloor \frac{t - D_j}{T_j} \right\rfloor \right\rfloor$.

Given that $\forall x \in \mathbb{R}^+$, $\forall r$, $0 \leq r < 1$, $0 \leq \lfloor x + r \rfloor \leq \lceil x \rceil$, we have: $0 \leq \delta_\upsilon^j(t) \leq \left\lceil \frac{\upsilon}{T_j} \right\rceil$.

Consequently, $\chi_\upsilon(t) \geq N_U(\tau)\upsilon - \sum_{j=1}^{n} \left\lceil \frac{\upsilon}{T_j} \right\rceil C_j$.

Let ν be the smallest strictly positive root of the equation $\nu N_U(\tau) = \sum_{j=1}^{n} \left\lceil \frac{\upsilon}{T_j} \right\rceil C_j$. Clearly, $\text{LCM}_j\{T_j\}$ is the smallest root of this equation.

Finally, $\forall t, 0 \leq t < \text{LCM}_j\{T_j\}, \Psi(t) \geq 0 \Rightarrow \forall t, t \geq 0, \Psi(t) \geq 0$. ❑

Let us now introduce $\sigma = \sum_{i=1}^{n} (T_i - D_i)\frac{C_i}{T_i}$. The following lemma can easily be derived:

Lemma 4 - $\forall t \geq Max(D_i), \frac{h(t)}{t} \leq N_U(\tau) + \frac{\sigma}{t}$

Proof

By definition $\frac{h(t)}{t} = \frac{1}{t}\sum_{j=1}^{n} Max\left\{0, 1 + \left\lfloor \frac{t - D_j}{T_j} \right\rfloor\right\} C_j$. $\forall t \geq Max(D_i)$, we have:

$$\frac{h(t)}{t} = \frac{1}{t}\sum_{j=1}^{n}\left\lfloor \frac{t + T_j - D_j}{T_j} \right\rfloor C_j \leq \frac{1}{t}\sum_{j=1}^{n}\left(\frac{t + T_j - D_j}{T_j}\right)C_j = N_U(\tau) + \frac{1}{t}\sum_{j=1}^{n}\left(\frac{T_j - D_j}{T_j}\right)C_j .$$

Finally, $\frac{h(t)}{t} \leq N_U(\tau) + \frac{\sigma}{t}$. ❑

This last lemma yields the following results:

- if $\sigma \leq 0$, $N_{EDF}(\tau) = Max\left(Max_{0 < t < Max(D_i)}\left(\frac{h(t)}{t}\right), N_U(\tau)\right)$, that is:

$$N_{EDF}(\tau) = Max\left(Max_{Min(D_i) \leq t < Max(D_i)}\left(\frac{h(t)}{t}\right), N_U(\tau)\right),$$

- if $\sigma > 0$, the following lemma shows that $N_{EDF}(\tau)$ can be found in a bounded interval.

Lemma 5 - *If* $\sigma \geq 0$ *and if* $\exists t_\sigma > 0, \frac{h(t_\sigma)}{t_\sigma} > N_U(\tau)$, *then*

$$\forall t \geq Max\left(Max(D_i), \frac{\sigma}{\frac{h(t_\sigma)}{t_\sigma} - N_U(\tau)}\right), \frac{h(t)}{t} \leq \frac{h(t_\sigma)}{t_\sigma} \qquad (10)$$

Proof

Let $t \geq Max\left(Max(D_i), \frac{\sigma}{\frac{h(t_\sigma)}{t_\sigma} - N_U(\tau)}\right)$. From Lemma 4, page 34, it follows that:

$$\frac{h(t)}{t} \leq N_U(\tau) + \frac{\sigma}{t} \leq N_U(\tau) + \frac{\sigma}{\frac{\sigma}{\frac{h(t_\sigma)}{t_\sigma} - N_U(\tau)}} = \frac{h(t_\sigma)}{t_\sigma}$$ ❑

2.1.5.2 Efficiency procedure

Combining the previous lemmata, we will now derive an effective procedure to compute exactly $\varepsilon(P)$, the efficiency of a particular scheduling algorithm P with respect to a periodic scheduling referential $\Sigma = (\Pi, \Upsilon)$ that contains EDF. This procedure can be costly but leads to the exact value of $\varepsilon(P)$ (a faster method to establish an underestimation of $\varepsilon(P)$ will be established in Section 3, page 60). In fact, by exact computation, we mean that the efficiency can be computed as precisely as possible. The precision is bounded by: $\delta C / \mathrm{Min}_j\{T_j, D_j\}$ where δC is the incrementation value of C. [1]

By definition, Υ is a particular PNTSC characterized by a set of (T_i, D_i) and $\Pi = \{\mathrm{EDF}, P, \ldots\}$. As N_{EDF} is a finest criterion, $\varepsilon(P)$, the efficiency of P with respect to Σ, is equal to $\alpha_{N_{EDF}}(P)$. Then $\varepsilon(P)$ is computed by an exhaustive examination of all possible tasks sets in Υ which are feasible by P (i.e., $P(\tau)$). This exhaustive examination as described in the following pseudo-code, can be done by nested loops if the number n of periodic tasks is fixed (an alternative way is to do it recursively).

According to Section 2.1.5.1, page 33, the computation of N_{EDF} is done in a bounded interval (cf. Function $N_{EDF}(\tau \in \Upsilon)$) and only when the task set is no longer feasible by P on the previous PNTS τ_b which was feasible (cf. Function $\mathrm{Efficiency}(P \in \Pi, \Sigma)$). The conditional test $\langle P(\tau) \rangle$ at the beginning of each loop can be optimized by first a very rough test using a sufficient condition (e.g. in Section 2.2, page 40, the Efficiency theorem will give an interesting sufficient condition using the processor load norm N_U). The exact test $\langle P(\tau) \rangle$ will be computed only when the previous one fails.

Function $\mathrm{Efficiency}(P \in \Pi, \Sigma)$
For $\langle (i = 1) \rangle$ **to n Do** $C_i \leftarrow 0$ **; EndFor;** $\varepsilon(P, \Sigma) \leftarrow \infty$ **;** $N_{EDF} \leftarrow \infty$ **;**
While $\langle P(\tau) \rangle$ **Do**
$\tau_b \leftarrow \tau$ **;** $C_1 \leftarrow C_1 + 1$ **;**
While $\langle P(\tau) \rangle$ **Do**
$\tau_b \leftarrow \tau$ **;** $C_2 \leftarrow C_2 + 1$ **;**
...
While $\langle P(\tau) \rangle$ **Do**
$\tau_b \leftarrow \tau$ **;** $C_n \leftarrow C_n + 1$ **;**
EndWhile;
$N_{EDF} \leftarrow \mathrm{CallFunctionN}_{EDF}(\tau_b)$ **;**
$\varepsilon(P, \Sigma) \leftarrow \mathrm{Min}(\varepsilon(P, \Sigma), N_{EDF})$ **;**
...
EndWhile;
$N_{EDF} \leftarrow \mathrm{CallFunctionN}_{EDF}(\tau_b)$ **;**
$\varepsilon(P, \Sigma) \leftarrow \mathrm{Min}(\varepsilon(P, \Sigma), N_{EDF})$ **;**
EndWhile;
$N_{EDF} \leftarrow \mathrm{CallFunctionN}_{EDF}(\tau_b)$ **;**
$\varepsilon(P, \Sigma) \leftarrow \mathrm{Min}(\varepsilon(P, \Sigma), N_{EDF})$ **;**
Return $\varepsilon(P, \Sigma)$ **;**
EndFunction;

1. This bound comes from the fact that $N_{EDF}(\tau + \delta\tau) \leq N_{EDF}(\tau) + N_{EDF}(\delta\tau)$.

Function $N_{EDF}(\tau \in \Upsilon)$

$\sigma \leftarrow \sum_{i=1}^{n} (T_i - D_i)\frac{C_i}{T_i}$;

$M_1 \leftarrow h(1)$;

$t \leftarrow Min(D_i)$; **/* with t=0 a critical instant */**

If $\sigma \leq 0$ **Then** **/* cf. Lemma 3 and Lemma 4*/**

While $\langle t < Min(Max(D_i), LCM(T_i)) \rangle$ **Do**

$M_1 \leftarrow Max\left(M_1, \frac{h(t)}{t}\right)$;

t <- next absolute deadline;

EndWhile;

If $\langle M_1 \leq N_U(\tau) \rangle$ **Then** $N_{EDF}(\tau) \leftarrow N_U(\tau)$;

Else **If** $\langle Max(D_i) \leq LCM(T_i) \rangle$ **Then** $N_{EDF}(\tau) \leftarrow M_1$;

Else $M_2 \leftarrow \frac{h(t)}{t}$;

t <- next absolute deadline;

While $\langle t < Max(D_i) \rangle$ **Do**

$M_2 \leftarrow Max\left(\frac{h(t)}{t}, M_2\right)$;

t <- next absolute deadline;

EndWhile;

$N_{EDF}(\tau) \leftarrow Max(M_1, M_2)$;

EndIf;

EndIf;

Else **/* cf. Lemma 3, Lemma 4 and Lemma 5*/**

If $\langle Max(D_i) \leq LCM(T_i) \rangle$ **Then**

While $\langle t < Max(D_i) \rangle$ **Do**

$M_1 \leftarrow Max\left(M_1, \frac{h(t)}{t}\right)$;

t <- next absolute deadline;

EndWhile;

$t_1 \leftarrow LCM(T_i)$;

$M_2 \leftarrow N_U(\tau)$;

While $\langle t \leq t_1 \rangle$ **Do**

If $\frac{h(t)}{t} > M_2$ **Then** $t_1 \leftarrow \frac{\sigma}{\frac{h(t)}{t} - N_U(\tau)}$;

$M_2 \leftarrow \frac{h(t)}{t}$;

EndIf;

t <- next absolute deadline;

EndWhile;

$N_{EDF}(\tau) \leftarrow Max(M_1, M_2)$;

Else While $\langle t < LCM(T_i) \rangle$ **Do**

$M_1 \leftarrow Max\left(M_1, \frac{h(t)}{t}\right)$;

t <- next absolute deadline;

EndWhile;

If $M_1 \leq N_U(\tau)$ **Then** $N_{EDF}(\tau) \leftarrow N_U(\tau)$;

Else $t_1 \leftarrow Max(D_i)$;

$M_2 \leftarrow N_U(\tau)$;

While $\langle t \leq t_1 \rangle$ **Do**

If $\frac{h(t)}{t} > M_2$ **Then**

$t_1 \leftarrow Max\left(Max(D_i), \frac{\sigma}{M_2 - N_U(\tau)}\right)$;

$M_2 \leftarrow \frac{h(t)}{t}$;

EndIf;

t <- next absolute deadline;

EndWhile;

$N_{EDF}(\tau) \leftarrow Max(M_1, M_2)$;

EndIf;

EndIf;

EndIf;

Return $N_{EDF}(\tau)$;

EndFunction;

Let us now illustrate the efficiency procedure on the following example. We consider the scheduling referential $\Sigma = (\Upsilon, \Pi)$, where:

- $\forall \tau \in \Upsilon, \tau = C_1 \cdot e_{7,5} + C_2 \cdot e_{10,7} + C_3 \cdot e_{13,10}$,
- $\Pi = \{EDF, HPF/DM\}$, (Earliest Deadline First and Highest Priority First / Deadline Monotonic).

We define the $\Sigma - region(P)$ of a scheduling algorithm P as follows:

$$\forall \tau \in \Upsilon, (\tau \in \Sigma - region(P)) \Leftrightarrow P(\tau)$$

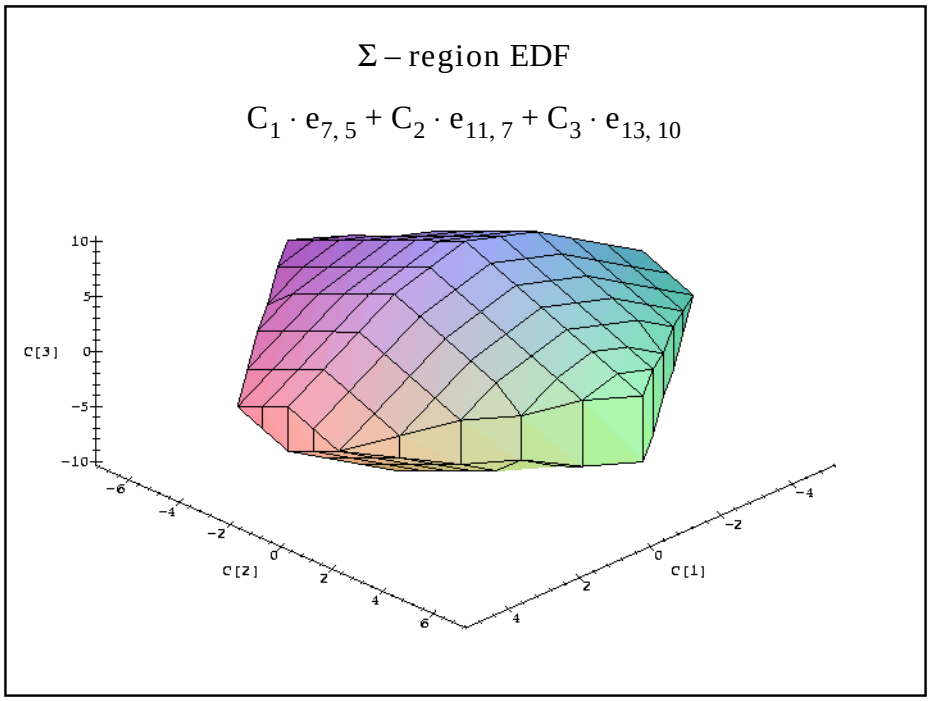
From Definition 4, page 24, and Definition 5, page 24, we have:

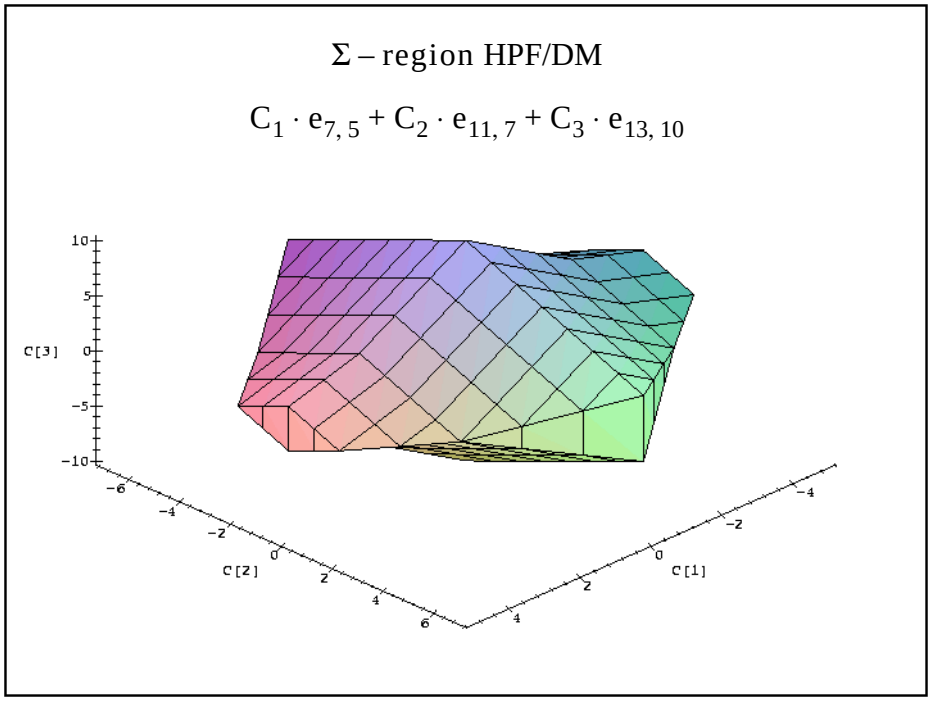
$\chi(\Sigma) = \{\tau \in \Upsilon, (\exists Q \in \Pi, Q(\tau))\} = \bigcup_{P \in \Pi} \Sigma - region(P)$, that is:

$\chi(\Sigma) = \{\tau \in \Upsilon, P(\tau)\} = \Sigma - region(P)$, where $P \in \Pi$ is $\Sigma - optimal$.

In this example, $\chi(\Sigma) = \{\tau \in \Upsilon, EDF(\tau)\}$, since $EDF \in \Pi$ is $\Sigma - optimal$.

The efficiency procedure computes $\varepsilon(P) = \alpha_{N_{EDF}}(P) \equiv Min_{\tau \in \text{ Polyhedron(P)}}\{N_{EDF}(\tau)\}$ where Polyhedron(P) denotes the set of the vertices of the $\Sigma - region(P)$.





It is important to point out that:

- the Σ-region $\chi(\Sigma) = \Sigma - \text{region(EDF)}$ is convex because there is a valid norm N_{EDF} characterizing such a region: $\chi(\Sigma) = \{\tau \in \Upsilon, N_{EDF}(\tau) \leq 1\}$,
- $\Sigma - \text{region(HPF/DM)} \subset \Sigma - \text{region(EDF)}$ since EDF is $\Sigma - \text{optimal}$.

The computation of $\varepsilon(P)$ is done very quickly but with a precision bounded by:

$$\frac{\delta C}{\text{Min}_j\{T_j, D_j\}} = \frac{1}{5} = 20\%$$

since $\delta C = 1$. To obtain (at a greater cost) a better precision (for example 1%), one must set δC, the incrementation value of C, to 0.05.

The following code, which is a Maple V Release 4 (computer algebra system) code, is the recursive implementation of Function $N_{EDF}(\tau)$.

Task set τ

```
> N:=3; C[1]:=2; T[1]:=7; d[1]:=5; C[2]:=3; T[2]:=11; d[2]:=7; C[3]:=5; T[3]:=13; d[3]:=10;
```

$LCM_j\{T_j\}$ (LCM), $Min_j\{D_j\}$ (Min) and $Max_j\{D_j\}$ (Max)

```
> LCM:=lcm(seq(T[j],j=1..N)); Min:=min(seq(d[j],j=1..N)); Max:=max(seq(d[j],j=1..N));
```

The processor utilization norm (rho), σ (sigma), and the h(t) demand processor function

```
> rho:=sum(abs(C[i])/T[i],i=1..N); sigma:=sum((T[i]-d[i])*abs(C[i])/T[i], i=1..N);
> h:=t->sum(max(0,1+floor((t-d[i])/T[i]))*abs(C[i]), i=1..N);
```

To set $N_{EDF}(\tau)$ to $N_U(\tau)$

```
> Nedf:=rho;
```

Next(t) computes the point of discontinuities which is greater than t

```
> Next := proc(t) local j; min(seq(d[j]+(1+floor((t-d[j])/T[j]))*T[j],j = 1 .. N)) end
```

Norme_EDF(a, b) computes $N_{EDF}(\tau) = Max\left\{Max_{t<a}\left\{\frac{h(t,\tau)}{t}\right\}, Max_{t\geq a}\left\{\frac{h(t,\tau)}{t}\right\}\right\}$

```
> Norme_EDF := proc(a,b) local t,norm; global Max,sigma,Nedf;
    t := a; norm := h(t)/t;
    while norm <= Nedf and t < b do  t := Next(t); norm := h(t)/t od;
    if b <= t then RETURN(Nedf)
    else Nedf := norm; RETURN(Norme_EDF(Next(t),max(Max,sigma/(Nedf-rho))))
    fi                                   (cf. Lemma 5, page 34)
  end
```

Main( ) computes $N_{EDF}(\tau)$ (cf. Lemma 3, page 33, and Lemma 4, page 34)

```
> Main := proc()
    local t,norme;
    global Min,Max,rho,sigma,LCM,Nedf;
      if 0 < sigma then RETURN(Norme_EDF(Min,LCM))
      else RETURN(Norme_EDF(Min,min(Max,LCM)))
      fi
    end
```

$$N_{EDF}(2 \cdot e_{7,5} + 3 \cdot e_{11,7} + 5 \cdot e_{13,10}) = 1$$

```
> Main();
```

1

2.2 Complexity framework

The purpose of this part is to introduce a general framework to compare the complexity of the feasibility tests associated to preemptive, fixed and dynamic priority driven scheduling algorithms in terms of basic operations (i.e., addition, multiplication and function). To this end, we will consider a periodic scheduling referential $\Sigma = (\Upsilon, \Pi)$, where Υ is a PNTSC (i.e., a class of general or particular task sets) and Π is a set of non-idling preemptive scheduling algorithms, which contains the Σ-optimal Earliest Deadline First scheduling algorithm and a set of Fixed Priority based scheduling algorithms.

The existing pseudo-polynomial necessary and sufficient feasibility tests for Fixed Priority based scheduling algorithms and for the Earliest Deadline First scheduling algorithm are introduced in the state-of-the-art in Section 1.2, page 11. These tests have been established separately, which makes the comparison of their respective complexity difficult. Therefore, in order to make a coherent and significant comparison, the first step will show that, whatever the scheduling algorithm, it is possible to use a similar approach to establish tractable and optimized feasibility tests. Then, the second step will show how to establish their complexity in terms of basic operations.

More precisely, we will first state these feasibility tests under their intractable forms in Section 2.2.1, page 41. Then Section 2.2.2, page 42, will formally explain how the concept of busy period enables us to make these feasibility tests tractable. Indeed, this concept leads to a restriction of the number of points where the feasibility checking should be done. Moreover, as will be seen in Section 2.2.3, page 49, the tractable feasibility tests based upon calculation of the worst-case response time of a task might be improved according to a faster convergence computation of $w_{i,q}$ and $L_i(a)$. Then, we will express the feasibility tests in their optimized form (cf. Section 2.2.4, page 49) and we will show how to establish and to compare their complexity in terms of iterations and basic operations (cf. Section 2.2.5, page 58).

By means of this general framework, which is summarized in Section 2.2.6, page 59, we will compare the complexity of the Σ-optimal EDF algorithm with Fixed Priority based scheduling algorithms in presence of general task sets, as well as particular task sets (cf. Section 4, page 68).

2.2.1 The necessary and sufficient feasibility tests

The necessary and sufficient feasibility tests for scheduling algorithms can be expressed in:

- a « $\forall t \in S, P(t)$ » form, where P is a predicate and S is the set of points where the predicate P should be checked,
- a « $\forall i \in [1, n], r_i \leq D_i$ » form, where r_i (resp. D_i) is the worst-case response time (resp. the relative deadline) of a task τ_i.

It is important to point out that the necessary and sufficient feasibility tests which are based upon calculation of the worst-case response time of a task yield not only the feasibility of the task set but also the worst-case response time r_i of a task τ_i. The computation of r_i is important if we want to analyze a global distributed system according to the holistic approach introduced by Tindell et al. [TC95].

2.2.1.1 The feasibility tests in a « $\forall t \in S, P(t)$ » form

2.2.1.1.1 Fixed Priority based scheduling algorithms

From Theorem 20, page 32, we know that the necessary and sufficient feasibility test for Fixed Priority based scheduling algorithms cannot be written:

$$((\forall \tau \in \Upsilon) \wedge (\forall t \in S(\tau)))(f(t, \tau) \leq t)$$

where f is a positive function which is convex for τ and $S(\tau)$ is a set of real numbers.

2.2.1.1.2 The Earliest Deadline First scheduling algorithm

The necessary and sufficient feasibility test for the Earliest Deadline First scheduling algorithm in a « $\forall t \in S, P(t)$ » form is as follows:

$$\forall t > 0, \sum_{j=1}^{n} \mathrm{Max}\left\{0, 1 + \left\lfloor \frac{t - D_j}{T_j} \right\rfloor\right\} C_j \leq t$$

This test is strictly equivalent to: $N_{EDF}\{\tau\} \leq 1, \tau \in \Upsilon$. It is also called the norm based feasibility test for EDF.

2.2.1.2 The feasibility tests in a « $\forall i \in [1, n], r_i \leq D_i$ » form

2.2.1.2.1 Fixed Priority based scheduling algorithms

The necessary and sufficient feasibility test for Fixed Priority based scheduling algorithms is as follows:

$$\forall i \in [1, n], r_i \leq D_i$$

where r_i, the worst-case response time of a task τ_i, is given by: $r_i = \mathrm{Max}_{q \geq 0}\{w_{i,q} - qT_i\}$.

The response time $r_{i,q} = w_{i,q} - qT_i$ corresponding to the given level-i busy period starting qT_i before the current activation of τ_i is given by the difference of $w_{i,q}$ and qT_i. The width $w_{i,q}$ of such a level-i busy period is given by:

$$w_{i,q} = (q + 1)C_i + \sum_{j < i} \left\lceil \frac{w_{i,q}}{T_j} \right\rceil C_j$$

2.2.1.2.2 The Earliest Deadline First scheduling algorithm

The necessary and sufficient feasibility test for the Earliest Deadline First algorithm is as follows:

$$\forall i \in [1, n], r_i \leq D_i$$

where r_i, the worst-case response time of a task τ_i, is given by: $r_i = Max_{a \geq 0}\{L_i(a) - a\}$.

The response time $r_i(a) = L_i(a) - a$ corresponding to the resulting busy period relative to the absolute deadline $a + D_i$ starting at time a before the current activation of τ_i is given by the difference of $L_i(a)$ and a. The length $L_i(a)$ of such a busy period is given by:

$$L_i(a) = \left(1 + \left\lfloor \frac{a}{T_i} \right\rfloor\right) C_i + \sum_{j \neq i} \min\left\{ \left\lceil \frac{L_i(a)}{T_j} \right\rceil, \max\left\{0, 1 + \left\lfloor \frac{a + D_i - D_j}{T_j} \right\rfloor\right\}\right\} C_j$$

These last two feasibility tests are based upon calculation of worst-case response time of a task and are quite similar. More precisely, for Fixed Priority based scheduling algorithms, the worst-case response time r_i of a task τ_i is given by the maximum over $q \in \mathbb{N}$ of the difference of $w_{i,q}$ and qT_i, where $w_{i,q}$ is the width of the resulting level-i busy period starting at time qT_i before the release of the current invocation of τ_i. For the Earliest Deadline First scheduling algorithm, the worst-case response time r_i of a task τ_i is given by the maximum over $a \in \mathbb{R}$ of the difference of $L_i(a)$ and a, where $L_i(a)$ is the length of the resulting busy period relative to the absolute deadline $a+D_i$ starting at time a before the current invocation of τ_i.

In a general manner, these feasibility tests are intractable. Indeed, the number of points where the feasibility checking should be done is infinite. Consequently, we need to render these tests tractable. This is the purpose of following section, Section 2.2.2, page 42.

2.2.2 From intractable to tractable feasibility tests

If we want to obtain tractable feasibility tests, we need to introduce and to use the notion of busy period. This enables us to restrict the set of points where the feasibility checking should be done.

Let $\tau \in \Upsilon$. In a general manner, the busy period $\lambda(\tau)$ is defined as the maximum interval of time during which the processor runs tasks. λ is determined through the following recursive equation:

$\lambda = \sum_j \left\lceil \frac{\lambda}{T_j} \right\rceil C_j$ and is bounded by: $\text{Min}\left\{ \sum_j \frac{C_j}{T_j} \cdot \text{LCM}_l\{T_l\}, \frac{\sum_j C_j}{1 - \sum_j \frac{C_j}{T_j}} \right\}$ (cf. Appendix A).

The busy period $\lambda(\tau)$ is important and very useful to obtain the tractable necessary and sufficient feasibility tests for the Earliest Deadline First scheduling algorithm both in the « $\forall t \in S, P(t)$ » form and in the « $\forall i \in [1, n], r_i \leq D_i$ » form as we will see further on.

This notion can easily be extended to deal with Fixed Priority based scheduling algorithms. In that case, a level-i busy period $\lambda_i(\tau)$ is defined as the maximum interval of time during which the processor runs tasks of higher priority than task τ_i. λ_i is determined through the following recursive equation:

$\lambda_i = \sum_{j \leq i} \left\lceil \frac{\lambda_i}{T_j} \right\rceil C_j$ and is bounded by: $\text{Min}\left\{ \sum_{j \leq i} \frac{C_j}{T_j} \cdot \text{LCM}_{l \leq i}\{T_l\}, \frac{\sum_{j \leq i} C_j}{1 - \sum_{j \leq i} \frac{C_j}{T_j}} \right\}$ (cf. Appendix A).

2.2.2.1 The feasibility test for EDF in a « $\forall t \in S, P(t)$ » form

The necessary and sufficient feasibility test for the Earliest Deadline First scheduling algorithm in a « $\forall t \in S, P(t)$ » form is intractable at first glance. Indeed, the set of points where the predicate P should be checked is $\mathbb{R}^{+*}$.

To obtain a tractable feasibility test, we have to restrict the set of points where the predicate P should be checked. To achieve that purpose, we demonstrate the following lemmata (Lemma 6, page 43, and Lemma 7, page 44) and corollary (Corollary 1, page 44).

Wherever $t \geq 0$, let $h(t) = \sum_{j=1}^{n} \mathrm{Max}\left\{0, 1 + \left\lfloor \frac{t - D_j}{T_j} \right\rfloor\right\} C_j$ and $\Phi(t) = t - h(t)$.

Lemma 6 - $\forall t, 0 \leq t < \lambda, \Phi(t) \geq 0 \Rightarrow \forall t, t \geq 0, \Phi(t) \geq 0$

where λ ***is determined through the following equation:*** $\lambda = \sum_{j=1}^{n} \left\lceil \frac{\lambda}{T_j} \right\rceil C_j$

Proof.

Let λ be a positive real and $\Delta_\lambda(t)$ be the difference of $\Phi(t + \lambda)$ and $\Phi(t)$. We have:

$$\Delta_\lambda(t) = \lambda - \sum_{j=1}^{n} \left(\mathrm{Max}\left\{0, 1 + \left\lfloor \frac{t + \lambda - D_j}{T_j} \right\rfloor\right\} - \mathrm{Max}\left\{0, 1 + \left\lfloor \frac{t - D_j}{T_j} \right\rfloor\right\} \right) C_j .$$

Given that $\forall x \in \mathbb{R}$, $\forall y \in \mathbb{R}$, $\mathrm{Max}\{0, x\} - \mathrm{Max}\{0, y\} \leq \mathrm{Max}\{0, x - y\}$, we have:

$$\Delta_\lambda(t) \geq \lambda - \sum_{j=1}^{n} \mathrm{Max}\left\{0, \left\lfloor \frac{t + \lambda - D_j}{T_j} \right\rfloor - \left\lfloor \frac{t - D_j}{T_j} \right\rfloor\right\} C_j$$

Let $\delta_\lambda^j(t) = \left\lfloor \frac{t + \lambda - D_j}{T_j} \right\rfloor - \left\lfloor \frac{t - D_j}{T_j} \right\rfloor = \left\lfloor \frac{\lambda}{T_j} + \frac{t - D_j}{T_j} - \left\lfloor \frac{t - D_j}{T_j} \right\rfloor \right\rfloor$.

Given that $\forall x \in \mathbb{R}^{+}$, $\forall r$, $0 \leq r < 1$, $0 \leq \lfloor x + r \rfloor \leq \lceil x \rceil$, we have: $0 \leq \delta_\lambda^j(t) \leq \left\lceil \frac{\lambda}{T_j} \right\rceil$.

Consequently, $\Delta_\lambda(t) \geq \lambda - \sum_{j=1}^{n} \left\lceil \frac{\lambda}{T_j} \right\rceil C_j$.

Let λ be the smallest positive root of the following equation: $\lambda = \sum_{j=1}^{n} \left\lceil \frac{\lambda}{T_j} \right\rceil C_j$.

Thus, we have:
$\forall t, t \geq 0, \Delta_\lambda(t) \geq 0 \Leftrightarrow \forall t, t \geq 0, \Phi(t + \lambda) \geq \Phi(t)$.

Finally, this last yields:
$\forall t, 0 \leq t < \lambda, \Phi(t) \geq 0 \Rightarrow \forall t, t \geq 0, \Phi(t) \geq 0$. ❑

Lemma 7 - $\forall t, 0 \le t < \mu, \Phi(t) \ge 0 \Rightarrow \forall t, t \ge 0, \Phi(t) \ge 0$

$$\textit{where } \mu = Max\left\{ Max_{1 \le j \le n}\{D_j\}, \frac{\sum_{j=1}^{n} (T_j - D_j)\frac{C_j}{T_j}}{1 - \sum_{j=1}^{n} \frac{C_j}{T_j}} \right\}$$

Proof.

Let $t \ge \mu \ge Max_{1 \le j \le n}\{D_j\}$. From this assumption, we have:

$$\Phi(t) = t - \sum_{j=1}^{n} \left(1 + \left\lfloor \frac{t - D_j}{T_j} \right\rfloor\right) C_j .$$

Given that $\forall x \in \mathbb{R}, \lfloor x \rfloor \le x$, we have:

$$\Phi(t) \ge t - \sum_{j=1}^{n} \left(1 + \frac{t - D_j}{T_j}\right) C_j \ge \left(1 - \sum_{j=1}^{n} \frac{C_j}{T_j}\right) t - \sum_{j=1}^{n} (T_j - D_j)\frac{C_j}{T_j} \ge 0 .$$

Finally, this last yields:
$\forall t, 0 \le t < \mu, \Phi(t) \ge 0 \Rightarrow \forall t, t \ge 0, \Phi(t) \ge 0$. ❑

Corollary 1 - $\forall t, 0 \le t < Min\{\lambda, \mu\}, \Phi(t) \ge 0 \Rightarrow \forall t, t \ge 0, \Phi(t) \ge 0$

The proof of this corollary is immediate from Lemma 6, page 43, and Lemma 7, page 44.

Corollary 1, page 44, enables us to restrict the set of points where the predicate P should be checked. We have:

$$\forall t, 0 \le t < Min\{\lambda, \mu\}, P(t) \Rightarrow \forall t, t \ge 0, P(t)$$

where $\lambda = \sum_{j=1}^{n} \left\lceil \frac{\lambda}{T_j} \right\rceil C_j$ and $\mu = Max\left\{ Max_{1 \le j \le n}\{D_j\}, \frac{\sum_{j=1}^{n} (T_j - D_j)\frac{C_j}{T_j}}{1 - \sum_{j=1}^{n} \frac{C_j}{T_j}} \right\}$.

In a general manner, the predicate P should be checked over the busy period λ.

2.2.2.2 The feasibility tests in a « $\forall i \in [1, n], r_i \le D_i$ » form

2.2.2.2.1 Fixed Priority based scheduling algorithms

The necessary and sufficient feasibility test for Fixed Priority based scheduling algorithms in a « $\forall i \in [1, n], r_i \le D_i$ » form, is intractable at first glance. Indeed, the set of points where the predicate P should be checked is $\mathbb{N}$.

To obtain a tractable feasibility test, we have to restrict the set of points where the computation of r_i should be done. To achieve that purpose, we demonstrate the following lemma (Lemma 8, page 45) and corollary (Corollary 2, page 46).

Lemma 8 - $\forall q \in \mathbb{N}$, $w_{i,\,q+Q_i+1} - w_{i,\,q} \le \lambda_i$ *where* $Q_i = \left\lceil \frac{\lambda_i}{T_i} \right\rceil - 1$ *and* $\lambda_i = \sum_{j \le i} \left\lceil \frac{\lambda_i}{T_j} \right\rceil C_j$

Proof.

Let us consider a level-i busy period, which is defined as the maximum interval of time during which a processor runs tasks of higher or equal priorities than τ_i. The length λ_i of such a busy period is given by the following equation:

$$\lambda_i = \sum_{j \le i} \left\lceil \frac{\lambda_i}{T_j} \right\rceil C_j$$

Let Q_i be such that $Q_i T_i < \lambda_i \le (Q_i + 1) T_i$, that is: $Q_i + 1 = \left\lceil \frac{\lambda_i}{T_i} \right\rceil$.

Let us consider the width $w_{i,\,Q_i}$ of the given level-i busy period starting $Q_i T_i$ before the current activation of τ_i. It is determined through the following equation:

$$w_{i,\,Q_i} = \left\lceil \frac{\lambda_i}{T_i} \right\rceil C_i + \sum_{j < i} \left\lceil \frac{w_{i,\,Q_i}}{T_j} \right\rceil C_j$$

This recursive equation can be solved by successive iterations starting from $w_{i,\,Q_i} = 0$. It is easy to show that the series:

$$w_{i,\,Q_i}^{(k+1)} = \left\lceil \frac{\lambda_i}{T_i} \right\rceil C_i + \sum_{j < i} \left\lceil \frac{w_{i,\,Q_i}^{(k)}}{T_j} \right\rceil C_j \text{ where } w_{i,\,Q_i}^{(0)} = 0$$

is increasing and that it converges to λ_i.

Hence, we have the following remarkable equality: $w_{i,\,Q_i} = \lambda_i$.

Let us now consider the sum of $w_{i,\,q}$ and $w_{i,\,Q_i}$. By definition, we have:

$$w_{i,\,q} + w_{i,\,Q_i} = (q + Q_i + 2) C_i + \sum_{j < i} \left(\left\lceil \frac{w_{i,\,q}}{T_j} \right\rceil + \left\lceil \frac{w_{i,\,Q_i}}{T_j} \right\rceil \right) C_j .$$

Given that $\forall (x, y) \in \mathbb{R} \times \mathbb{R}$, $\lceil x + y \rceil \le \lceil x \rceil + \lceil y \rceil$, we have:

$$(q + Q_i + 2) C_i + \sum_{j < i} \left\lceil \frac{w_{i,\,q} + w_{i,\,Q_i}}{T_j} \right\rceil C_j \le w_{i,\,q} + w_{i,\,Q_i} .$$

By definition, we have: $w_{i,\,q+Q_i+1} = (q + Q_i + 2) C_i + \sum_{j < i} \left\lceil \frac{w_{i,\,q+Q_i+1}}{T_j} \right\rceil C_j$.

This recursive equation can be solved by successive iterations starting from $w_{i,\,q+Q_i+1} = 0$. It is easy to show that the series:

$$w_{i,\,q+Q_i+1}^{(k+1)} = (q + Q_i + 2) C_i + \sum_{j < i} \left\lceil \frac{w_{i,\,q+Q_i+1}^{(k)}}{T_j} \right\rceil C_j \text{ where } w_{i,\,q+Q_i+1}^{(0)} = 0$$

is increasing and that it converges to $w_{i,\,q+Q_i+1}$.

For every w, such that: $0 \le w < w_{i,\, q+Q_i+1}$, we have: $(q + Q_i + 2)C_i + \sum_{j<i} \left\lceil \frac{w}{T_j} \right\rceil C_j > w$.

Therefore, we finally have: $w_{i,\,q} + w_{i,\,Q_i} \ge w_{i,\,q+Q_i+1}$, that is: $w_{i,\,q+Q_i+1} - w_{i,\,q} \le \lambda_i$. ❑

Corollary 2 - $\forall q \in \mathbb{N},\ r_{i,\,q+Q_i+1} \le r_{i,\,q}$ *where* $Q_i = \left\lceil \frac{\lambda_i}{T_i} \right\rceil - 1$ *and* $\lambda_i = \sum_{j \le i} \left\lceil \frac{\lambda_i}{T_j} \right\rceil C_j$

Proof.

Let us consider the difference of $r_{i,\,q+Q_i+1}$ and $r_{i,\,q}$. We have:

$$r_{i,\,q+Q_i+1} - r_{i,\,q} = w_{i,\,q+Q_i+1} - w_{i,\,q} - (Q_i + 1)T_i = w_{i,\,q+Q_i+1} - w_{i,\,q} - \left\lceil \frac{\lambda_i}{T_i} \right\rceil T_i .$$

From Lemma 8, page 45, we immediately have: $r_{i,\,q+Q_i+1} - r_{i,\,q} \le \lambda_i - \left\lceil \frac{\lambda_i}{T_i} \right\rceil T_i \le 0$. ❑

Corollary 2, page 46, enables us to restrict the set of points where the computation of r_i should be done for Fixed Priority based scheduling algorithms. We have:

$$\forall q,\ q \ge 0,\ r_{i,\,q+\left\lceil \frac{\lambda_i}{T_i} \right\rceil} \le r_{i,\,q}, \text{ where } r_{i,\,q} = w_{i,\,q} - qT_i$$

It follows that:

$$r_i = \mathrm{Max}_{q \ge 0}\{w_{i,\,q} - qT_i\} = \mathrm{Max}_{0 \le q \le \left\lceil \frac{\lambda_i}{T_i} \right\rceil - 1}\{w_{i,\,q} - qT_i\}$$

In a general manner, the computation of r_i should be done over the level i busy period λ_i.

2.2.2.2.2 The Earliest Deadline First scheduling algorithm

The necessary and sufficient feasibility test for the Earliest Deadline First scheduling algorithm in a « $\forall i \in [1, n],\ r_i \le D_i$ » form, is intractable at first glance. Indeed, the set of points where the predicate P should be checked is $\mathbb{R}$.

To obtain a tractable feasibility test, we have to restrict the set of points where the computation of r_i should be done. To achieve that purpose, we demonstrate the following lemma (Lemma 9, page 46) and corollary (Corollary 3, page 48).

Lemma 9 - $\forall a,\ a \ge 0,\ L_i(a + \lambda) - L_i(a) \le \lambda$ *where* $\lambda = \sum_j \left\lceil \frac{\lambda}{T_j} \right\rceil C_j$

Proof.

The length $L_i(a)$ is determined through the following equation:

$$L_i(a) = \left(1 + \left\lfloor \frac{a}{T_i} \right\rfloor\right)C_i + \sum_{j \ne i} \min\left\{\left\lceil \frac{L_i(a)}{T_j} \right\rceil, \max\left\{0, 1 + \left\lfloor \frac{a + D_i - D_j}{T_j} \right\rfloor\right\}\right\}C_j$$

This recursive equation can be solved by successive iterations starting from $L_i^{(0)}(a) = 0$. It is easy to show that the series:

$$L_i^{(k+1)}(a) = \left(1 + \left\lfloor \frac{a}{T_i} \right\rfloor\right) C_i + \sum_{j \neq i} \min\left\{ \left\lceil \frac{L_i^{(k)}(a)}{T_j} \right\rceil, \max\left\{0, 1 + \left\lfloor \frac{a + D_i - D_j}{T_j} \right\rfloor\right\}\right\} C_j$$

is increasing and that it converges to $L_i(a)$.

Let us demonstrate by induction that $\forall k \in \mathbb{N}$, $L_i^{(k)}(a+\lambda) - L_i^{(k)}(a) \leq \lambda$ where $\lambda = \sum_j \left\lceil \frac{\lambda}{T_j} \right\rceil C_j$.

By definition, we have $L_i^{(0)}(a+\lambda) = L_i^{(0)}(a) = 0$. Thus, $L_i^{(0)}(a+\lambda) - L_i^{(0)}(a) \leq \lambda$.

Let us assume that $L_i^{(k)}(a+\lambda) - L_i^{(k)}(a) \leq \lambda$ is true at rank k.

By definition, we have:

$$L_i^{(k+1)}(a+\lambda) = \left(1 + \left\lfloor \frac{a+\lambda}{T_i} \right\rfloor\right) C_i + \sum_{j \neq i} \min\left\{ \left\lceil \frac{L_i^{(k)}(a+\lambda)}{T_j} \right\rceil, \max\left\{0, 1 + \left\lfloor \frac{a+\lambda+D_i-D_j}{T_j} \right\rfloor\right\}\right\} C_j$$

We have:

- $L_i^{(k)}(a+\lambda) \leq L_i^{(k)}(a) + \lambda$ (by assumption);
- $1 + \left\lfloor \frac{a+\lambda}{T_i} \right\rfloor \leq 1 + \left\lfloor \frac{a}{T_i} \right\rfloor + \left\lceil \frac{\lambda}{T_i} \right\rceil$;
- $\max\left\{0, 1 + \left\lfloor \frac{a+\lambda+D_i-D_j}{T_j} \right\rfloor\right\} \leq \max\left\{0, 1 + \left\lfloor \frac{a+D_i-D_j}{T_j} \right\rfloor\right\} + \left\lceil \frac{\lambda}{T_j} \right\rceil$.

It follows that:

$$L_i^{(k+1)}(a+\lambda) \leq \left(1 + \left\lfloor \frac{a}{T_i} \right\rfloor\right) C_i + \left\lceil \frac{\lambda}{T_i} \right\rceil C_i + \sum_{j \neq i} \min\left\{ \left\lceil \frac{L_i^{(k)}(a)+\lambda}{T_j} \right\rceil, \max\left\{0, 1 + \left\lfloor \frac{a+D_i-D_j}{T_j} \right\rfloor\right\} + \left\lceil \frac{\lambda}{T_j} \right\rceil\right\} C_j$$

$$L_i^{(k+1)}(a+\lambda) \leq \left(1 + \left\lfloor \frac{a}{T_i} \right\rfloor\right) C_i + \left\lceil \frac{\lambda}{T_i} \right\rceil C_i + \sum_{j \neq i} \min\left\{ \left\lceil \frac{L_i^{(k)}(a)}{T_j} \right\rceil + \left\lceil \frac{\lambda}{T_j} \right\rceil, \max\left\{0, 1 + \left\lfloor \frac{a+D_i-D_j}{T_j} \right\rfloor\right\} + \left\lceil \frac{\lambda}{T_j} \right\rceil\right\} C_j$$

$$L_i^{(k+1)}(a+\lambda) \le \left(1+\left\lfloor \frac{a}{T_i} \right\rfloor\right)C_i + \left\lceil \frac{\lambda}{T_i} \right\rceil C_i + \sum_{j \neq i} \min\left\{ \left\lceil \frac{L_i^{(k)}(a)}{T_j} \right\rceil, \max\left\{0, 1+\left\lfloor \frac{a+D_i-D_j}{T_j} \right\rfloor\right\}\right\} C_j + \sum_{j \neq i} \left\lceil \frac{\lambda}{T_j} \right\rceil C_j$$

Hence,

$L_i^{(k+1)}(a+\lambda) - L_i^{(k+1)}(a) \le \lambda$.

By induction, we have:

$\forall k \in \mathbb{N}$, $L_i^{(k)}(a+\lambda) - L_i^{(k)}(a) \le \lambda$ where $\lambda = \sum_j \left\lceil \frac{\lambda}{T_j} \right\rceil C_j$.

Let $L_i(a) = \lim_{k \to +\infty} L_i^{(k)}(a)$ for every a. Finally, we have: $L_i(a+\lambda) - L_i(a) \le \lambda$. ❑

Corollary 3 - $\forall a, a \ge 0, r_i(a+\lambda) \le r_i(a)$ *where* $\lambda = \sum_j \left\lceil \frac{\lambda}{T_j} \right\rceil C_j$

Proof.

Let us consider the difference of $r_i(a+\lambda)$ and $r_i(a)$. We have:

$r_i(a+\lambda) - r_i(a) = L_i(a+\lambda) - L_i(a) - \lambda$.

From Lemma 9, page 46, we immediately have: $r_i(a+\lambda) - r_i(a) \le 0$. ❑

Corollary 3, page 48, enables us to restrict the set of points where the computation of r_i should be done for the Earliest Deadline First scheduling algorithm. We have:

$$\forall a, a \ge 0, r_i(a+\lambda) \le r_i(a), \text{ where } r_i(a) = L_i(a) - a$$

It follows that:

$$r_i = \text{Max}_{a \ge 0}\{L_i(a) - a\} = \text{Max}_{0 \le a < \lambda}\{L_i(a) - a\}$$

In a general manner, the computation of r_i should be done over the busy period λ.

For the Fixed Priorities policy, the worst-case response time r_i of a task i is based upon calculation of $w_{i,q}$, the width of the resulting level i busy period starting at time qT_i before the release of the current invocation of task i. For the Earliest Deadline First policy, the worst-case response time r_i of a task i is based upon calculation of $L_i(a)$, the length of the resulting busy period relative to the absolute deadline $a+D_i$ starting at time a before the current invocation of task i. The calculation of $w_{i,q}$ (resp. $L_i(a)$) is done by successive iterations starting from $w_{i,q} = 0$ (resp. $L_i(a) = 0$).

All these tractable feasibility tests based upon calculation of the worst-case response time of a task can be improved. The purpose of Section 2.2.3, page 49, is to study $w_{i,q}$ and $L_i(a)$ to obtain a faster convergence.

2.2.3 Improvements of the feasibility tests

The study of $w_{i,q}$ and $L_i(a)$ is done in Appendix B, page 127. We know that $w_{i,q}$ (resp. $L_i(a)$) increases as q (resp. a) increases.

Thus, a faster convergence can be obtained with $w_{i,q+1}^{(0)} = w_{i,q}$ (resp. $L_i(a_{l+1})^{(0)} = L_i(a_l)$).

In a general manner, to determine $w_{i,q}$, we consider the sequence $\{w_{i,q}^{(k)}\}_{k \geq 1}$ defined by the following recursive relation:

$$w_{i,q}^{(k+1)} = (q+1)C_i + \sum_{j<i} \left\lceil \frac{w_{i,q}^{(k)}}{T_j} \right\rceil C_j \text{ where } w_{i,q}^{(1)} = C_i + w_{i,q-1} \text{ and } w_{i,-1} = \sum_{j<i} C_j$$

As well, to determine $L_i(a)$, we consider the sequence $\{L_i(a_l)^{(k)}\}_{k \geq 0}$ defined by the following recursive relation:

$$L_i(a_l)^{(k+1)} = \left(1 + \left\lfloor \frac{a_l}{T_i} \right\rfloor\right) C_i + \sum_{j \neq i} \min\left\{ \left\lceil \frac{L_i(a_l)^{(k)}}{T_j} \right\rceil, \max\left\{0, 1 + \left\lfloor \frac{a_l + D_i - D_j}{T_j} \right\rfloor\right\}\right\} C_j$$

$$\text{where } L_i(a_l)^{(0)} = L_i(a_{l-1}) \text{ and } L_i(a_0) = 0$$

2.2.4 The feasibility tests in their optimized form

2.2.4.1 The optimized feasibility test for EDF in a « $\forall t \in S, P(t)$ » form

Theorem 21 - ***The feasibility test for the Earliest Deadline First scheduling algorithm in a « $\forall t \in S, P(t)$ » form may be revisited as follows:***

$$\forall t \in S, \sum_{j=1}^{n} Max\left\{0, 1 + \left\lfloor \frac{t - D_j}{T_j} \right\rfloor\right\} C_j \leq t$$

where:

- $S = \bigcup_{j=1}^{n} S_j$, $S_j = \left\{ D_j + k_j T_j, 0 \leq k_j \leq \left\lceil \frac{Min\{\lambda, \mu\} - D_j}{T_j} \right\rceil - 1 \right\}$
- $\lambda = \sum_{j=1}^{n} \left\lceil \frac{\lambda}{T_j} \right\rceil C_j$ (λ *is the smallest positive root of this equation)*
- $\mu = Max\left\{ Max_{1 \leq j \leq n}\{D_j\}, \frac{\sum_{j=1}^{n} (T_j - D_j)\frac{C_j}{T_j}}{1 - \sum_{j=1}^{n} \frac{C_j}{T_j}} \right\}$

Proof.

From Corollary 1, page 44, we have the following relationships:
$\forall t \in [0, Min\{\lambda, \mu\}[\ , h(t) \leq t \Leftrightarrow \forall t \geq 0\ , h(t) \leq t$

Since $h(t)$ is discontinuous and since $h(t)/t$ decreases as t increases between two consecutive points of discontinuities, we only need to check if $h(t) \leq t$ holds on the set $S = \bigcup_{j=1}^{n} S_j$, $S_j = \{D_j + T_j \cdot \mathbb{N}\} \cap [0, Min\{\lambda, \mu\}[= \left\{ D_j + k_j T_j,\ 0 \leq k_j \leq \left\lceil \frac{Min\{\lambda, \mu\} - D_j}{T_j} \right\rceil - 1 \right\}$. ❑

The following equivalence relations hold.

$$\left(Max_{0 < t < Min\{\lambda, \mu\}} \left\{ \frac{h(t)}{t} \right\} \leq 1 \right) \Leftrightarrow \left(Max_{t > 0} \left\{ \frac{h(t)}{t} \right\} \leq 1 \right) \Leftrightarrow (N_{EDF}(\tau) \leq 1)$$

However, it is important to point out that:

- $Max_{t > 0} \left\{ \frac{h(t)}{t} \right\} = N_{EDF}(\tau)$ is clearly a norm,
- $Max_{0 < t < Min\{\lambda, \mu\}} \left\{ \frac{h(t)}{t} \right\}$ is not a norm since $Min\{\lambda, \mu\}$, the upper bound on t, depends on $\{C_j\}_{j \in [1, n]}$.

Consequently, the optimized feasibility test for EDF in a « $\forall t \in S, P(t)$ » form will not be referred to as the norm based feasibility test for EDF in the sequel. The norm based feasibility test for EDF is only $N_{EDF}(\tau) \leq 1$.

Let us now consider the following example. Three tasks τ_1, τ_2 and τ_3 are considered. We have: $\tau_1 = (C_1, T_1, D_1) = (2, 7, 5)$, $\tau_2 = (C_2, T_2, D_2) = (3, 11, 7)$ and $\tau_3 = (C_3, T_3, D_3) = (5, 13, 10)$.

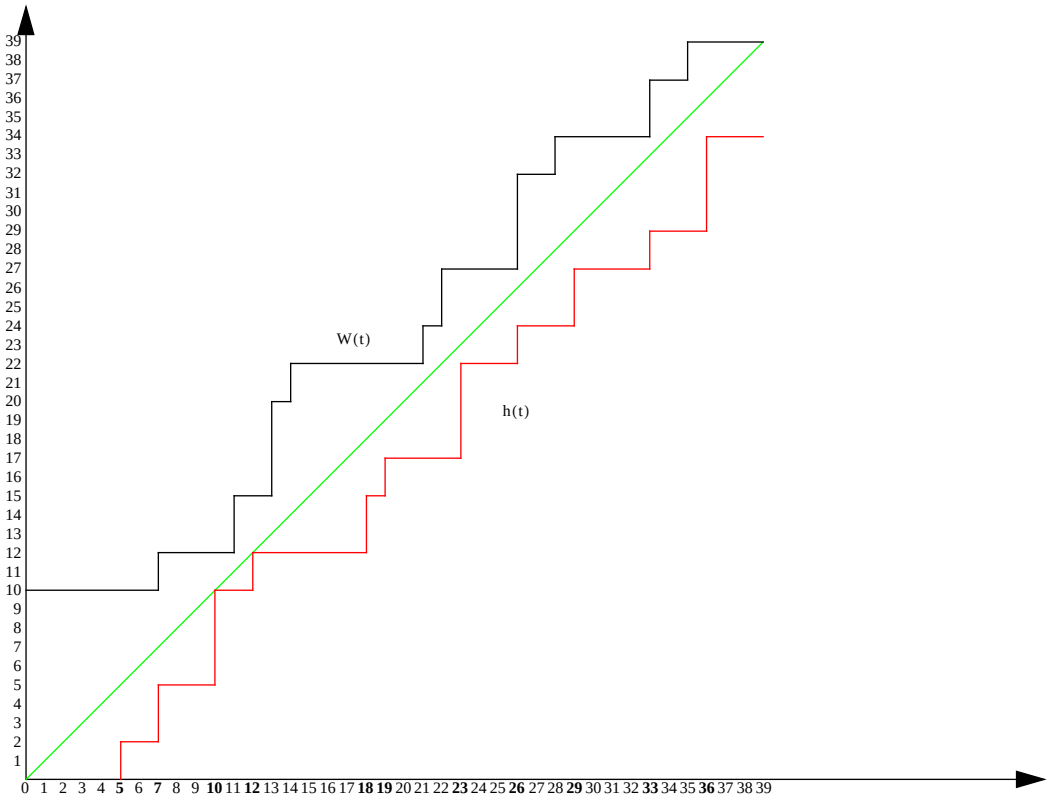


Figure 1: The W(t), t and h(t) functions over [0, λ[.

Task τ_1: $(C_1, T_1, D_1) = (2, 7, 5)$

$$S_1 = \left\{ D_1 + k_1 T_1, 0 \le k_1 \le \left\lceil \frac{\text{Min}\{\lambda, \mu\} - D_1}{T_1} \right\rceil - 1 \right\} = \{5, 12, 19, 26, 33\}$$

Task τ_2: $(C_2, T_2, D_2) = (3, 11, 7)$

$$S_2 = \left\{ D_2 + k_2 T_2, 0 \le k_2 \le \left\lceil \frac{\text{Min}\{\lambda, \mu\} - D_2}{T_2} \right\rceil - 1 \right\} = \{7, 18, 29\}$$

Task τ_3: $(C_3, T_3, D_3) = (5, 13, 10)$

$$S_3 = \left\{ D_3 + k_3 T_3, 0 \le k_3 \le \left\lceil \frac{\text{Min}\{\lambda, \mu\} - D_3}{T_3} \right\rceil - 1 \right\} = \{10, 23, 36\}$$

The following relation holds: $\forall t \in S, h(t) \le t$.

Consequently, the task set $\tau = \tau_1 + \tau_2 + \tau_3$ is EDF feasible.

2.2.4.2 The optimized feasibility tests in a « $\forall i \in [1, n], r_i \leq D_i$ » form

2.2.4.2.1 Fixed Priority based scheduling algorithms

Theorem 22 - ***The necessary and sufficient feasibility test for Fixed Priority based scheduling algorithms may be revisited as follows:***

$$\forall i \in [1, n],\ r_i \leq D_i,$$

$$r_i = Max_{0 \leq q \leq Q_i}\{w_{i,q} - qT_i\}$$

where:

- $Q_i = \left\lceil \frac{\lambda_i}{T_i} \right\rceil - 1$ *is the smallest value of q that satisfies* $w_{i,q} \leq (q+1)T_i$,
- $\lambda_i = \sum_{j \leq i} \left\lceil \frac{\lambda_i}{T_j} \right\rceil C_j$ *is the limit* $\lambda_i = \lim_{k \to +\infty} \lambda_i^{(k)}$ *of the series:*

$$\lambda_i^{(k+1)} = \sum_{j \leq i} \left\lceil \frac{\lambda_i^{(k)}}{T_j} \right\rceil C_j,\ \lambda_i^{(0)} = 1,$$

- $w_{i,q} = (q+1)C_i + \sum_{j < i} \left\lceil \frac{w_{i,q}}{T_j} \right\rceil C_j$ *is the limit* $w_{i,q} = \lim_{k \to +\infty} w_{i,q}^{(k)}$ *of the series:*

$$w_{i,q}^{(k+1)} = (q+1)C_i + \sum_{j < i} \left\lceil \frac{w_{i,q}^{(k)}}{T_j} \right\rceil C_j,\ w_{i,q}^{(1)} = C_i + w_{i,q-1},\ w_{i,-1} = \sum_{j < i} C_j.$$

Proof.

The worst-case response time r_i of a task τ_i, is given by: $r_i = Max_{q \geq 0}\{r_{i,q}\}$, $r_{i,q} = w_{i,q} - qT_i$.

From Corollary 2, page 46, we have: $\forall q \in \mathbb{N}$, $r_{i,q+Q_i+1} \leq r_{i,q}$ where:

$Q_i = \left\lceil \frac{\lambda_i}{T_i} \right\rceil - 1$ and $\lambda_i = \sum_{j \leq i} \left\lceil \frac{\lambda_i}{T_j} \right\rceil C_j$. Hence, $r_i = Max_{q \geq 0}\{r_{i,q}\} = Max_{0 \leq q \leq Q_i}\{r_{i,q}\}$.

For every q, $0 \leq q < Q_i$, we have: $\lambda_i > w_{i,q} > (q+1)T_i$. We also have:

$w_{i,Q_i} \leq (Q_i + 1)T_i \Leftrightarrow \lambda_i \leq \left\lceil \frac{\lambda_i}{T_i} \right\rceil T_i$.

- $Q_i = \left\lceil \frac{\lambda_i}{T_i} \right\rceil - 1$ is the smallest value of q that satisfies $w_{i,q} \leq (q+1)T_i$.

Appendix A, page 120, provides a practical way of computing λ_i:

- $\lambda_i = \sum_{j \le i} \left\lceil \frac{\lambda_i}{T_j} \right\rceil C_j$ is the limit $\lambda_i = \lim_{k \to +\infty} \lambda_i^{(k)}$ of the series:

$$\lambda_i^{(k+1)} = \sum_{j \le i} \left\lceil \frac{\lambda_i^{(k)}}{T_j} \right\rceil C_j, \ \lambda_i^{(0)} = 1.$$

Appendix B, page 127, provides a practical way of computing $w_{i,q}$:

- $w_{i,q} = (q+1)C_i + \sum_{j < i} \left\lceil \frac{w_{i,q}}{T_j} \right\rceil C_j$ is the limit $w_{i,q} = \lim_{k \to +\infty} w_{i,q}^{(k)}$ of the series:

$$w_{i,q}^{(k+1)} = (q+1)C_i + \sum_{j < i} \left\lceil \frac{w_{i,q}^{(k)}}{T_j} \right\rceil C_j, \ w_{i,q}^{(1)} = C_i + w_{i,q-1}, \ w_{i,-1} = \sum_{j < i} C_j. \quad \square$$

Let us now consider the following example. Three tasks τ_1, τ_2 and τ_3 are considered. We have: $\tau_1 = (C_1, T_1, D_1) = (2, 7, 5)$, $\tau_2 = (C_2, T_2, D_2) = (3, 11, 7)$ and $\tau_3 = (C_3, T_3, D_3) = (5, 13, 10)$. HPF/DM (Highest Priority First / Deadline Monotonic) is the scheduling algorithm.

Task τ_1: $(C_1, T_1, D_1) = (2, 7, 5)$

- $\lambda_1 = \left\lceil \frac{\lambda_1}{T_1} \right\rceil C_1 \Rightarrow \lambda_1 = 2$ and $Q_1 = \left\lceil \frac{\lambda_1}{T_1} \right\rceil - 1 \Rightarrow Q_1 = 0$,
- $r_1 = w_{1,0} = w_{1,Q_1} = \lambda_1 = 2$.

Task τ_2: $(C_2, T_2, D_2) = (3, 11, 7)$

- $\lambda_2 = \left\lceil \frac{\lambda_2}{T_1} \right\rceil C_1 + \left\lceil \frac{\lambda_2}{T_2} \right\rceil C_2 \Rightarrow \lambda_2 = 5$ and $Q_2 = \left\lceil \frac{\lambda_2}{T_2} \right\rceil - 1 \Rightarrow Q_2 = 0$,
- $r_2 = w_{2,0} = w_{2,Q_2} = \lambda_2 = 5$.

Task τ_3: $(C_3, T_3, D_3) = (5, 13, 10)$

- $\lambda_3 = \left\lceil \frac{\lambda_3}{T_1} \right\rceil C_1 + \left\lceil \frac{\lambda_3}{T_2} \right\rceil C_2 + \left\lceil \frac{\lambda_3}{T_3} \right\rceil C_3 \Rightarrow \lambda_3 = 39$ and $Q_3 = \left\lceil \frac{\lambda_3}{T_3} \right\rceil - 1 \Rightarrow Q_3 = 2$,
- $r_3 = \text{Max}\{w_{3,0}, w_{3,1} - 13, w_{3,2} - 26\} = 17$,
  - $w_{3,0} = C_3 + \left\lceil \frac{w_{3,0}}{T_1} \right\rceil C_1 + \left\lceil \frac{w_{3,0}}{T_2} \right\rceil C_2 \Rightarrow w_{3,0} = 17$,

- $w_{3,1} = 2C_3 + \left\lceil \frac{w_{3,1}}{T_1} \right\rceil C_1 + \left\lceil \frac{w_{3,1}}{T_2} \right\rceil C_2 \Rightarrow w_{3,1} = 27$,

- $w_{3,2} = w_{3,Q_3} = \lambda_3 = 39$.

The following relation does not hold: $\forall i \in [1, n], r_i \leq D_i$, since $r_3 > D_3$.

Consequently, the task set $\tau = \tau_1 + \tau_2 + \tau_3$ is not HPF/DM feasible. From Section 2.2.4.1, page 49, we know that τ is EDF feasible

2.2.4.2.2 The Earliest Deadline First scheduling algorithm

Theorem 23 - ***The necessary and sufficient feasibility test for the Earliest Deadline First algorithm may be revisited as follows:***

$$\forall i \in [1, n], r_i \leq D_i,$$

$$r_i = Max_{a \in A_i^{\dagger}}\{L_i(a) - a\}$$

where:

- $A_i^{\dagger} = \bigcup_j A_{i,j}^{\dagger}$, $A_{i,j}^{\dagger} = A_{i,j} \cap [0, \lambda - C_i]$, $A_{i,j} = \{D_j + k_j T_j - D_i,\ k_j \in \mathbb{N}\}$,

  $$A_{i,j}^{\dagger} = \left\{ D_j + k_j T_j - D_i,\ max\left\{0, \left\lceil \frac{D_i - D_j}{T_j} \right\rceil\right\} \leq k_j \leq \left\lfloor \frac{\lambda - C_i + D_i - D_j}{T_j} \right\rfloor \right\},$$

  $A_i^{\dagger} = \{a_1, a_2, \ldots, a_{|A_i^{\dagger}|}\}$ ***is ordered in a non-decreasing order;***

- $\lambda = \sum_j \left\lceil \frac{\lambda}{T_j} \right\rceil C_j$ ***is the limit*** $\lambda = \lim_{k \to +\infty} \lambda^{(k)}$ ***of the series:***

  $$\lambda^{(k+1)} = \sum_j \left\lceil \frac{\lambda^{(k)}}{T_j} \right\rceil C_j,\ \lambda^{(0)} = 1;$$

- $L_i(a_l) = \left(1 + \left\lfloor \frac{a_l}{T_i} \right\rfloor\right) C_i + \sum_{j \neq i} min\left\{ \left\lceil \frac{L_i(a_l)}{T_j} \right\rceil, max\left\{0, 1 + \left\lfloor \frac{a_l + D_i - D_j}{T_j} \right\rfloor\right\}\right\} C_j,$

  $a_l \in A_i^{\dagger}$, ***is the limit*** $L_i(a_l) = \lim_{k \to +\infty} L_i^{(k)}(a_l)$ ***of the series:***

  $$L_i^{(k+1)}(a_l) = \left(1 + \left\lfloor \frac{a_l}{T_i} \right\rfloor\right) C_i + \sum_{j \neq i} min\left\{ \left\lceil \frac{L_i^{(k)}(a_l)}{T_j} \right\rceil, max\left\{0, 1 + \left\lfloor \frac{a_l + D_i - D_j}{T_j} \right\rfloor\right\}\right\} C_j$$

  $L_i^{(0)}(a_l) = L_i(a_{l-1}),\ L_i(a_0) = 0.$

Proof.

The worst-case response time r_i of a task τ_i, is given by: $r_i = \mathrm{Max}_{a \geq 0}\{r_i(a)\}$, $r_i(a) = L_i(a) - a$.

From Corollary 3, page 48, we have: $\forall a,\ a \geq 0,\ r_i(a + \lambda) \leq r_i(a)$ where $\lambda = \sum_j \left\lceil \frac{\lambda}{T_j} \right\rceil C_j$.

It follows that: $r_i = \mathrm{Max}_{a \geq 0}\{r_i(a)\} = \mathrm{Max}_{0 \leq a < \lambda}\{r_i(a)\}$.

For every a, $0 \leq a < \lambda$, we have: $1 + \left\lfloor \frac{a}{T_i} \right\rfloor \leq \left\lceil \frac{\lambda}{T_i} \right\rceil$. It follows that:

$L_i^{(k+1)}(a) \leq \left\lceil \frac{\lambda}{T_i} \right\rceil C_i + \sum_{j \neq i} \left\lceil \frac{L_i^{(k)}(a)}{T_j} \right\rceil C_j$ where $L_i^{(0)}(a) = 0$.

Consequently, $L_i(a) = \lim_{k \to +\infty} L_i^{(k)}(a)$ is bounded by: $\lambda = \sum_j \left\lceil \frac{\lambda}{T_j} \right\rceil C_j$.

For every a, $\lambda - C_i < a < \lambda$, we have: $r_i(a) < L_i(a) - \lambda + C_i \leq C_i$, that is: $r_i(a) < C_i$.

$r_i(0) = C_i + \sum_{j \neq i} \min\left\{ \left\lceil \frac{L_i(0)}{T_j} \right\rceil, \max\left\{0, 1 + \left\lfloor \frac{D_i - D_j}{T_j} \right\rfloor\right\}\right\} C_j$ being greater than or equal to C_i, we finally have: $r_i = \mathrm{Max}_{a \geq 0}\{r_i(a)\} = \mathrm{Max}_{0 \leq a < \lambda}\{r_i(a)\} = \mathrm{Max}_{0 \leq a \leq \lambda - C_i}\{r_i(a)\}$.

For notational convenience, we define the $L_i(a, t)$ function as:

$$\left(1 + \left\lfloor \frac{a}{T_i} \right\rfloor\right) C_i + \sum_{j \neq i} \min\left\{ \left\lceil \frac{t}{T_j} \right\rceil, \max\left\{0, 1 + \left\lfloor \frac{a + D_i - D_j}{T_j} \right\rfloor\right\}\right\} C_j .$$

We have: $L_i(a) = \lim_{k \to +\infty} L_i^{(k)}(a) = L_i(a, L_i(a))$.

The $L_i(a, t)$ function is discontinuous (with respect to a). Since it remains constant between two consecutive points of discontinuities, we only need to compute $r_i(a)$ on the set $A_i^{\dagger}$:

- $A_i^{\dagger} = \bigcup_j A_{i,j}^{\dagger}$, $A_{i,j}^{\dagger} = A_{i,j} \cap [0, \lambda - C_i]$, $A_{i,j} = \{D_j + k_j T_j - D_i,\ k_j \in \mathbb{N}\}$,

  $A_{i,j}^{\dagger} = \left\{ D_j + k_j T_j - D_i, \max\left\{0, \left\lceil \frac{D_i - D_j}{T_j} \right\rceil\right\} \leq k_j \leq \left\lfloor \frac{\lambda - C_i + D_i - D_j}{T_j} \right\rfloor \right\}$,

  $A_i^{\dagger} = \{a_1, a_2, \ldots, a_{|A_i^{\dagger}|}\}$ is ordered in a non-decreasing order.

Appendix A, page 120, provides a practical way of computing λ:

- $\lambda = \sum_j \left\lceil \frac{\lambda}{T_j} \right\rceil C_j$ is the limit $\lambda = \lim_{k \to +\infty} \lambda^{(k)}$ of the series:

$$\lambda^{(k+1)} = \sum_j \left\lceil \frac{\lambda^{(k)}}{T_j} \right\rceil C_j, \lambda^{(0)} = 1.$$

Appendix B, page 127, provides a practical way of computing $L_i(a)$:

- $L_i(a_l) = \left(1 + \left\lfloor \frac{a_l}{T_i} \right\rfloor\right) C_i + \sum_{j \neq i} \min\left\{ \left\lceil \frac{L_i(a_l)}{T_j} \right\rceil, \max\left\{0, 1 + \left\lfloor \frac{a_l + D_i - D_j}{T_j} \right\rfloor\right\}\right\} C_j,$

$a_l \in A_i^{\dagger}$, is the limit $L_i(a_l) = \lim_{k \to +\infty} L_i^{(k)}(a_l)$ of the series:

$$L_i^{(k+1)}(a_l) = \left(1 + \left\lfloor \frac{a_l}{T_i} \right\rfloor\right) C_i + \sum_{j \neq i} \min\left\{ \left\lceil \frac{L_i^{(k)}(a_l)}{T_j} \right\rceil, \max\left\{0, 1 + \left\lfloor \frac{a_l + D_i - D_j}{T_j} \right\rfloor\right\}\right\} C_j$$

$L_i^{(0)}(a_l) = L_i(a_{l-1}), \quad L_i(a_0) = 0.$ ❑

Let us now consider the following example, which was introduced previously in Section 2.2.4.1, page 49. Three tasks τ_1, τ_2 and τ_3 are considered. We have:
$\tau_1 = (C_1, T_1, D_1) = (2, 7, 5)$, $\tau_2 = (C_2, T_2, D_2) = (3, 11, 7)$ and $\tau_3 = (C_3, T_3, D_3) = (5, 13, 10)$.

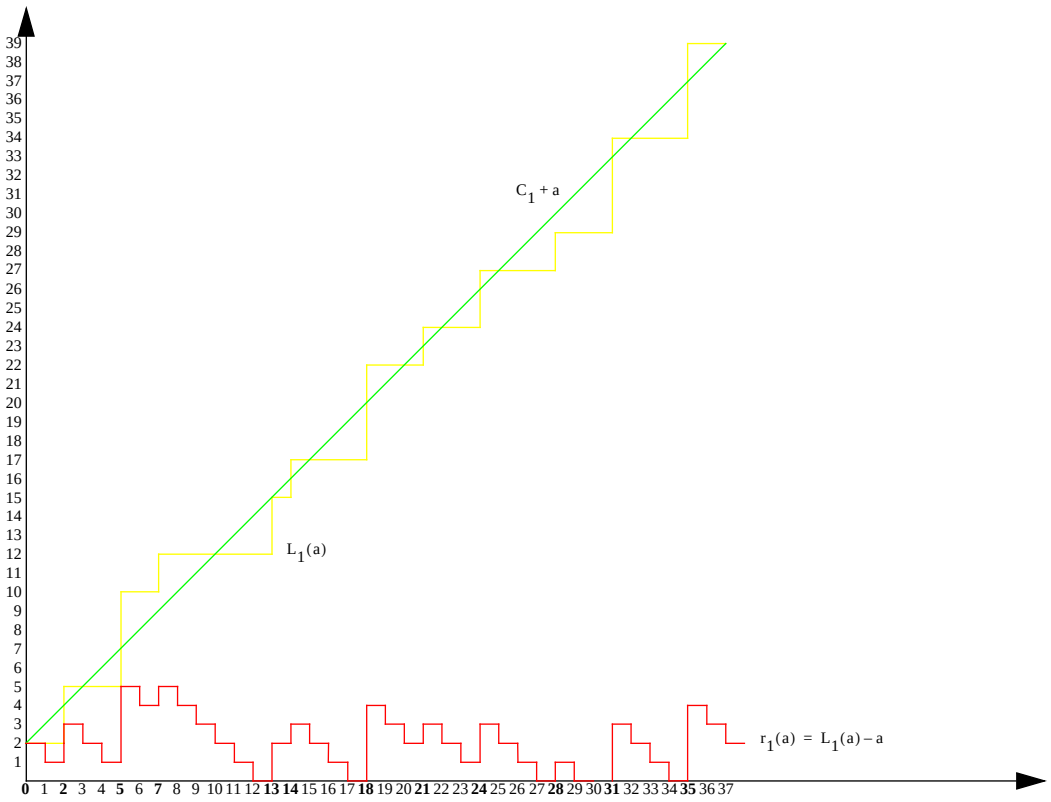


Figure 2: The L_1(floor(a)), C_1 + a and r_1(floor(a)) = L_1(floor(a)) - floor(a) functions over [0, λ - C_1].

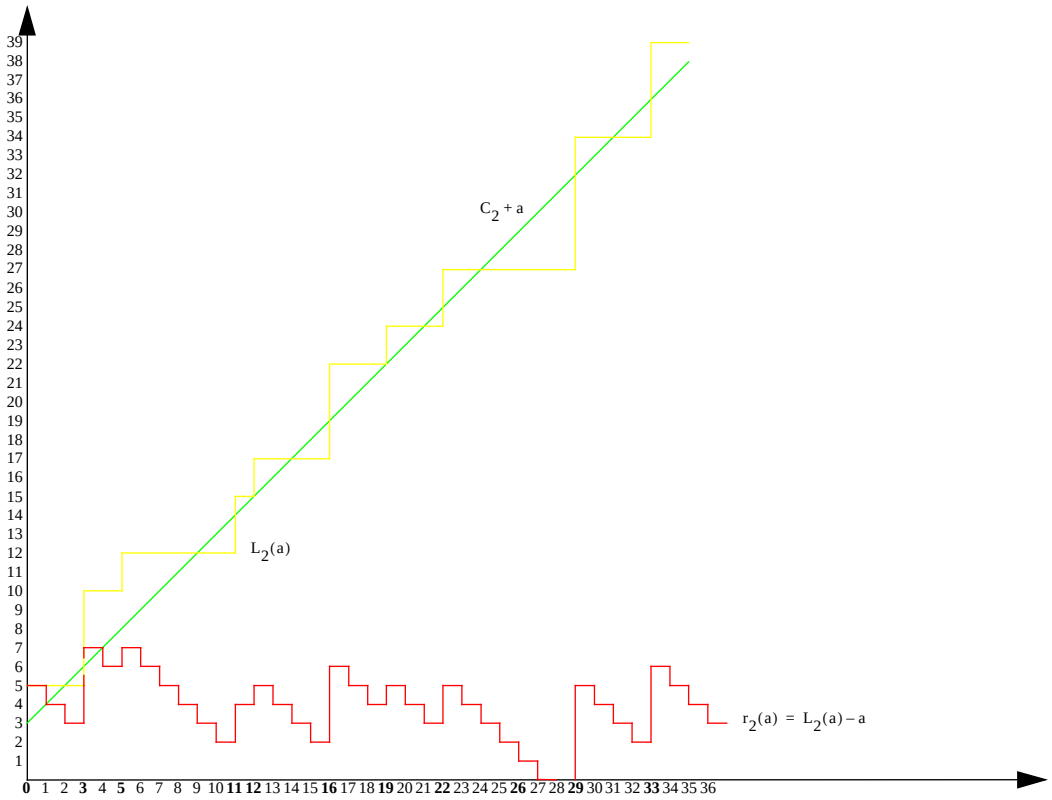


Figure 3: The $L_2(\text{floor}(a))$, $C_2 + a$ and $r_2(\text{floor}(a)) = L_2(\text{floor}(a)) - \text{floor}(a)$ functions over $[0, \lambda - C_2]$.

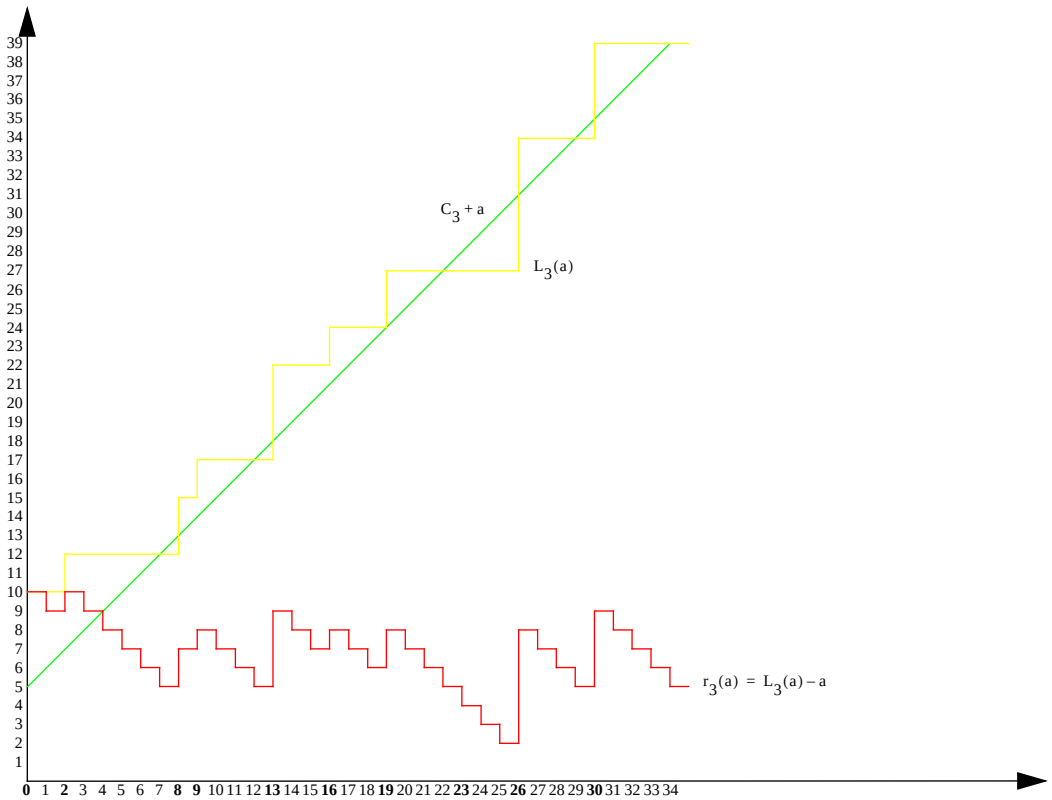


Figure 4: The $L_3(\text{floor}(a))$, $C_3 + a$ and $r_3(\text{floor}(a)) = L_3(\text{floor}(a)) - \text{floor}(a)$ functions over $[0, \lambda - C_3]$.

Task τ_1: $(C_1, T_1, D_1) = (2, 7, 5)$

$A_1^{\dagger} = \{0, 2, 5, 7, 13, 14, 18, 21, 24, 28, 31, 35\}$

$r_1 = Max_{a \in A_1^{\dagger}}\{L_1(a) - a\} = L_1(a) - a\big|_{a = 5} = 5$

Task τ_2: $(C_2, T_2, D_2) = (3, 11, 7)$

$A_2^{\dagger} = \{0, 3, 5, 11, 12, 16, 19, 22, 26, 29, 33\}$

$r_2 = Max_{a \in A_2^{\dagger}}\{L_2(a) - a\} = L_2(a) - a\big|_{a = 3} = 7$

Task τ_3: $(C_3, T_3, D_3) = (5, 13, 10)$

$A_3^{\dagger} = \{0, 2, 8, 9, 13, 16, 19, 23, 26, 30\}$

$r_3 = Max_{a \in A_3^{\dagger}}\{L_3(a) - a\} = L_3(a) - a\big|_{a = 0} = 10$

The following relation holds: $\forall i \in [1, n], r_i \leq D_i$. Consequently, the task set τ is EDF feasible.

2.2.5 Complexity comparison of the optimized feasibility tests

The foregoing necessary and sufficient feasibility tests are both tractable and optimized. They are expressed in:

- a « $\forall t \in S, P(t)$ » and « $\forall i \in [1, n], r_i \leq D_i$ » form for EDF,
- a « $\forall i \in [1, n], r_i \leq D_i$ » form for Fixed Priority based scheduling algorithms.

In Section 4, page 68, we will compare the complexity of these feasibility tests. To this end, we will first establish upper and lower bounds on the complexity of these feasibility tests. Tight bounds will be given in terms of basic operations, such as addition, multiplication and elementary function. Then, with these tight bounds, we will be able to set the bounds of the following two ratios:

- the cost of the feasibility test for EDF in its « $\forall i \in [1, n], r_i \leq D_i$ » form over the cost of the feasibility test for Fixed Priority based scheduling algorithms in its « $\forall i \in [1, n], r_i \leq D_i$ » form,
- the cost of the feasibility test for Fixed Priority based scheduling algorithms in its « $\forall i \in [1, n], r_i \leq D_i$ » form over the cost of the feasibility test for EDF in its « $\forall t \in S, P(t)$ » form.

These ratios will be denoted $\frac{C_{EDF, (\forall i \in [1, n], r_i \leq D_i)}}{C_{FP, (\forall i \in [1, n], r_i \leq D_i)}}$ and $\frac{C_{FP, (\forall i \in [1, n], r_i \leq D_i)}}{C_{EDF, (\forall t \in S, P(t))}}$ respectively.

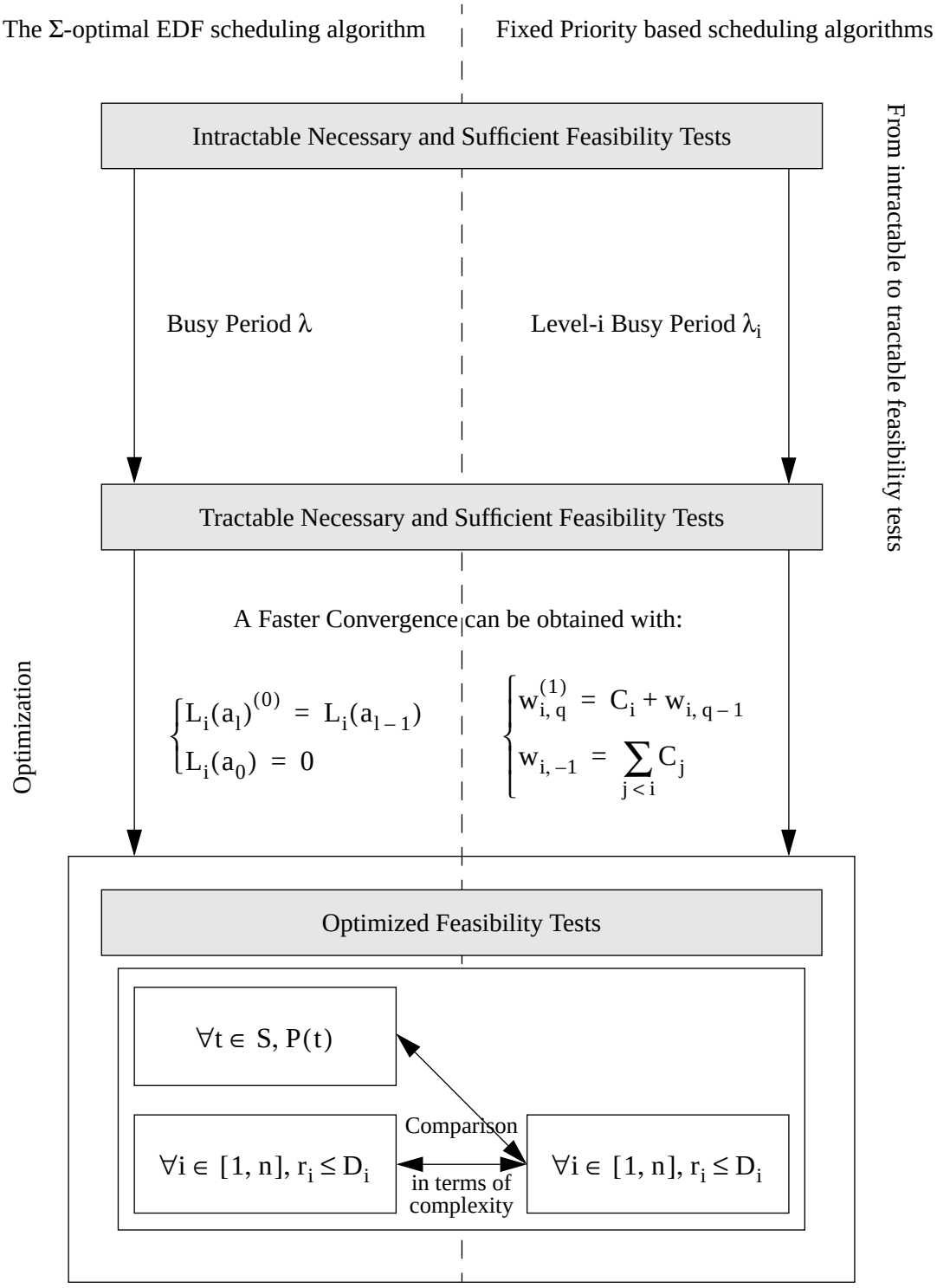
We will see that:

- $C_{EDF, (\forall i \in [1, n], r_i \leq D_i)} / C_{FP, (\forall i \in [1, n], r_i \leq D_i)}$ is pseudo-polynomial in $O(n)$,
- $C_{FP, (\forall i \in [1, n], r_i \leq D_i)} / C_{EDF, (\forall t \in S, P(t))}$ is pseudo-polynomial in $O(n^2)$.

This comparison analysis will be made in presence of particular task sets as well.

2.2.6 Summary

A general framework for the comparison of the Σ-optimal EDF scheduling algorithm and Fixed Priority based scheduling algorithms in terms of complexity



3 Efficiency of fixed and dynamic priority driven scheduling algorithms

In this section, we consider a periodic scheduling referential $\Sigma = (\Upsilon,\Pi)$, where Υ is a particular PNTSC characterized by a set of (T_i, D_i) and Π contains non-idling, preemptive versions of EDF and Fixed Priority scheduling algorithms. In such a scheduling referential we know, from Section 2.1, page 24, that N_{EDF} is the finest scheduling criterion and that EDF is Σ-optimal following that the efficiency of EDF, $\varepsilon(EDF) = \alpha_{N_{EDF}}(EDF) = 1$.

On the other hand, P being a given non-idling, preemptive Fixed Priority scheduling algorithm, Section 2.1.5, page 32, gives us an exact (but costly) algorithmic procedure to compute the efficiency of P $\varepsilon(P) = \alpha_{N_{EDF}}(P)$. The goal of this section is now to establish a general efficiency theorem that will enable us to compare P and EDF rapidly in presence of several PNTSC. More precisely, an efficiency theorem can be seen as a fast procedure based on a simple valid norm that leads to an underestimation of $\varepsilon(P)$. Indeed, finding a fast underestimation of $\varepsilon(P)$ based on N_{EDF} (the finest criterion) is an open question that we leave for a further study. However we know from Theorem 18, page 30, that given any scheduling algorithm $P \in \Pi$ and any valid norm N, $\alpha_N(P) \leq \alpha_{N_{EDF}}(P) = \varepsilon(P)$. Therefore, in this section, we will establish an efficiency theorem in two steps:

- first, we will use the simple processor load norm N_U to find an upper bound of the N_U-efficiency of EDF and then derive N'_U, a finer valid norm than N_U, for the study (cf. Section 3.1, page 60).
- second, we will find a lower bound of the N'_U-efficiency of P that is also a lower bound of the efficiency of P since $\alpha_{N'_U}(P) \leq \varepsilon(P)$ (cf. Section 3.2, page 61).

This result will be applied in Section 3.3, page 63, to compare EDF and fixed priority scheduling algorithms in presence of general, as well as particular, task sets. In particular. it will be shown that $\varepsilon(P)$ has a lower bound that cannot be equal to zero and that the value of this lower bound is PNTSC dependent (i.e. vary according to the authorized relations between deadlines and periods).

Let us note that we consider the efficiency theorem proposed in this section as a straight, but limited, application of our efficiency framework (introduced in Section 2.1, page 24). In order to apply this efficiency framework with more refinements for further study, we will derive, in Section 3.4, page 66, a fairly general method based on any valid norm to compute a lower bound of a particular scheduling algorithm, whenever the use of a finest scheduling criterion is not feasible.

3.1 Upper bound on the N_U-efficiency of EDF

The necessary and sufficient test for any $\tau \in \Upsilon$ with EDF is given by eq. (2), page 15. From this NSC, the following Necessary Condition can easily be derived:

$$\forall t \qquad t \geq \sum_{i=1}^{n} \mathrm{Max}\left\{0, \left\lfloor 1 + \frac{t - \mathrm{Max}(D_i)}{T_i} \right\rfloor\right\} C_i . \tag{11}$$

With $t = \mathrm{Max}(D_i)$, the following necessary test is derived:

$$\mathrm{Max}(D_i) \geq \sum_{i \leq n} C_i \Rightarrow \frac{\mathrm{Max}(D_i)}{\mathrm{Min}(T_i)} \geq \frac{\sum_{i \leq n} C_i}{\mathrm{Min}(T_i)} \geq \sum_{i \leq n} \frac{C_i}{T_i} = U = N_U. \quad (12)$$

Thus, an upper bound of the N_U-efficiency of EDF is $\frac{\mathrm{Max}(D_i)}{\mathrm{Min}(T_i)}$ if $\mathrm{Max}(D_i) \leq \mathrm{Min}(T_i)$. In addition to that, by making t increase towards infinity, we find that the efficiency of EDF is also bounded by 1. This is of course not a surprise, since this property is true for every algorithm. So, an upper bound for the efficiency of EDF is 1 if $\mathrm{Max}(D_i) \geq \mathrm{Min}(T_i)$.

Moreover, it also means that when $\mathrm{Max}(D_i) \leq \mathrm{Min}(T_i)$, $N'_U = \frac{\mathrm{Min}(T_i)}{\mathrm{Max}(D_i)} N_U$ is a valid norm since when $N_{EDF}(\tau) \leq 1$, then the necessary condition established in eq. (12), page 61, leads to $N_{U'}(\tau) \leq 1$. On the other hand, when $\mathrm{Max}(D_i) \geq \mathrm{Min}(T_i)$, $N'_U = N_U$ is a valid norm which is not surprising. Finally, whatever $\mathrm{Max}(D_i)$ and $\mathrm{Min}(T_i)$, we have $N'_U = \frac{\mathrm{Max}(\mathrm{Max}(D_i), \mathrm{Min}(T_i))}{\mathrm{Max}(D_i)} N_U$ which is a valid norm.

This property will be needed further on to find a lower bound of the efficiency of a fixed priority scheduling algorithm.

3.2 Lower bound on the efficiency of any fixed priority scheduling algorithm

3.2.1 Preliminary results

The equation giving $w_{i,q}$ (eq. (6), page 20) is recursive and can be solved by iterations. It is easy to show from the equation, that the resulting series converges if and only if the processor utilization norm N_U is less than one, which we will suppose implicitly from now on. For a given q:

$$(q+1)C_i + \sum_{j<i} \frac{w_{i,q}}{T_j} C_j \leq w_{i,q} \leq (q+1)C_i + \sum_{j<i} \left(\frac{w_{i,q}}{T_j} + 1 \right) C_j \quad (13)$$

$$\text{ie: } \frac{(q+1)C_i}{1 - \sum_{j<i} \frac{C_j}{T_j}} - qT_i \leq w_{i,q} - qT_i \leq \frac{(q+1)C_i + \sum_{j<i} C_j}{1 - \sum_{j<i} \frac{C_j}{T_j}} - qT_i \quad (14)$$

$$\text{ie: } \frac{qT_i \left(\sum_{j \leq i} \frac{C_j}{T_j} - 1 \right) + C_i}{1 - \sum_{j<i} \frac{C_j}{T_j}} \leq w_{i,q} - qT_i < \frac{qT_i \left(\sum_{j \leq i} \frac{C_j}{T_j} - 1 \right) + \sum_{j \leq i} C_j}{1 - \sum_{j<i} \frac{C_j}{T_j}} \quad (15)$$

Since $r_i = Max(w_{i,q} - qT_i) \le D_i$, $\dfrac{C_i}{1 - \sum_{j<i} \frac{C_j}{T_j}} \le r_i$ and $qT_i\left(\sum_{j \le i} \frac{C_j}{T_j} - 1\right) < 0$, then:

$$r_i < \frac{\sum_{j \le i} C_j}{1 - \sum_{j<i} \frac{C_j}{T_j}} \quad \text{and so:} \frac{C_i}{1 - \sum_{j<i} \frac{C_j}{T_j}} \le r_i < \frac{\sum_{j \le i} C_j}{1 - \sum_{j<i} \frac{C_j}{T_j}}$$

Theorem 24 - *Feasibility test for fixed priority scheduling algorithms*

The feasibility test for a periodic non-concrete task set and a fixed priorities scheduling algorithm can be written in the following manner:

$$\forall i \in \{1, \ldots, n\} f_i(C_1, \ldots, C_i, T_1, \ldots, T_{i-1}) \le D_i \tag{16}$$

with:
$$\frac{C_i}{1 - \sum_{j<i} \frac{C_j}{T_j}} \le f_i(C_1, \ldots, C_i, T_1, \ldots, T_{i-1}) < \frac{\sum_{j \le i} C_j}{1 - \sum_{j<i} \frac{C_j}{T_j}} \tag{17}$$

The lower bound of formula (17) can be reached if and only if:

$$\exists\left(k_1^i, \ldots, k_j^i, \ldots, k_{i-1}^i\right) \in N^{i-1} \qquad \forall j \in \{1, \ldots, i-1\} \qquad C_i = k_j^i T_j\left(1 - \sum_{k<i} \frac{C_k}{T_j}\right)$$

The upper bound is never reached but f_i can be as close as possible to it.

3.2.2 Efficiency theorem

Theorem 25 - *Efficiency of fixed priorities scheduling algorithms*

The efficiency of a fixed scheduling algorithm for each periodic non-concrete traffic class is greater than:

$$\frac{Max(Max(D_i), Min(T_i))}{Max(T_i) + Min(D_i)} \frac{Min(D_i)}{Max(D_i)}$$

Proof. Let r_{imin} and r_{imax} be respectively the lower bound and the upper bound in formula (17). A sufficient condition for $\forall i \quad r_{imax} \le D_i$ is that:

$$\frac{\mathrm{Max}(T_i)\sum_{j \le n}\frac{C_j}{T_j}}{1-\sum_{j<n}\frac{C_j}{T_j}} \le \mathrm{Min}(D_i) \tag{18}$$

It is therefore sufficient that: $N_U \le \frac{\mathrm{Min}(D_i)}{\mathrm{Max}(T_i)+\mathrm{Min}(D_i)}$, which is equivalent to: $N'_U \le \frac{\mathrm{Max}(\mathrm{Max}(D_i), \mathrm{Min}(T_i))}{\mathrm{Max}(T_i)+\mathrm{Min}(D_i)}\frac{\mathrm{Min}(D_i)}{\mathrm{Max}(D_i)}$ since, from Section 3.1, page 60, we know that N'_U is a valid norm. The theorem is proved. ❑

3.3 Applications of the efficiency theorem

The efficiency theorem established in Theorem 25, page 62, gives us a lower bound on $\varepsilon(P)$, i.e., an approximation of the measure of the non-optimality of any Fixed Priority scheduling algorithms P in a scheduling referential that contains EDF. Before applying this efficiency theorem, let us highlight its limitations:

- first, any PNTSC is simplified according to the dispersion of the task parameters.

- second, considering an “optimal” fixed priority scheduling algorithm, the lower bound on $\varepsilon(P)$ will be pessimistic since our efficiency theorem is valid whatever P, a “good” or a “bad” fixed priority scheduling algorithm. This is not surprising since, from the state of the art, cf. eq. (6), page 20, the existing fixed priority feasibility condition that we used to establish the efficiency theorem is independent of any fixed priority scheduling algorithm.

In spite of these limitations, some preliminary results will now be derived interpreting the efficiency theorem in the following situations, characterized by a kind of PNTSC (cf. Section 5, page 101, for an illustration of these results on some numerical examples):

- homogeneous case: $\forall i$, D_i=D, T_i=T.

- Liu and Layland’s case: $\forall i, D_i = T_i$.

- $\{D_i\} \ll \{T_i\}$. All the deadlines are supposed to be dominated by all the periods.

- $\{D_i\} \gg \{T_i\}$. All the periods are supposed to be dominated by all the deadlines.

- general case (i.e. relative deadline and periods are not related).

let us recall that in all that cases, the efficiency of EDF, $\varepsilon(EDF) = 1$.

■ Homogeneous case: $\forall i$, D_i=D, T_i=T.

In that case, every particular PNTS is deadline periodic equivalent to a monotask PNTS (i.e., is characterized by a unique couple (T, D) and a global computation time $C = \sum C_i$). This is a good illustra-

tion of the power of our framework from which we can derive a very simple admission test for EDF and for fixed priority scheduling algorithms:

- $C \leq T$ when $D \geq T$ (i.e. $U \leq 1$),
- $C \leq D$ when $D < T$ (i.e. $U \leq \frac{D}{T}$),

The feasibility conditions are the same for fixed priority scheduling algorithms and EDF. In other words, for homogeneous PNTSCs, fixed priority scheduling algorithms are optimal and their efficiency is equal to one.

Note that a direct application of our efficiency theorem shows that $\varepsilon(P)$ is underestimated by $\frac{T}{T+D} = \frac{1}{1+\frac{D}{T}}$ when D is less than T and by $\frac{D}{T+D}$ otherwise. From that, we can deduce a characterization of the error made by the approximation induced by Theorem 25, page 62. For example, in the particular case where D=T, the relative error is 50%.

■ Liu and Layland's case: $\forall i,\ D_i = T_i$.

In that particular case, we know from Section 2.2, page 40, that $\alpha_{N_U}(P) = \alpha_{N'_U}(P) = \varepsilon(P)$. Moreover, according to Theorem 9, page 18, Liu and Layland proved that $U \leq n\left(2^{\frac{1}{n}} - 1\right)$ is a sufficient condition when the scheduling algorithm is rate-monotonic (RM) and that $U \leq 1$ is a necessary and sufficient condition for EDF. It also means that $\alpha_{N_U}(RM)$, the efficiency of RM, is underestimated by $n\left(2^{\frac{1}{n}} - 1\right)$.

As a comparison, our efficiency theorem shows that $\varepsilon(P)$ is underestimated by $\frac{Min(T_i)}{Max(T_i) + Min(T_i)} = \frac{1}{\delta T + 1}$, where δT represents the dispersion of the periods in the PNTSC (note that in that Liu and Layland's case $\delta D = \delta T$).

If we consider that n, the number of periodic tasks in the PNTSC, can also be interpreted as a kind of measure of the dispersion, it is remarkable to see that both theorems give an underestimation which is decreasing with the dispersion. In other words, P is less efficient when the dispersion grows.

In our precise case « $\forall i,\ D_i = T_i$ » and P = RM, our efficiency theorem is weaker than Theorem 9, page 18. However, it is much more general since it is valid whatever P and whatever the relations between deadlines and periods. For all that cases, it gives the indication that EDF and Fixed Priority scheduling algorithms are equivalent when the dispersion of the tasks is very low and EDF becomes more efficient when the dispersion grows.

■ $\{D_i\} \gg \{T_i\}$ case

In that particular case, we know from Section 2.2, page 40, that $\alpha_{N_U}(P) = \alpha_{N'_U}(P) = \varepsilon(P)$. Moreover, we find that, $\varepsilon(P)$, is underestimated by: $\frac{Min(D_i)}{Max(T_i) + Min(D_i)}$. This gives us the indication that the more $MAX(T_i)$ is small compared to $MIN(D_i)$, the more the efficiency of P is close to 1. This seems intuitively normal since when the deadlines are very big, the choice of a particular policy is not so important.

■ $\{D_i\} \ll \{T_i\}$ case

In that case, $\varepsilon(P)$ is underestimated by: $\frac{Min(T_i)}{Max(T_i) + Min(D_i)} \frac{Min(D_i)}{Max(D_i)}$. Furthermore, when $MIN(D_i)$ becomes negligible in comparison to $MAX(T_i)$, $\varepsilon(P)$ is underestimated by $\frac{Min(T_i)}{Max(T_i)} \frac{Min(D_i)}{Max(D_i)} = \frac{1}{\delta T \delta D}$ where δT (resp. δD) represents the dispersion of the periods (resp. of the relative deadlines) of the PNTSC.

This gives us the indication that, the closer $MAX(D_i)$ is to $MIN(T_i)$, the smaller $\varepsilon(P)$ is. Moreover, this is even more true when the dispersion (in periods or in deadlines) increases. In other words, any (even a "bad") fixed priority scheduling algorithm P might be closer to EDF when the deadlines are largely smaller than the periods and when the dispersion is small. This seems intuitively normal since in that case, the scheduling decisions are not very important. Note that:

- the indications given by the efficiency theorem in that case (where $\{D_i\} \ll \{T_i\}$)clearly depend on the use of the valid norm N'_U that is a better approximation of the finest criterion N_{EDF} than the valid norm N_U (in the other cases $N'_U = N_U$).

- considering the limitations of the efficiency theorem explained at the beginning of this section, we can only conjecture that the use of DM (Deadline Monotonic), an "optimal" fixed priority scheduling algorithm in that case (cf. Section 1.2.2.1, page 18) will anticipate the indications given by the efficiency theorem. In other words DM might be rapidly comparable to EDF when all the deadlines become smaller than all the periods and the dispersion is not too large.

■ General case (i.e. when the periods are not related to the deadlines)

In the general case, due to the simplification of any PNTSC according to the dispersion of the task parameters, there is no clear interpretation of the efficiency theorem. It only gives a lower bound of $\varepsilon(P)$ that is not equal to zero but might be pessimistic.

To conclude, this part gives us some preliminary new results in several situations to compare the efficiency of fixed versus dynamic priority driven scheduling algorithms (cf. Section 5, page 101, for an illustration of these results on some numerical examples). We are fully aware, however, that a lot of work remains to be done to hold more precisely the measure of the non-optimality of Fixed Priority scheduling algorithms P in a scheduling referential that contains EDF. To that end, let us see now how it might be possible to refine the analysis.

3.4 A general method to compare the efficiency of scheduling algorithms.

In Section 3.2, page 61, we underestimated $\varepsilon(P)$, the efficiency of any fixed priority algorithm P, in a scheduling referential where EDF is allowed and the use of the finest criterion (N_{EDF}) is rather hard. More precisely, starting from a more simple valid norm (N_U), we demonstrated that:

$$\frac{Max(Max(D_i), Min(T_i))}{Max(T_i) + Min(D_i)} \frac{Min(D_i)}{Max(D_i)} \leq \alpha_{N'_U}(P) \leq \alpha_{N_{EDF}}(P) = \varepsilon(P)$$

where the valid norm N'_U is a better approximation of the finest criterion N_{EDF} than the valid norm N_U. In Section 3.3, page 63, we derived from this efficiency theorem some preliminary results concerning the behavior of $\varepsilon(P)$ in presence of several kinds of PNTSC. On the other hand, we saw the limitations of this efficiency theorem:

- first to handle clearly the efficiency of general task sets since it simplifies any PNTSC according to the dispersion of the task parameters.

- second to compute a tight lower bound on $\varepsilon(P)$ when P is an “optimal” fixed priority scheduling algorithm, since it doesn’t enable us to distinguish between several fixed priority driven scheduling algorithms.

In other words, our preliminary results give us some preliminary indications to compare fixed versus dynamic priority driven scheduling algorithms in terms of efficiency but the lower bound that we obtain on $\alpha_{N'_U}(P)$ might be a strong worst case.

In order to refine the analysis, it is possible to establish a finer efficiency theorem than the one we established. It is important to notice that the resulting classification of scheduling algorithms will depend on the choice of a particular valid norm. Indeed, we can imagine that the more the valid norm is close to the finest criterion, the more exact the classification will be. We leave this conjecture and the question to find a finer efficiency theorem for a further study.

Let us just derive now, from this particular efficiency theorem, a rather general method in order to underestimate the efficiency of any given scheduling algorithm P in a given scheduling referential where an optimal algorithm P_{opt} is allowed and where the use of a finest criterion is rather hard. To that end, let us choose a more simple valid norm N. Therefore (cf. Figure 5):

- in a first step, we derive a necessary condition on N for a PNTSC to be P_{opt}-feasible. This condition can be written: $\forall \tau \in \Upsilon$, $N(\tau) \leq \alpha$ where $\alpha \geq \alpha_N(P_{opt})$ i.e. α is a (tighter as possible) upper bound of the N-efficiency of P_{opt}.

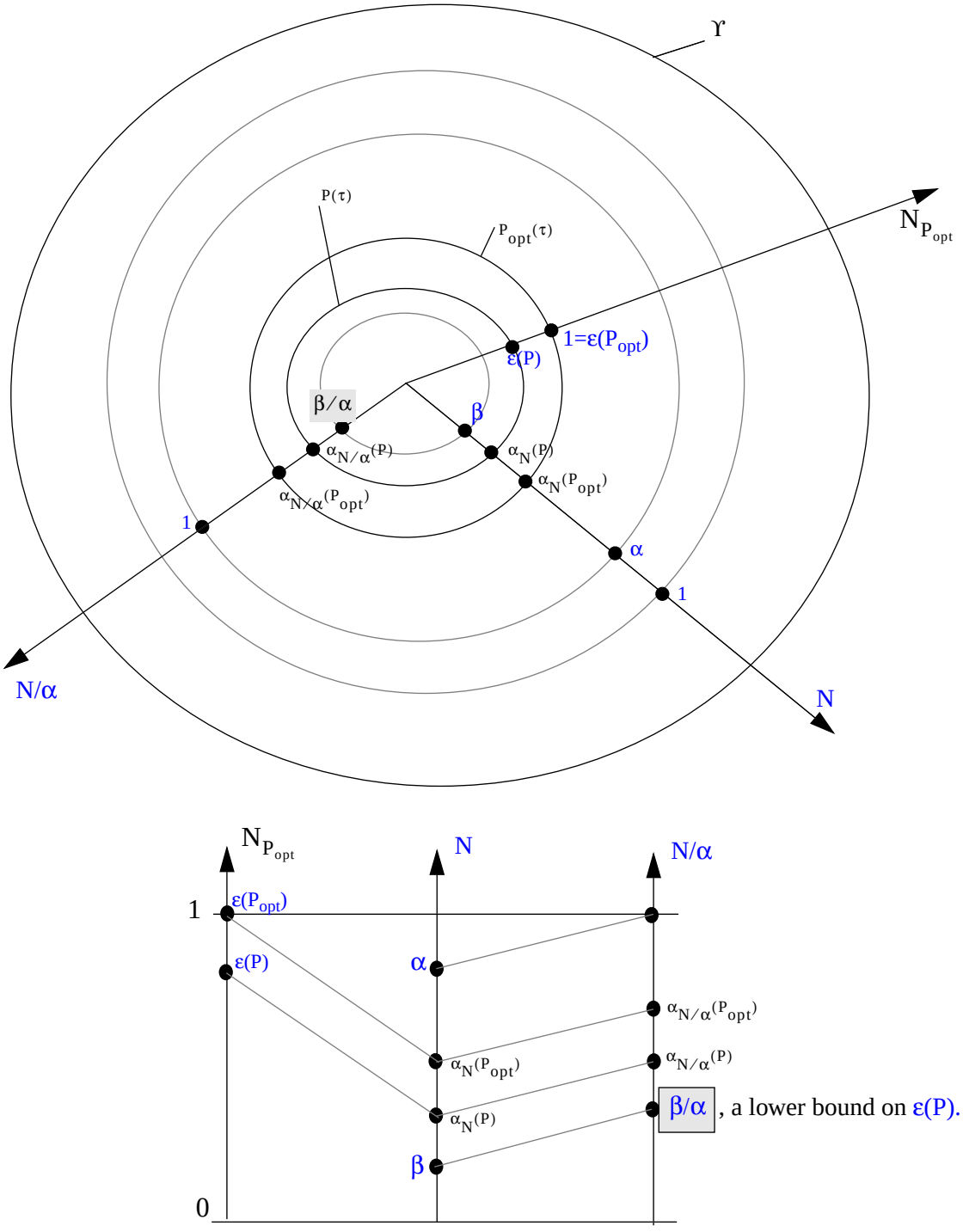
  N/α is still a valid norm since when τ is P_{opt}-feasible then the necessary condition $N(\tau) \leq \alpha$ holds. It should be used instead of N when $\alpha < 1$. Therefore, $N(\tau)/\alpha \leq 1$.

- in a second step, we derive a sufficient condition for a PNTSC to be P-feasible. This condition can be written: $\forall \tau \in \Upsilon$, $N(\tau) \leq \beta \leq \alpha_N(P)$, i.e. β is a (tighter as possible) lower bound of the N-efficiency of P.

  It follows that $\frac{N(\tau)}{\alpha} \leq \frac{\beta}{\alpha}$ is still a sufficient condition for the P-feasibility where N/α is the

valid norm established in the first step of the method that is a finer approximation of the finest criterion than N (since $\alpha \leq 1$).

Finally $\frac{\beta}{\alpha}$ is an underestimation of the efficiency of P that is better than β.



, a lower bound on ε(P).

Figure 5: General method to underestimate the efficiency of a partiular scheduling algorithm

4 Complexity of fixed and dynamic priority driven scheduling algorithms

The purpose of this section is to compare Fixed Priority based scheduling algorithms with the Earliest Deadline First scheduling algorithm in terms of complexity. To this end, we will use the general complexity framework introduced in Section 2.2, page 40. This framework will enable us to obtain:

- tractable feasibility tests, on the one hand,
- optimized feasibility tests, on the other hand.

In Section 2.1, page 24, we give a general definition of $\varepsilon(P)$, the efficiency of a scheduling algorithm P with respect to a scheduling referential Σ. In a general manner, we can define $\kappa(P)$, the complexity of the feasibility test for P, as follows:

$$\kappa(P) = \mathrm{Min}_{\mu \in M}\{\kappa_{\mu}(P)\}$$

where:

- $\kappa_{\mu}(P)$ denotes the complexity of the feasibility test for P, which is optimized according to μ, a particular method of optimization,
- M denotes the set of all the possible methods of optimization, which is infinite, a priori.

Here, we will use only $\kappa_{\mu^*}(P)$ to measure the complexity of the feasibility test for P, where μ^* is the optimization method introduced in the general complexity framework. This method is described in depth in Appendix B, page 127.

The complexity of the optimized feasibility tests will be given in terms of basic operations, such as addition, multiplication and elementary function. Elementary functions are functions such as “Min”, “Max”, “floor”, “ceil” or “less than or equal to”. In the sequel, we will assume that the cost of one addition (add), one multiplication (mult), one division (div) and one elementary function (function) is equal to ξ, the cost of one instruction cycle. Of course, such an assumption depends explicitly on the material architecture where the feasibility tests will be implemented. This assumption appears quite reasonable.

Unfortunately, we cannot directly compare the complexity of the optimized feasibility tests as they are stated in Section 2.2, page 40. The reasons are as follows:

- Theorem 21, page 49, exploits μ, which is used only for the feasibility test for EDF in a «$\forall t \in S, P(t)$» form,
- Theorem 22, page 52, exploits the notion of level-i busy period λ_i, which is used only for the time being for Fixed Priority based scheduling algorithms (cf. [GRS96] for possible uses of λ_i for deadline driven scheduling algorithms).

Therefore, in order to compare the complexity of the optimized feasibility tests, we will have to adjust them at a level of refinement where the comparison appears possible. To achieve that purpose, we will assume the following:

- Theorem 21, page 49:

$$S_j = \{D_j + T_j \cdot \mathbb{N}\} \cap [0, \lambda[= \left\{ D_j + k_j T_j, 0 \le k_j \le \left\lceil \frac{\lambda - D_j}{T_j} \right\rceil - 1 \right\},$$

- Theorem 22, page 52:
  $\lambda_i = \lambda$,

- Theorem 23, page 54:
  $A_i^{\dagger} = \bigcup_j A_{i,j}^{\dagger}$, $A_{i,j}^{\dagger} = A_{i,j} \cap [0, \lambda[$, $A_{i,j} = \{D_j + k_j T_j - D_i,\ k_j \in Z\}$,

$$A_{i,j}^{\dagger} = \left\{ D_j + k_j T_j - D_i, \left\lceil \frac{D_i - D_j}{T_j} \right\rceil \le k_j \le \left\lceil \frac{\lambda + D_i - D_j}{T_j} \right\rceil - 1 \right\},$$

Then, we will establish upper and lower bounds on the complexity of the feasibility tests. Tight bounds will be given:

- $\alpha' \le C_{EDF, (\forall t \in S, P(t))} \le \beta'$ where: $\alpha' = \Omega(n)$ and $\beta' = O(n^2)$,
- $\alpha \le C_{FP, (\forall i \in [1, n], r_i \le D_i)} \le \beta$ where: $\alpha = \Omega(n^2)$ and $\beta = O(n^3)$,
- $\gamma \le C_{EDF, (\forall i \in [1, n], r_i \le D_i)} \le \delta$ where: $\gamma = \Omega(n^3)$ and $\delta = O(n^3)$.

With these tight bounds, we will be able to set the bounds of the following two ratios:

- $\frac{\gamma}{\beta} \le \frac{C_{EDF, (\forall i \in [1, n], r_i \le D_i)}}{C_{FP, (\forall i \in [1, n], r_i \le D_i)}} \le \frac{\delta}{\alpha}$, where: $\frac{\gamma}{\beta} = \Omega(1)$ and $\frac{\delta}{\alpha} = O(n)$,
- $\frac{\alpha}{\beta'} \le \frac{C_{FP, (\forall i \in [1, n], r_i \le D_i)}}{C_{EDF, (\forall t \in S, P(t))}} \le \frac{\beta}{\alpha'}$, where: $\frac{\alpha}{\beta'} = \Omega(1)$ and $\frac{\beta}{\alpha'} = O(n^2)$.

We will see that:

- the ratio $\frac{C_{EDF, (\forall i \in [1, n], r_i \le D_i)}}{C_{FP, (\forall i \in [1, n], r_i \le D_i)}}$ is pseudo-polynomial in $O(n)$,
- the ratio $\frac{C_{FP, (\forall i \in [1, n], r_i \le D_i)}}{C_{EDF, (\forall t \in S, P(t))}}$ is pseudo-polynomial in:
  - $O(n)$, if $\lambda \ge \mathrm{Max}_j\{D_j\}$,
  - $O(n^2)$, otherwise.

4.1 Complexity analysis

4.1.1 The optimized feasibility test for EDF in a « $\forall t \in S, P(t)$ » form

The optimized feasibility test for EDF is given by Theorem 21, page 49.

A task set τ is EDF feasible if and only if $\forall t \in S,\ \sum_{j=1}^{n} \mathrm{Max}\left\{0, 1 + \left\lfloor \frac{t - D_j}{T_j} \right\rfloor\right\} C_j \leq t$.

Let add, mult and function denote respectively the cost of one addition, the cost of one multiplication and the cost of one elementary function such as “floor”, “max” or “less than or equal to”.

Let C denote the cost of the optimized feasibility test. We have:

C	The cost of the optimized feasibility test for EDF

$$test \leftarrow -\infty \quad ;$$

$$\forall t \in S, \quad test \leftarrow \mathrm{Max}\left\{test, \frac{1}{t}\sum_{j} \mathrm{Max}\left\{0, 1 + \left\lfloor \frac{t - D_j}{T_j} \right\rfloor\right\} C_j\right\} \quad ;$$

$$\text{is } test \leq 1 \ ?$$

$$C = |S| \cdot (C\_Computation + 1 \cdot function) + 1 \cdot function$$

where C_Computation denotes the cost of the computation of $\frac{1}{t}\sum_{j} \mathrm{Max}\left\{0, 1 + \left\lfloor \frac{t - D_j}{T_j} \right\rfloor\right\} C_j$.

By definition, we have:

$|S| = \sum_{j} |S_j|$; $|S_j| = \mathrm{Max}\left\{0, \left\lceil \frac{\mathrm{Min}\{\lambda, \mu\} - D_j}{T_j} \right\rceil\right\}$ (cf. Theorem 21, page 49).

It is easy to show that:

$C\_Computation = (3n - 1) \cdot add + (2n + 1) \cdot mult + 2n \cdot function$.

Finally, we have:

$$C = \sum_{j} \mathrm{Max}\left\{0, \left\lceil \frac{\mathrm{Min}\{\lambda, \mu\} - D_j}{T_j} \right\rceil\right\} \cdot ((3n - 1) \cdot add + (2n + 1) \cdot mult + (2n + 1) \cdot function) + 1 \cdot function$$

$$C \leq \sum_{j} \left\lceil \frac{\lambda}{T_j} \right\rceil \cdot ((3n - 1) \cdot add + (2n + 1) \cdot mult + (2n + 1) \cdot function) + 1 \cdot function$$

Assuming that $add = mult = function = \xi$, the cost of one instruction cycle, we have:

$$\boxed{C \leq \left((7n+1) \cdot \sum_j \left\lceil \frac{\lambda}{T_j} \right\rceil + 1\right) \cdot \xi}$$

From Appendix A, page 120, we have:

$\sum_j \left\lceil \frac{\lambda}{T_j} \right\rceil \leq n \cdot \left\lceil \frac{\rho}{1-\rho} \cdot \delta\{T\} \right\rceil$, where $\rho = \sum_j \frac{C_j}{T_j} = N_U(\tau)$ and $\delta\{T\} = \frac{Max_l\{T_l\}}{Min_l\{T_l\}}$.

Consequently, we have:

$$\boxed{C \leq \left((7n+1) \cdot n \cdot \left\lceil \frac{\rho}{1-\rho} \cdot \delta\{T\} \right\rceil + 1\right) \cdot \xi}$$

Therefore, C is pseudo-polynomial in $O(n^2)$ with a constant factor of approximately:
$7 \cdot \left\lceil \frac{\rho}{1-\rho} \cdot \delta\{T\} \right\rceil$, which is bounded by: $7 \cdot \left\lceil \frac{P}{1-P} \cdot \Delta\{T\} \right\rceil$ when $\rho \leq P < 1$; $\delta\{T\} \leq \Delta\{T\}$.

To be very rigorous, we must add to C the costs of the off-line computations of λ and μ since the interval of points where the predicate P should be checked is bounded by: $Min\{\lambda, \mu\}$.

These costs are as follows:

- the cost C_λ of the computation of λ is pseudo-polynomial in $O(n^2)$ with a constant factor of approximately: $4 \cdot \left\lceil \frac{\rho}{1-\rho} \cdot \delta\{T\} \right\rceil - 4$, which is bounded by: $4 \cdot \left\lceil \frac{P}{1-P} \cdot \Delta\{T\} \right\rceil - 4$ when $\rho \leq P < 1$ and $\delta\{T\} \leq \Delta\{T\}$ (cf. Appendix A, page 120) ;
- the cost C_μ of the computation of μ is polynomial in $O(n)$ with a constant factor of approximately: 7.

Finally, C remains pseudo-polynomial in $O(n^2)$ but with a constant factor of approximately:
$11 \cdot \left\lceil \frac{\rho}{1-\rho} \cdot \delta\{T\} \right\rceil - 4 \leq 11 \cdot \left\lceil \frac{P}{1-P} \cdot \Delta\{T\} \right\rceil - 4$ when $\rho \leq P < 1$ and $\delta\{T\} \leq \Delta\{T\}$.

4.1.2 The optimized feasibility tests in a « $\forall i \in [1, n], r_i \leq D_i$ » form

4.1.2.1 Fixed Priority based scheduling algorithms

From Theorem 22, page 52, the worst-case response time of a task τ_i is given by:

$$r_i = \mathrm{Max}_{0 \leq q \leq Q_i}\{w_{i,q} - qT_i\}$$

where $Q_i = \left\lceil \frac{\lambda_i}{T_i} \right\rceil - 1$ is the smallest value of q that satisfies $w_{i,q} \leq (q+1)T_i$.

The width $w_{i,q}$ of the given level-i busy period starting qT_i before the current activation of τ_i is determined through the following equation:

$$w_{i,q} = (q+1)C_i + \sum_{j<i} \left\lceil \frac{w_{i,q}}{T_j} \right\rceil C_j$$

For notational convenience, we define the $W_{i,q}(t)$ function as $(q+1)C_i + \sum_{j<i} \left\lceil \frac{t}{T_j} \right\rceil C_j$.

Appendix B, page 127, provides a practical way of solving the recursive equation $w_{i,q} = W_{i,q}(w_{i,q})$.
The series:

$$w_{i,q}^{(k+1)} = W_{i,q}(w_{i,q}^{(k)}) \text{ where } w_{i,q}^{(0)} = w_{i,q-1} \text{ and } w_{i,-1} = \sum_{j<i} C_j$$

is increasing and it converges to $w_{i,q}$.

The $W_{i,q}(t)$ function is discontinuous. If $N_i(0, \alpha)$ (resp. $N_i(\alpha, \beta)$) denotes the number of steps of $W_{i,q}(t)$ over $[0, \alpha[$ (resp. $[\alpha, \beta[$) then we have:

$$N_i(0, \alpha) = \sum_{j<i} \left\lceil \frac{\alpha}{T_j} \right\rceil \text{ and } N_i(\alpha, \beta) = N_i(0, \beta) - N_i(0, \alpha) = \sum_{j<i} \left\lceil \frac{\beta}{T_j} \right\rceil - \sum_{j<i} \left\lceil \frac{\alpha}{T_j} \right\rceil.$$

Thus, the number of successive iterations that is needed to converge to $w_{i,q} = \lim_{k \to +\infty} w_{i,q}^{(k)}$ starting from $w_{i,q}^{(1)} = W_{i,q}(w_{i,q}^{(0)}) = C_i + w_{i,q}^{(0)} = C_i + w_{i,q-1}$ is bounded by:

$$N_i(w_{i,q}^{(0)}, w_{i,q}) = N_i(w_{i,q-1}, w_{i,q}) = \sum_{j<i} \left\lceil \frac{w_{i,q}}{T_j} \right\rceil - \sum_{j<i} \left\lceil \frac{w_{i,q-1}}{T_j} \right\rceil.$$

Consequently, the number of successive iterations that is needed to determine $w_{i,q} = \lim_{k \to +\infty} w_{i,q}^{(k)}$ (i.e., to converge to $w_{i,q} = w_{i,q}^{(k)}$ and to verify that $w_{i,q}^{(k+1)} = W_{i,q}(w_{i,q}^{(k)}) = w_{i,q}^{(k)}$) is bounded by:

$$\sum_{j<i} \left\lceil \frac{w_{i,q}}{T_j} \right\rceil - \sum_{j<i} \left\lceil \frac{w_{i,q-1}}{T_j} \right\rceil + 1.$$

Let $C_{i,q}$ denote the cost we have to pay to solve $w_{i,q} = W_{i,q}(w_{i,q})$ by successive iterations starting from $w_{i,q}^{(1)}$. We have:

$C_{i,q} \le C_{i,q}^{(1)} + C_{i,q}^{(2)} + C_{i,q}^{(3)}$	The cost we have to pay to solve $w_{i,q} = W_{i,q}(w_{i,q})$ by successive iterations starting from $w_{i,q}^{(1)}$

$C_{i,q}^{(1)}$	The cost of the computation of $w_{i,q}^{(1)} = C_i + w_{i,q-1}$
$C_{i,q}^{(2)}$	The cost of $\sum_{j<i} \left\lceil \frac{w_{i,q}}{T_j} \right\rceil - \sum_{j<i} \left\lceil \frac{w_{i,q-1}}{T_j} \right\rceil + 1$ iterations
$C_{i,q}^{(3)}$	The cost of the test «is $w_{i,q} \le (q+1)T_i$?»

Note that $C_{i,q}^{(1)}$, the cost of the computation of $w_{i,q}^{(1)} = W_{i,q}(w_{i,q}^{(0)})$, is generally equal to the cost of one iteration. In our particular case, it is only equal to the cost of one single addition since $w_{i,q}^{(1)} = C_i + w_{i,q-1}$.

Let add, mult and function denote respectively the cost of one addition, the cost of one multiplication and the cost of one elementary function such as “ceil”, “max” or “less than or equal to”.

Let $C\_Computation_{i,q}$ denote the cost of the computation of $w_{i,q}^{(k+1)} = W_{i,q}(w_{i,q}^{(k)})$.

Let $C\_Comparison_{i,q}$ denote the cost of the test «is $w_{i,q}^{(k+1)} = w_{i,q}^{(k)}$?».

Let $C\_Iteration_{i,q}$ denote the cost of one iteration.

We have: $C\_Iteration_{i,q} = C\_Computation_{i,q} + C\_Comparison_{i,q}$ where:

$C\_Computation_{i,q} = i \cdot add + (2i-1) \cdot mult + (i-1) \cdot function$,

$C\_Comparison_{i,q} = 1 \cdot function$.

Hence, $C\_Iteration_{i,q} = i \cdot add + (2i-1) \cdot mult + i \cdot function$.

Therefore, we have:

$C_{i,q}^{(2)} = \left(\sum_{j<i} \left\lceil \frac{w_{i,q}}{T_j} \right\rceil - \sum_{j<i} \left\lceil \frac{w_{i,q-1}}{T_j} \right\rceil + 1 \right) \cdot C\_Iteration_{i,q}$, that is:

$C_{i,q}^{(2)} = \left(\sum_{j<i} \left\lceil \frac{w_{i,q}}{T_j} \right\rceil - \sum_{j<i} \left\lceil \frac{w_{i,q-1}}{T_j} \right\rceil + 1 \right) \cdot (i \cdot add + (2i-1) \cdot mult + i \cdot function)$.

$C_{i,q}^{(3)}$, the cost of the test «is $w_{i,q} \le (q+1)T_i$?» is equal to $1 \cdot add + 1 \cdot mult + 1 \cdot function$.

Finally, we have:

$$C_{i,q} \le 1 \cdot add + \left(\sum_{j<i} \left\lceil \frac{w_{i,q}}{T_j} \right\rceil - \sum_{j<i} \left\lceil \frac{w_{i,q-1}}{T_j} \right\rceil + 1 \right) \cdot (i \cdot add + (2i-1) \cdot mult + i \cdot function) + 1 \cdot add + 1 \cdot mult + 1 \cdot function$$

Let C_i denote the cost of the computation of $r_i = \mathrm{Max}_{0 \le q \le Q_i}\{w_{i,q} - qT_i\}$. We have:

C_i	The cost of the computation of $r_i = \mathrm{Max}_{0 \le q \le Q_i}\{w_{i,q} - qT_i\}$

$$r_i \leftarrow -\infty\ ;\quad \forall q,\ 0 \le q \le Q_i,\ r_i \leftarrow \mathrm{Max}\{r_i, w_{i,q} - qT_i\}$$

$$C_i \le \sum_{0 \le q \le Q_i} (C_{i,q} + 1 \cdot \mathrm{add} + 1 \cdot \mathrm{mult} + 1 \cdot \mathrm{function})$$

Let us recall that $Q_i + 1 = \left\lceil \frac{\lambda_i}{T_i} \right\rceil$ and $w_{i,Q_i} = \lambda_i$. (cf. Lemma 8, page 45). We have:

$$C_i \le \sum_{0 \le q \le Q_i} C_{i,q} + (Q_i + 1)(1 \cdot \mathrm{add} + 1 \cdot \mathrm{mult} + 1 \cdot \mathrm{function}), \text{ that is:}$$

$$C_i \le \sum_{0 \le q \le Q_i} \left(\sum_{j<i} \left\lceil \frac{w_{i,q}}{T_j} \right\rceil - \sum_{j<i} \left\lceil \frac{w_{i,q-1}}{T_j} \right\rceil + 1 \right) \cdot (i \cdot \mathrm{add} + (2i-1) \cdot \mathrm{mult} + i \cdot \mathrm{function}) +$$
$$(Q_i + 1)(2 \cdot \mathrm{add} + 1 \cdot \mathrm{mult} + 1 \cdot \mathrm{function}) + (Q_i + 1)(1 \cdot \mathrm{add} + 1 \cdot \mathrm{mult} + 1 \cdot \mathrm{function})$$

$$C_i \le \left(\sum_{j<i} \left\lceil \frac{w_{i,Q_i}}{T_j} \right\rceil - \sum_{j<i} \left\lceil \frac{w_{i,-1}}{T_j} \right\rceil + (Q_i + 1) \right) \cdot (i \cdot \mathrm{add} + (2i-1) \cdot \mathrm{mult} + i \cdot \mathrm{function}) +$$
$$(Q_i + 1)(3 \cdot \mathrm{add} + 2 \cdot \mathrm{mult} + 2 \cdot \mathrm{function})$$

Finally, we have:

$$C_i \le \sum_{j \le i} \left\lceil \frac{\lambda_i}{T_j} \right\rceil \cdot (i \cdot \mathrm{add} + (2i-1) \cdot \mathrm{mult} + i \cdot \mathrm{function}) + \left\lceil \frac{\lambda_i}{T_i} \right\rceil \cdot (3\mathrm{add} + 2\mathrm{mult} + 2\mathrm{function})$$

Let C denote the cost of the optimized feasibility test. We have:

C	The cost of the optimized feasibility test $\forall i \in [1, n],\ r_i \le D_i$

$$\mathrm{test} \leftarrow -\infty\ ;\ \forall i \in [1, n],\ \mathrm{test} \leftarrow \mathrm{Max}\{\mathrm{test}, r_i - D_i\}\ ;\ \text{is test} \le 0?$$

$$C \le \sum_{i=1}^{n} (C_i + 1 \cdot \mathrm{add} + 1 \cdot \mathrm{function}) + 1 \cdot \mathrm{function}$$

Finally, we have:

$$C \le \sum_i \sum_{j \le i} \left\lceil \frac{\lambda_i}{T_j} \right\rceil \cdot (i \cdot \mathrm{add} + (2i-1) \cdot \mathrm{mult} + i \cdot \mathrm{function}) +$$
$$\sum_i \left\lceil \frac{\lambda_i}{T_i} \right\rceil \cdot (3 \cdot \mathrm{add} + 2 \cdot \mathrm{mult} + 2 \cdot \mathrm{function}) + n \cdot \mathrm{add} + (n+1) \cdot \mathrm{function}$$

Assuming that $\mathrm{add} = \mathrm{mult} = \mathrm{function} = \xi$, the cost of one instruction cycle, we have:

$$C \leq \left(\sum_i \sum_{j \leq i} \left\lceil \frac{\lambda_i}{T_j} \right\rceil \cdot (4i - 1) + 7 \sum_i \left\lceil \frac{\lambda_i}{T_i} \right\rceil + (2n + 1) \right) \cdot \xi$$

Furthermore, we have: $\lambda_i \leq \lambda_n$ for every $i \in [1, n]$. For notational convenience, let $\lambda_n = \lambda$.

We have:

$$C \leq \left((2n^2 + n + 7) \cdot \sum_j \left\lceil \frac{\lambda}{T_j} \right\rceil + (2n + 1) \right) \cdot \xi$$

From Appendix A, page 120, we have:

$$\sum_j \left\lceil \frac{\lambda}{T_j} \right\rceil \leq n \cdot \left\lceil \frac{\rho}{1 - \rho} \cdot \delta\{T\} \right\rceil, \text{ where } \delta\{T\} = \frac{\mathrm{Max}_l\{T_l\}}{\mathrm{Min}_l\{T_l\}}.$$

Consequently, we have:

$$C \leq \left((2n^2 + n + 7) \cdot n \cdot \left\lceil \frac{\rho}{1 - \rho} \cdot \delta\{T\} \right\rceil + (2n + 1) \right) \cdot \xi$$

Therefore, C is pseudo-polynomial in $O(n^3)$ with a constant factor of approximately:
$2 \cdot \left\lceil \frac{\rho}{1 - \rho} \cdot \delta\{T\} \right\rceil$, which is bounded by: $2 \cdot \left\lceil \frac{P}{1 - P} \cdot \Delta\{T\} \right\rceil$ when $\rho \leq P < 1$; $\delta\{T\} \leq \Delta\{T\}$.

Contrary to Section 4.1.1, page 70, here we must not add to C the costs of the off-line computations of $\{\lambda_i\}_{i \in [1, n]}$. These costs have already been taken into account. Indeed, we know that:

- $Q_i = \left\lceil \frac{\lambda_i}{T_i} \right\rceil - 1$ is the smallest value of q that satisfies $w_{i, q} \leq (q + 1)T_i$,
- $w_{i, Q_i} = \lambda_i$.

These last two properties are obviously taken into account during the computation of r_i and its associated cost C_i.

4.1.2.2 The Earliest Deadline First scheduling algorithm

From Theorem 23, page 54, the worst-case response time of a task τ_i is given by:

$$r_i = \mathrm{Max}_{a \in A_i^{\dagger}}\{L_i(a) - a\}$$

where

$$A_i^{\dagger} = \bigcup_j A_{i,j}^{\dagger},\ A_{i,j}^{\dagger} = A_{i,j} \cap [0, \lambda - C_i],\ A_{i,j} = \{D_j + k_j T_j - D_i,\ k_j \in \mathbb{N}\},$$
$$\lambda = \sum_j \left\lceil \frac{\lambda}{T_j} \right\rceil C_j .$$

The set $A_i^{\dagger} = \{a_1, a_2, \ldots, a_{|A_i^{\dagger}|}\}$ is ordered in a non-decreasing order.

The length $L_i(a)$ of the resulting busy period relative to the absolute deadline $a + D_i$ starting at time a before the current activation of τ_i is determined through the following equation:

$$L_i(a) = \left(1 + \left\lfloor \frac{a}{T_i} \right\rfloor\right) C_i + \sum_{j \neq i} \min\left\{\left\lceil \frac{L_i(a)}{T_j} \right\rceil, \max\left\{0, 1 + \left\lfloor \frac{a + D_i - D_j}{T_j} \right\rfloor\right\}\right\} C_j$$

For notational convenience, we define the $L_i(a, t)$ function as:

$$\left(1 + \left\lfloor \frac{a}{T_i} \right\rfloor\right) C_i + \sum_{j \neq i} \min\left\{\left\lceil \frac{t}{T_j} \right\rceil, \max\left\{0, 1 + \left\lfloor \frac{a + D_i - D_j}{T_j} \right\rfloor\right\}\right\} C_j .$$

Appendix B, page 127, provides a practical way of solving the recursive equation:
$L_i(a_l) = L_i(a_l, L_i(a_l))$.

The series:

$$L_i^{(k+1)}(a_l) = L_i(a_l, L_i^{(k)}(a_l)) \text{ where } L_i^{(0)}(a_l) = L_i(a_{l-1}) \text{ and } L_i(a_0) = 0$$

is increasing and converges to $L_i(a_l)$.

The $L_i(a_l, t)$ function is discontinuous. If $N_i(0, \alpha)$ (resp. $N_i(\alpha, \beta)$) denotes the number of steps of $L_i(a_l, t)$ over $[0, \alpha[$ (resp. $[\alpha, \beta[$) then we have:

$$N_i(0, \alpha) = \sum_{j \neq i} \min\left\{\left\lceil \frac{\alpha}{T_j} \right\rceil, \max\left\{0, 1 + \left\lfloor \frac{a_l + D_i - D_j}{T_j} \right\rfloor\right\}\right\} \text{ and,}$$

$N_i(\alpha, \beta) = N_i(0, \beta) - N_i(0, \alpha)$, that is:

$$\sum_{j \neq i} \min\left\{\left\lceil \frac{\beta}{T_j} \right\rceil, \max\left\{0, 1 + \left\lfloor \frac{a_l + D_i - D_j}{T_j} \right\rfloor\right\}\right\} - \sum_{j \neq i} \min\left\{\left\lceil \frac{\alpha}{T_j} \right\rceil, \max\left\{0, 1 + \left\lfloor \frac{a_l + D_i - D_j}{T_j} \right\rfloor\right\}\right\}$$

Thus, the number of successive iterations that is needed to converge to $L_i(a_l) = \lim_{k \to +\infty} L_i^{(k)}(a_l)$ starting from $L_i^{(0)}(a_l) = L_i(a_{l-1})$ is bounded by:

$N_i(L_i^{(0)}(a_l), L_i(a_l)) = N_i(L_i(a_{l-1}), L_i(a_l))$, that is:

$$\sum_{j \neq i} \min\left\{\left\lceil \frac{L_i(a_l)}{T_j} \right\rceil, \max\left\{0, 1 + \left\lfloor \frac{a_l + D_i - D_j}{T_j} \right\rfloor\right\}\right\} - \sum_{j \neq i} \min\left\{\left\lceil \frac{L_i(a_{l-1})}{T_j} \right\rceil, \max\left\{0, 1 + \left\lfloor \frac{a_l + D_j}{T}\right.\right.\right.$$

Consequently, the number of successive iterations that is need to determine $L_i(a_l) = \lim_{k \to +\infty} L_i^{(k)}(a_l)$ (i.e., to converge to $L_i(a_l) = \lim_{k \to +\infty} L_i^{(k)}(a_l)$ and to verify that $L_i^{(k+1)}(a_l) = L_i(a_l, L_i^{(k)}(a_l)) = L_i^{(k)}(a_l)$) is bounded by: $N_i(L_i(a_{l-1}), L_i(a_l)) + 1$.

Let $C_i(a_l)$ denote the cost we have to pay to solve $L_i(a_l) = L_i(a_l, L_i(a_l))$ by successive iterations starting from $L_i^{(0)}(a_l)$. We have:

$C_i(a_l) \leq C_i^{(1)}(a_l) + C_i^{(2)}(a_l)$	The cost we have to pay to solve $L_i(a_l) = L_i(a_l, L_i(a_l))$ by successive iterations starting from $L_i^{(0)}(a_l)$

$C_i^{(1)}(a_l)$	The cost of the computation of $L_i^{(1)}(a_l) = L_i(a_l, L_i^{(0)}(a_l))$
$C_i^{(2)}(a_l)$	The cost of $N_i(L_i(a_{l-1}), L_i(a_l)) + 1$ iterations

$C_i^{(1)}(a_l)$, the cost of the computation of $L_i^{(1)}(a_l) = L_i(a_l, L_i^{(0)}(a_l))$ is equal to the cost of one iteration.

Let add, mult and function denote respectively the cost of one addition, the cost of one multiplication and the cost of one elementary function such as “floor”, “ceil”, “min”, “max” or “less than or equal to”.

Let $C\_Computation_i(a_l)$ denote the cost of the computation of $L_i^{(k+1)}(a_l) = L_i(a_l, L_i^{(k)}(a_l))$.

Let $C\_Comparison_i(a_l)$ denote the cost of the test «is $L_i^{(k+1)}(a_l) = L_i^{(k)}(a_l)$?».

Let $C\_Iteration_i = C\_Iteration_i(a_l)$ denote the cost of one iteration.

We have: $C\_Iteration_i(a_l) = C\_Computation_i(a_l) + C\_Comparison_i(a_l)$ where:

$C\_Computation_i(a_l) = (4n-3) \cdot add + (3n-1) \cdot mult + (4n-3) \cdot function$,

$C\_Comparison_i(a_l) = 1 \cdot function$.

Hence, $C\_Iteration_i(a_l) = (4n-3) \cdot add + (3n-1) \cdot mult + 2(2n-1) \cdot function$.

Therefore, we have:

$C_i^{(2)}(a_l) = (N_i(L_i(a_{l-1}), L_i(a_l)) + 1) \cdot C\_Iteration_i$.

Finally, we have:

$C_i(a_l) \leq (N_i(L_i(a_{l-1}), L_i(a_l)) + 2) \cdot C\_Iteration_i$.

Let C_i denote the cost of the computation of $r_i = Max_{a \in A_i^\dagger}\{L_i(a) - a\}$. We have:

C_i	The cost of the computation of $r_i = Max_{a \in A_i^\dagger}\{L_i(a) - a\}$

$r_i \leftarrow -\infty$; $\forall a \in A_i^\dagger$, $r_i \leftarrow Max\{r_i, L_i(a) - a\}$

$C_i \leq \sum_{a \in A_i^\dagger} (C_i(a_l) + 1 \cdot add + 1 \cdot function)$

We have:

$C_i \leq \sum_{a \in A_i^\dagger} (C_i(a_l) + 1 \cdot add + 1 \cdot function)$, that is:

$C_i \leq \sum_{a \in A_i^\dagger} N_i(L_i(a_{l-1}), L_i(a_l)) \cdot C\_Iteration_i + (2C\_Iteration_i + add + function) \cdot |A_i^\dagger|$

$$C_i \leq \sum_{a \in A_i^\dagger} \sum_{j \neq i} \min\left\{\left\lceil \frac{L_i(a_l)}{T_j} \right\rceil, \max\left\{0, 1 + \left\lfloor \frac{a_l + D_i - D_j}{T_j} \right\rfloor\right\}\right\} \cdot C\_Iteration_i -$$

$$\sum_{a \in A_i^\dagger} \sum_{j \neq i} \min\left\{\left\lceil \frac{L_i(a_{l-1})}{T_j} \right\rceil, \max\left\{0, 1 + \left\lfloor \frac{a_l + D_i - D_j}{T_j} \right\rfloor\right\}\right\} \cdot C\_Iteration_i +$$

$$(2 \cdot C\_Iteration_i + 1 \cdot add + 1 \cdot function) \cdot |A_i^\dagger|$$

$$C_i \leq \sum_{a \in A_i^\dagger} \sum_{j \neq i} \min\left\{\left\lceil \frac{L_i(a_l)}{T_j} \right\rceil, \max\left\{0, 1 + \left\lfloor \frac{a_l + D_i - D_j}{T_j} \right\rfloor\right\}\right\} \cdot C\_Iteration_i -$$

$$\sum_{a \in A_i^\dagger} \sum_{j \neq i} \min\left\{\left\lceil \frac{L_i(a_{l-1})}{T_j} \right\rceil, \max\left\{0, 1 + \left\lfloor \frac{a_{l-1} + D_i - D_j}{T_j} \right\rfloor\right\}\right\} \cdot C\_Iteration_i +$$

$$(2 \cdot C\_Iteration_i + 1 \cdot add + 1 \cdot function) \cdot |A_i^\dagger|$$

$$C_i \leq \sum_{j \neq i} \min\left\{\left\lceil \frac{L_i(a_{|A_i^\dagger|})}{T_j} \right\rceil, \max\left\{0, 1 + \left\lfloor \frac{a_{|A_i^\dagger|} + D_i - D_j}{T_j} \right\rfloor\right\}\right\} \cdot C\_Iteration_i +$$

$$(2 \cdot C\_Iteration_i + 1 \cdot add + 1 \cdot function) \cdot |A_i^\dagger|$$

$$C_i \le \sum_{j \neq i} \min\left\{\left\lceil \frac{\lambda}{T_j} \right\rceil, \max\left\{0, 1 + \left\lfloor \frac{\lambda - C_i + D_i - D_j}{T_j} \right\rfloor\right\}\right\} \cdot C\_Iteration_i +$$
$$(2 \cdot C\_Iteration_i + 1 \cdot add + 1 \cdot function) \cdot \left|A_i^{\dagger}\right|$$

Finally, we have:

$$C_i \le \sum_{j \neq i} \min\left\{\left\lceil \frac{\lambda}{T_j} \right\rceil, \max\left\{0, 1 + \left\lfloor \frac{\lambda - C_i + D_i - D_j}{T_j} \right\rfloor\right\}\right\} \cdot C\_Iteration_i +$$
$$2 \cdot \left|A_i^{\dagger}\right| \cdot C\_Iteration_i + \left|A_i^{\dagger}\right| \cdot (1 \cdot add + 1 \cdot function)$$

Let C denote the cost of the optimized feasibility test. We have:

C	The cost of the optimized feasibility test $\forall i \in [1, n],\ r_i \le D_i$
	$test \leftarrow -\infty$; $\forall i \in [1, n]$, $test \leftarrow Max\{test, r_i - D_i\}$; is $test \le 0$?
	$C \le \sum_{i=1}^{n} (C_i + 1 \cdot add + 1 \cdot function) + 1 \cdot function$

Finally, we have:

$$C \le \sum_i \sum_{j \neq i} \min\left\{\left\lceil \frac{\lambda}{T_j} \right\rceil, \max\left\{0, 1 + \left\lfloor \frac{\lambda - C_i + D_i - D_j}{T_j} \right\rfloor\right\}\right\} \cdot C\_Iteration_i +$$
$$2 \cdot \sum_i \left|A_i^{\dagger}\right| \cdot C\_Iteration_i + \sum_i \left|A_i^{\dagger}\right| \cdot (1 \cdot add + 1 \cdot function)$$

By definition, we have:

$$A_i^{\dagger} = \bigcup_j A_{i,j}{}^{\dagger},\ A_{i,j}{}^{\dagger} = A_{i,j} \cap [0, \lambda - C_i],\ A_{i,j} = \{D_j + k_j T_j - D_i,\ k_j \in \mathbb{N}\},$$

$$A_{i,j}{}^{\dagger} = \left\{D_j + k_j T_j - D_i, Max\left\{0, \left\lceil \frac{D_i - D_j}{T_j} \right\rceil\right\} \le k_j \le \left\lfloor \frac{\lambda - C_i + D_i - D_j}{T_j} \right\rfloor\right\}.$$

We have: $A_{i,j}{}^{\dagger} \subset (A_{i,j} \cap [0, \lambda[)$, where:

$$A_{i,j} \cap [0, \lambda[= \left\{D_j + k_j T_j - D_i, Max\left\{0, \left\lceil \frac{D_i - D_j}{T_j} \right\rceil\right\} \le k_j \le \left\lceil \frac{\lambda + D_i - D_j}{T_j} \right\rceil - 1\right\}.$$

Hence, $\left|A_{i,\,j}{}^{\dagger}\right| \le \mathrm{Max}\left\{0, \left\lceil \frac{\lambda + D_i - D_j}{T_j} \right\rceil - \mathrm{Max}\left\{0, \left\lceil \frac{D_i - D_j}{T_j} \right\rceil\right\}\right\} \le \left\lceil \frac{\lambda}{T_j} \right\rceil$ and $\left|A_i{}^{\dagger}\right| \le \sum_j \left\lceil \frac{\lambda}{T_j} \right\rceil$.

Finally, $\sum_i \left|A_i{}^{\dagger}\right| \le n \cdot \sum_j \left\lceil \frac{\lambda}{T_j} \right\rceil$.

We have:

$$C \le (3 \cdot (2n-1)^2 \cdot \mathrm{add} + (3n-1)^2 \cdot \mathrm{mult} + (12n^2 - 9n + 2) \cdot \mathrm{function}) \cdot \sum_j \left\lceil \frac{\lambda}{T_j} \right\rceil \cdot \xi$$

Assuming that $\mathrm{add} = \mathrm{mult} = \mathrm{function} = \xi$, the cost of one instruction cycle, we have:

$$C \le (33n^2 - 27n + 6) \cdot \sum_j \left\lceil \frac{\lambda}{T_j} \right\rceil \cdot \xi$$

From Appendix A, page 120, we have:

$\sum_j \left\lceil \frac{\lambda}{T_j} \right\rceil \le n \cdot \left\lceil \frac{\rho}{1-\rho} \cdot \delta\{T\} \right\rceil$, where $\delta\{T\} = \frac{\mathrm{Max}_l\{T_l\}}{\mathrm{Min}_l\{T_l\}}$.

Consequently, we have:

$$C \le (33n^2 - 27n + 6) \cdot n \cdot \left\lceil \frac{\rho}{1-\rho} \cdot \delta\{T\} \right\rceil \cdot \xi$$

Therefore, C is pseudo-polynomial in $O(n^3)$ with a constant factor of approximately:

$33 \cdot \left\lceil \frac{\rho}{1-\rho} \cdot \delta\{T\} \right\rceil$, which is bounded by: $33 \cdot \left\lceil \frac{P}{1-P} \cdot \Delta\{T\} \right\rceil$ when $\rho \le P < 1 ; \delta\{T\} \le \Delta\{T\}$.

To be very rigorous, we must add to C the cost C_λ of the off-line computation of λ since the interval of points where $r_i(a) = L_i(a) - a$ should be computed is bounded by: λ.

This cost C_λ is pseudo-polynomial in $O(n^2)$ with a constant factor of approximately:

$4 \cdot \left\lceil \frac{\rho}{1-\rho} \cdot \delta\{T\} \right\rceil - 4$, which is bounded by: $4 \cdot \left\lceil \frac{P}{1-P} \cdot \Delta\{T\} \right\rceil - 4$ when $\rho \le P < 1$ and $\delta\{T\} \le \Delta\{T\}$ (cf. Appendix A, page 120). It is therefore negligible beside $O(n^3)$.

Finally, C remains pseudo-polynomial in $O(n^3)$ with a constant factor of approximately:

$33 \cdot \left\lceil \frac{\rho}{1-\rho} \cdot \delta\{T\} \right\rceil$, which is bounded by: $33 \cdot \left\lceil \frac{P}{1-P} \cdot \Delta\{T\} \right\rceil$ when $\rho \le P < 1 ; \delta\{T\} \le \Delta\{T\}$.

4.2 Complexity comparison

4.2.1 The optimized feasibility test for EDF in a « $\forall \mathbf{t} \in \mathbf{S}, \mathbf{P}(\mathbf{t})$ » form

4.2.1.1 Upper bound on the cost of the optimized feasibility test

Let β' denote an upper bound on the cost of the optimized feasibility test for the Earliest Deadline First scheduling algorithm. From Section 4.1.1, page 70, we have:

$$\boxed{\beta' = O(n^2)}$$

β' is pseudo-polynomial with a constant factor of approximately: $11 \cdot \left\lceil \frac{\rho}{1-\rho} \cdot \delta\{T\} \right\rceil - 4$, which is bounded by: $11 \cdot \left\lceil \frac{P}{1-P} \cdot \Delta\{T\} \right\rceil - 4$ when $\rho \le P < 1$ and $\delta\{T\} \le \Delta\{T\}$.

4.2.1.2 Lower bound on the cost of the optimized feasibility test

Let α' denote a lower bound on the cost of the optimized feasibility test for the Earliest Deadline First scheduling algorithm. From Section 4.1.1, page 70, and without taking into account of μ, we have:

$$\boxed{\alpha' = (7n+1) \cdot \sum_j \mathrm{Max}\left\{0, \left\lceil \frac{\lambda - D_j}{T_j} \right\rceil\right\} \cdot \xi}$$

We can distinguish three cases:

■ Case I: $\lambda < \mathrm{Min}_j\{D_j\}$

In that case, we have: $\alpha' = 0$. We must add to α' the cost C_λ of the off-line computation of λ.

From Appendix A, page 120, we have: $\Omega(n) \le C_\lambda \le O(n^2)$. Hence, $\Omega(n) \le \alpha' \le O(n^2)$.

■ Case II: $\mathrm{Min}_j\{D_j\} \le \lambda < \mathrm{Max}_j\{D_j\}$

In that case, we have: $(7n+1) \cdot \xi \le \alpha' \le (7n+1) \cdot \sum_j \left\lceil \frac{\lambda}{T_j} \right\rceil \cdot \xi$, namely: $\Omega(n) \le \alpha' \le O(n^2)$.

We must add to α' the cost C_λ of the off-line computation of λ.

From Appendix A, page 120, we have: $\Omega(n) \le C_\lambda \le O(n^2)$. Hence, $\Omega(n) \le \alpha' \le O(n^2)$.

■ Case III: $\lambda \ge \mathrm{Max}_j\{D_j\}$

In that case, we have: $(7n+1) \cdot n \cdot \xi \le \alpha' \le (7n+1) \cdot \sum_j \left\lceil \frac{\lambda}{T_j} \right\rceil \cdot \xi$, namely: $\Omega(n^2) \le \alpha' \le O(n^2)$.

We must add to α' the cost C_λ of the off-line computation of λ.

From Appendix A, page 120, we have: $\Omega(n) \le C_\lambda \le O(n^2)$. Hence, $\Omega(n^2) \le \alpha' \le O(n^2)$.

4.2.1.3 The ratio β'/α'

From Section 4.2.1.1, page 81, and Section 4.2.1.2, page 81, it follows that:

- if $\lambda < \mathrm{Max}_j\{D_j\}$, then $\Omega(1) \leq \dfrac{\beta'}{\alpha'} \leq O(n)$. (cf. Case I and Case II),

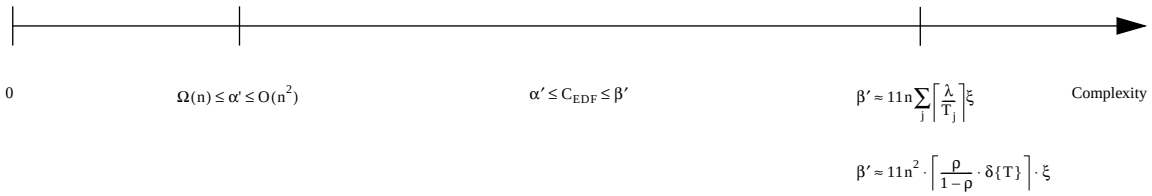


Figure 6: Complexity of the optimized feasibility test for EDF when $\lambda < \mathrm{Max}_j\{D_j\}$

- if $\lambda \geq \mathrm{Max}_j\{D_j\}$, then $\Omega(1) \leq \dfrac{\beta'}{\alpha'} \leq O(1)$. (cf. Case III).

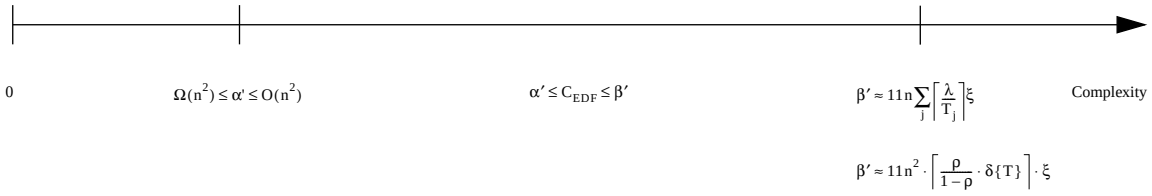


Figure 7: Complexity of the optimized feasibility test for EDF when $\lambda \geq \mathrm{Max}_j\{D_j\}$

4.2.2 The optimized feasibility tests in a « $\forall \mathbf{i} \in [\mathbf{1}, \mathbf{n}]$, $\mathbf{r_i} \leq \mathbf{D_i}$ » form

4.2.2.1 Fixed Priority based scheduling algorithms

In this section, we use the same notations as those introduced previously in Section 4.1.2.1, page 72.

4.2.2.1.1 Upper bound on the cost of the optimized feasibility test

Let β denote an upper bound on the cost of the optimized feasibility test for Fixed Priority based scheduling algorithms. From Section 4.1.2.1, page 72, we have:

$$\boxed{\beta = O(n^3)}$$

β is pseudo-polynomial with a constant factor of approximately: $2 \cdot \left\lceil \dfrac{\rho}{1-\rho} \cdot \delta\{T\} \right\rceil$, which is bounded by: $2 \cdot \left\lceil \dfrac{P}{1-P} \cdot \Delta\{T\} \right\rceil$ when $\rho \leq P < 1$ and $\delta\{T\} \leq \Delta\{T\}$.

4.2.2.1.2 Lower bound on the cost of the optimized feasibility test

Let $\underline{C}_{i,q}$ denote the lower bound on the cost we have to pay to solve $w_{i,q} = W_{i,q}(w_{i,q})$ by successive iterations starting from $w_{i,q}^{(1)}$. We have: $\underline{C}_{i,q} = 1 \cdot add + 1 \cdot C\_Iteration_{i,q}$.

Proof. The cost $C_{i,q}$ is minimum when $w_{i,q}^{(2)} = w_{i,q}^{(1)} = C_i + w_{i,q}^{(0)}$. ❑

We have:
$\underline{C}_{i,q} = (i + 1) \cdot add + (2i - 1) \cdot mult + i \cdot function$.

Let $\underline{C}_i$ denote the lower bound on the cost of the computation of $r_i = Max_{0 \le q \le Q_i'}\{w_{i,q} - qT_i\}$. We have:

$\underline{C}_i = \sum_{0 \le q \le Q_i'} (\underline{C}_{i,q} + 1 \cdot add + 1 \cdot mult + 1 \cdot function)$ where $Q_i' + 1 = \left\lceil \frac{\lambda}{T_i} \right\rceil$, that is:

$\underline{C}_i = ((i + 2) \cdot add + 2i \cdot mult + (i + 1) \cdot function) \cdot \left\lceil \frac{\lambda}{T_i} \right\rceil$.

Let $\underline{C}$ denote the lower bound on the cost of the optimized feasibility test. We have:

$\underline{C} = \sum_{i=1}^{n} (\underline{C}_i + 1 \cdot add + 1 \cdot function) + 1 \cdot function$, that is:

$\underline{C} = \sum_{i=1}^{n} \left(((i + 2) \cdot add + 2i \cdot mult + (i + 1) \cdot function) \cdot \left\lceil \frac{\lambda}{T_i} \right\rceil \right) + n \cdot add + (n + 1) \cdot function$.

Assuming that $add = mult = function = \xi$, the cost of one instruction cycle, we have:

$\underline{C} = \left(\sum_{i=1}^{n} \left((4i + 3) \cdot \left\lceil \frac{\lambda}{T_i} \right\rceil \right) + (2n + 1) \right) \cdot \xi$.

From Appendix A, page 120, we have: $\rho \cdot Min_l\{T_l\} \le \lambda$.

It follows that:

$\left(\sum_{i=1}^{n} (4i + 3) \cdot \left\lceil \frac{\rho}{\delta\{T\}} \right\rceil + (2n + 1) \right) \cdot \xi \le \underline{C}$ where $\delta\{T\} = \frac{Max_l\{T_l\}}{Min_l\{T_l\}}$.

Given that: $0 < \rho \le 1$ and $\delta\{T\} \ge 1$, we have: $\left\lceil \frac{\rho}{\delta\{T\}} \right\rceil = 1$.

Finally, we have: $(2n^2 + 7n + 1) \cdot \xi \le \underline{C}$.

Let α denote a lower bound on the cost of the optimized feasibility test for Fixed Priority based scheduling algorithms. We have:

$$\alpha = \Omega(n^2)$$

α is pseudo-polynomial with a constant factor of approximately: 2.

4.2.2.1.3 The ratio β/α

From Section 4.2.2.1.1, page 82, and Section 4.2.2.1.2, page 83, we immediately have:

$$\frac{\beta}{\alpha} = O(n)$$

$\frac{\beta}{\alpha}$ is pseudo-polynomial with a constant factor of approximately: $\left\lceil \frac{\rho}{1-\rho} \cdot \delta\{T\} \right\rceil$, which is bounded by: $\left\lceil \frac{P}{1-P} \cdot \Delta\{T\} \right\rceil$ when $\rho \leq P < 1$ and $\delta\{T\} \leq \Delta\{T\}$.

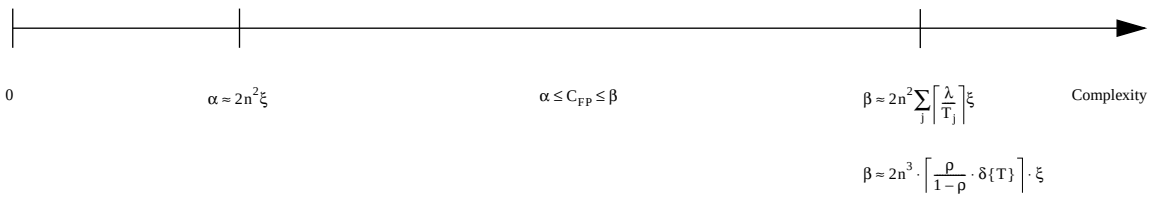


Figure 8: Complexity of the optimized feasibility test for Fixed Priority based scheduling algorithms

4.2.2.2 The Earliest Deadline First scheduling algorithm

In this section, we use the same notations as those introduced previously in Section 4.1.2.2, page 76.

4.2.2.2.1 Upper bound on the cost of the optimized feasibility test

Let δ denote an upper bound on the cost of the optimized feasibility test for the Earliest Deadline First scheduling algorithm. From Section 4.1.2.2, page 76, we have:

$$\delta = O\left(n^2 \cdot \sum_j \left\lceil \frac{\lambda}{T_j} \right\rceil\right)$$

δ is pseudo-polynomial with a constant factor of approximately: 33.

4.2.2.2.2 Lower bound on the cost of the optimized feasibility test

Let $\underline{C}_i(a_l)$ denote the lower bound on the cost we have to pay to solve $L_i(a_l) = L_i(a_l, L_i(a_l))$ by successive iterations starting from $L_i^{(0)}(a_l)$. We have: $\underline{C}_i(a_l) = C\_Iteration_i(a_l)$.

Proof. The cost $C_i(a_l)$ is minimum when $L_i^{(1)}(a_l) = L_i^{(0)}(a_l)$. ❑

We have: $\underline{C}_i(a_l) = (4n-3) \cdot add + (3n-1) \cdot mult + 2(2n-1) \cdot function$.

Let $\underline{C}_i$ denote the lower bound on the cost of the computation of $r_i = Max_{a \in A_i^\dagger}\{L_i(a) - a\}$.
We have:

$\underline{C}_i = \sum_{a \in A_i^\dagger} (C_i(a_l) + 1 \cdot add + 1 \cdot function)$, that is:

$\underline{C}_i = (2(2n-1) \cdot add + (3n-1) \cdot mult + (4n-1) \cdot function) \cdot \left|A_i^\dagger\right|$.

Let $\underline{C}$ denote the lower bound on the cost of the optimized feasibility test. We have:

$\underline{C} = \sum_{i=1}^{n} (\underline{C}_i + 1 \cdot add + 1 \cdot function) + 1 \cdot function$, that is:

$$\underline{C} = (2(2n-1) \cdot add + (3n-1) \cdot mult + (4n-1) \cdot function) \cdot \sum_{i=1}^{n} \left|A_i^\dagger\right| + n \cdot add + (n+1) \cdot function$$

By definition, we have:

$A_i^\dagger = \bigcup_j A_{i,j}^\dagger$, $A_{i,j}^\dagger = A_{i,j} \cap [0, \lambda[$, $A_{i,j} = \{D_j + k_j T_j - D_i,\ k_j \in Z\}$,

$$A_{i,j}^\dagger = \left\{ D_j + k_j T_j - D_i, \left\lceil \frac{D_i - D_j}{T_j} \right\rceil \le k_j \le \left\lceil \frac{\lambda + D_i - D_j}{T_j} \right\rceil - 1 \right\}.$$

Hence, $\left|A_{i,j}^\dagger\right| = \left\lceil \frac{\lambda + D_i - D_j}{T_j} \right\rceil - \left\lceil \frac{D_i - D_j}{T_j} \right\rceil$.

It follows that:

$\left|A_{i,i}^\dagger\right| = \left\lceil \frac{\lambda}{T_i} \right\rceil$ and $\forall j \neq i$, $\left\lfloor \frac{\lambda}{T_j} \right\rfloor \le \left|A_{i,j}^\dagger\right| \le \left\lceil \frac{\lambda}{T_j} \right\rceil$.

Hence,

$$\left\lceil \frac{\lambda}{T_i} \right\rceil + \sum_{j \neq i} \left\lfloor \frac{\lambda}{T_j} \right\rfloor \le \left|A_i^\dagger\right| \le \sum_j \left\lceil \frac{\lambda}{T_j} \right\rceil \text{ and } \sum_j \left\lceil \frac{\lambda}{T_j} \right\rceil + (n-1) \cdot \sum_j \left\lfloor \frac{\lambda}{T_j} \right\rfloor \le \sum_i \left|A_i^\dagger\right| \le n \cdot \sum_j \left\lceil \frac{\lambda}{T_j} \right\rceil.$$

Finally, we have:

$$\underline{C} \geq (2(2n-1)\cdot add + (3n-1)\cdot mult + (4n-1)\cdot function)\cdot\left(\sum_j\left\lceil\frac{\lambda}{T_j}\right\rceil + (n-1)\cdot\sum_j\left\lfloor\frac{\lambda}{T_j}\right\rfloor\right)$$
$$+ n\cdot add + (n+1)\cdot function$$

Assuming that $add = mult = function = \xi$, the cost of one instruction cycle, we have:

$$\left((11n-4)\cdot\left(\sum_j\left\lceil\frac{\lambda}{T_j}\right\rceil + (n-1)\cdot\sum_j\left\lfloor\frac{\lambda}{T_j}\right\rfloor\right) + (2n+1)\right)\cdot\xi \leq \underline{C}\,.$$

Let γ denote a lower bound on the cost of the optimized feasibility test for the Earliest Deadline First scheduling algorithm.

We have to consider two cases.

Case I: $\lambda \geq \mathrm{Min}_j\{T_j\}$

In that case, $(11n-4)\cdot n\cdot\sum_j\left\lfloor\frac{\lambda}{T_j}\right\rfloor\cdot\xi \leq \underline{C}$. Hence, $\gamma = \Omega\left(n^2\cdot\sum_j\left\lfloor\frac{\lambda}{T_j}\right\rfloor\right)$.

γ is pseudo-polynomial with a constant factor of approximately: 11.

Case II: $\lambda < \mathrm{Min}_j\{T_j\}$

In that case, $(11n-4)\cdot\sum_j\left\lceil\frac{\lambda}{T_j}\right\rceil\cdot\xi \leq \underline{C}$. Hence, $\gamma = \Omega\left(n\cdot\sum_j\left\lceil\frac{\lambda}{T_j}\right\rceil\right)$.

γ is pseudo-polynomial with a constant factor of approximately: 11.

4.2.2.2.3 The ratio δ/γ

From Section 4.2.2.2.1, page 84, and Section 4.2.2.2.2, page 85, we immediately have:

Case I: $\lambda \geq \mathrm{Min}_j\{T_j\}$

In that case, $\frac{\delta}{\gamma} = O\left(\frac{\sum_j\left\lceil\frac{\lambda}{T_j}\right\rceil}{\sum_j\left\lfloor\frac{\lambda}{T_j}\right\rfloor}\right)$.

$\frac{\delta}{\gamma}$ is pseudo-polynomial with a constant factor of approximately: 3.

Note that: $\frac{\sum_j\left\lceil\frac{\lambda}{T_j}\right\rceil}{\sum_j\left\lfloor\frac{\lambda}{T_j}\right\rfloor} \leq 1 + n/\left(\sum_j\left\lfloor\frac{\lambda}{T_j}\right\rfloor\right) \leq 1 + n$. Hence, $\frac{\delta}{\gamma} = O(n)$.

Case II: $\lambda < \mathrm{Min}_j\{T_j\}$

In that case, $\frac{\delta}{\gamma} = O(n)$.

$\frac{\delta}{\gamma}$ is pseudo-polynomial with a constant factor of approximately: 3.

Here, it is important to point out that the ratio δ/γ is at most 3n, but it is much closer to 3 in most cases. This means that δ (resp. γ), the upper (resp. lower) bound on the cost of the optimized feasibility test for the Earliest Deadline First scheduling algorithm, is rather tight and therefore relevant.

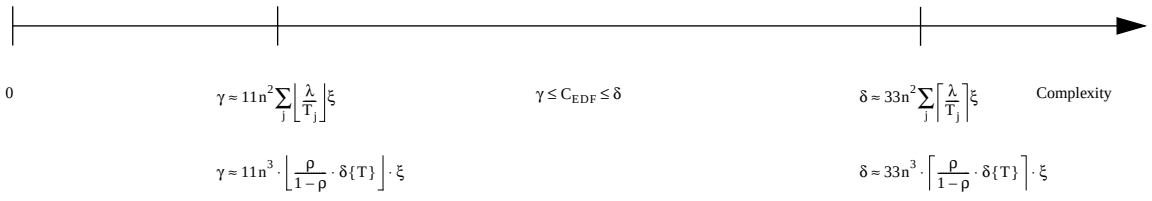


Figure 9: Complexity of the optimized feasibility test for EDF ; $\lambda \ge \mathrm{Min}_j\{T_j\}$

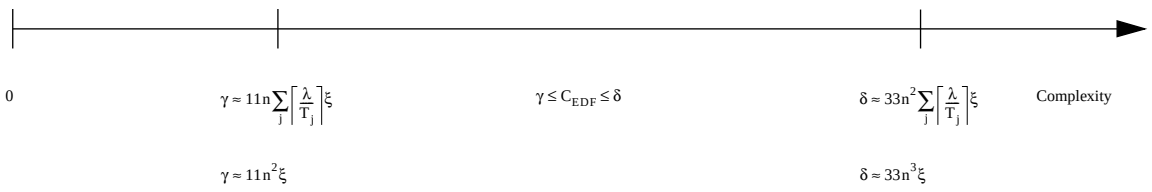


Figure 10: Complexity of the optimized feasibility test for EDF ; $\lambda < \mathrm{Min}_j\{T_j\}$

The second case, where $\lambda < \mathrm{Min}_j\{T_j\}$, is a pathological one. In such a case, we have:

- $\lambda = \sum_j C_j$,
- $\forall j \in [1, n], \left\lfloor \frac{\lambda}{T_j} \right\rfloor = 0$ and $\left\lceil \frac{\lambda}{T_j} \right\rceil = 1$ ($\sum_j \left\lfloor \frac{\lambda}{T_j} \right\rfloor = 0$ and $\sum_j \left\lceil \frac{\lambda}{T_j} \right\rceil = n$).

4.2.3 Complexity Theorems

4.2.3.1 The ratio $C_{\mathbf{FP},\,(\forall \mathbf{i} \in [\mathbf{1},\,\mathbf{n}],\,\mathbf{r_i} \le \mathbf{D_i})} / C_{\mathbf{EDF},\,(\forall \mathbf{t} \in \mathbf{S},\,\mathbf{P}(\mathbf{t}))}$

We have to distinguish two cases:

■ Case I: $\lambda < \mathrm{Max}_j\{D_j\}$

In that case, from Section 4.2.1, page 81, we have: $\alpha' \le C_{EDF} \le \beta'$, where:

$$\Omega(n) \le \alpha' \le O(n^2) \text{ and } \beta' \approx 11n \cdot \sum_j \left\lceil \frac{\lambda}{T_j} \right\rceil \cdot \xi \approx 11n^2 \cdot \left\lceil \frac{\rho}{1-\rho} \cdot \delta\{T\} \right\rceil \cdot \xi$$

From Section 4.2.2.1, page 82, we have: $\alpha \le C_{FP} \le \beta$, where:

$$\alpha \approx 2n^2\xi$$

$$\beta \approx 2n^2 \cdot \sum_j \left\lceil \frac{\lambda}{T_j} \right\rceil \cdot \xi \approx 2n^3 \cdot \left\lceil \frac{\rho}{1-\rho} \cdot \delta\{T\} \right\rceil \cdot \xi$$

Theorem 26 - $\dfrac{\alpha}{\beta'} \le \dfrac{C_{FP,\,(\forall i \in [1,\,n],\,r_i \le D_i)}}{C_{EDF,\,(\forall t \in S,\,P(t))}} \le \dfrac{\beta}{\alpha'}$ *where:* $\dfrac{\alpha}{\beta'} = \Omega(1)$ *and* $\dfrac{\beta}{\alpha'} = O(n^2)$

Proof.

We have: $\dfrac{\alpha}{\beta'} \approx \dfrac{2n^2 \cdot \xi}{11n^2 \cdot \left\lceil \dfrac{\rho}{1-\rho} \cdot \delta\{T\} \right\rceil \cdot \xi} \approx \dfrac{2}{11} \cdot \dfrac{1}{\left\lceil \dfrac{\rho}{1-\rho} \cdot \delta\{T\} \right\rceil}$. Hence, $\dfrac{\alpha}{\beta'} = \Omega(1)$.

We have: $\dfrac{\beta}{\alpha'} \approx \dfrac{2n^2 \cdot \sum_j \left\lceil \dfrac{\lambda}{T_j} \right\rceil \cdot \xi}{n \cdot \xi} \approx 2n^2 \cdot \left\lceil \dfrac{\rho}{1-\rho} \cdot \delta\{T\} \right\rceil$. Hence, $\dfrac{\beta}{\alpha'} = O(n^2)$. ❑

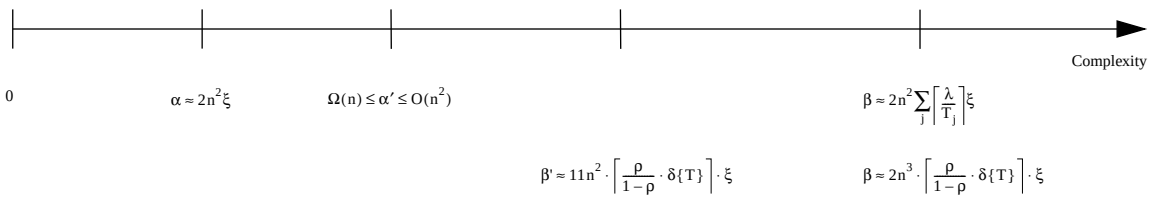


Figure 11: Complexity comparison : the ratio $C_{FP,\,(\forall i \in [1,\,n],\,r_i \le D_i)} / C_{EDF,\,(\forall t \in S,\,P(t))}$

The ratio C_{FP}/C_{EDF} is bounded by: β/α', which is pseudo-polynomial in $O(n^2)$.

■ Case II: $\lambda \geq \mathrm{Max}_j\{D_j\}$

In that case, from Section 4.2.1, page 81, we have: $C_{EDF} \approx \alpha' \approx \beta'$, where:

$$\alpha' \approx \beta' \approx 11n \cdot \sum_j \left\lceil \frac{\lambda}{T_j} \right\rceil \cdot \xi \approx 11n^2 \cdot \left\lceil \frac{\rho}{1-\rho} \cdot \delta\{T\} \right\rceil \cdot \xi$$

From Section 4.2.2.1, page 82, we have: $\alpha \leq C_{FP} \leq \beta$, where:

$$\alpha \approx 2n^2\xi$$

$$\beta \approx 2n^2 \cdot \sum_j \left\lceil \frac{\lambda}{T_j} \right\rceil \cdot \xi \approx 2n^3 \cdot \left\lceil \frac{\rho}{1-\rho} \cdot \delta\{T\} \right\rceil \cdot \xi$$

Theorem 27 - $\frac{\alpha}{\beta'} \leq \frac{C_{FP,\,(\forall i \in [1,n],\, r_i \leq D_i)}}{C_{EDF,\,(\forall t \in S,\, P(t))}} \leq \frac{\beta}{\alpha'}$ *where:* $\frac{\alpha}{\beta'} = \Omega(1)$ *and* $\frac{\beta}{\alpha'} = O(n)$

Proof.

We have: $\frac{\alpha}{\beta'} \approx \frac{2n^2 \cdot \xi}{11n^2 \cdot \left\lceil \frac{\rho}{1-\rho} \cdot \delta\{T\} \right\rceil \cdot \xi} \approx \frac{2}{11} \cdot \frac{1}{\left\lceil \frac{\rho}{1-\rho} \cdot \delta\{T\} \right\rceil}$. Hence, $\frac{\alpha}{\beta'} = \Omega(1)$.

We have: $\frac{\beta}{\alpha'} \approx \frac{2n^2 \cdot \sum_j \left\lceil \frac{\lambda}{T_j} \right\rceil \cdot \xi}{11n \cdot \sum_j \left\lceil \frac{\lambda}{T_j} \right\rceil \cdot \xi} \approx \frac{2}{11} \cdot n$. Hence, $\frac{\beta}{\alpha'} = O(n)$. ❑

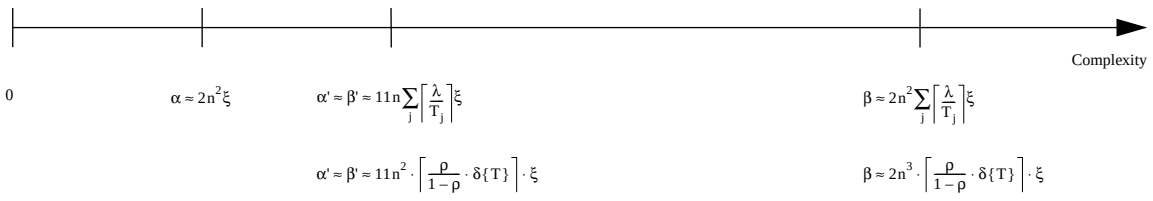


Figure 12: Complexity comparison : the ratio $C_{FP,\,(\forall i \in [1,n],\, r_i \leq D_i)} / C_{EDF,\,(\forall t \in S,\, P(t))}$

The ratio C_{FP}/C_{EDF} is bounded by: β/α', which is pseudo-polynomial in $O(n)$.

4.2.3.2 The ratio $C_{\mathbf{EDF},\,(\forall \mathbf{i} \in [\mathbf{1},\,\mathbf{n}],\,\mathbf{r_i} \le \mathbf{D_i})} / C_{\mathbf{FP},\,(\forall \mathbf{i} \in [\mathbf{1},\,\mathbf{n}],\,\mathbf{r_i} \le \mathbf{D_i})}$

From Section 4.2.2.1, page 82, we have: $\alpha \le C_{FP} \le \beta$, where:

$$\alpha \approx 2n^2 \cdot \xi$$
$$\beta \approx 2n^2 \cdot \sum_j \left\lceil \frac{\lambda}{T_j} \right\rceil \cdot \xi \approx 2n^3 \cdot \left\lceil \frac{\rho}{1-\rho} \cdot \delta\{T\} \right\rceil \cdot \xi$$

From Section 4.2.2.2, page 84, we have: $\gamma \le C_{EDF} \le \delta$, where:

$$\gamma \approx 11n^2 \cdot \sum_j \left\lfloor \frac{\lambda}{T_j} \right\rfloor \cdot \xi \approx 11n^3 \cdot \left\lfloor \frac{\rho}{1-\rho} \cdot \delta\{T\} \right\rfloor \cdot \xi$$
$$\delta \approx 33n^2 \cdot \sum_j \left\lceil \frac{\lambda}{T_j} \right\rceil \cdot \xi \approx 33n^3 \cdot \left\lceil \frac{\rho}{1-\rho} \cdot \delta\{T\} \right\rceil \cdot \xi$$

Theorem 28 - $\frac{\gamma}{\beta} \le \frac{C_{EDF,\,(\forall i \in [1,\,n],\,r_i \le D_i)}}{C_{FP,\,(\forall i \in [1,\,n],\,r_i \le D_i)}} \le \frac{\delta}{\alpha}$ *where:* $\frac{\gamma}{\beta} = \Omega(1)$ *and* $\frac{\delta}{\alpha} = O(n)$

Proof.

We have: $\frac{\gamma}{\beta} \approx \frac{11}{2} \cdot \frac{n^2 \cdot \sum_j \left\lfloor \frac{\lambda}{T_j} \right\rfloor \cdot \xi}{n^2 \cdot \sum_j \left\lceil \frac{\lambda}{T_j} \right\rceil \cdot \xi} \approx \frac{11}{2}$. Hence, $\frac{\gamma}{\beta} = \Omega(1)$.

We have: $\frac{\delta}{\alpha} \approx \frac{33}{2} \cdot \sum_j \left\lceil \frac{\lambda}{T_j} \right\rceil \approx \frac{33}{2} \cdot n \cdot \left\lceil \frac{\rho}{1-\rho} \cdot \delta\{T\} \right\rceil$. Hence, $\frac{\delta}{\alpha} = O(n)$. ❑

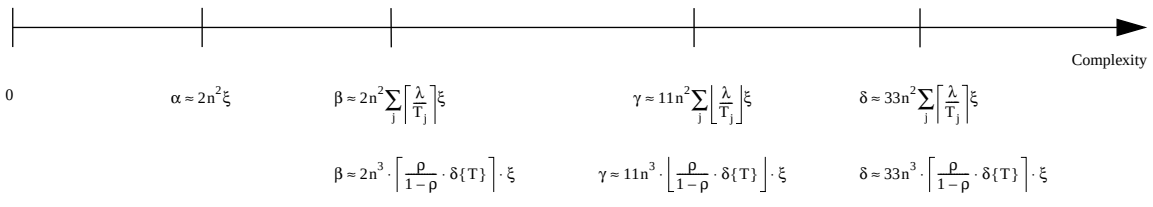


Figure 13: Complexity comparison : the ratio $C_{EDF,\,(\forall i \in [1,\,n],\,r_i \le D_i)} / C_{FP,\,(\forall i \in [1,\,n],\,r_i \le D_i)}$

The ratio C_{EDF} / C_{FP} is bounded by: δ / α, which is pseudo-polynomial in $O(n)$.

4.2.4 Summary

■ **The optimized feasibility test for EDF in a « $\forall \mathbf{t} \in \mathbf{S}, \mathbf{P}(\mathbf{t})$ » form**

From Section 4.1.1, page 70, we know that:

- $C_{EDF, (\forall t \in S, P(t))}$ is pseudo-polynomial in $O(n^2)$,
- $\alpha' \leq C_{EDF, (\forall t \in S, P(t))} \leq \beta'$, where $\alpha' = \Omega(n)$ and $\beta' = O(n^2)$.

■ **The optimized feasibility test for FP scheduling algorithms in a « $\forall \mathbf{i} \in [\mathbf{1}, \mathbf{n}], \mathbf{r_i} \leq \mathbf{D_i}$ » form**

From Section 4.1.2.1, page 72, we know that:

- $C_{FP, (\forall i \in [1, n], r_i \leq D_i)}$ is pseudo-polynomial in $O(n^3)$,
- $\alpha \leq C_{FP, (\forall i \in [1, n], r_i \leq D_i)} \leq \beta$, where $\alpha = \Omega(n^2)$ and $\beta = O(n^3)$.

■ **The optimized feasibility test for EDF in a « $\forall \mathbf{i} \in [\mathbf{1}, \mathbf{n}], \mathbf{r_i} \leq \mathbf{D_i}$ » form**

From Section 4.1.2.2, page 76, we know that:

- $C_{EDF, (\forall i \in [1, n], r_i \leq D_i)}$ is pseudo-polynomial in $O(n^3)$,
- $\gamma \leq C_{EDF, (\forall i \in [1, n], r_i \leq D_i)} \leq \delta$, where $\gamma = \Omega(n^3)$ and $\delta = O(n^3)$.

■ **The ratio $\mathbf{C_{FP, (\forall i \in [1, n], r_i \leq D_i)} / C_{EDF, (\forall t \in S, P(t))}}$**

Case I: $\lambda < \mathrm{Max}_j\{D_j\}$. From Theorem 26, page 88, we know that:

- $C_{FP, (\forall i \in [1, n], r_i \leq D_i)} / C_{EDF, (\forall t \in S, P(t))}$ is pseudo-polynomial in $O(n^2)$,
- $\frac{\alpha}{\beta'} \leq \frac{C_{FP, (\forall i \in [1, n], r_i \leq D_i)}}{C_{EDF, (\forall t \in S, P(t))}} \leq \frac{\beta}{\alpha'}$, where $\frac{\alpha}{\beta'} = \Omega(1)$ and $\frac{\beta}{\alpha'} = O(n^2)$.

Case II: $\lambda \geq \mathrm{Max}_j\{D_j\}$. From Theorem 27, page 89, we know that:

- $C_{FP, (\forall i \in [1, n], r_i \leq D_i)} / C_{EDF, (\forall t \in S, P(t))}$ is pseudo-polynomial in $O(n)$,
- $\frac{\alpha}{\beta'} \leq \frac{C_{FP, (\forall i \in [1, n], r_i \leq D_i)}}{C_{EDF, (\forall t \in S, P(t))}} \leq \frac{\beta}{\alpha'}$, where $\frac{\alpha}{\beta'} = \Omega(1)$ and $\frac{\beta}{\alpha'} = O(n)$.

■ **The ratio $\mathbf{C_{EDF, (\forall i \in [1, n], r_i \leq D_i)} / C_{FP, (\forall i \in [1, n], r_i \leq D_i)}}$**

From Theorem 28, page 90, we know that:

- $C_{EDF, (\forall i \in [1, n], r_i \leq D_i)} / C_{FP, (\forall i \in [1, n], r_i \leq D_i)}$ is pseudo-polynomial in $O(n)$,
- $\frac{\gamma}{\beta} \leq \frac{C_{EDF, (\forall i \in [1, n], r_i \leq D_i)}}{C_{FP, (\forall i \in [1, n], r_i \leq D_i)}} \leq \frac{\delta}{\alpha}$, where $\frac{\gamma}{\beta} = \Omega(1)$ and $\frac{\delta}{\alpha} = O(n)$.

4.3 Application of the complexity theorems

4.3.1 Homogeneous traffics

We consider homogeneous traffics. We have: $T_j = T$ and $D_j = D$ for every $j \in [1, n]$.

We also have: $\lambda = \sum_j C_j$ and $\lambda_i = \sum_{j \le i} C_j$ for every $i \in [1, n]$.

4.3.1.1 The feasibility test for EDF in a « $\forall t \in S, P(t)$ » form

Theorem 29 - $N_{EDF}(\tau) = N_U(\tau) \cdot \dfrac{T}{\mathrm{Min}\{T, D\}}$

Proof.

We have: $N_U(\tau) = \dfrac{1}{T}\sum_j C_j$ and $N_{EDF}(\tau) = N_U(\tau) \cdot T \cdot \mathrm{Max}_{t>0}\left\{\dfrac{1}{t}\mathrm{Max}\left\{0, 1 + \left\lfloor \dfrac{t-D}{T} \right\rfloor\right\}\right\}$.

Thus, $N_{EDF}(\tau) = N_U(\tau) \cdot T \cdot \mathrm{Max}_{k \ge 0}\left\{\dfrac{1+k}{D+kT}\right\}$.

Let $f(k) = \dfrac{1+k}{D+kT}$. We have: $f'(k) = \dfrac{D-T}{(D+kT)^2}$.

We have to consider three cases.

Case I: $D < T$

In that case, $f'(k) < 0$. It follows that: $\mathrm{Max}_{k \ge 0}\{f(k)\} = f(0) = \dfrac{1}{D}$.

Case II: $D = T$

In that case, $f(k) = \dfrac{1}{T} = \dfrac{1}{D}$ for every integer k. Hence, $\mathrm{Max}_{k \ge 0}\{f(k)\} = \dfrac{1}{T} = \dfrac{1}{D}$.

Case III: $D > T$

In that case, $f'(k) > 0$. It follows that: $\mathrm{Max}_{k \ge 0}\{f(k)\} = \lim_{k \to +\infty} f(k) = \dfrac{1}{T}$.

Consequently, we have: $\mathrm{Max}_{k \ge 0}\{f(k)\} = \dfrac{1}{\mathrm{Min}\{T, D\}}$.

Finally, we have: $N_{EDF}(\tau) = N_U(\tau) \cdot \dfrac{T}{\mathrm{Min}\{T, D\}}$. ❑

From Theorem 29, page 92, we have the following equivalence relations:

$$(N_{EDF}(\tau) \leq 1) \Leftrightarrow (\lambda \leq \text{Min}\{T, D\}) \Leftrightarrow ((\lambda \leq T) \wedge (\lambda \leq D)) \Leftrightarrow ((N_U(\tau) \leq 1) \wedge (\lambda \leq D))$$

In other words, τ is EDF feasible if and only if $\sum_j C_j \leq \text{Min}\{T, D\}$.

Let add and function denote respectively the cost of one addition and the cost of one elementary function such as “min” or “less than or equal to”.

Let C denote the complexity for this feasibility test. C is just equal to $(n-1) \cdot \text{add} + 2 \cdot \text{function}$.

Assuming that $\text{add} = \text{function} = \xi$, the cost of one instruction cycle, we have:

$$C = (n+1)\xi, \text{ which is polynomial in O(n)}$$

4.3.1.2 The feasibility tests in a « $\forall i \in [1, n]$, $r_i \leq D_i$ » form

4.3.1.2.1 Fixed Priority based scheduling algorithms

Theorem 30 - $\forall i \in [1, n-1],\ r_{i+1} = r_i + C_{i+1},\ r_1 = C_1$

Proof.

Let $\rho_i = \frac{1}{T}\sum_{j \leq i} C_j$. By definition, $\rho_i \leq 1$ for every $i \in [1, n]$, which is equivalent to $\rho_n \leq 1$.

Given that $\lambda_i = \sum_{j \leq i} C_j$, we have $\rho_i = \frac{\lambda_i}{T}$, which is less than or equal to one.

From Theorem 22, page 52, the worst-case response time of a task τ_i is given by:

$$r_i = \text{Max}_{0 \leq q \leq Q_i}\{w_{i,q} - qT_i\}$$

where $Q_i = \left\lceil \frac{\lambda_i}{T_i} \right\rceil - 1$ is the smallest value of q that satisfies $w_{i,q} \leq (q+1)T_i$.

For homogeneous traffics, we have: $Q_i = \left\lceil \frac{\lambda_i}{T} \right\rceil - 1 = \lceil \rho_i \rceil - 1 = 0$ since $0 < \rho_i \leq 1$.

Therefore, the worst-case response time of a task τ_i is given by:

$$r_i = w_{i,0} = w_{i,Q_i} = \lambda_i = \sum_{j \leq i} C_j$$

Finally, we have: $\forall i \in [1, n-1],\ r_{i+1} = r_i + C_{i+1},\ r_1 = C_1$. ❑

From Theorem 30, page 93, we have the following equivalence relations:

$$(\forall i \in [1, n], ((\rho_i \le 1) \wedge (r_i \le D))) \Leftrightarrow ((\rho_n \le 1) \wedge (r_n \le D)) \Leftrightarrow ((\lambda \le T) \wedge (\lambda \le D))$$
$$(\forall i \in [1, n], ((\rho_i \le 1) \wedge (r_i \le D))) \Leftrightarrow (\lambda \le \mathrm{Min}\{T, D\}) \Leftrightarrow (N_{EDF}(\tau) \le 1)$$

In other words, τ is feasible if and only if $r_n = \sum_j C_j = \lambda$ is less than or equal to $\mathrm{Min}\{T, D\}$.

Let add and function denote respectively the cost of one addition and the cost of one elementary function such as “min” or “less than or equal to”.

Let C denote the complexity for this feasibility test. C is just equal to $(n-1) \cdot \mathrm{add} + 2 \cdot \mathrm{function}$.

Assuming that $\mathrm{add} = \mathrm{function} = \xi$, the cost of one instruction cycle, we have:

$$C = (n+1)\xi, \text{ which is polynomial in O(n)}$$

4.3.1.2.2 The Earliest Deadline First scheduling algorithm

Theorem 31 - $\forall i \in [1, n],\ r_i = \lambda$

Proof.

From Theorem 23, page 54, the worst-case response time of a task τ_i is given by:

$$r_i = \mathrm{Max}_{a \ge 0}\{L_i(a) - a\}$$

where:

$$L_i(a) = \left(1 + \left\lfloor \frac{a}{T_i} \right\rfloor\right) C_i + \sum_{j \ne i} \min\left\{ \left\lceil \frac{L_i(a)}{T_j} \right\rceil, \max\left\{0, 1 + \left\lfloor \frac{a + D_i - D_j}{T_j} \right\rfloor \right\} \right\} C_j$$

For homogeneous traffics, we have: $L_i(a) = \left(1 + \left\lfloor \frac{a}{T} \right\rfloor\right) C_i + \min\left\{ \left\lceil \frac{L_i(a)}{T} \right\rceil, 1 + \left\lfloor \frac{a}{T} \right\rfloor \right\} \cdot \sum_{j \ne i} C_j$.

We have: $r_i = \mathrm{Max}_{a \ge 0}\{L_i(a) - a\} = \mathrm{Max}_{\substack{a \ge 0 \\ L_i(a) - a > 0}} \{L_i(a) - a\}$.

If $L_i(a) - a > 0$ then $1 + \left\lfloor \frac{a}{T} \right\rfloor \le \left\lceil \frac{L_i(a)}{T} \right\rceil$, which involves $L_i(a) = \left(1 + \left\lfloor \frac{a}{T} \right\rfloor\right) \cdot \lambda$.

It follows that:

$$r_i = \mathrm{Max}_{\substack{a \ge 0 \\ L_i(a) - a > 0}} \left\{ \left(1 + \left\lfloor \frac{a}{T} \right\rfloor\right) \cdot \lambda - a \right\} = \lambda + \mathrm{Max}_{\substack{k \ge 0 \\ L_i(kT) - kT > 0}} \{k\lambda - kT\}.$$

$$r_i = \lambda + T \cdot (1-\rho) \cdot \underset{\substack{k \geq 0 \\ L_i(kT) - kT > 0}}{\mathrm{Max}} \{-k\} = \lambda. \qquad \square$$

From Theorem 31, page 94, we have the following equivalence relations:

$$((\rho \leq 1) \wedge (\forall i \in [1, n], r_i \leq D)) \Leftrightarrow ((\lambda \leq T) \wedge (\lambda \leq D)) \Leftrightarrow (\lambda \leq \mathrm{Min}\{T, D\})$$
$$((\rho \leq 1) \wedge (\forall i \in [1, n], r_i \leq D)) \Leftrightarrow (N_{EDF}(\tau) \leq 1)$$

In other words, τ is feasible if and only if $\lambda = \sum_j C_j$ is less than or equal to $\mathrm{Min}\{T, D\}$.

Let add and function denote respectively the cost of one addition and the cost of one elementary function such as “min” or “less than or equal to”.

Let C denote the complexity for this feasibility test. C is just equal to $(n-1) \cdot \mathrm{add} + 2 \cdot \mathrm{function}$.

Assuming that $\mathrm{add} = \mathrm{function} = \xi$, the cost of one instruction cycle, we have:

$$C = (n+1)\xi, \text{ which is polynomial in O(n)}$$

4.3.1.3 Summary

■ **The optimized feasibility test for EDF in a « $\forall t \in S, P(t)$ » form**

From Section 4.3.1.1, page 92, we know that:

- $(N_{EDF}(\tau) \leq 1) \Leftrightarrow (\lambda \leq \mathrm{Min}\{T, D\})$ where $\lambda = \sum_j C_j$,
- $C_{EDF, (\forall t \in S, P(t))} = (n+1)\xi$.

■ **The optimized feasibility test for FP scheduling algorithms in a « $\forall i \in [1, n], r_i \leq D_i$ » form**

From Section 4.3.1.2.1, page 93, we know that:

- $(\forall i \in [1, n], r_i \leq D_i) \Leftrightarrow (N_{EDF}(\tau) \leq 1)$ where $\forall i \in [1, n], r_i = \lambda_i = \sum_{j \leq i} C_j$,
- $C_{FP, (\forall i \in [1, n], r_i \leq D_i)} = (n+1)\xi$.

■ **The optimized feasibility test for EDF in a « $\forall i \in [1, n], r_i \leq D_i$ » form**

From Section 4.3.1.2.2, page 94, we know that:

- $(\forall i \in [1, n], r_i \leq D_i) \Leftrightarrow (N_{EDF}(\tau) \leq 1)$ where $\forall i \in [1, n], r_i = \lambda = \sum_j C_j$,
- $C_{EDF, (\forall i \in [1, n], r_i \leq D_i)} = (n+1)\xi$.

■ **The ratios** $\frac{C_{\mathbf{FP},\,(\forall \mathbf{i} \in [\mathbf{1},\,\mathbf{n}],\,\mathbf{r_i} \le \mathbf{D_i})}}{C_{\mathbf{EDF},\,(\forall \mathbf{t} \in \mathbf{S},\,\mathbf{P}(\mathbf{t}))}}$ **and** $\frac{C_{\mathbf{EDF},\,(\forall \mathbf{i} \in [\mathbf{1},\,\mathbf{n}],\,\mathbf{r_i} \le \mathbf{D_i})}}{C_{\mathbf{FP},\,(\forall \mathbf{i} \in [\mathbf{1},\,\mathbf{n}],\,\mathbf{r_i} \le \mathbf{D_i})}}$

From above, we immediately have: $\frac{C_{FP,\,(\forall i \in [1,\,n],\,r_i \le D_i)}}{C_{EDF,\,(\forall t \in S,\,P(t))}} = \frac{C_{EDF,\,(\forall i \in [1,\,n],\,r_i \le D_i)}}{C_{FP,\,(\forall i \in [1,\,n],\,r_i \le D_i)}} = 1$.

As a conclusion, we can say that the feasibility conditions are the same for Fixed Priority based scheduling algorithms and EDF, which is not surprising at all. (cf. Section 3.3, page 63).

4.3.2 Heterogeneous traffics

4.3.2.1 The feasibility test for EDF in a « $\forall \mathbf{t} \in \mathbf{S}, \mathbf{P}(\mathbf{t})$ » form

Theorem 32 - $\sum_j \frac{C_j}{T_j} \le N_{EDF}(\tau) \le \sum_j \frac{C_j}{\mathrm{Min}\{T_j, D_j\}}$

Proof.

Let us consider a task set $\tau \in \Upsilon$, $\tau = \sum_j \tau_j$.

From Theorem 29, page 92, we have: $N_{EDF}(\tau_j) = N_U(\tau_j) \cdot \frac{T_j}{\mathrm{Min}\{T_j, D_j\}} = \frac{C_j}{\mathrm{Min}\{T_j, D_j\}}$.

If follows that: $N_{EDF}(\tau) \le \sum_j N_{EDF}(\tau_j)$ since N_{EDF} is a norm. Furthermore, $N_U(\tau) \le N_{EDF}(\tau)$.

Finally, we have: $\sum_j \frac{C_j}{T_j} \le N_{EDF}(\tau) \le \sum_j \frac{C_j}{\mathrm{Min}\{T_j, D_j\}}$. ❑

4.3.2.1.1 $T_j \le D_j$, for every $j \in [1, n]$

From Theorem 32, page 96, we immediately have: $\boxed{N_{EDF}(\tau) = N_U(\tau) = \sum_j \frac{C_j}{T_j}}$

Let add, mult and function denote respectively the cost of one addition, the cost of one multiplication and the cost of one elementary function such as “less than or equal to”.

Let C denote the complexity for this feasibility test.

C is just equal to $(n-1) \cdot \mathrm{add} + n \cdot \mathrm{mult} + 1 \cdot \mathrm{function}$.

Assuming that $\mathrm{add} = \mathrm{mult} = \mathrm{function} = \xi$, the cost of one instruction cycle, we have:

$\boxed{C = 2n\xi, \text{ which is polynomial in } O(n)}$

4.3.2.1.2 $T_j > D_j$**, for every** $j \in [1, n]$

From the assumption $D_j < T_j$, for every $j \in [1, n]$, Lemma 7, page 44, may be revisited as follows:

$$\forall t,\ 0 \le t < \mu',\ \Phi(t) \ge 0 \Rightarrow \forall t,\ t \ge 0,\ \Phi(t) \ge 0 \text{ where } \mu' = \frac{\sum_{j=1}^{n} (T_j - D_j)\frac{C_j}{T_j}}{1 - \sum_{j=1}^{n} \frac{C_j}{T_j}}$$

Proof.

From the assumption $D_j < T_j$, for every $j \in [1, n]$, we immediately have:

$\Phi(t) = t - \sum_{j=1}^{n} \left(1 + \left\lfloor \frac{t - D_j}{T_j} \right\rfloor\right) C_j$ without assuming $t \ge \text{Max}_{1 \le j \le n}\{D_j\}$.

The end of the proof is trivial (cf. Lemma 7, page 44). ❑

Finally, Theorem 21, page 49, may be revisited as follows:

$$\text{Max}_{t \in S}\left\{\frac{1}{t}\sum_{j=1}^{n} \text{Max}\left\{0, 1 + \left\lfloor \frac{t - D_j}{T_j} \right\rfloor\right\} C_j\right\} \le 1$$

where:

- $S = \bigcup_{j=1}^{n} S_j$, $S_j = \left\{D_j + k_j T_j,\ 0 \le k_j \le \left\lceil \frac{\text{Min}\{\lambda, \mu'\} - D_j}{T_j} \right\rceil - 1\right\}$
- $\lambda = \sum_{j=1}^{n} \left\lceil \frac{\lambda}{T_j} \right\rceil C_j$ (λ is the smallest positive root of this equation)
- $\mu' = \frac{\sum_{j=1}^{n} (T_j - D_j)\frac{C_j}{T_j}}{1 - \sum_{j=1}^{n} \frac{C_j}{T_j}}$

Let C denote the complexity for this feasibility test. From Section 4.1.1, page 70, we know that C is pseudo-polynomial in $O(n^2)$ with a constant factor of approximately: $11 \cdot \left\lceil \frac{\rho}{1-\rho} \cdot \delta\{T\} \right\rceil - 4$, which is bounded by: $11 \cdot \left\lceil \frac{P}{1-P} \cdot \Delta\{T\} \right\rceil - 4$ when $\rho \le P < 1$ and $\delta\{T\} \le \Delta\{T\}$.

4.3.2.2 The feasibility tests in a « $\forall i \in [1, n], r_i \leq D_i$ » form

4.3.2.2.1 Fixed Priority based scheduling algorithms

4.3.2.2.1.1 $T_j \geq D_j$, for every $j \in [1, n]$

From Theorem 22, page 52, the worst-case response time of a task τ_i is given by:

$$r_i = \text{Max}_{0 \leq q \leq Q_i}\{w_{i,q} - qT_i\}$$

where $Q_i = \left\lceil \frac{\lambda_i}{T_i} \right\rceil - 1$ is the smallest value of q that satisfies $w_{i,q} \leq (q+1)T_i$.

The task set $\tau = \sum_i \tau_i$ is feasible if and only if: $\forall i \in [1, n], r_i \leq D_i$, that is:

$$\forall i \in [1, n], \forall q \in [0, Q_i], w_{i,q} \leq qT_i + D_i$$

Assuming that $T_i \geq D_i$, for every $i \in [1, n]$, we immediately have:

$$\forall i \in [1, n], \forall q \in [0, Q_i], w_{i,q} \leq (q+1)T_i$$

Hence, $Q_i = 0$ and $\lambda_i \leq T_i$. It follows that: $r_i = w_{i,0} = w_{i,Q_i} = \lambda_i$.

Finally, we have:

$$\boxed{\forall i \in [1, n],\ r_i = \lambda_i \text{ where: } \lambda_i = \sum_{j \leq i} \left\lceil \frac{\lambda_i}{T_j} \right\rceil C_j}$$

A practical way of solving this recursive equation is given in Appendix A, page 120.

Let add, mult and function denote respectively the cost of one addition, the cost of one multiplication and the cost of one elementary function such as “ceil”.

Let C_i denote the cost we have to pay to solve $\lambda_i = \sum_{j \leq i} \left\lceil \frac{\lambda_i}{T_j} \right\rceil C_j$ by successive iterations starting from $\lambda_i^{(1)} = \sum_{j \leq i} C_j$.

From Appendix A, page 120, we have:

$$\boxed{C_i \leq (i-1) \cdot \text{add} + \left(\sum_{j \leq i} \left\lceil \frac{\lambda_i}{T_j} \right\rceil - i + 1 \right) \cdot ((i-1) \cdot \text{add} + (2i) \cdot \text{mult} + (i+1) \cdot \text{function})}$$

Assuming that $\text{add} = \text{mult} = \text{function} = \xi$, the cost of one instruction cycle, we have:

$$\boxed{C_i \leq \left(4i \sum_{j \leq i} \left\lceil \frac{\lambda_i}{T_j} \right\rceil - (4i-1)(i-1) \right) \xi}$$

Let C denote the complexity for the feasibility test $\forall i \in [1, n], r_i \leq D_i$. We have:

C	The cost of the optimized feasibility test $\forall i \in [1, n],\ r_i \leq D_i$
	$test \leftarrow -\infty\ ;\ \forall i \in [1, n],\ test \leftarrow Max\{test, r_i - D_i\}\ ;$ is $test \leq 0$? $C \leq \sum_{i=1}^{n} (C_i + 2\xi) + \xi$

It follows that:

$$C \leq \left(4\sum_i i \sum_{j \leq i} \left\lceil \frac{\lambda_i}{T_j} \right\rceil - \frac{1}{6}(n+1)(8n^2 - 11n - 6)\right)\xi .$$

Furthermore, we have: $\lambda_i \leq \lambda_n$ for every $i \in [1, n]$. For notational convenience, let $\lambda_n = \lambda$.

We have:

$$C \leq \frac{1}{6}\left(12n\sum_j \left\lceil \frac{\lambda}{T_j} \right\rceil - 8n^2 + 11n + 6\right)(n+1)\xi .$$

From Appendix A, page 120, we also have:

$$\sum_j \left\lceil \frac{\lambda}{T_j} \right\rceil \leq n \cdot \left\lceil \frac{\rho}{1-\rho} \cdot \delta\{T\} \right\rceil, \text{ where } \rho = \sum_j \frac{C_j}{T_j} \text{ and } \delta\{T\} = \frac{Max_l\{T_l\}}{Min_l\{T_l\}}.$$

Finally, we have:

$$C \leq \left(2\left(\left\lceil \frac{\rho}{1-\rho} \cdot \delta\{T\} \right\rceil - 1\right)n^2 + \frac{1}{6}(4n^2 + 11n + 6)\right)(n+1)\xi$$

Therefore, C is pseudo-polynomial in $O(n^3)$ with a constant factor of approximately:

$2 \cdot \left(\left\lceil \frac{\rho}{1-\rho} \cdot \delta\{T\} \right\rceil - 1\right) + \frac{2}{3}$, which is bounded by: $2 \cdot \left(\left\lceil \frac{P}{1-P} \cdot \Delta\{T\} \right\rceil - 1\right) + \frac{2}{3}$ when:

$\rho \leq P < 1$ and $\delta\{T\} \leq \Delta\{T\}$.

4.3.2.2.1.2 $T_j < D_j$, for every $j \in [1, n]$

In that case, we do not have any particular results and we apply those of Section 4.1.2.1, page 72. We have:

$$C \leq \left((2n^2 + n + 7) \cdot n \cdot \left\lceil \frac{\rho}{1-\rho} \cdot \delta\{T\} \right\rceil + (2n+1)\right) \cdot \xi$$

C is pseudo-polynomial in $O(n^3)$ with a constant factor of approximately:
$2 \cdot \left\lceil \frac{\rho}{1-\rho} \cdot \delta\{T\} \right\rceil$, which is bounded by: $2 \cdot \left\lceil \frac{P}{1-P} \cdot \Delta\{T\} \right\rceil$ when $\rho \le P < 1$; $\delta\{T\} \le \Delta\{T\}$.

4.3.2.2.2 The Earliest Deadline First scheduling algorithm

In that case, we do not have any particular results and we apply those of Section 4.1.2.2, page 76.
We have:

$$C \le (33n^2 - 27n + 6) \cdot n \cdot \left\lceil \frac{\rho}{1-\rho} \cdot \delta\{T\} \right\rceil \cdot \xi$$

C is pseudo-polynomial in $O(n^3)$ with a constant factor of approximately:
$33 \cdot \left\lceil \frac{\rho}{1-\rho} \cdot \delta\{T\} \right\rceil$, which is bounded by: $33 \cdot \left\lceil \frac{P}{1-P} \cdot \Delta\{T\} \right\rceil$ when $\rho \le P < 1$; $\delta\{T\} \le \Delta\{T\}$.

5 Numerical examples

The goal of this section is to verify our theoretical results in presence of numerical examples. In Section 5.1, page 101, considering particular scheduling referentials, we will briefly comment on the possible measures. In Section 5.2, page 105 (resp. Section 5.3, page 111), we will focus on the efficiency (resp. complexity) analysis in order to highlight the trends and the limitations of our preliminary results established in Section 3, page 60 (resp. Section 4, page 68).

5.1 Particular scheduling referential

Let $\Sigma = (\Upsilon, \Pi)$ be a scheduling referential with Π = {EDF, DM) and Υ be a PNTSC characterized by all the deadline periodic equivalent task sets of τ, a particular PNTS (where $\tau_i = (C_i, D_i, T_i)$,

- (1) τ ={(3, 20, 12) ; (4, 20, 12) ; (1, 20, 12) ; (1, 20, 12) ; (1, 20, 12) ; (2, 20, 12)}.
- (2) τ ={(12, 20, 12)}.
- (3) τ ={(15, 30, 40) ; (15, 30, 40)}.
- (4) τ ={(2, 20, 12) ; (2, 20, 12) ; (2, 20, 12) ; (15, 30, 40)}.
- (5) τ ={(2, 5, 7) ; (3, 7, 11) ; (5, 10, 13)}.
- (6) τ ={(1, 5, 10) ; (1, 13, 18) ; (5, 20, 45) ; (9, 40, 30) ; (7, 45, 32) ; (11, 80, 150) ; (4, 180, 50)}.
- (7) τ ={(2227, 5000, 200000); (1423, 12000, 25000); (420, 14199, 40000);
  (496, 19199, 20000); (552, 19199, 160000); (3096, 50000, 50000);
  (7880, 59000, 59000); (3220, 87199, 800000); (3220, 98399, 100000);
  (1996, 100000, 50000); (520, 100000, 200000); (1990, 193499, 1000000);
  (1120, 197598, 200000); (954, 197598, 2000000); (1124, 198545, 200000);
  (3345, 200000, 200000)}

The following table examines these PNTSs in terms of feasibility, response time and Complexity. More precisely it gives:

- $\lambda = \lambda_n$, the processor Busy period (cf. Appendix A, page 120): size and real cost.

- r_{DM}, the worst-case response time obtained by DM (cf. Theorem 22, page 52).

- cost1, the r_{DM} computation complexity on λ_i (cf. Section 4, page 68).

- r_{EDF}, the worst-case response time obtained by EDF (cf. Theorem 21, page 49).

- cost2, the r_{EDF} computation complexity on λ (cf. Section 4, page 68).

- cost3, the EDF feasibility complexity on λ (cf. Section 4, page 68).

- cost2/cost1, the response time complexity ratio (EDF/DM).

- cost1/cost3, the feasibility complexity ratio (DM/EDF).

TABLE 2. Feasibility, response time and Complexity

τ	λ	r_{DM}	cost1	r_{EDF}	cost2	cost3	cost2/cost1	cost1/cost3
(1)	12 , 29	r_1=3, r_2=7, r_3=8, r_4=9, r_5=10, r_6=12 (SUCCESS)	133	r_1=12, r_2=12, r_3=12, r_4=12, r_5=12, r_6=12, (SUCCESS)	1105	1	8.31	133
(2)	12, 4	r_1=12 (SUCCESS)	13	r_1=12 (SUCCESS)	15	1	1.15	13
(3)	30, 4,	r_1=30 (SUCCESS)	13	r_1=30 (SUCCESS)	15	1	1.15	13
(4)	33, 25	r_1=6, r_2=33 (FAIL)	43	r_1=15, r_2=25 (SUCCESS)	191	46	4.44	0.94
(5)	39, 122	r_1=2, r_2=5, r_3=17 (FAIL)	173	r_1=5, r_2=7, r_3=10 (SUCCESS)	1963	243	11.35	0.71
(6)	147, 342	r_1=1, r_2=2, r_3=7, r_4=17, r_5=26, r_6=83, r_7=87 (FAIL)	683	r_1=1, r_2=2, r_3=7, r_4=24, r_5=29, r_6=64, r_7=87 (SUCCESS)	26092	1601	38.20	0.43
(7)	35502 143	r_1=2227, r_2=3650, r_3=4070, r_4=4566, r_5=5118, r_6=8214, r_7=16094, r_8=19314 r_9=23030, r_{10}=26449 r_{11}=26969, r_{12}=28959 r_{13}=30079, r_{14}=31033 r_{15}=32157, r_{16}=35502 (SUCCESS)	1104	r_1=2227, r_2=3650, r_3=4070, r_4=4566, r_5=5118, r_6=8214, r_7=16094, r_8=19314 r_9=25368, r_{10}=26969 r_{11}=26969r_{12}=29001 r_{13}=33100r_{14}=33100 r_{15}=34047r_{16}=35502 (SUCCESS)	27425	453	24.84	2.44

The following table examines these PNTSs according to several valid norms and the associated efficiency of scheduling algorithms. More precisely it gives:

- $N_{EDF}(\tau)$ the EDF norm on τ (cf. Theorem 13, page 26).

- $N_U(\tau)$, the processor utilization norm on τ (cf. Theorem 13, page 26).

- $N'_U(\tau) = \frac{Max(Max(D_i), Min(T_i))}{Max(D_i)} N_U(\tau)$ the valid norm used by the efficiency theorem on τ (cf. Section 2.2, page 40).

- $\varepsilon(EDF)$, the EDF-efficiency in Σ.

- $\varepsilon(DM) = \alpha_{N_{EDF}}(DM)$, the exact computation of the N_{EDF}-efficiency of DM in Σ (cf. Section 2.1.5.2, page 35, for the algorithmic procedure).

- $\alpha_{N_U}(DM)$, the exact computation of the N_U-efficiency of DM in Σ.

- $\alpha_{N'_U}(DM)$, the exact computation of the N'_U-efficiency of DM in Σ.

- $\frac{Max(Max(D_i), Min(T_i))}{Max(T_i) + Min(D_i)} \frac{Min(D_i)}{Max(D_i)} \leq \alpha_{N'_U}(P) \leq \alpha_{N_{EDF}}(P) = \varepsilon(P)$, a lower bound of the efficiency of any fixed priority scheduling algorithm P in Σ (cf. Section 3.2.2, page 62).

TABLE 3. Valid Norms and efficiency

τ	$N_{EDF}(\tau)$	$N_U(\tau)$	$N'_U(\tau)$	$\varepsilon(EDF)$	$\varepsilon(DM)$	$\alpha_{N_U}(DM)$	$\alpha_{N'_U}(DM)$	lower bound on $\varepsilon(P)$
(1)	1	1	1	1	1	1	1	0.625
(2)	1	1	1	1	1	1	1	0.625
(3)	1	0.75	1	1	1	0.75	1	0.571429
(4)	0.875	0.875	0.875	1	< 0.81667	< 0.775	<0.775	0.33333
(5)	1	0.943057	0.943057	1	< 0.916667	< 0.7272	<0.7272	0.277778
(6)	0.93875	0.93875	0.93875	1	< 0.773561	< 0.46667	<0.46667	0.03226
(7)	0.4454	0.411441	0.411441	1				0.002494

Let us now briefly comment on these tables:

- PNTS(2) is deadline periodic equivalent with PNTS(1) (cf. Definition 9, page 25). It follows that the results concerning feasibility, valid norms and efficiency are the same. The only differences are that the response time of the unique task of PNTS(2) is equal to the maximum worst task response time of PNTS(1) and that the complexity analysis is lower for PNTS(2). According to this simplifying property, we will always aggregate deadline periodic task in the sequel.
  Moreover, PNTS(2) is homogeneous (i.e. contains only one couple (T, D)) and can be seen, for example, as a set of audio streams. To this end, let us consider that the time unit is 1 msec (for example, D=20msec for PNTS(2). This is quite reasonable since for teleconferencing, the maximum lifetime for audio messages is 250 ms [JSS94]). As planned in Section 3.3, page 63, in that case, EDF has no interest since $\varepsilon(DM) = \varepsilon(EDF) = 1$. This result holds whatever P, a fixed priority scheduling algorithm is (DM in our case) and $\forall(\tau \in \Upsilon)$ (i.e. for any task set where $\forall i$, $T_i=12$, $D_i=20$).
  Finally, since the unique deadline (D=20) is greater than the unique period (T=12), we have then $N_{EDF}(\tau) = N_U(\tau) = N'_U(\tau)$ and the feasibility is simply checked by $\sum C_i = C \leq T$ (i.e. $N_U(\tau) \leq 1$).

- PNTS(3) is homogenous too and can be seen, for example, as a set of video streams that we simplify by $\tau = \{(30, 30, 40)\}$ due to the deadline periodic equivalence property. For the same reasons than PNTSs (1) and (2), EDF is out of interest. However, in that case, the unique deadline (D=30) is smaller than the unique period (T=40) following that the feasibility analysis is now simplified by $\sum C_i = C \leq D$ (cf. Section 3.3, page 63) and that $\alpha_{N_U}(DM) \leq \varepsilon(DM) = 1$ since $\forall(\tau \in \Upsilon)$ $N_U(\tau) \leq N_{EDF}(\tau)$. In other words, the fact that N_U is not a finest criterion when deadlines are smaller than the periods, and then not the best measure of the efficiency of DM, is clearly visible on this simple example ($\varepsilon(DM) = 1$ when $\alpha_{N_U}(DM) = 0.75$). On the other hand, in that case where all the deadlines are smaller than all the periods, N'_U, the valid norm that we introduced for the efficiency theorem (cf. Section 3.3, page 63), leads to an easy and more precise evaluation of the efficiency of DM than N_U ($\varepsilon(DM) = \alpha_{N'_U}(DM) = 1$).

- PNTS(4) considers half computation time of PNTSs (2) and (3), i.e., can be seen as a set of voice and video streams. Due to the deadline periodic equivalence property, we simplify it by τ={(6, 20, 12); (15, 30, 40)} = $(PNTS(2) + PNTS(3))/2$. As planned in Section 2.1, page 24, τ is still EDF-feasible.

  On the other hand, τ being heterogeneous, DM must be dominated by EDF (cf. Section 3.3, page 63). This is verified by the efficiency measure, $\varepsilon(EDF) = 1 \geq \varepsilon(DM) = 0.8667$ (in particular τ, is not DM-feasible since $r_2 = 33 > 30$).

  Let us consider now τ'={(6, 20, 12) ; (12, 30, 40)}, a deadline periodic equivalent modification of PNTS(4) that considers a lower computation time for τ_2. τ' becomes DM-feasible and it is remarkable that using N_{EDF} (the finest criterion) this was foreseeable since the computation of $N_{EDF}(\tau') = 0.8$ is smaller than $\varepsilon(DM) < 0.81667$. On the other hand, using the valid norm N_U (that is not a finest criterion), this was not foreseeable since the computation of $N_U(\tau') = 0.8 > 0.775 = \alpha_{N_U}(DM)$ (the same remark holds using the valid norm N'_U since for this PNTSC all the deadlines are not smaller than all the periods, and then $N'_U = N_U$).

- PNTS(5) and PNTS(6) can be seen, for example, as small real time embedded applications (note that PNTS(5) is the example used in the previous sections to illustrate our approach). In both cases the processor utilization is close to 1, $N'_U = N_U$ (since all the deadlines are not smaller than all the periods) and τ is EDF-feasible but not DM-feasible. This last point is not surprising since in both cases $N_{EDF}(\tau)$ (resp. $N_U(\tau)$) is greater than $\varepsilon(DM)$ (resp. $\alpha_{N_U}(DM)$).

  A more general remark, planned in Section 3.3, page 63, and verified by the lower bound on $\varepsilon(P)$, is that PNTS(6) being more heterogeneous than PNTS(5) (see the dispersion in deadlines and periods), the efficiency of DM must be lower. Clearly, this is also verified by the exact computation of $\varepsilon(DM)$ and $\alpha_{N'_U}(DM)$. The drawback is that the lower bound on $\varepsilon(P)$ is a strong worst case.

  Let us recall however (cf. Section 2.2, page 40), that we consider the efficiency theorem more as a fast procedure to evaluate the quality of fixed priority scheduling algorithms than as a tight lower bound (cf. Section 5.2, page 105, for a more detailed discussion on that point).

- PNTS(7) is inspired from [TC95] and [SPU96-2] that describe a hypothetical aircraft control system. Unfortunately, when the number of task is high, the procedure that we proposed to compute exactly the efficiency of DM is no more usable since it becomes too costly (cf. Section 3.2.2, page 62). Let us notice however that the (strong) lower bound on $\varepsilon(P)$ obtained by the efficiency theorem in that case shows that DM seems to be dominated by EDF in terms of efficiency.

- Finally, as said previously, the complexity analysis of PNTS(1) is out of interest since it is deadline periodic equivalent with PNTS(2) but more costly. For the other PNTSs, let us recall that our measure is made on λ_i for DM and λ for EDF (we let the use of λ_i for EDF, established by [GRS96], for further works). On these particular examples it appears that the use of EDF leads to a clear difference, in terms of complexity, if the feasibility alone of the task set is needed and not a full worst-case response time analysis. On the other hand, the unique possible measure of the complexity analysis, considering a fixed priority scheduling algorithm (DM in our case) is intermediate with those obtained, considering EDF. This was planned in Section 2.2, page 40, and Section 4, page 68, and we refer the reader to Section 5.3, page 111, for an illustration of the general asymptotic complexity performance analysis.

Let us now examine in more details how the efficiency vary when the scheduling referential vary.

5.2 Efficiency performances

The goal of this section is to verify if the preliminary results obtained by the application of the efficiency theorem are valids. More precisely, we will compare $\varepsilon(P)$ the exact computation efficiency of some fixed priority scheduling algorithms P (cf. Section 2.1.5.2, page 35) with the lower bound given by our efficiency theorem (cf. Section 3, page 60). Let us recall that this efficiency theorem gives us an underestimation of the efficiency of any fixed priority scheduling algorithm with regard to the dispersion of deadlines and periods. This will be done in presence of some numerical examples where the scheduling referential vary.

First (cf. Figure 14), let us consider a very simple example that considers Υ, a PNTSC characterized by only two couples (T_i, D_i) and where $\forall i$, $D_i = T_i$ (i.e. $N_{EDF}(\tau) = N_U(\tau) = N'_U(\tau)$). Now, let Υ vary according to the dispersion of the periods.

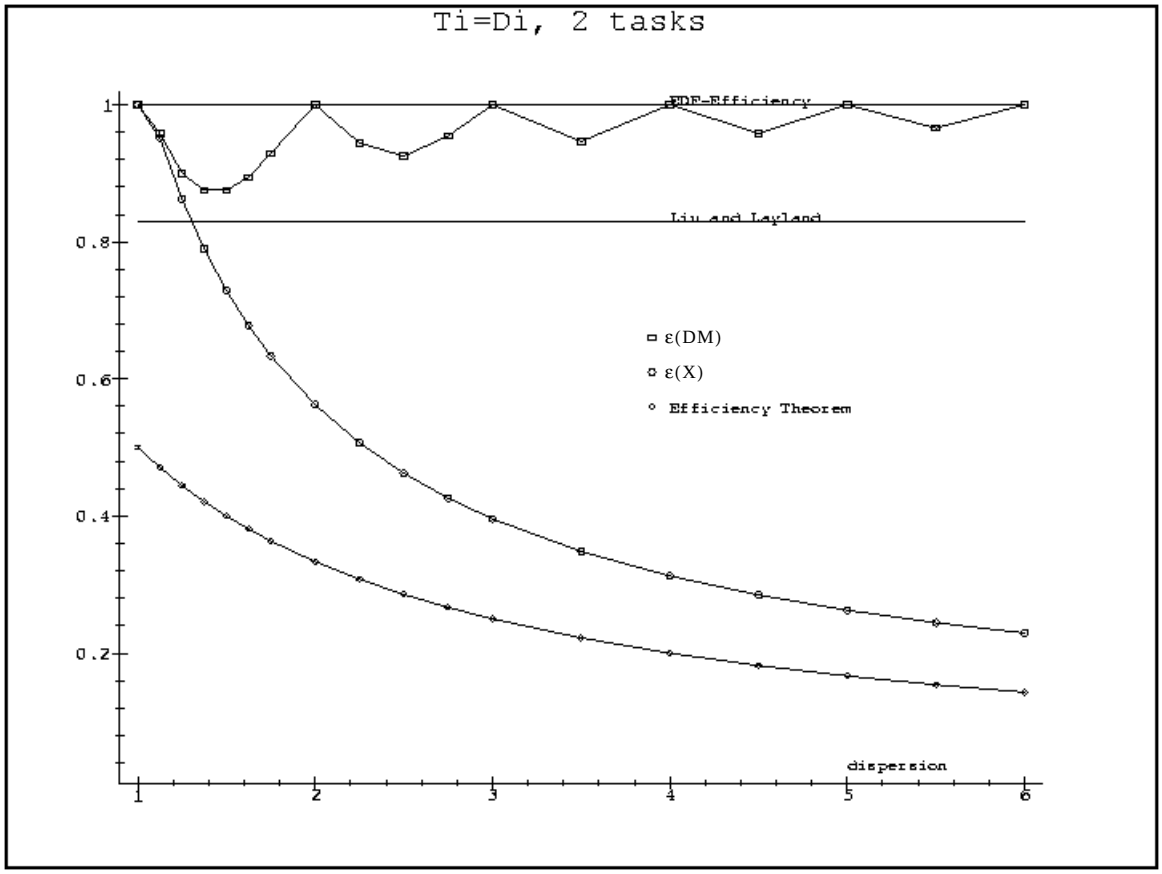


Figure 14: Liu and Layland case, 2 tasks

In Figure 14,we note that:

- The efficiency theorem gives us a lower bound on $\varepsilon(P)$, the efficiency of any fixed priority driven scheduling algorithm P, that is decreasing with the dispersion. This concurs with the theoretical results.

- Let us consider now X, a “bad” fixed priority driven scheduling algorithm where the priority assigned to tasks is proportional to their relative deadlines. Clearly, we verify that $\varepsilon(X)$ decreases with the dispersion and that the efficiency theorem leads to a tight lower bound. This is not surprising since the efficiency theorem is valid whatever the fixed priority scheduling algorithm.

- On the other hand, let us consider now RM (Rate Monotonic, cf. Section 1.2.2.1, page 18) an “optimal” fixed priority driven scheduling algorithm in this context. Clearly, the efficiency theorem is not representative of the behavior of $\varepsilon(RM)$. More precisely, in that particular case, we know that a finer underestimation of $\varepsilon(RM)$ is given by the result of [LL73]. Let us recall, however, that the result of [LL73] is valid only for RM and when $\forall i,\ D_i = T_i$. On the other hand, our efficiency theorem is more general.
  Commenting further using RM, we note that when the periods of the two tasks are multiples of one another, we can see that $\varepsilon(RM)$ is equal to one. This is not surprising from the result of [LEH90] (cf. Section 1.2.2.2, page 18). Indeed, in that case where $\forall i,\ D_i = T_i$ and when the periods are multiple of each other, Theorem 10, page 19, leads to:

$$\forall i \in [1, n],\ r_i = C_i + \sum_{j \in hp(i)} \left\lceil \frac{r_i}{T_j} \right\rceil C_j \le D_i \Rightarrow r_i \le C_i + \sum_{j \in hp(i)} \left(\frac{T_i}{T_j} \right) C_j \le T_i$$

  Therefore $\sum_{j \in hp(i) + i} \frac{C_j}{T_j} \le 1$ => the CNS is $N_u(\tau) \le 1$.

Let us now extend (cf. Figure 15) the context to five couples (T_i, D_i), where $\forall i,\ D_i = T_i$ and where we ensure that the periods are never multiples of each other (except, of course, when Dispersion = 1).

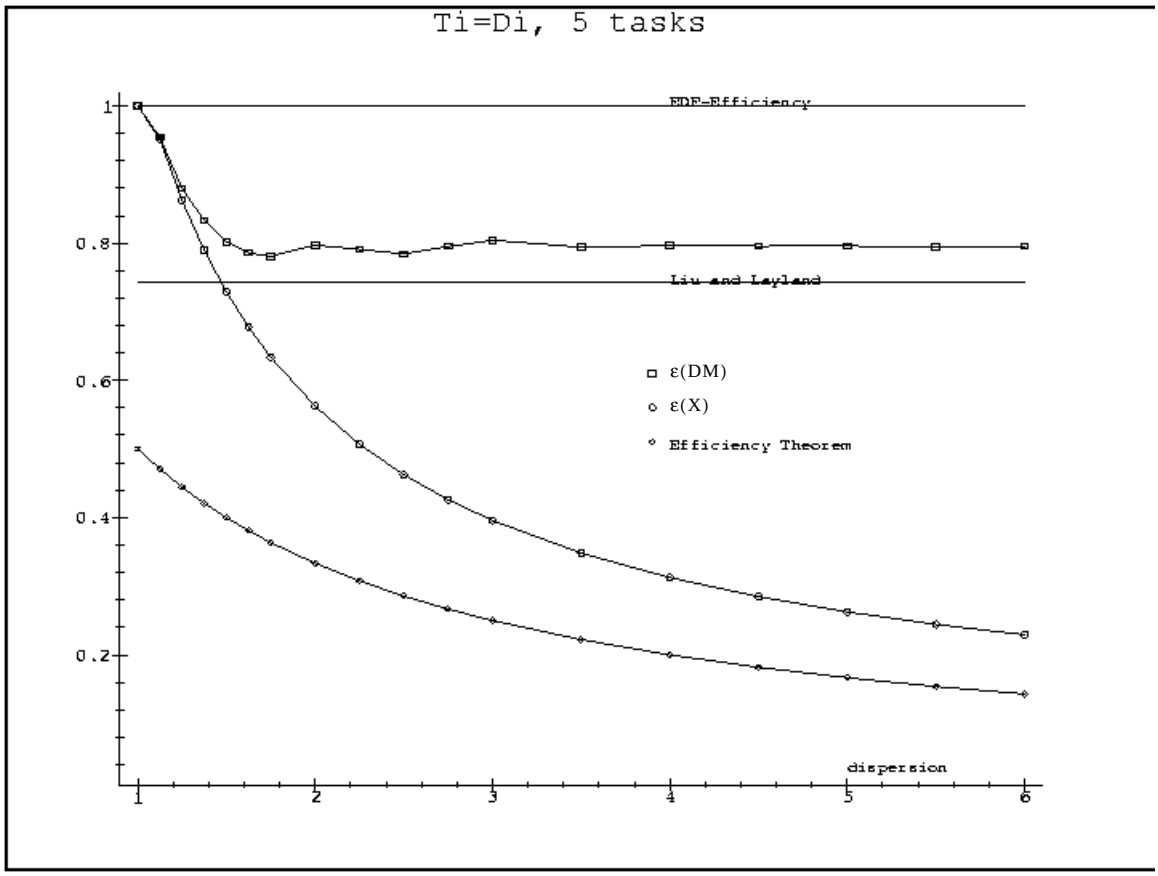


Figure 15: Liu and Layland case, 5 tasks

The conclusions are the same as those in the presence of two couples (T_i, D_i) except on two points. First the underestimation of $\varepsilon(RM)$ given by [LL73] is lower since the number of couples is greater. Second $\varepsilon(RM)$ cannot be equal to one, as seen previously, since the parameters are not multiples of each other (except of course when the dispersion is equal to one).

Using DM only, let us now consider (cf. Figure 16) an example characterized by Υ, a PNTSC where the dispersion in deadlines and in periods is a constant equal to 2. Let us modify Υ increasing or decreasing the deadlines. More precisely, $x = 0$ means that $\forall i, D_i = T_i$, $x < 0$ means that $\forall i, D_i < T_i$ and $x > 0$ means that $\forall i, D_i > T_i$. As a consequence, we either have tasks with deadlines greater than the periods or tasks with deadlines smaller than the periods (but never simultaneously at the same time). Furthermore, $x < -15$ means that all the deadlines are smaller than all the periods and $x > 30$ means that all the deadlines are greater than all the periods (note that in any cases, the lower bound of Liu & Layland given in the following figures has to be considered as an asymptote for the efficiency theorem).

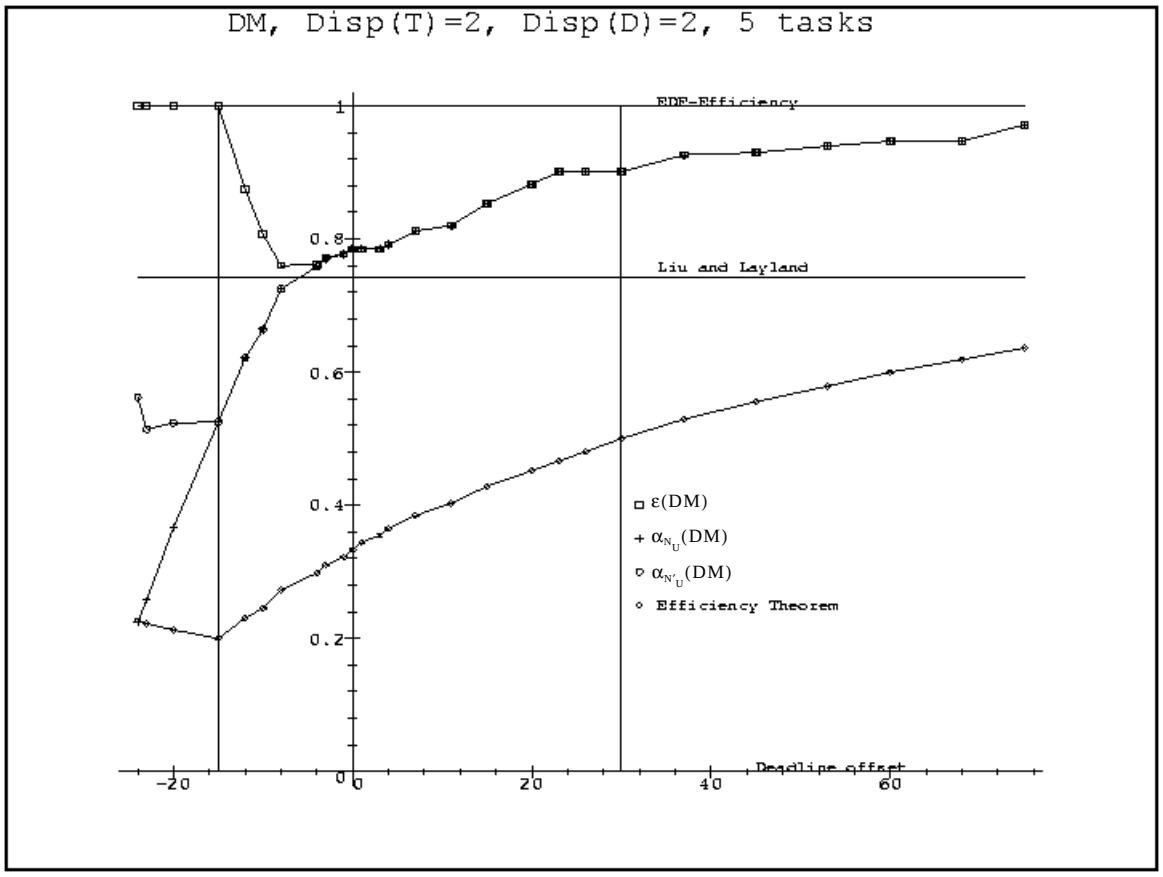


Figure 16: General case, 5 tasks

In Figure 16, we note that:

- When $x \geq 0$ (i.e., $\forall i, D_i \geq T_i$), $N_{EDF}(\tau) = N_U(\tau) = N'_U(\tau)$. Therefore $\varepsilon(DM)$ is equal to $\alpha_{N_{edf}}(DM) = \alpha_{N'_U}(DM) = \alpha_{N_U}(DM)$. Furthermore, as planned in Section 3.3, page 63, the

more the deadlines increase, the more $\varepsilon(DM)$ increase.

- When $x \leq 0$ (i.e., $\forall i, D_i \leq T_i$), $\varepsilon(DM)$ is first at its lowest value when $\forall i$, D_i is close to T_i. After that, when the deadlines continue to decrease, $\varepsilon(DM)$ increases swiftly to become equal to 1 when all the deadlines become smaller than all the periods.
  Considering $\alpha_{N_U}(DM)$, this behavior of $\varepsilon(DM)$ is undetectable. On the other hand $\alpha_{N'_U}(DM)$ is stabilized when all the deadlines become smaller than all the periods. This confirms that N'_U (that is the valid norm used by the efficiency theorem in Section 3, page 60) is a finer approximation of N_{EDF} than N_U in that case.The drawback is that the lower bound on $\varepsilon(P)$ given by the efficiency theorem is a strong worst case in comparison to the real behavior of $\varepsilon(DM)$ (as said previously, we leave the question of finding a fast computation of the exact efficiency or a finer efficiency theorem for further works). Let us recall, from Section 3.3, page 63, that we consider the efficiency theorem as a fast procedure to evaluate the quality of fixed priority scheduling algorithms but we also identify its limitation since it simplifies any PNTSC according to the dispersion and it does not enable us to distinguish between several fixed priority driven scheduling algorithms.

In order to illustrate that particular point, let us now consider (cf. Figure 17) the same example but using X, a “bad” fixed priority scheduling algorithm where the priority assigned to tasks is proportional to their relative deadlines, instead of DM.

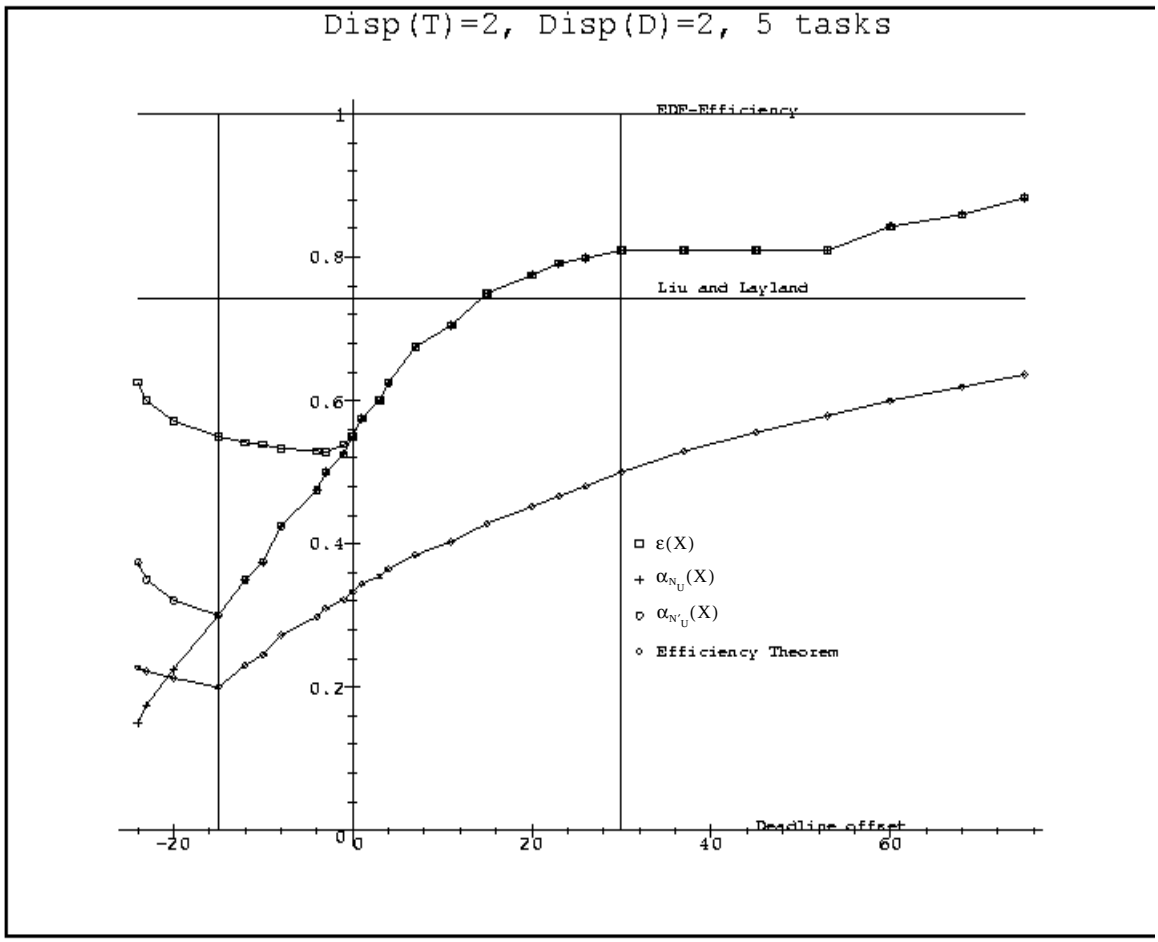


Figure 17: General case, 5 tasks

The conclusions are the same as in presence of DM except that the measures of $\varepsilon(X)$, $\alpha_{N'_U}(X)$ and $\alpha_{N_U}(X)$ are lower than for DM (especially on the left side of the figure). This is not surprising since, from Section 1.2.2.1, page 18, X is a rather "bad" fixed priority scheduling algorithm in comparison to DM. The important point is that the lower bound given by the efficiency theorem is still valid (since it is valid whatever the fixed priority scheduling algorithm) and is a tighter lower bound in that case.

Using DM again, let us now consider (cf. Figure 18) a similar example where the dispersion in periods is still a constant but where the dispersion in deadlines increases on the left side of the figure.

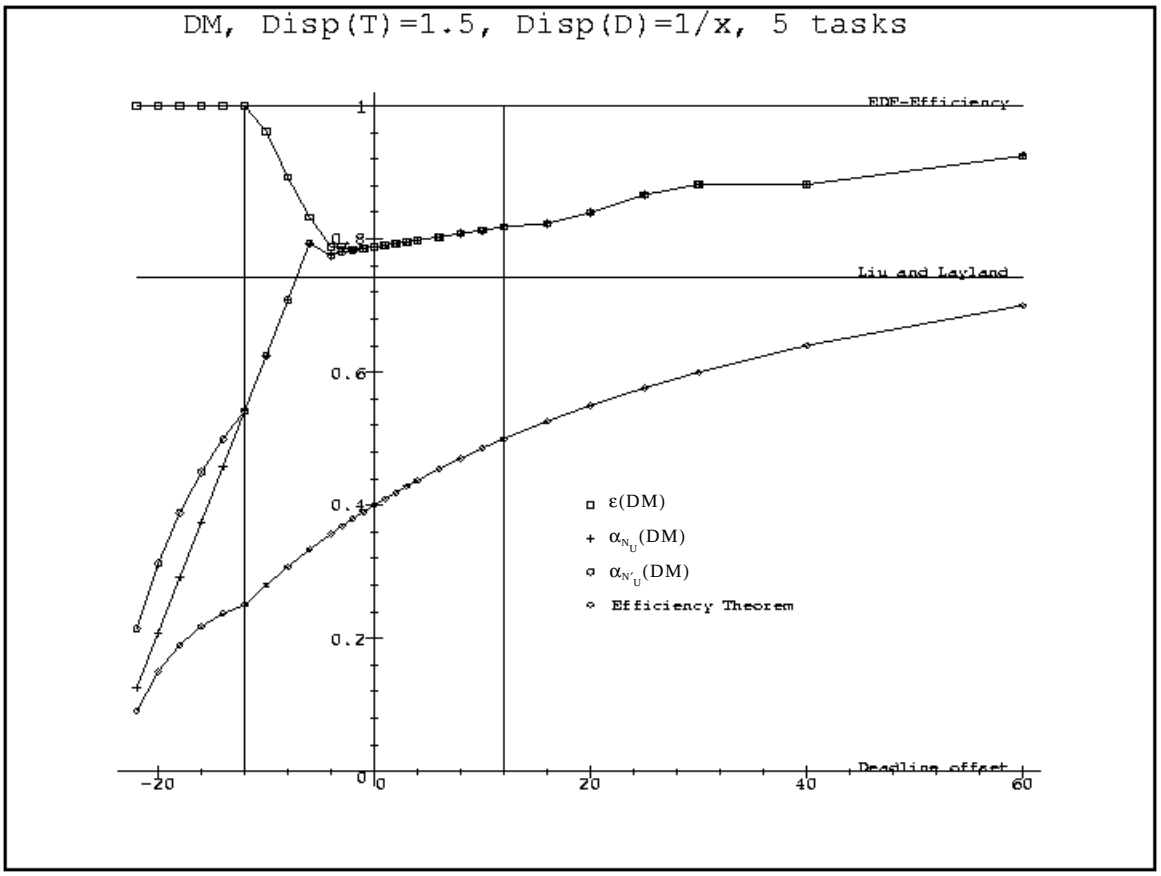


Figure 18: General case, 5 tasks

The conclusions are the same as for Figure 16, except that, due to the greater dispersion of the deadlines to the left side of the figure, the measure of $\alpha_{N_U}(DM)$, $\alpha_{N'_U}(DM)$ and the lower bound of $\varepsilon(P)$ given by the efficiency theorem decreases more swiftly in that case. This was planned in Section 3.3, page 63, but is not verified by $\varepsilon(DM)$, the exact computation of the efficiency of DM. More precisely, at this time, we do not know if this is example dependent or if our efficiency theorem is a too strong approximation of N_{EDF} to handle the behavior of $\varepsilon(DM)$ in that case. As said previously, we leave the question of finding a better efficiency theorem for further works.

Finally, let us consider now (cf. Figure 19) a more general case where, from Section 1.2.1.1, page 12, DM is not an "optimal" fixed priority scheduling algorithm. To that end, let us consider a scheduling referential with Υ a general PNTSC where periods and deadlines are not related. More precisely, let us arbitrarily set a constant dispersion for the periods equal to 8 and a constant dispersion for the deadline

equal to 1 (meaning that all the deadlines are equal to a unique value D). Let us now modify Υ increasing or decreasing the value of D. $x < -70$ means that D is smaller than all the periods, $x = 0$ means that D is equal to the average of the periods and $x > -70$ means that D is greater than all the periods. As a consequence, we never have $\forall i, D_i = T_i$ and during (-70, 70), we are in presence of tasks having their deadlines greater than their periods and at the same time of tasks having their deadlines smaller than their periods).

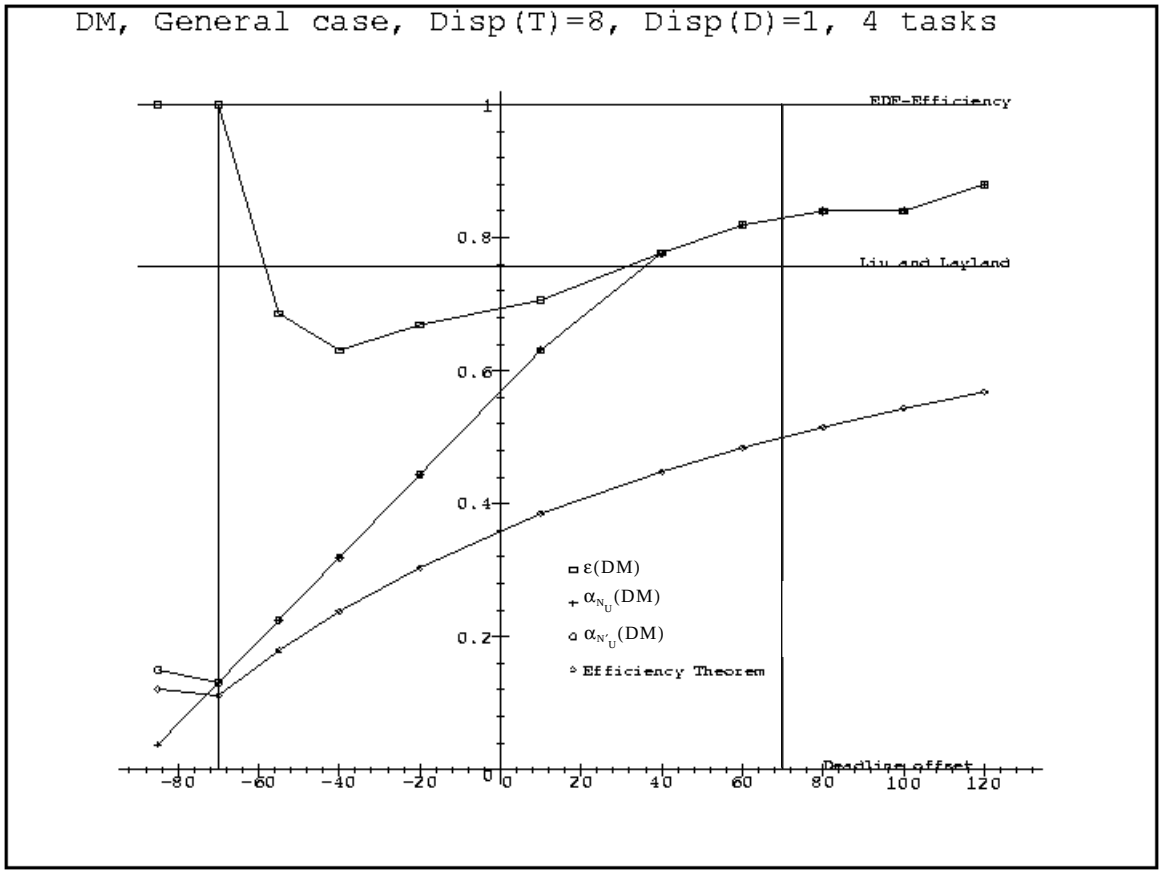


Figure 19: General case, 4 tasks

In Figure 19, we note that $\varepsilon(DM)$, the exact computation of the efficiency of DM, varies:

- when $x \geq 0$, the more D increases, the more $\varepsilon(DM)$ increases. This is verified by the lower bound given by the efficiency theorem and concurs with the theoretical results. Moreover, since x = 0 do not represent the case, $\forall i, D_i \geq T_i$, we do not immediately have $N_{EDF}(\tau) = N_U(\tau) = N'_U(\tau)$. After a while, however, $\varepsilon(DM)$ becomes equal to $\alpha_{N_{edf}}(DM) = \alpha_{N'_U}(DM) = \alpha_{N_U}(DM)$.

- during the interval [-50, 40], $\varepsilon(DM)$ is at its lowest value that is below the lower bound established by [LL73]. This seems to indicate that EDF dominates DM in the general case which is not surprising for two reasons. The first one is obvious since the lower bound of [LL73] is not valid for that case. The second one is in some ways subtler since we do not even know if DM is the best fixed priority driven scheduling algorithm in the general case. Indeed, let us recall from the state of the art (cf. Section 1.2.2.1, page 18) that in the general case, the optimal fixed priority assignment is obtained by the Audsley procedure [AUD91] that is in $O(n^2)$, an extra computation cost.

- when $x \leq -50$, $\varepsilon(DM)$ increases swiftly to become equal to 1 when x=-70, i.e. when D, the unique deadline, becomes smaller than all the periods. That is not obvious considering the lower

bound given by the efficiency theorem. In fact, in that case, this efficiency theorem gives us a strong underestimation of $\varepsilon(DM)$. However, note that it stops decreasing when x=-70 meaning that $\varepsilon(DM)$ cannot be equal to zero. Once more, the reason is because of the use of the valid norm N'_U instead of N_U that is a better approximation of the finest criterion when all the deadline are smaller than the periods

To conclude this part, we have compared $\varepsilon(P)$, the exact computation of the efficiency of any fixed priority scheduling algorithms P with the theoretical results planned by our efficiency theorem. In particular, we have verified that the efficiency of any fixed priority scheduling algorithm has a lower bound that is not equal to zero and that the more the task set is homogeneous or the more all the deadlines are greater (or smaller) than all the periods, the more DM is close to EDF in terms of efficiency. In other words, the more the task set is heterogeneous or the more the PNTSC is in the general case, the more EDF dominates fixed priority scheduling algorithms.
On the other hand, the limitations of our comparison are first that these results have been established on relatively small PNTSC (since the procedure that we have established in Section 2.1.5, page 32, to compute exactly the efficiency of scheduling algorithm is too costly when the number of couples (T_i, D_i) is large). Second, we have confirmed that our efficiency theorem is a straight, but limited, application of Section 2.1, page 24. In particular, it need to be improved to hold more clearly the distance of the optimality of a specific fixed priority driven scheduling algorithm. To that end, let us recall that we have proposed in Section 3.4, page 66, a general method to establish finer efficiency theorems.

In order to complete the performance analysis, let us now examine in more details the complexity of the associated feasibility conditions.

5.3 Computational Complexity

The purpose of this section is to illustrate our complexity results with a numerical example. An asymptotic analysis of the computational complexity of the feasibility tests associated to Fixed Priority based scheduling algorithms and EDF is given in Section 4, page 68. This asymptotic analysis is done under the following assumptions:

- Theorem 22, page 52:
  $\lambda_i = \lambda$,
- Theorem 23, page 54:
  $A_i^{\dagger} = \bigcup_j A_{i,j}^{\dagger}$, $A_{i,j}^{\dagger} = A_{i,j} \cap [0, \lambda[$, $A_{i,j} = \{D_j + k_j T_j - D_i,\ k_j \in Z\}$,
  $$A_{i,j}^{\dagger} = \left\{ D_j + k_j T_j - D_i, \left\lceil \frac{D_i - D_j}{T_j} \right\rceil \le k_j \le \left\lceil \frac{\lambda + D_i - D_j}{T_j} \right\rceil - 1 \right\}.$$

The reasons why these assumptions are made are given at the beginning of Section 4, page 68. Roughly speaking, these assumptions are made in order to compare the complexity of the optimized feasibility tests at a level of refinement where the comparison appears possible. Here, we will not take into account of these assumptions. In other words, we will only consider the optimized feasibility tests at their finest level of refinement as they are given at the end of the complexity framework (cf. Section 2.2, page 40). We will see that the asymptotic analysis always holds.

We will consider the following example:

$$\tau = \sum_{1 \le i \le 100} C_i \cdot e_{T_i, D_i}, \text{ where:}$$

- $C_i = \text{ithprime}(1 + ((i - 1) \bmod 4))$, that is:
  $C_{4k+1} = 2$, $C_{4k+2} = 3$, $C_{4k+3} = 5$, $C_{4k+4} = 7$, $k \geq 0$,
- $T_i = D_i = \text{ithprime}(i + 45)$, that is:
  $\{199, 211, 223, 227, \ldots, 821, 823, 827, 829\}$.

This example is described in depth in Appendix C, page 131.

We will consider the following scheduling algorithms: HFP/DM (Highest Priority First / Deadline Monotonic) and EDF (Earliest Deadline First).

Let us compute the processor utilization norm. We have: $N_u(\tau) = \sum_{1 \leq i \leq 100} \frac{C_i}{T_i} = 96.5\%$.

Since we are in a "Liu and Layland" case ($T_i = D_i$), we have: $N_{EDF}(\tau) = N_U(\tau)$. It follows that τ is EDF feasible. In the sequel, we will see that τ is not HPF/DM feasible.

The computational complexity of the feasibility tests based upon calculation of the worst-case response time of a task for both HPF/DM and EDF are given in Appendix C, page 131, for every task set $\tau_{[1,k]} = \sum_{1 \leq i \leq k} \tau_i$, $1 \leq k \leq 100$.

■ Highest Priority First / Deadline Monotonic

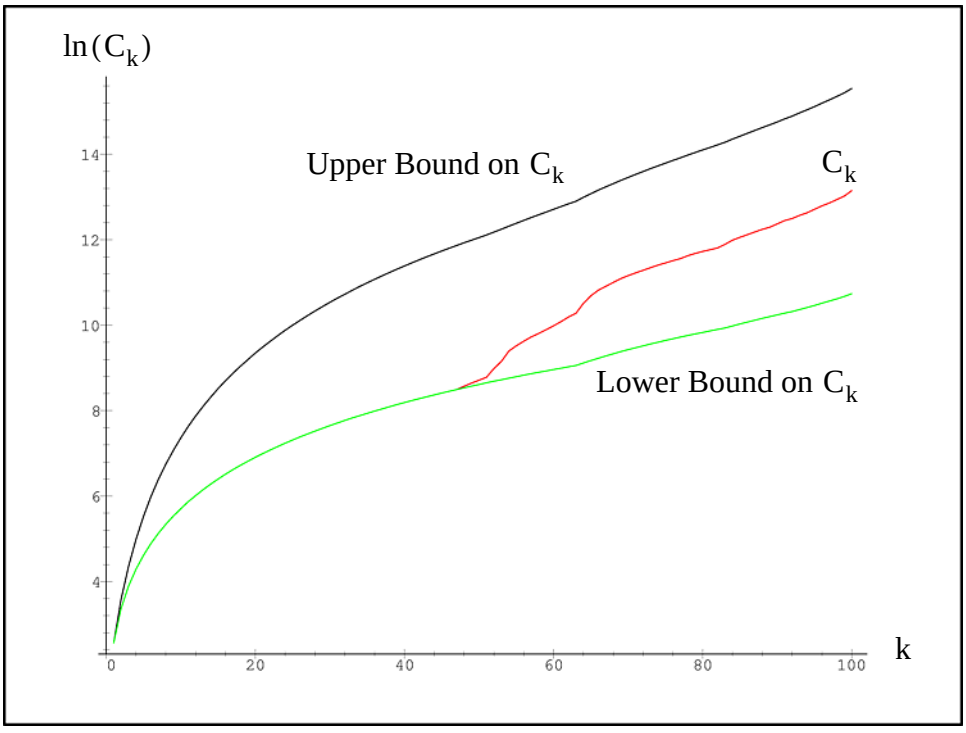


Figure 20: Lower Bound on C_k, Real Overall Cost C_k, Upper Bound on C_k

Let α_k = Lower Bound on C_k and β_k = Upper Bound on C_k. We can distinguish three cases:

Case I : $1 \leq k \leq 47$

In that case, all the task sets $\tau_{[1,\,k]} = \sum_{1 \leq i \leq k} \tau_i$ are HPF/DM feasible.

Furthermore, we have the following property (cf. Appendix C, page 131):

$$\forall k,\ 1 \leq k \leq 47,\ \lambda_k \leq \mathrm{Min}_{i \leq k}\{T_i\}, \text{ which is equivalent to: } \lambda_k = \sum_{i \leq k} C_i .$$

This is the reason why $\alpha_k = C_k$ for every k, $1 \leq k \leq 47$. Indeed, we have:

$$Q_k = 0,\ r_k = r_{k,\,0} = w_{k,\,0} = w_{k,\,Q_k} = \lambda_k = \sum_{i \leq k} C_i, \text{ and } r_k \leq D_k .$$

Case II : $48 \leq k \leq 63$

In that case, all the task sets $\tau_{[1,\,k]} = \sum_{1 \leq i \leq k} \tau_i$ are HPF/DM feasible. We have: $\alpha_k \leq C_k \leq \beta_k$, $r_k = \mathrm{Max}_{0 \leq q \leq Q_k}\{r_{k,\,q}\}$, where: $r_{k,\,q} = w_{k,\,q} - qT_k$, and $r_k \leq D_k$.

Case III : $64 \leq k \leq 100$

In that case, all the task sets $\tau_{[1,\,k]} = \sum_{1 \leq i \leq k} \tau_i$ are not HPF/DM feasible. We have: $\alpha_k \leq C_k \leq \beta_k$ $r_k = \mathrm{Max}_{0 \leq q \leq Q_k}\{r_{k,\,q}\}$, where: $r_{k,\,q} = w_{k,\,q} - qT_k$, and $r_k > D_k$.

■ **Earliest Deadline First**

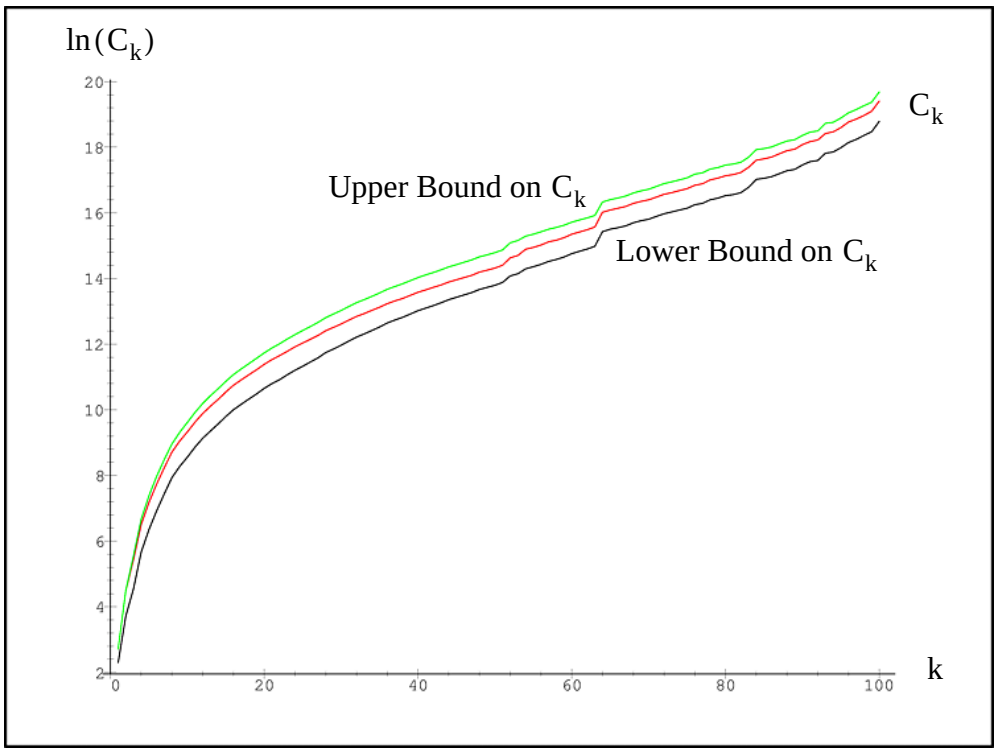


Figure 21: Lower Bound on C_k, Real Overall Cost C_k, Upper Bound on C_k

Let γ_k = Lower Bound on C_k and δ_k = Upper Bound on C_k. We have:

$$1.5 \leq \mathrm{Min}_{1 \leq k \leq 100}\left\{\frac{\delta_k}{\gamma_k}\right\} \leq \frac{\delta_k}{\gamma_k} \leq \mathrm{Max}_{1 \leq k \leq 100}\left\{\frac{\delta_k}{\gamma_k}\right\} \leq \mathbf{2.98}$$

This result is not surprising at all (cf. Section 4.2.2.2, page 84). Furthermore, we have:

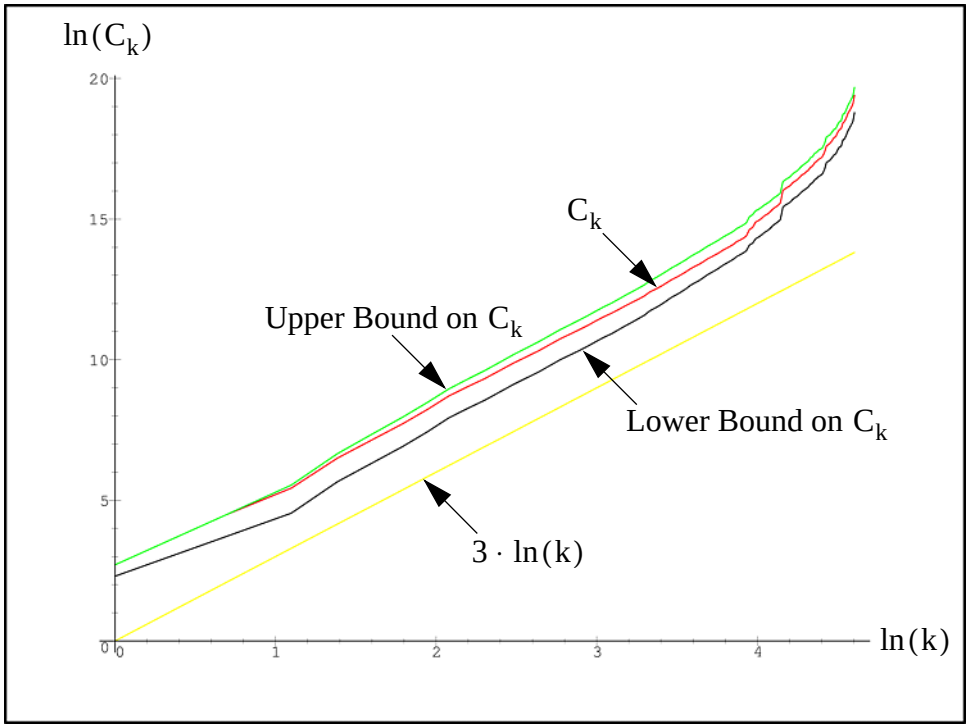


Figure 22: Lower Bound on C_k, Real Overall Cost C_k, Upper Bound on C_k

$$\gamma_k = \Omega(k^3) \text{ and } \delta_k = O(k^3)$$

This result is not surprising at all (cf. Section 4.2.2.2, page 84).

Let $\varphi(k) = \frac{1}{k} \cdot \frac{C_{EDF,\,(\forall i \in [1,\,k],\,r_i \leq D_i),\,k}}{C_{FP,\,(\forall i \in [1,\,k],\,r_i \leq D_i),\,k}}$. We have:

$$1.15 \leq \mathrm{Min}_{1 \leq k \leq 100}\{\varphi(k)\} \leq \varphi(k) \leq \mathrm{Max}_{1 \leq k \leq 100}\{\varphi(k)\} \leq 5.78$$

This result is not surprising at all (cf. Section 4.2.3.2, page 90). We know that:

$$\frac{C_{\mathbf{EDF},\,(\forall i \in [1,\,k],\,r_i \leq D_i),\,k}}{C_{\mathbf{FP},\,(\forall i \in [1,\,k],\,r_i \leq D_i),\,k}} = O(k)$$

This is illustrated on the following figure:

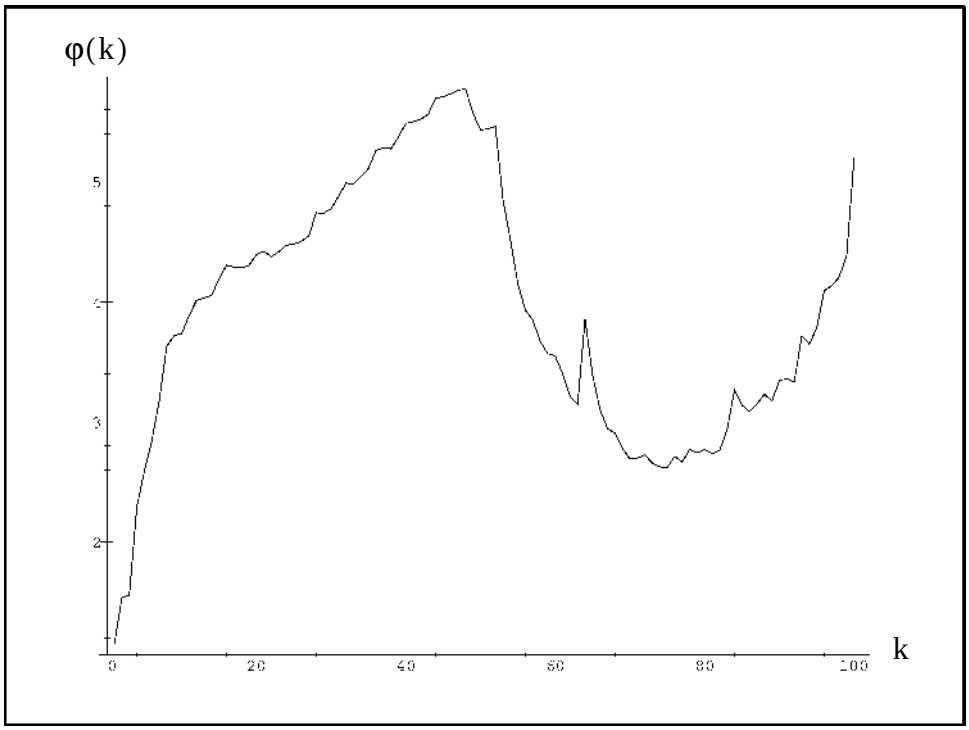


Figure 23: $\varphi(k) = \frac{1}{k} \cdot \frac{C_{EDF, (\forall i \in [1, k], r_i \leq D_i), k}}{C_{FP, (\forall i \in [1, k], r_i \leq D_i), k}}$

6 Synthesis

Let us now summarize our results concerning the efficiency (resp. complexity) comparison established in Section 3, page 60 (resp. Section 4, page 68) in the following cases.

■ Homogeneous case: $\forall i \in [1, n], T_i = T, D_i = D$

- $\varepsilon(P)$, the efficiency of any Fixed Priority based scheduling algorithm P, is equal to $\varepsilon(EDF)$, namely: $\varepsilon(P) = \varepsilon(EDF) = 1$ (cf. Section 3.3, page 63),
- $\frac{C_{FP, (\forall i \in [1, n], r_i \leq D_i)}}{C_{EDF, (\forall t \in S, P(t))}} = \frac{C_{EDF, (\forall i \in [1, n], r_i \leq D_i)}}{C_{FP, (\forall i \in [1, n], r_i \leq D_i)}} = 1$ (cf. Section 4.3.1.3, page 95).

In that case, the feasibility tests are the same both for any fixed priority based scheduling algorithms P and EDF. In other words, for homogeneous PNTSCs, fixed priority based scheduling algorithms are Σ-optimal and their efficiency is equal to one. Clearly, in that case, EDF has no interest.

■ General case (i.e., when periods are not related to deadlines)

- the lower bound on $\varepsilon(P)$, the efficiency of any fixed priority based scheduling algorithm P, is underestimated by:

  $\frac{Max(Max(D_i), Min(T_i))}{Max(T_i) + Min(D_i)} \frac{Min(D_i)}{Max(D_i)}$ (cf. Theorem 25, page 62, and Section 3.3, page 63)

- the ratio $\frac{C_{FP, (\forall i \in [1, n], r_i \leq D_i)}}{C_{EDF, (\forall t \in S, P(t))}}$ is bounded by:

  - Case I: $\lambda < Max_j\{D_j\}$,

    $\frac{\alpha}{\beta'} \leq \frac{C_{FP, (\forall i \in [1, n], r_i \leq D_i)}}{C_{EDF, (\forall t \in S, P(t))}} \leq \frac{\beta}{\alpha'}$ where: $\frac{\alpha}{\beta'} = \Omega(1)$ and $\frac{\beta}{\alpha'} = O(n^2)$,
    (cf. Theorem 26, page 88),

  - Case II: $\lambda \geq Max_j\{D_j\}$,

    $\frac{\alpha}{\beta'} \leq \frac{C_{FP, (\forall i \in [1, n], r_i \leq D_i)}}{C_{EDF, (\forall t \in S, P(t))}} \leq \frac{\beta}{\alpha'}$ where: $\frac{\alpha}{\beta'} = \Omega(1)$ and $\frac{\beta}{\alpha'} = O(n)$,
    (cf. Theorem 27, page 89),

- the ratio $\frac{C_{EDF, (\forall i \in [1, n], r_i \leq D_i)}}{C_{FP, (\forall i \in [1, n], r_i \leq D_i)}}$ is bounded by:

$$\frac{\gamma}{\beta} \leq \frac{C_{EDF,\ (\forall i \in [1, n],\ r_i \leq D_i)}}{C_{FP,\ (\forall i \in [1, n],\ r_i \leq D_i)}} \leq \frac{\delta}{\alpha} \text{ where: } \frac{\gamma}{\beta} = \Omega(1) \text{ and } \frac{\delta}{\alpha} = O(n),$$

(cf. Theorem 28, page 90).

In that case, due to the simplification of any PNTSC according to the dispersion of the task parameters, there is no clear interpretation of the efficiency theorem. It only gives a lower bound of $\varepsilon(P)$ that is not equal to zero but might be pessimistic. Moreover, the efficiency performances analysis that we made in Section 5.2, page 105, shows that $\varepsilon(P)$ is dominated by $\varepsilon(EDF)$ in this general case. This last point is not clear according to our efficiency theorem and we leave for further works the possible refinements of our analysis in that case.
On the other hand, the complexity analysis is slightly in favor of fixed (resp. dynamic) priority driven algorithms when the worst-case response time (resp. feasibility only) information is needed. This is the case in particular when the worst-case time computation is part of a holistic analysis on a distributed system.

■ $\{D_i\} \gg \{T_i\}$ case (i.e., when all the periods are supposed to be dominated by all the deadlines)

- we know from Section 2.2, page 40, and Section 3.3, page 63, that $\alpha_{N_U}(P) = \alpha_{N'_U}(P) = \varepsilon(P)$. Moreover, the lower bound on $\varepsilon(P)$, the efficiency of any fixed priority scheduling algorithm P, is underestimated by (cf. Section 3.3, page 63): $\frac{Min(D_i)}{Max(T_i) + Min(D_i)}$.

In that case, the bigger MIN (D_i) is compared to MAX(T_i), the higher and closer to one the efficiency of P is. This seems intuitively normal since when the deadlines are big, the choice of a particular policy is not so important.

■ Case $\forall i \in [1, n], D_i \geq T_i$

- we just know from Section 2.2, page 40, and Section 3.3, page 63, that $\alpha_{N_U}(P) = \alpha_{N'_U}(P) = \varepsilon(P)$.

In that case, due to the simplification of any PNTSC according to the dispersion of the task parameters, there is no clear interpretation of the efficiency theorem. We can just conjecture that the behavior of the efficiency of any fixed priority scheduling algorithm P might be intermediate between the indications given by the general case and the $\{D_i\} \gg \{T_i\}$ case.

■ Liu and Layland's case: $\forall i, D_i = T_i$.

- we know from Section 2.2, page 40, and Section 3.3, page 63, that $\alpha_{N_U}(P) = \alpha_{N'_U}(P) = \varepsilon(P)$. Moreover, the lower bound on $\varepsilon(P)$, the efficiency of any fixed priority scheduling algorithm P, is underestimated by (cf. Section 3.3, page 63): $\frac{Min(T_i)}{Max(T_i) + Min(T_i)} = \frac{1}{\delta T + 1}$, where

δT represents the dispersion of the periods in the PNTSC.
Note that, according to Theorem 9, page 18, Liu and Layland proved in that case that $U \leq n\left(2^{\frac{1}{n}} - 1\right)$ is a sufficient condition when the scheduling algorithm is rate-monotonic (RM) and that $U \leq 1$ is a necessary and sufficient condition for EDF. It also means that $\alpha_{N_U}(RM)$, the efficiency of RM, is underestimated by $n\left(2^{\frac{1}{n}} - 1\right)$ that is a finer lower bound on $\varepsilon(RM)$ that the one given by our efficiency theorem.

In that case, if we consider that n, the number of periodic tasks in the PNTSC, can also be interpreted as a kind of measure of the dispersion, it is remarkable to see that the efficiency theorems and Liu and Layland lower bound give us an underestimation which is decreasing with the dispersion.

■ **Case** $\forall i \in [1, n]$, $D_i \leq T_i$

- we just know from Section 2.2, page 40, and Section 3.3, page 63, that $\alpha_{N_U}(P) \leq \alpha_{N'_U}(P) \leq \varepsilon(P)$.

- see the general case and the $\{D_i\} \ll \{T_i\}$ case for the efficiency behavior.

In that case, due to the simplification of any PNTSC according to the dispersion of the task parameters, there is no clear interpretation of the efficiency theorem. We can just conjecture that the behavior of the efficiency of any fixed priority scheduling algorithm P might be intermediate between the indications given by the general case and the $\{D_i\} \ll \{T_i\}$ case.

■ $\{D_i\} \ll \{T_i\}$ case (i.e., when all the deadlines are supposed to be dominated by all the periods)

- we know from Section 2.2, page 40, and Section 3.3, page 63, that $\alpha_{N_U}(P) \leq \alpha_{N'_U}(P) \leq \varepsilon(P)$. Moreover, the lower bound on $\varepsilon(P)$, the efficiency of any fixed priority scheduling algorithm P, is underestimated by (cf. Section 3.3, page 63): $\frac{Min(T_i)}{Max(T_i) + Min(D_i)} \frac{Min(D_i)}{Max(D_i)}$. Therefore, when $MIN(D_i)$ becomes negligible in comparison to $MAX(T_i)$, $\varepsilon(P)$ is underestimated by $\frac{Min(T_i)}{Max(T_i)} \frac{Min(D_i)}{Max(D_i)} = \frac{1}{\delta T \delta D}$ where δT (resp. δD) represent the dispersion of the periods (resp. of the relative deadlines) of the PNTSC.

In that case, the efficiency theorem gives us the indication that the more the dispersion in periods or in deadlines decreases, the more the lower bound on $\varepsilon(P)$ increases. Moreover, the efficiency performance analysis that we made in Section 5.2, page 105, shows that $\varepsilon(P)$ increases swiftly when $MAX(D_i)$ decreases. This last point is not clear according to our efficiency theorem and we leave for further works the possible refinements of our analysis.

7 Conclusion

In this paper, we have initiated a comparison of uniprocessor, preemptive, fixed and dynamic priority driven scheduling algorithms. To this end, we saw that a lot of results such as optimality, feasibility, worst-case response time were already proved, but also that only a few things had been established to compare fixed versus dynamic algorithms. Then, as a first step, we proposed a general framework to compare scheduling algorithms, using two criteria (cf. Section 2, page 24):

- $\varepsilon(P)$, the efficiency of any given scheduling algorithm $P \in \Pi$ in a given periodic scheduling referential $\Sigma=(\Upsilon,\Pi)$ where Υ is a PNTSC (a class of non-concrete task sets) and Π is a set of non-idling preemptive scheduling algorithms.
  $\varepsilon(P)$ was formally defined in Section 2.1.3, page 28, and can be seen as the maximum of the f-efficiencies, where f is in the set of valid norms for the vector space $(\Upsilon,+,\cdot)$.
- the complexity of the existing fixed and dynamic feasibility tests in terms of basic operations (such as addition, multiplication and functions). To achieve that purpose, we showed that it was possible to make these tests consistent, i.e., tractable and optimized in the same way. In particular, the concept of busy period was examined in details.

As a second step, we initiated the comparison of fixed versus dynamic priority driven scheduling algorithms in terms of efficiency (cf. Section 3, page 60) and complexity (cf. Section 4, page 68). More precisely, considering several kinds of traffics, we proposed a straight, but limited (in the sense that many refinements or other results might be derived from Section 2, page 24) application of our general framework. It appears notably that, except for the implementation and market criteria, the choice of a scheduling algorithm for a given real-time problem is more a question of scheduling context (for which efficiency and complexity criteria are relevant to find the right trade-off). As a consequence, rather than drawing a general conclusion stating finally what is the best scheduling algorithm, we would prefer to give a short list of pragmatic pieces of advice:

- 1: from a complexity point of view, if the worst-case response time information is needed, then it is advantageous, but not a determining factor, to use fixed priority schedulers. This is the case in particular when the worst-case time computation is part of a holistic analysis on a distributed system.
- 2: from an efficiency point of view, advice given in 1 is even truer in presence of Deadline Monotonic when the traffic class is homogeneous in terms of deadlines and periods and/or when the deadlines are much bigger than the periods. This is the case, for example, if we consider a voice and image traffics as might be used in large distributed systems.
- 3: on the other hand, from an efficiency point of view, EDF or other deadline driven scheduling algorithms should be used when the traffics are heterogeneous and/or when the deadlines might be arbitrarily smaller or greater than the periods. This is even truer, from a complexity point of view, if the individual worst-case response times of the tasks are not needed. This is the case, for example, if we consider a traditional real-time problem as might be used for embedded systems.

Note that these results seem intuitively natural since any scheduling decision is more critical if the pending activations differ one from the other or if the busy periods contain several activations and absolute deadline of the tasks. Finally, although we are fully aware that a lot of work remains to be done on that subject, we think that our framework provides a good basis for unifying existing state-of-the-art results and for introducing new ones. In particular, we used it in order to start a comparison between several uniprocessor preemptive scheduling algorithms. Our hope is that the framework that we propose might be general enough (1) to refine our preliminary results, (2) to define and evaluate the efficiency of scheduling algorithms in terms of response time (cf. (Q2), Section 1.3, page 21) and (3) to be extended in other scheduling contexts such as non-preemptive scheduling, transactional and/or distributed systems, admission-control mechanisms.

Appendix A

In this appendix, we study the sequence $\{\lambda_i^{(k)}\}_{k \geq 0}$ defined by the following recursive equation:

$$\lambda_i^{(k+1)} = \sum_{j \leq i} \left\lceil \frac{\lambda_i^{(k)}}{T_j} \right\rceil C_j, \ \lambda_i^{(0)} = 1$$

where $\forall j,\ 1 \leq j \leq i,\ (C_j, T_j) \in \mathbb{N}^* \times \mathbb{N}^*$ and where $\rho_i = \sum_{j \leq i} \frac{C_j}{T_j}$ is less than or equal to one.

We define the $W_i(t)$ function as $\sum_{j \leq i} \left\lceil \frac{t}{T_j} \right\rceil C_j$ for any integer t.

This appendix is organized in three sections as follows:

- in section 1, we show by induction that the sequence $\{\lambda_i^{(k)}\}_{k \geq 0}$ is increasing, bounded, therefore convergent. To this end, we demonstrate Lemma 1 and Lemma 2.
- in section 2, we show that $\lim_{k \to +\infty} \lambda_i^{(k)}$ is bounded by $\mathrm{Min}\left\{\rho_i \cdot \mathrm{LCM}_{l \leq i}\{T_l\}, \frac{1}{1-\rho_i} \cdot \sum_{j \leq i} C_j\right\}$.
  To this end, we consider the set of the positive roots of $\lambda_i = W_i(\lambda_i)$.
- in section 3, we provide a practical way of solving the recursive equation $\lambda_i = W_i(\lambda_i)$ by successive iterations and we express its computational complexity, which is pseudo-polynomial in $O(i^2)$.

A.1 The sequence $\{\lambda_i^{(k)}\}_{k \geq 0}$ is increasing, bounded, therefore convergent.

We show by induction that the sequence $\{\lambda_i^{(k)}\}_{k \geq 0}$ is increasing, bounded, therefore convergent.
To this end, we demonstrate the following lemmata.

Lemma 1 - *If* $\lambda_i^{(k+1)} \geq \lambda_i^{(k)}$ *then* $\lambda_i^{(k+2)} \geq \lambda_i^{(k+1)}$.

By assumption, we have $\lambda_i^{(k+1)} \geq \lambda_i^{(k)}$. It follows that $\left\lceil \frac{\lambda_i^{(k+1)}}{T_j} \right\rceil \geq \left\lceil \frac{\lambda_i^{(k)}}{T_j} \right\rceil$ for every $j,\ 1 \leq j \leq i$.

Hence, $\lambda_i^{(k+2)} \geq \lambda_i^{(k+1)}$. ❑

We have $\lambda_i^{(1)} = \sum_{j \leq i} C_j$, which yields $\lambda_i^{(1)} \geq \lambda_i^{(0)}$.

By induction, we can deduce that the sequence $\{\lambda_i^{(k)}\}_{k \geq 0}$ is increasing.

Lemma 2 - *If* $\lambda_i^{(k)} \le \rho_i \cdot \mathrm{LCM}_{l \le i}\{T_l\}$ *then* $\lambda_i^{(k+1)} \le \rho_i \cdot \mathrm{LCM}_{l \le i}\{T_l\}$.

By assumption (resp. by definition), we have $\lambda_i^{(k)} \le \rho_i \cdot \mathrm{LCM}_{l \le i}\{T_l\}$ (resp. $\rho_i \le 1$).

It follows that $\lambda_i^{(k+1)} \le \sum_{j \le i} \left\lceil \frac{\rho_i \cdot \mathrm{LCM}_{l \le i}\{T_l\}}{T_j} \right\rceil C_j \le \sum_{j \le i} \left\lceil \frac{\mathrm{LCM}_{l \le i}\{T_l\}}{T_j} \right\rceil C_j$.

Hence, $\lambda_i^{(k+1)} \le \rho_i \cdot \mathrm{LCM}_{l \le i}\{T_l\}$. ❑

We have $\lambda_i^{(1)} = \sum_{j \le i} C_j$, which yields $\lambda_i^{(1)} \le \rho_i \cdot \mathrm{LCM}_{l \le i}\{T_l\}$.

By induction, we can deduce that the sequence $\{\lambda_i^{(k)}\}_{k \ge 0}$ is bounded by $\rho_i \cdot \mathrm{LCM}_{l \le i}\{T_l\}$.

From the previous lemmata, we know that the sequence $\{\lambda_i^{(k)}\}_{k \ge 0}$ is increasing and bounded. Therefore, it converges to a single limit $\lambda_i = \lim_{k \to +\infty} \lambda_i^{(k)}$.

This limit λ_i is such that $\lambda_i = W_i(\lambda_i)$. Furthermore, we have $\lambda_i \le \rho_i \cdot \mathrm{LCM}_{l \le i}\{T_l\}$.

A.2 The limit $\lambda_i = \lim_{k \to +\infty} \lambda_i^{(k)}$ is bounded by $\mathrm{Min}\left\{\rho_i \cdot \mathrm{LCM}_{l \le i}\{T_l\}, \frac{1}{1-\rho_i} \cdot \sum_{j \le i} C_j\right\}$.

Let $\lambda_{i,\,p\_th}$ denote the p_th positive root of $\lambda_i = W_i(\lambda_i)$.

It is easy to show that the sequence $\{\lambda_i^{(k)}\}_{k \ge 0}$ converges to the limit $\lambda_i = \lim_{k \to +\infty} \lambda_i^{(k)} = \lambda_{i,\,1\_st}$.

Furthermore, it is interesting to point out that the number of the positive roots of $\lambda_i = W_i(\lambda_i)$ is either finite or infinite. It is finite (resp. infinite) if and only if $\rho_i < 1$ (resp. $\rho_i = 1$).

The positive roots of $\lambda_i = W_i(\lambda_i)$ belong to the set $\{p \cdot \mathrm{LCM}_{l \le i}\{T_l\},\ p \in \mathbb{N}^*\}$ if and only if $\rho_i = 1$. In this case, we have $\lambda_{i,\,p\_th} = p \cdot \mathrm{LCM}_{l \le i}\{T_l\}$ for any natural p.

We can bound all the positive roots of $\lambda_i = W_i(\lambda_i)$ by $\frac{1}{1-\rho_i} \cdot \sum_{j \le i} C_j$ and all the more so λ_i.

Of course, this bound tends to infinity as ρ_i tends to one, but it remains relevant when ρ_i is rather small compared to one.

From section 1 (resp. from section 2), we have $\lambda_i \le \rho_i \cdot \mathrm{LCM}_{l \le i}\{T_l\}$ (resp. $\lambda_i \le \frac{1}{1-\rho_i} \cdot \sum_{j \le i} C_j$).

It follows that $\lambda_i = \lim_{k \to +\infty} \lambda_i^{(k)}$ is bounded by $\mathrm{Min}\left\{\rho_i \cdot \mathrm{LCM}_{l \le i}\{T_l\}, \frac{1}{1-\rho_i} \cdot \sum_{j \le i} C_j\right\}$.

A.3 How much worth is it to solve $\lambda_i = W_i(\lambda_i)$ by successive iterations?

A practical way of solving the recursive equation $\lambda_i = W_i(\lambda_i)$ by successive iterations is to consider the sequence $\{\lambda_i^{(k)}\}_{k \geq 1}$, where $\lambda_i^{(k+1)} = W_i(\lambda_i^{(k)})$ and $\lambda_i^{(1)} = \sum_{j \leq i} C_j$.

The $W_i(t)$ function is discontinuous. If $N_i(0, \alpha)$ (resp. $N_i(\alpha, \beta)$) denotes the number of steps of $W_i(t)$ over $[0, \alpha[$ (resp. $[\alpha, \beta[$) then we have:

$$N_i(0, \alpha) = \sum_{j \leq i} \left\lceil \frac{\alpha}{T_j} \right\rceil \text{ and } N_i(\alpha, \beta) = N_i(0, \beta) - N_i(0, \alpha) = \sum_{j \leq i} \left\lceil \frac{\beta}{T_j} \right\rceil - \sum_{j \leq i} \left\lceil \frac{\alpha}{T_j} \right\rceil.$$

Thus, the number of successive iterations that is needed to converge to $\lambda_i = \lim_{k \to +\infty} \lambda_i^{(k)}$ starting from $\lambda_i^{(1)} = W_i(\lambda_i^{(0)})$ is bounded by $N_i(\lambda_i^{(0)}, \lambda_i) = \sum_{j \leq i} \left\lceil \frac{\lambda_i}{T_j} \right\rceil - \sum_{j \leq i} \left\lceil \frac{\lambda_i^{(0)}}{T_j} \right\rceil = \sum_{j \leq i} \left\lceil \frac{\lambda_i}{T_j} \right\rceil - i$.

Consequently, the number of successive iterations that is need to determine $\lambda_i = \lim_{k \to +\infty} \lambda_i^{(k)}$ (i.e., to converge to $\lambda_i = \lambda_i^{(k)}$ and to verify that $\lambda_i = \lambda_i^{(k)} = \lambda_i^{(k+1)}$) is bounded by $\sum_{j \leq i} \left\lceil \frac{\lambda_i}{T_j} \right\rceil - i + 1$.

Let C_i denote the cost we have to pay to solve $\lambda_i = W_i(\lambda_i)$ by successive iterations starting from $\lambda_i^{(1)}$. We have:

$C_i \leq C_i^{(1)} + C_i^{(2)}$	The cost we have to pay to solve $\lambda_i = W_i(\lambda_i)$ by successive iterations starting from $\lambda_i^{(1)}$
$C_i^{(1)}$	The cost of the computation of $\lambda_i^{(1)} = W_i(\lambda_i^{(0)}) = \sum_{j \leq i} C_j$
$C_i^{(2)}$	The cost of $\sum_{j \leq i} \left\lceil \frac{\lambda_i}{T_j} \right\rceil - i + 1$ iterations

Note that $C_i^{(1)}$, the cost of the computation of $\lambda_i^{(1)} = W_i(\lambda_i^{(0)})$, is generally equal to the cost of one iteration. In our particular case, it is only equal to the cost of $(i-1)$ additions since $\lambda_i^{(1)} = \sum_{j \leq i} C_j$.

Let add, mult and function denote respectively the cost of one addition, the cost of one multiplication and the cost of one elementary function such as “ceil”.

Let $C\_Computation_i$ denote the cost of the computation of $\lambda_i^{(k)}$ and let $C\_Comparison_i$ denote the cost of the comparison of $\lambda_i^{(k)}$ and $\lambda_i^{(k+1)}$.

The cost of each iteration, denoted $C\_Iteration_i$, is equal to the sum of $C\_Computation_i$ and $C\_Comparison_i$.

We have:

- $C\_Computation_i = (i-1)\cdot add + (2i)\cdot mult + i\cdot function$,
- $C\_Comparison_i = 1\cdot function$.

Hence, $C\_Iteration_i = (i-1)\cdot add + (2i)\cdot mult + (i+1)\cdot function$.

Therefore, we have:

$$C_i \le C_i^{(1)} + C_i^{(2)} \le (i-1)\cdot add + \left(\sum_{j\le i}\left\lceil\frac{\lambda_i}{T_j}\right\rceil - i + 1\right)\cdot C\_Iteration_i .$$

Finally, we have:

$$\boxed{C_i \le (i-1)\cdot add + \left(\sum_{j\le i}\left\lceil\frac{\lambda_i}{T_j}\right\rceil - i + 1\right)\cdot ((i-1)\cdot add + (2i)\cdot mult + (i+1)\cdot function)}$$

Assuming that $add = mult = function = \xi$, the cost of one instruction cycle, we have:

$$\boxed{C_i \le \left(4i\sum_{j\le i}\left\lceil\frac{\lambda_i}{T_j}\right\rceil - (4i-1)(i-1)\right)\xi}$$

From the definition of ρ_i, we immediately have:

$\frac{\sum_{j\le i} C_j}{Max_{l\le i}\{T_l\}} \le \rho_i \le \frac{\sum_{j\le i} C_j}{Min_{l\le i}\{T_l\}}$, namely: $\rho_i \cdot Min_{l\le i}\{T_l\} \le \sum_{j\le i} C_j \le \rho_i \cdot Max_{l\le i}\{T_l\}$.

From section 1 and section 2, we have: $\sum_{j\le i} C_j \le \lambda_i^{(1)} \le \lambda_i \le \lim_{k\to+\infty}\lambda_i^{(k)} \le \frac{1}{1-\rho_i}\cdot\sum_{j\le i} C_j$.

It follows that:

$$\rho_i \cdot Min_{l\le i}\{T_l\} \le \lambda_i \le \frac{\rho_i}{1-\rho_i}\cdot Max_{l\le i}\{T_l\} .$$

Thus, we have:

$$i \le \sum_{j\le i}\left\lceil\frac{\lambda_i}{T_j}\right\rceil \le \sum_{j\le i}\left\lceil\frac{\rho_i}{1-\rho_i}\cdot\frac{Max_{l\le i}\{T_l\}}{T_j}\right\rceil \le \sum_{j\le i}\left\lceil\frac{\rho_i}{1-\rho_i}\cdot\frac{Max_{l\le i}\{T_l\}}{Min_{l\le i}\{T_l\}}\right\rceil$$

$i \leq \sum_{j \leq i} \left\lceil \frac{\lambda_i}{T_j} \right\rceil \leq i \cdot \left\lceil \frac{\rho_i}{1-\rho_i} \cdot \delta_i\{T\} \right\rceil$, where $\rho_i = \sum_{j \leq i} \frac{C_j}{T_j}$ and $\delta_i\{T\} = \frac{Max_{l \leq i}\{T_l\}}{Min_{l \leq i}\{T_l\}}$.

Consequently, $\sum_{j \leq i} \left\lceil \frac{\lambda_i}{T_j} \right\rceil$ is pseudo-polynomial in O(i) with a constant factor of approximately:

$\left\lceil \frac{\rho_i}{1-\rho_i} \cdot \delta_i\{T\} \right\rceil$, which is bounded by: $\left\lceil \frac{P_i}{1-P_i} \cdot \Delta_i\{T\} \right\rceil$ when $\rho_i \leq P_i < 1$ and $\delta_i\{T\} \leq \Delta_i\{T\}$.

Finally, we have:

$$C_i \leq \left(4i^2 \cdot \left\lceil \frac{\rho_i}{1-\rho_i} \cdot \delta_i\{T\} - 1 \right\rceil + 5i - 1\right)\xi$$

We can distinguish two cases:

■ Case I: $\frac{\rho_i}{1-\rho_i} \cdot \delta_i\{T\} \leq 1$, that is: $\rho_i \leq \frac{1}{1+\delta_i\{T\}} \leq 50\%$

In that case, we have:

$\forall j \in [1, i], 1 \leq \left\lceil \frac{\lambda_i}{T_j} \right\rceil \leq \left\lceil \frac{\rho_i}{1-\rho_i} \cdot \delta_i\{T\} \right\rceil \leq 1$, which is equivalent to: $0 < \lambda_i \leq Min_{l \leq i}\{T_l\}$.

Since the following equivalence relation holds:

$$(\lambda_i \leq Min_{l \leq i}\{T_l\}) \Leftrightarrow \left(\lambda_i = \sum_{j \leq i} C_j\right)$$

we have: $\lambda_i = \sum_{j \leq i} C_j$ and the cost $C_i = (i-1) \cdot \xi$, namely: $C_i = O(i)$.

■ Case II: $\frac{\rho_i}{1-\rho_i} \cdot \delta_i\{T\} > 1$, that is: $\rho_i > \frac{1}{1+\delta_i\{T\}}$

In that case, we have:

$C_i \leq \left(4i^2 \left\lceil \frac{\rho_i}{1-\rho_i} \cdot \delta_i\{T\} - 1 \right\rceil + 5i - 1\right)\xi$, namely: $C_i = O(i^2)$.

C_i is pseudo-polynomial in $O(i^2)$ with at most a constant factor of approximately:

$4 \cdot \left\lceil \frac{\rho_i}{1-\rho_i} \cdot \delta_i\{T\} \right\rceil - 4 \leq 4 \cdot \left\lceil \frac{P_i}{1-P_i} \cdot \Delta_i\{T\} \right\rceil - 4$ when $\rho_i \leq P_i < 1$ and $\delta_i\{T\} \leq \Delta_i\{T\}$.

This is summarized on the following figure.

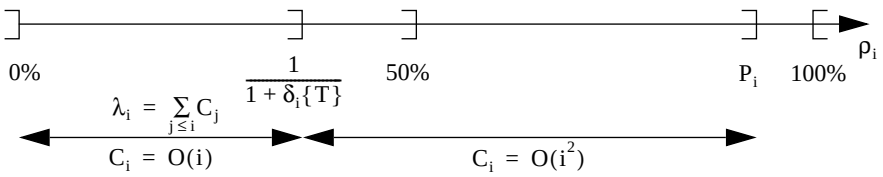


Figure 24: Complexity C_i for general traffics where $\delta_i\{T\} \geq 1$

Let us now consider the following example. Three tasks τ_1, τ_2 and τ_3 are considered. We have: $\tau_1 : (C_1, T_1) = (2, 7)$, $\tau_2 : (C_2, T_2) = (3, 11)$ and $\tau_3 : (C_3, T_3) = (5, 13)$. We seek to solve $\lambda_3 = W_3(\lambda_3)$. For notational convenience, λ_3 (resp. W_3) will be referred simply as λ (resp. W) in Figure 2.

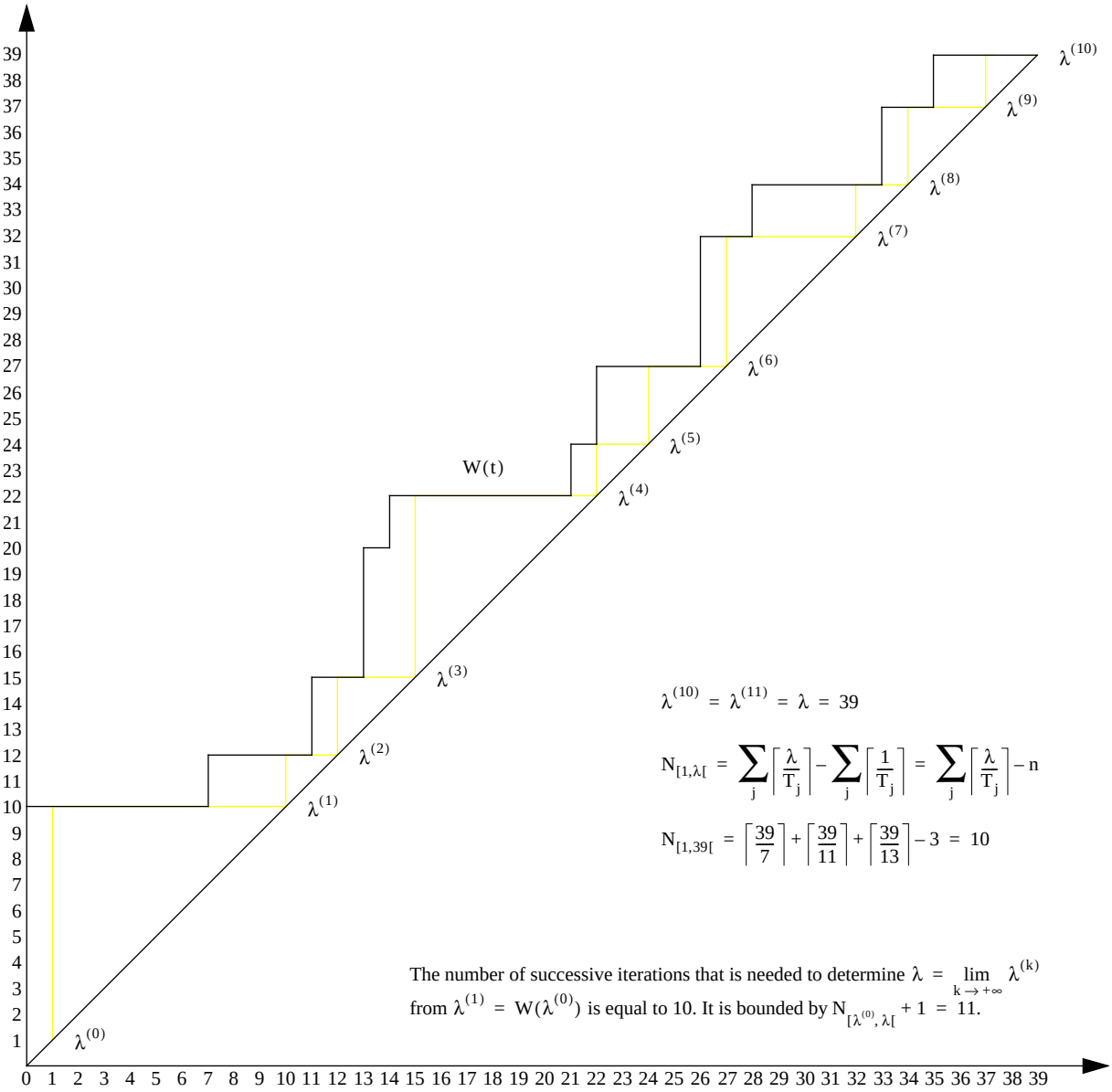


Figure 25: Solving $\lambda = W(\lambda)$ by sucessive iterations.

In this example, C_3 , the cost we have to pay to solve $\lambda_3 = W_3(\lambda_3)$ by successive iterations starting from $\lambda_3^{(1)} = W_3(\lambda_3^{(0)}) = \sum_{j=1}^{3} C_j = 10$, is just equal to $2\text{add} + 10(2\text{add} + 6\text{mult} + 4\text{function})$.

Hence,

$$\boxed{C_3 = 22\text{add} + 60\text{mult} + 40\text{function}}$$

Let $\bar{C}_3$ denote the upper bound on C_3. We have:

$\bar{C}_3 = C_3^{(1)} + C_3^{(2)} = 2\text{add} + 11(2\text{add} + 6\text{mult} + 4\text{function})$.

Hence,

$$\boxed{\bar{C}_3 = 24\text{add} + 66\text{mult} + 44\text{function}}$$

Assuming that $\text{add} = \text{mult} = \text{function} = \xi$, the cost of one instruction cycle, we get a relative error $\frac{\Delta C_3}{C_3} = \frac{\bar{C}_3 - C_3}{C_3} = \frac{2\text{add} + 6\text{mult} + 4\text{function}}{22\text{add} + 60\text{mult} + 40\text{function}} = \frac{12\xi}{122\xi} = \frac{6}{61}$, which is less than 10%.

Appendix B

In this appendix, we study the sequences $\{w_{i,q}^{(k)}\}_{k \geq 0}$ and $\{L_i^{(k)}(a_l)\}_{k \geq 0}$ defined by the following recursive equations:

$$w_{i,q}^{(k+1)} = (q+1)C_i + \sum_{j<i} \left\lceil \frac{w_{i,q}^{(k)}}{T_j} \right\rceil C_j, \; w_{i,q}^{(0)} = 0$$

$$L_i^{(k+1)}(a_l) = \left(1 + \left\lfloor \frac{a_l}{T_i} \right\rfloor\right) C_i + \sum_{j \neq i} \min\left\{ \left\lceil \frac{L_i^{(k)}(a_l)}{T_j} \right\rceil, \max\left\{0, 1 + \left\lfloor \frac{a_l + D_i - D_j}{T_j} \right\rfloor\right\}\right\} C_j,$$

$$L_i^{(0)}(a_l) = 0$$

where:

$$q \in Q_i^{\dagger}, \; Q_i^{\dagger} = \mathbb{N} \cap [0, \lambda_i / T_i[,$$

$$a \in A_i^{\dagger}, \; A_i^{\dagger} = \bigcup_j A_{i,j}{}^{\dagger}, \; A_{i,j}{}^{\dagger} = A_{i,j} \cap [0, \lambda[, \; A_{i,j} = \{D_j + k_j T_j - D_i, \; k_j \in \mathbb{N}\},$$

$$\lambda_i = \sum_{j \leq i} \left\lceil \frac{\lambda_i}{T_j} \right\rceil C_j, \; \lambda = \sum_j \left\lceil \frac{\lambda}{T_j} \right\rceil C_j,$$

$$\forall j, \; 1 \leq j \leq n, \; (C_j, T_j) \in \mathbb{N}^* \times \mathbb{N}^* \text{ and } \sum_j \frac{C_j}{T_j} \leq 1.$$

The set $A_i^{\dagger} = \{a_1, a_2, \ldots, a_{|A_i^{\dagger}|}\}$ is ordered in a non-decreasing order.

For notational convenience, we define the $W_{i,q}(t)$ function as $(q+1)C_i + \sum_{j<i} \left\lceil \frac{t}{T_j} \right\rceil C_j$ and the $L_i(a, t)$ function as $\left(1 + \left\lfloor \frac{a}{T_i} \right\rfloor\right) C_i + \sum_{j \neq i} \min\left\{ \left\lceil \frac{t}{T_j} \right\rceil, \max\left\{0, 1 + \left\lfloor \frac{a + D_i - D_j}{T_j} \right\rfloor\right\}\right\} C_j$.

The purpose of this appendix is to show how to obtain a faster convergence.

Appendix B is organized in two sections as follows:

- in section 1, we show by induction that $w_{i,q+1}^{(k)} \geq w_{i,q}^{(k)}$. It involves $w_{i,q+1} \geq w_{i,q}$, where $w_{i,q} = W_{i,q}(w_{i,q}) = \lim_{k \to +\infty} w_{i,q}^{(k)}$ for every $q \in Q_i^{\dagger}$. It is used to obtain a faster convergence.
- in section 2, we show by induction that $L_i^{(k)}(a_{l+1}) \geq L_i^{(k)}(a_l)$. It involves $L_i(a_{l+1}) \geq L_i(a_l)$, where $L_i(a_l) = L_i(a_l, L_i(a_l)) = \lim_{k \to +\infty} L_i^{(k)}(a_l)$ for every $a_l \in A_i^{\dagger}$. It is used to obtain a faster convergence.

B.1 The sequence $\{w_{i,q}^{(k)}\}_{k \geq 0}$, $w_{i,q}^{(0)} = 0$. How to obtain a faster convergence?

Let us demonstrate by induction the following lemma.

Lemma 1 - $\forall k \geq 0,\ w_{i,q+1}^{(k)} \geq w_{i,q}^{(k)}$.

By definition, we have $w_{i,q+1}^{(0)} = w_{i,q}^{(0)} = 0$. Thus, $w_{i,q+1}^{(0)} \geq w_{i,q}^{(0)}$.

Let us assume that $w_{i,q+1}^{(k)} \geq w_{i,q}^{(k)}$ is true at rank k.

We have $w_{i,q+1}^{(k+1)} - w_{i,q}^{(k+1)} = C_i + \sum_{j<i} \left\{ \left\lceil \frac{w_{i,q+1}^{(k)}}{T_j} \right\rceil - \left\lceil \frac{w_{i,q}^{(k)}}{T_j} \right\rceil \right\} C_j$.

By assumption, we have $w_{i,q+1}^{(k)} \geq w_{i,q}^{(k)}$. It follows that $\left\lceil \frac{w_{i,q+1}^{(k)}}{T_j} \right\rceil \geq \left\lceil \frac{w_{i,q}^{(k)}}{T_j} \right\rceil$ for every j, $1 \leq j < i$.

Hence, $w_{i,q+1}^{(k+1)} \geq w_{i,q}^{(k+1)}$.

By induction, we have $\forall k \geq 0,\ w_{i,q+1}^{(k)} \geq w_{i,q}^{(k)}$. ❑

Let $w_{i,q} = \lim_{k \to +\infty} w_{i,q}^{(k)}$ for every $q \in Q_i^{\dagger}$. From Lemma 1, we have $w_{i,q+1} \geq w_{i,q}$.

It follows that a faster convergence can be obtained with $w_{i,q+1}^{(0)} = w_{i,q}$.

Furthermore, with $w_{i,q+1}^{(0)} = w_{i,q}$, we have:

$$w_{i,q+1}^{(1)} = (q+2)C_i + \sum_{j<i} \left\lceil \frac{w_{i,q+1}^{(0)}}{T_j} \right\rceil C_j = C_i + (q+1)C_i + \sum_{j<i} \left\lceil \frac{w_{i,q}}{T_j} \right\rceil C_j = C_i + w_{i,q}.$$

Therefore, a faster convergence can be obtained with $w_{i,q+1}^{(1)} = C_i + w_{i,q}$.

By definition, we have $w_{i,0}^{(k+1)} = C_i + \sum_{j<i} \left\lceil \frac{w_{i,0}^{(k)}}{T_j} \right\rceil C_j$, $w_{i,0}^{(0)} = 0$.

This yields $w_{i,0}^{(1)} = C_i$ and $w_{i,0}^{(2)} = C_i + \sum_{j<i} \left\lceil \frac{C_i}{T_j} \right\rceil C_j$. We have $w_{i,0}^{(2)} \geq C_i + \sum_{j<i} C_j$.

It follows that a faster convergence can be obtained with $w_{i,0}^{(0)} = 1$.

Furthermore, with $w_{i,0}^{(0)} = 1$, we have $w_{i,0}^{(1)} = C_i + \sum_{j<i} C_j$.

Therefore, a faster convergence can be obtained with $w_{i,0}^{(1)} = C_i + \sum_{j<i} C_j$.

In a general manner, to solve $w_{i,q} = W_{i,q}(w_{i,q})$ by successive iterations, we consider the sequence $\{w_{i,q}^{(k)}\}_{k \geq 1}$ defined by the following recursive relation:

$$w_{i,q}^{(k+1)} = W_{i,q}(w_{i,q}^{(k)}), \quad w_{i,q}^{(1)} = C_i + w_{i,q-1}, \quad w_{i,-1} = \sum_{j<i} C_j$$

B.2 The sequence $\{L_i^{(k)}(a_l)\}_{k \geq 0}$, $L_i^{(0)}(a_l) = 0$. How to obtain a faster convergence?

Let us demonstrate by induction the following lemma.

Lemma 2 - $\forall k \geq 0,\ L_i^{(k)}(a_{l+1}) \geq L_i^{(k)}(a_l)$.

By definition, we have $L_i^{(0)}(a_{l+1}) = L_i^{(0)}(a_l) = 0$. Thus, $L_i^{(0)}(a_{l+1}) \geq L_i^{(0)}(a_l)$.

Let us assume that $L_i^{(k)}(a_{l+1}) \geq L_i^{(k)}(a_l)$ is true at rank k.

We have:

$$L_i^{(k+1)}(a_{l+1}) = \left(1 + \left\lfloor \frac{a_{l+1}}{T_i} \right\rfloor\right) C_i + \sum_{j \neq i} \min\left\{ \left\lceil \frac{L_i^{(k)}(a_{l+1})}{T_j} \right\rceil, \max\left\{0, 1 + \left\lfloor \frac{a_{l+1} + D_i - D_j}{T_j} \right\rfloor\right\}\right\} C_j$$

By assumption (resp. by definition), we have $L_i^{(k)}(a_{l+1}) \geq L_i^{(k)}(a_l)$ (resp. $a_{l+1} \geq a_l$). It follows that:

$$L_i^{(k+1)}(a_{l+1}) \geq \left(1 + \left\lfloor \frac{a_l}{T_i} \right\rfloor\right) C_i + \sum_{j \neq i} \min\left\{ \left\lceil \frac{L_i^{(k)}(a_l)}{T_j} \right\rceil, \max\left\{0, 1 + \left\lfloor \frac{a_l + D_i - D_j}{T_j} \right\rfloor\right\}\right\} C_j .$$

Hence, $L_i^{(k+1)}(a_{l+1}) \geq L_i^{(k+1)}(a_l)$.

By induction, we have $\forall k \geq 0,\ L_i^{(k)}(a_{l+1}) \geq L_i^{(k)}(a_l)$. ❑

Let $L_i(a_l) = \lim_{k \to +\infty} L_i^{(k)}(a_l)$ for every $a_l \in A_i^{\dagger}$. From Lemma 2, we have $L_i(a_{l+1}) \geq L_i(a_l)$.

It follows that a faster convergence can be obtained with $L_i^{(0)}(a_{l+1}) = L_i(a_l)$.

In a general manner, to solve $L_i(a_l) = L_i(a_l, L_i(a_l))$ by successive iterations, we consider the sequence $\{L_i^{(k)}(a_l)\}_{k \geq 0}$ defined by the following recursive relation:

$$L_i^{(k+1)}(a_l) = L_i(a_l, L_i^{(k)}(a_l)), \quad L_i^{(0)}(a_l) = L_i(a_{l-1}), \quad L_i(a_0) = 0$$

Appendix C

In this appendix, we describe in depth the task sets used in Section 5.3, page 111, and the computational complexity of the feasibility tests based upon calculation of the worst-case response time of a task for HPF/DM (Highest Priority First / Deadline Monotonic) and EDF (Earliest Deadline First).

C.1 Task parameters

TABLE 4. Tasks τ_i -- 100 tasks

Task τ_i	C_i	T_i	D_i
τ_1	2	199	199
τ_2	3	211	211
τ_3	5	223	223
τ_4	7	227	227
τ_5	2	229	229
τ_6	3	233	233
τ_7	5	239	239
τ_8	7	241	241
τ_9	2	251	251
τ_{10}	3	257	257
τ_{11}	5	263	263
τ_{12}	7	269	269
τ_{13}	2	271	271
τ_{14}	3	277	277
τ_{15}	5	281	281
τ_{16}	7	283	283
τ_{17}	2	293	293
τ_{18}	3	307	307
τ_{19}	5	311	311
τ_{20}	7	313	313
τ_{21}	2	317	317
τ_{22}	3	331	331
τ_{23}	5	337	337
τ_{24}	7	347	347
τ_{25}	2	349	349
τ_{26}	3	353	353
τ_{27}	5	359	359
τ_{28}	7	367	367
τ_{29}	2	373	373
τ_{30}	3	379	379
τ_{31}	5	383	383
τ_{32}	7	389	389
τ_{33}	2	397	397
τ_{34}	3	401	401
τ_{35}	5	409	409
τ_{36}	7	419	419
τ_{37}	2	421	421
τ_{38}	3	431	431
τ_{39}	5	433	433
τ_{40}	7	439	439
τ_{41}	2	443	443
τ_{42}	3	449	449
τ_{43}	5	457	457
τ_{44}	7	461	461
τ_{45}	2	463	463

TABLE 4. Tasks τ_i -- 100 tasks

Task τ_i	C_i	T_i	D_i
τ_{46}	3	467	467
τ_{47}	5	479	479
τ_{48}	7	487	487
τ_{49}	2	491	491
τ_{50}	3	499	499
τ_{51}	5	503	503
τ_{52}	7	509	509
τ_{53}	2	521	521
τ_{54}	3	523	523
τ_{55}	5	541	541
τ_{56}	7	547	547
τ_{57}	2	557	557
τ_{58}	3	563	563
τ_{59}	5	569	569
τ_{60}	7	571	571
τ_{61}	2	577	577
τ_{62}	3	587	587
τ_{63}	5	593	593
τ_{64}	7	599	599
τ_{65}	2	601	601
τ_{66}	3	607	607
τ_{67}	5	613	613
τ_{68}	7	617	617
τ_{69}	2	619	619
τ_{70}	3	631	631
τ_{71}	5	641	641
τ_{72}	7	643	643
τ_{73}	2	647	647
τ_{74}	3	653	653
τ_{75}	5	659	659
τ_{76}	7	661	661
τ_{77}	2	673	673
τ_{78}	3	677	677
τ_{79}	5	683	683
τ_{80}	7	691	691
τ_{81}	2	701	701
τ_{82}	3	709	709
τ_{83}	5	719	719
τ_{84}	7	727	727
τ_{85}	2	733	733
τ_{86}	3	739	739
τ_{87}	5	743	743
τ_{88}	7	751	751
τ_{89}	2	757	757
τ_{90}	3	761	761
τ_{91}	5	769	769
τ_{92}	7	773	773
τ_{93}	2	787	787
τ_{94}	3	797	797
τ_{95}	5	809	809
τ_{96}	7	811	811
τ_{97}	2	821	821
τ_{98}	3	823	823
τ_{99}	5	827	827
τ_{100}	7	829	829

C.2 Computational Complexity -- Highest Priority First / Deadline Monotonic

TABLE 5. Computational Complexity -- HPF/DM

Task set τ	Lower Bound on C (ξ) **(off-line computation)**	Real Overall Cost C (ξ) **(on-line computation)**	Upper Bound on C (ξ) **(off-line computation)**
$\tau = \Sigma_{i \leq 1} \tau_i$	13	13	13
$\tau = \Sigma_{i \leq 2} \tau_i$	29	29	36
$\tau = \Sigma_{i \leq 3} \tau_i$	49	49	78
$\tau = \Sigma_{i \leq 4} \tau_i$	73	73	147
$\tau = \Sigma_{i \leq 5} \tau_i$	101	101	251
$\tau = \Sigma_{i \leq 6} \tau_i$	133	133	398
$\tau = \Sigma_{i \leq 7} \tau_i$	169	169	596
$\tau = \Sigma_{i \leq 8} \tau_i$	209	209	853
$\tau = \Sigma_{i \leq 9} \tau_i$	253	253	1 177
$\tau = \Sigma_{i \leq 10} \tau_i$	301	301	1 576
$\tau = \Sigma_{i \leq 11} \tau_i$	353	353	2 058
$\tau = \Sigma_{i \leq 12} \tau_i$	409	409	2 631
$\tau = \Sigma_{i \leq 13} \tau_i$	469	469	3 303
$\tau = \Sigma_{i \leq 14} \tau_i$	533	533	4 082
$\tau = \Sigma_{i \leq 15} \tau_i$	601	601	4 976
$\tau = \Sigma_{i \leq 16} \tau_i$	673	673	5 993
$\tau = \Sigma_{i \leq 17} \tau_i$	749	749	7 141
$\tau = \Sigma_{i \leq 18} \tau_i$	829	829	8 428
$\tau = \Sigma_{i \leq 19} \tau_i$	913	913	9 862
$\tau = \Sigma_{i \leq 20} \tau_i$	1 001	1 001	11 451
$\tau = \Sigma_{i \leq 21} \tau_i$	1 093	1 093	13 203
$\tau = \Sigma_{i \leq 22} \tau_i$	1 189	1 189	15 126
$\tau = \Sigma_{i \leq 23} \tau_i$	1 289	1 289	17 228
$\tau = \Sigma_{i \leq 24} \tau_i$	1 393	1 393	19 517
$\tau = \Sigma_{i \leq 25} \tau_i$	1 501	1 501	22 001
$\tau = \Sigma_{i \leq 26} \tau_i$	1 613	1 613	24 688
$\tau = \Sigma_{i \leq 27} \tau_i$	1 729	1 729	27 586
$\tau = \Sigma_{i \leq 28} \tau_i$	1 849	1 849	30 703
$\tau = \Sigma_{i \leq 29} \tau_i$	1 973	1 973	34 047
$\tau = \Sigma_{i \leq 30} \tau_i$	2 101	2 101	37 626
$\tau = \Sigma_{i \leq 31} \tau_i$	2 233	2 233	41 448
$\tau = \Sigma_{i \leq 32} \tau_i$	2 369	2 369	45 521
$\tau = \Sigma_{i \leq 33} \tau_i$	2 509	2 509	49 853
$\tau = \Sigma_{i \leq 34} \tau_i$	2 653	2 653	54 452
$\tau = \Sigma_{i \leq 35} \tau_i$	2 801	2 801	59 326
$\tau = \Sigma_{i \leq 36} \tau_i$	2 953	2 953	64 483
$\tau = \Sigma_{i \leq 37} \tau_i$	3 109	3 109	69 931
$\tau = \Sigma_{i \leq 38} \tau_i$	3 269	3 269	75 678
$\tau = \Sigma_{i \leq 39} \tau_i$	3 433	3 433	81 732
$\tau = \Sigma_{i \leq 40} \tau_i$	3 601	3 601	88 101
$\tau = \Sigma_{i \leq 41} \tau_i$	3 773	3 773	94 793
$\tau = \Sigma_{i \leq 42} \tau_i$	3 949	3 949	101 816
$\tau = \Sigma_{i \leq 43} \tau_i$	4 129	4 129	109 178
$\tau = \Sigma_{i \leq 44} \tau_i$	4 313	4 313	116 887
$\tau = \Sigma_{i \leq 45} \tau_i$	4 501	4 501	124 951
$\tau = \Sigma_{i \leq 46} \tau_i$	4 693	4 693	133 378
$\tau = \Sigma_{i \leq 47} \tau_i$	4 889	4 889	142 176
$\tau = \Sigma_{i \leq 48} \tau_i$	5 089	5 280	151 544

TABLE 5. Computational Complexity -- HPF/DM

Task set τ	Lower Bound on C (ξ) **(off-line computation)**	Real Overall Cost C (ξ) **(on-line computation)**	Upper Bound on C (ξ) **(off-line computation)**
$\tau = \Sigma_{i \leq 49} \tau_i$	5 293	5 679	161 303
$\tau = \Sigma_{i \leq 50} \tau_i$	5 501	6 086	171 461
$\tau = \Sigma_{i \leq 51} \tau_i$	5 713	6 501	182 229
$\tau = \Sigma_{i \leq 52} \tau_i$	5 929	7 959	194 865
$\tau = \Sigma_{i \leq 53} \tau_i$	6 149	9 445	208 167
$\tau = \Sigma_{i \leq 54} \tau_i$	6 373	12 034	223 441
$\tau = \Sigma_{i \leq 55} \tau_i$	6 601	13 576	239 218
$\tau = \Sigma_{i \leq 56} \tau_i$	6 833	15 146	255 729
$\tau = \Sigma_{i \leq 57} \tau_i$	7 069	16 744	273 444
$\tau = \Sigma_{i \leq 58} \tau_i$	7 309	18 139	291 702
$\tau = \Sigma_{i \leq 59} \tau_i$	7 553	19 793	310 981
$\tau = \Sigma_{i \leq 60} \tau_i$	7 801	21 714	331 783
$\tau = \Sigma_{i \leq 61} \tau_i$	8 053	23 910	353 662
$\tau = \Sigma_{i \leq 62} \tau_i$	8 309	26 636	376 642
$\tau = \Sigma_{i \leq 63} \tau_i$	8 569	29 155	401 249
$\tau = \Sigma_{i \leq 64} \tau_i$	9 095	36 566	438 750
$\tau = \Sigma_{i \leq 65} \tau_i$	9 629	43 834	478 393
$\tau = \Sigma_{i \leq 66} \tau_i$	10 171	49 899	519 174
$\tau = \Sigma_{i \leq 67} \tau_i$	10 721	54 721	561 109
$\tau = \Sigma_{i \leq 68} \tau_i$	11 279	59 886	605 569
$\tau = \Sigma_{i \leq 69} \tau_i$	11 845	65 402	652 335
$\tau = \Sigma_{i \leq 70} \tau_i$	12 419	70 161	700 339
$\tau = \Sigma_{i \leq 71} \tau_i$	13 001	74 988	751 012
$\tau = \Sigma_{i \leq 72} \tau_i$	13 591	79 596	804 410
$\tau = \Sigma_{i \leq 73} \tau_i$	14 189	84 559	860 007
$\tau = \Sigma_{i \leq 74} \tau_i$	14 795	89 885	917 843
$\tau = \Sigma_{i \leq 75} \tau_i$	15 409	94 386	977 659
$\tau = \Sigma_{i \leq 76} \tau_i$	16 031	99 856	1 042 214
$\tau = \Sigma_{i \leq 77} \tau_i$	16 661	104 784	1 108 542
$\tau = \Sigma_{i \leq 78} \tau_i$	17 299	111 953	1 180 710
$\tau = \Sigma_{i \leq 79} \tau_i$	17 945	117 954	1 255 066
$\tau = \Sigma_{i \leq 80} \tau_i$	18 599	124 031	1 333 556
$\tau = \Sigma_{i \leq 81} \tau_i$	19 261	128 892	1 413 676
$\tau = \Sigma_{i \leq 82} \tau_i$	19 931	134 467	1 497 731
$\tau = \Sigma_{i \leq 83} \tau_i$	20 609	146 068	1 594 399
$\tau = \Sigma_{i \leq 84} \tau_i$	21 637	161 166	1 715 357
$\tau = \Sigma_{i \leq 85} \tau_i$	22 677	172 376	1 838 776
$\tau = \Sigma_{i \leq 86} \tau_i$	23 729	183 375	1 966 052
$\tau = \Sigma_{i \leq 87} \tau_i$	24 793	195 543	2 102 446
$\tau = \Sigma_{i \leq 88} \tau_i$	25 869	207 500	2 248 134
$\tau = \Sigma_{i \leq 89} \tau_i$	26 957	218 528	2 398 677
$\tau = \Sigma_{i \leq 90} \tau_i$	28 057	236 142	2 567 789
$\tau = \Sigma_{i \leq 91} \tau_i$	29 169	255 041	2 751 853
$\tau = \Sigma_{i \leq 92} \tau_i$	30 293	267 175	2 939 413
$\tau = \Sigma_{i \leq 93} \tau_i$	31 807	287 981	3 169 834
$\tau = \Sigma_{i \leq 94} \tau_i$	33 337	306 011	3 405 364
$\tau = \Sigma_{i \leq 95} \tau_i$	34 883	331 434	3 668 420
$\tau = \Sigma_{i \leq 96} \tau_i$	36 835	360 196	3 971 793
$\tau = \Sigma_{i \leq 97} \tau_i$	38 807	384 614	4 301 554
$\tau = \Sigma_{i \leq 98} \tau_i$	40 799	417 886	4 667 958
$\tau = \Sigma_{i \leq 99} \tau_i$	43 213	452 295	5 072 087
$\tau = \Sigma_{i \leq 100} \tau_i$	46 057	513 792	5 613 980

C.3 Computational Complexity -- Earliest Deadline First

TABLE 6. Computational Complexity -- EDF

Task set τ	Lower Bound on C (ξ) **(off-line computation)**	Real Overall Cost C (ξ) **(on-line computation)**	Upper Bound on C (ξ) **(off-line computation)**
$\tau = \Sigma_{i \leq 1} \tau_i$	10	15	15
$\tau = \Sigma_{i \leq 2} \tau_i$	41	89	89
$\tau = \Sigma_{i \leq 3} \tau_i$	94	229	256
$\tau = \Sigma_{i \leq 4} \tau_i$	289	669	783
$\tau = \Sigma_{i \leq 5} \tau_i$	572	1 307	1 601
$\tau = \Sigma_{i \leq 6} \tau_i$	1 005	2 265	2 865
$\tau = \Sigma_{i \leq 7} \tau_i$	1 694	3 753	4 818
$\tau = \Sigma_{i \leq 8} \tau_i$	2 789	6 069	7 791
$\tau = \Sigma_{i \leq 9} \tau_i$	3 914	8 471	11 075
$\tau = \Sigma_{i \leq 10} \tau_i$	5 215	11 247	14 991
$\tau = \Sigma_{i \leq 11} \tau_i$	7 043	15 093	20 268
$\tau = \Sigma_{i \leq 12} \tau_i$	9 241	19 699	26 629
$\tau = \Sigma_{i \leq 13} \tau_i$	11 564	24 579	33 621
$\tau = \Sigma_{i \leq 14} \tau_i$	14 279	30 263	41 807
$\tau = \Sigma_{i \leq 15} \tau_i$	17 902	37 777	52 246
$\tau = \Sigma_{i \leq 16} \tau_i$	22 049	46 359	64 209
$\tau = \Sigma_{i \leq 17} \tau_i$	26 021	54 619	76 339
$\tau = \Sigma_{i \leq 18} \tau_i$	30 495	63 903	90 015
$\tau = \Sigma_{i \leq 19} \tau_i$	35 709	74 685	105 744
$\tau = \Sigma_{i \leq 20} \tau_i$	42 161	87 957	124 551
$\tau = \Sigma_{i \leq 21} \tau_i$	48 848	101 498	144 473
$\tau = \Sigma_{i \leq 22} \tau_i$	55 261	114 497	164 529
$\tau = \Sigma_{i \leq 23} \tau_i$	64 040	130 977	190 010
$\tau = \Sigma_{i \leq 24} \tau_i$	73 369	149 479	217 333
$\tau = \Sigma_{i \leq 25} \tau_i$	82 977	168 250	245 991
$\tau = \Sigma_{i \leq 26} \tau_i$	94 241	188 881	278 761
$\tau = \Sigma_{i \leq 27} \tau_i$	107 000	212 342	315 356
$\tau = \Sigma_{i \leq 28} \tau_i$	125 913	245 505	365 097
$\tau = \Sigma_{i \leq 29} \tau_i$	139 919	271 066	405 969
$\tau = \Sigma_{i \leq 30} \tau_i$	157 193	301 049	454 301
$\tau = \Sigma_{i \leq 31} \tau_i$	178 336	337 796	511 326
$\tau = \Sigma_{i \leq 32} \tau_i$	202 253	378 367	574 895
$\tau = \Sigma_{i \leq 33} \tau_i$	221 929	412 210	631 051
$\tau = \Sigma_{i \leq 34} \tau_i$	246 489	455 145	698 025
$\tau = \Sigma_{i \leq 35} \tau_i$	274 391	501 033	772 776
$\tau = \Sigma_{i \leq 36} \tau_i$	310 145	559 355	864 335
$\tau = \Sigma_{i \leq 37} \tau_i$	339 401	607 670	944 109
$\tau = \Sigma_{i \leq 38} \tau_i$	367 709	655 285	1 023 201
$\tau = \Sigma_{i \leq 39} \tau_i$	406 379	718 976	1 124 210
$\tau = \Sigma_{i \leq 40} \tau_i$	451 341	790 295	1 239 051
$\tau = \Sigma_{i \leq 41} \tau_i$	488 207	850 882	1 339 047
$\tau = \Sigma_{i \leq 42} \tau_i$	527 243	915 755	1 444 715
$\tau = \Sigma_{i \leq 43} \tau_i$	572 736	987 432	1 564 644
$\tau = \Sigma_{i \leq 44} \tau_i$	631 769	1 081 089	1 713 005
$\tau = \Sigma_{i \leq 45} \tau_i$	677 671	1 155 913	1 836 601
$\tau = \Sigma_{i \leq 46} \tau_i$	728 495	1 237 495	1 971 495
$\tau = \Sigma_{i \leq 47} \tau_i$	782 933	1 324 082	2 115 110
$\tau = \Sigma_{i \leq 48} \tau_i$	861 553	1 462 897	2 325 241
$\tau = \Sigma_{i \leq 49} \tau_i$	916 019	1 549 756	2 472 912
$\tau = \Sigma_{i \leq 50} \tau_i$	979 625	1 652 553	2 641 545
$\tau = \Sigma_{i \leq 51} \tau_i$	1 063 973	1 803 788	2 872 718
$\tau = \Sigma_{i \leq 52} \tau_i$	1 299 689	2 261 323	3 568 217

TABLE 6. Computational Complexity -- EDF

Task set τ	Lower Bound on C (ξ) **(off-line computation)**	Real Overall Cost C (ξ) **(on-line computation)**	Upper Bound on C (ξ) **(off-line computation)**
$\tau = \Sigma_{i \leq 53} \tau_i$	1 389 128	2 429 459	3 830 415
$\tau = \Sigma_{i \leq 54} \tau_i$	1 618 479	2 936 775	4 373 259
$\tau = \Sigma_{i \leq 55} \tau_i$	1 717 769	3 102 058	4 637 295
$\tau = \Sigma_{i \leq 56} \tau_i$	1 852 637	3 335 547	4 971 567
$\tau = \Sigma_{i \leq 57} \tau_i$	2 028 603	3 671 769	5 395 044
$\tau = \Sigma_{i \leq 58} \tau_i$	2 144 939	3 862 715	5 696 779
$\tau = \Sigma_{i \leq 59} \tau_i$	2 314 379	4 168 148	6 107 436
$\tau = \Sigma_{i \leq 60} \tau_i$	2 566 393	4 626 493	6 685 285
$\tau = \Sigma_{i \leq 61} \tau_i$	2 754 833	4 957 978	7 138 513
$\tau = \Sigma_{i \leq 62} \tau_i$	2 945 357	5 296 485	7 598 941
$\tau = \Sigma_{i \leq 63} \tau_i$	3 207 422	5 778 176	8 228 705
$\tau = \Sigma_{i \leq 64} \tau_i$	5 007 929	9 029 805	12 349 493
$\tau = \Sigma_{i \leq 65} \tau_i$	5 388 089	9 664 777	13 292 021
$\tau = \Sigma_{i \leq 66} \tau_i$	5 656 281	10 212 441	13 980 201
$\tau = \Sigma_{i \leq 67} \tau_i$	5 972 619	10 784 061	14 778 976
$\tau = \Sigma_{i \leq 68} \tau_i$	6 557 753	11 837 825	16 236 401
$\tau = \Sigma_{i \leq 69} \tau_i$	7 000 499	12 547 850	17 360 273
$\tau = \Sigma_{i \leq 70} \tau_i$	7 322 335	13 211 247	18 187 179
$\tau = \Sigma_{i \leq 71} \tau_i$	7 972 163	14 363 588	19 838 188
$\tau = \Sigma_{i \leq 72} \tau_i$	8 657 901	15 609 285	21 619 041
$\tau = \Sigma_{i \leq 73} \tau_i$	9 126 325	16 398 950	22 845 883
$\tau = \Sigma_{i \leq 74} \tau_i$	9 661 019	17 467 107	24 251 883
$\tau = \Sigma_{i \leq 75} \tau_i$	10 233 916	18 539 395	25 772 803
$\tau = \Sigma_{i \leq 76} \tau_i$	11 375 257	20 579 127	28 878 297
$\tau = \Sigma_{i \leq 77} \tau_i$	11 851 892	21 499 844	30 109 161
$\tau = \Sigma_{i \leq 78} \tau_i$	13 192 749	24 211 665	33 653 529
$\tau = \Sigma_{i \leq 79} \tau_i$	13 890 329	25 558 952	35 326 386
$\tau = \Sigma_{i \leq 80} \tau_i$	14 969 249	27 485 803	38 075 187
$\tau = \Sigma_{i \leq 81} \tau_i$	15 511 132	28 552 492	39 321 172
$\tau = \Sigma_{i \leq 82} \tau_i$	16 456 913	30 430 929	41 553 873
$\tau = \Sigma_{i \leq 83} \tau_i$	19 291 874	35 512 662	48 288 664
$\tau = \Sigma_{i \leq 84} \tau_i$	24 648 809	44 248 109	60 949 283
$\tau = \Sigma_{i \leq 85} \tau_i$	25 495 606	45 988 417	62 977 969
$\tau = \Sigma_{i \leq 86} \tau_i$	26 740 727	48 666 227	66 045 887
$\tau = \Sigma_{i \leq 87} \tau_i$	29 282 053	53 558 230	72 555 406
$\tau = \Sigma_{i \leq 88} \tau_i$	32 014 617	58 967 933	79 120 871
$\tau = \Sigma_{i \leq 89} \tau_i$	33 424 154	61 701 480	82 348 540
$\tau = \Sigma_{i \leq 90} \tau_i$	38 224 443	71 071 347	94 066 443
$\tau = \Sigma_{i \leq 91} \tau_i$	42 145 367	77 914 622	104 043 322
$\tau = \Sigma_{i \leq 92} \tau_i$	43 808 873	81 812 535	107 730 113
$\tau = \Sigma_{i \leq 93} \tau_i$	54 696 031	99 598 615	136 265 533
$\tau = \Sigma_{i \leq 94} \tau_i$	56 628 559	105 011 379	139 760 863
$\tau = \Sigma_{i \leq 95} \tau_i$	64 564 052	119 037 783	159 574 368
$\tau = \Sigma_{i \leq 96} \tau_i$	75 791 533	141 537 283	186 581 233
$\tau = \Sigma_{i \leq 97} \tau_i$	83 381 915	154 266 264	206 092 931
$\tau = \Sigma_{i \leq 98} \tau_i$	93 939 755	172 304 027	232 487 179
$\tau = \Sigma_{i \leq 99} \tau_i$	105 175 759	196 587 457	258 322 789
$\tau = \Sigma_{i \leq 100} \tau_i$	143 873 217	266 743 639	353 721 015

Lower bounds (off-line computation) (cf. Section 4.2.2.2.2, page 85)

Let $\underline{C}_i(a_l)$ denote the lower bound on $C_i(a_l)$.

We have: $\underline{C}_i(a_l) = 1 \cdot C\_Iteration_i(a_l) = (11n - 6) \cdot \xi$.

Let $\underline{C}_i$ denote the lower bound on C_i.

We have: $\underline{C}_i = \sum_{a_l \in A_i^\dagger} (\underline{C}_i(a_l) + 2\xi) = (11n - 4) \cdot \left|A_i^\dagger\right| \cdot \xi$.

Let $\underline{C}$ denote the lower bound on C.

We have: $\underline{C} = \sum_i (\underline{C}_i + 2\xi) + \xi = (11n - 4) \cdot \sum_i \left|A_i^\dagger\right| \cdot \xi + (2n + 1) \cdot \xi$.

Real Costs (on-line computation) (cf. Section 4.1.2.2, page 76)

Let $Number\_of\_Iterations_i(a_l)$ denote the exact number of successive iterations that is needed to determine $L_i(a_l) = \lim_{k \to +\infty} L_i^{(k)}(a_l)$ starting from $L_i^{(0)}(a_l) = L_i(a_{l-1})$.

We have:

$C_i(a_l) = Number\_of\_Iterations_i(a_l) \cdot C\_Iteration_i(a_l)$;

$C_i = \sum_{a_l \in A_i^\dagger} (C_i(a_l) + 2\xi)$;

$C = \sum_i (C_i + 2\xi) + \xi$.

Upper bounds (off-line computation) (cf. Section 4.1.2.2, page 76)

Let $\overline{C}_i(a_l)$ denote the upper bound on $C_i(a_l)$.

We have: $\overline{C}_i(a_l) = (N_i(L_i(a_{l-1}), L_i(a_l)) + 2) \cdot C\_Iteration_i(a_l)$, where:

$$N_i(L_i(a_{l-1}), L_i(a_l)) = \sum_{j \neq i} \min\left\{\left\lceil \frac{L_i(a_l)}{T_j} \right\rceil, \max\left\{0, 1 + \left\lfloor \frac{a_l + D_i - D_j}{T_j} \right\rfloor\right\}\right\} - \sum_{j \neq i} \min\left\{\left\lceil \frac{L_i(a_{l-1})}{T_j} \right\rceil, \max\left\{0, 1 + \left\lfloor \frac{a_l + D_i - D_j}{T_j} \right\rfloor\right\}\right\}$$

Let $\overline{C}_i$ denote the upper bound on C_i.

We have: $\overline{C}_i = \sum_{a_l \in A_i^\dagger} (\overline{C}_i(a_l) + 2\xi)$.

Let $\overline{C}$ denote the upper bound on C.

We have: $\overline{C} = \sum_i (\overline{C}_i + 2\xi) + \xi$.

References

[AD89] H. Ahmadi, W. E. Denzel, “A survey of modern high-performance switching techniques”, IEEE Jour. on selected areas in Communications, Vol. 7, No. 7, pp. 1091-1102, Sept. 1989.

[AUD91] N. C. Audsley, “Optimal priority assignement and Feasibility of static priority tasks with arbitrary start times”, Univ. of York, Dept. of Computer Science, Report YCS 164, 1991.

[BA91] T. P. Baker, “Stack-based scheduling of real-time processes”, The Journal of Real-Time Systems, Vol. 3, pp. 67-99, 1991.

[BHR90] S. K. Baruah, R. R. Howell, L. E. Rosier, “Algorithms and Complexity Concerning the Preemptive Scheduling of Periodic Real-Time tasks on One Processor”, The Journal of Real-Time Systems, Vol. 2, pp. 301-324, 1990.

[BHR93] S. K. Baruah, R. R. Howell, and L. E. Rosier, “Feasibility Problems for Recurring Tasks on One Processor”. Theo. Comp. Sci., Vol. 118, pp. 3-20, 1993.

[BMR90] S. K. Baruah, A. K. Mok, L. E. Rosier, “Preemptively Scheduling Hard-Real-Time Sporadic Tasks on One Processor”, Proc. of the 11th IEEE Real-Time Systems Symposium, pp. 182-190, Dec. 1990.

[CL90] M. Chen, K. Link, “Dynamic priority ceiling: a concurrency control protocol for real-time systems”, The Journal of Real-Time Systems, Vol. 2, pp. 325-346, 1990.

[DER74] M. Dertouzos, “Control Robotics: the procedural control of physical prosessors”, Proc. of the IFIP congress, pp. 807-813, 1974.

[DRS89] J. W. Dolter, P. Ramanathan, K. G. Shin, “A microprogrammable VLSI routing controller for HARTS”, Int. Conf. on Comp. Design: VLSI in Computers, pp. 160-163, Oct. 1989.

[GRS96] L. George, N. Rivierre, M. Spuri, “Preemptive and non-preemptive real-time uniprocessor scheduling”, INRIA, Research Report 2966, Sept. 1996.

[HS96] J.-F. Hermant, M. Spuri, “End-To-End Response Times in Real-Time Distributed Systems”, PDCS’96, Proc. of the 9th ISCA/IEEE International Conference on Parallel and Distributed Computing Systems, pp. 413-417, Sept. 1996.

[IZ91] A. Indiresan, Q. Zheng, “Design and evaluating of a fast deadline scheduling switch for multicomputers”, RTCL working document, Dec. 91.

[JP86] M. Joseph, P. Pandya, “Finding response times in a real-time system”, BCS Comp. Jour., Vol. 29, No. 5, pp. 390-395, 1986.

[JSS94] K. Jeffay, D. L. Stone, F. D. Smith, “transport and display mechanisms for multimedia conferencing across packet switched networks”, Comp. Networks, 26:1281-1304, 1994.

[KLS93] D. I. Katcher, J. P. Lehoczky, J. K. Strosnider, “Scheduling models of dynamic priority schedulers”, Carnegie Mellon University, Research Report CMUCDS-93-4, Apr. 1993.

[KN80] K. H. Kim, M. Naghibdadeh, “Prevention of task overruns in real-time non-preemptive multiprogramming systems”, Proc. of Perf., ACM, pp. 267-276, 1980.

[LEH90] J. P. Lehoczky, “Fixed priority scheduling of periodic task sets with arbitrary deadlines”, Proc. of the 11th IEEE Real-Time Systems Symposium, pp. 201-209, Dec. 1990.

[LL73] C. L. Liu, J. W. Layland, “Scheduling Algorithms for Multiprogramming in a Hard-Real-Time Environment”, Journal of the Association for Computing Machinery, Vol. 20, No. 1, pp. 46-61, Jan. 1973.

[LM80] J. Y.-T. Leung and M. L. Merril, “A note on preemptive scheduling of periodic, real-time tasks”, Information Processing Letters, Vol. 11, No. 3, Nov 1980.

[LS95] L. Leboucher, Jean-Bernard Stéfani, “Admission control for end-to-end distributed bindings”, COST 231, Lectures Notes in Computer Science, Vol. 1052, pp. 192-208, Nov. 1995.

[LW82] J. Y.-T. Leung, J. Whitehead. “On the Complexity of Fixed-Priority Scheduling of Periodic, Real-Time Tasks”, Performance Evaluation, Vol. 2, pp. 237-250, 1982.

[MOK83] A. K. Mok, “Fundamental Design Problems for the Hard Real-Time Environments”, MIT, Ph. D. Dissertation, May 1983.

[RCM96] I. Ripoll, A. Crespo, A. K. Mok, “Improvement in Feasibility Testing for Real-Time Tasks,” The Journal of Real-Time Systems, Vol. 11, No. 1, pp 19-39, 1996.

[RSL90] R. Rajkumar, L. Sha, J. P. Lehoczky, “Priority inheritance protocols: an approach to real-time synchronisation”, IEEE Trans. on Comp., Vol. 39, No. 9, Sept. 1990.

[SNS89] H. Suzuki, H. Nagano, T. Suzuki, “Output buffer switch architecture for asynchronous transfer mode”, Proc. of Inter. Conf. on communications, pp. 411-415, Jun. 1989.

[SPU95] M. Spuri, “Earliest Deadline scheduling in real-time systems”, Doctorate Dissertation, Scuola Superiore S. Anna, Pisa, 1995.

[SPU96] M. Spuri, “Analysis of deadline scheduled real-time systems”, INRIA, Research Report 2772, Jan. 1996.

[SPU96-2] M. Spuri, “Holistic analysis for deadline scheduled real-time distributed systems”, INRIA, Research Report 2873, Apr. 1996.

[TBW94] K. W. Tindell, A. Burns, A. J. Wellings, “An Extendible Approach for Analyzing Fixed Priority Hard Real-Time Tasks”, The Journal of Real-Time Systems, Vol. 6, No. 2, pp. 133-152, Mar. 1994.

[TC95] K. W. Tindell, J. Clark, “Holistic Schedulability Analysis for Distributed Hard Real-Time Systems”, Microprocessing and Microprogramming, Vol. 40, No. 2-3, pp. 117-134, Apr. 1994.

[TO90] F. A. Tobagi, “Fast packet switch architectures for broadband integrated services digital networks”, Proc. of the IEEE, Vol. 78, No. 1, pp. 133-167, Jan. 1990.

[ZE93] Q. Zheng, “Real-Time fault tolerant communication in computer network”, Univ. of Michigan, Ph. D. Dissertation, 1993.

[ZS92] Q. Zheng, K. G. Shin, “Fault-tolerant real-time communication in distributed computing systems”, Proc. of the 22nd Symposium on Fault-Tolerant Computing, pp. 86-93, 1992.

[ZS94] Q. Zheng, K. G. Shin, “On the Ability of Establishing Real-Time Channels in Point-to-Point Packet-Switched Networks”, IEEE Transactions on Communications, Vol. 42, No. 2-3-4, 1994.

Unité de recherche INRIA Lorraine, technopôle de Nancy-Brabois, 615 rue du jardin botanique, BP 101, 54600 VILLERS-LÈS-NANCY
Unité de recherche INRIA Rennes, IRISA, Campus universitaire de Beaulieu, 35042 RENNES Cedex
Unité de recherche INRIA Rhône-Alpes, 46 avenue Félix Viallet, 38031 GRENOBLE Cedex 1
Unité de recherche INRIA Rocquencourt, domaine de Voluceau, Rocquencourt, BP 105, LE CHESNAY Cedex
Unité de recherche INRIA Sophia-Antipolis, 2004 route des Lucioles, BP 93, 06902 SOPHIA-ANTIPOLIS Cedex

Éditeur

Inria, Domaine de Voluceau, Rocquencourt, BP 105 LE CHESNAY Cedex (France)

ISSN 0249-6399