# An Algebraic Approach for Fixed-Priority Scheduling of Hard Real-time Systems with Exact Preemption Cost

Patrick Meumeu Yomsi, Yves Sorel

INSTITUT NATIONAL DE RECHERCHE EN INFORMATIQUE ET EN AUTOMATIQUE

# *An Algebraic Approach for Fixed-Priority Scheduling of Hard Real-Time Systems with Exact Preemption Cost*

Patrick Meumeu Yomsi — Yves Sorel

**N° 7702**

Août 2011

Embedded and Real Time Systems

Rapport de recherche

ISSN 0249-6399 ISRN INRIA/RR--7702--FR+ENG

INSTITUT NATIONAL DE RECHERCHE EN INFORMATIQUE ET EN AUTOMATIQUE

INRIA

centre de recherche PARIS - ROCQUENCOURT

# An Algebraic Approach for Fixed-Priority Scheduling of Hard Real-Time Systems with Exact Preemption Cost

Patrick Meumeu Yomsi , Yves Sorel

Theme : Embedded and Real Time Systems
Algorithmics, Programming, Software and Architecture
Équipe-Projet Aoste

Rapport de recherche n° 7702 — Août 2011 — 43 pages

**Abstract:** In this paper we study hard real-time systems composed of independent periodic preemptive tasks in the monoprocessor case. For such systems it is mandatory to satisfy all the constraints for all tasks. Although preemptive scheduling algorithms are able to successfully schedule some systems that cannot be scheduled by any non preemptive scheduling algorithm, the cost of preemption may not be negligible. Therefore, we propose to consider explicitly its exact cost in the schedulability conditions in order to avoid wasting resources and provide safety in terms of guaranteeing the right behavior of the system at run-time. Five main contributions are presented in this paper. First, we introduce a new model to describe and analyse hard real-time systems, which unifies in one framework different models such as Liu & Layland's and Mok's models. Second, we show the impact of considering the exact cost of preemption for each task on the schedulability analysis. Third, by using our model based on an algebraic approach we provide new schedulability conditions which take into account the exact cost due to the occurrence of each preemption. Fourth, in this case we propose an optimal algorithm in the sense of feasibility for choosing the fixed-priority of each task. Finally, we address the problem of reducing the number of preemptions.

**Key-words:** scheduling theory, schedulability analysis, hard real-time systems, exact preemption cost, real-time operating system cost, task models, fixed-priority, optimal priority assignement, number of preemptions

Centre de recherche INRIA Paris – Rocquencourt
Domaine de Voluceau, Rocquencourt, BP 105, 78153 Le Chesnay Cedex
Téléphone : +33 1 39 63 55 11 — Télécopie : +33 1 39 63 53 30

# Une approche algébrique pour l'ordonnancement à priorités fixes des systèmes temps réel durs avec prise en compte du coût exact de la préemption

**Résumé :** Dans ce papier nous étudions les systèmes temps réel durs composés de tâches préemptives indépendantes dans le cas mono-processeur. Pour de tels systèmes il est obligatoire de respecter toutes les contraintes auxquelles sont soumises toutes les tâches. Bien que des algorithmes d'ordonnancement préemptifs soient capables d'ordonnancer certains systèmes qui ne peuvent être ordonnancés avec aucun algorithme d'ordonnancement non préemptif, la préemption peut avoir un coût non négligeable. Nous proposons donc de considérer explicitement le coût exact de la préemption dans les analyses d'ordonnançabilité afin d'une part d'éviter du gaspillage de ressources et d'autre part de garantir un comportement correct lors de l'exécution en temps réel conforme aux analyses d'ordonnançabilité. Nous présentons dans ce papier cinq contributions. Premièrement nous introduisons un nouveau model pour décrire et analyser les systèmes temps réel durs qui unifie plusieurs modèles comme celui de Liu et Layland ou celui de Mok. Deuxièmement nous montrons l'impact de la prise en compte du coût exact de la préemption pour chaque tâche lors de l'analyse d'ordonnançabilité. Troisièmement en utilisant notre modèle fondé sur une approche algébrique nous proposons de nouvelles conditions d'ordonnançabilité qui prennent en compte le coût exact dû à l'occurence de chaque préemption. Quatrièmement, dans le cas ou l'on considère ce coût exact de la préemption, nous proposons un algorithme d'ordonnancement optimal, au sens de la faisabilité, pour choisir la priorité fixe de chaque tâche. Finalement nous étudions le problème consistant à réduire le nombre de préemptions.

**Mots-clés :** théorie de l'ordonnancement, analyse d'ordonnançabilité, système temps réel dur, coût exact de la préemption, coût du système d'exploitation temps réel, modèle de tâche, priorité fixe, assignation optimale de priorités, nombre de préemptions

# 1 Introduction

This paper focuses on hard real-time monoprocessor systems and addresses the scheduling problem of independent periodic preemptive tasks on specific hardware platforms without cache, pipeline, or complex internal architecture, when tasks are scheduled according to a fixed-priority scheduling policy. Up to now, many models and concepts necessary to describe and analyse hard real-time systems have been proposed. Rich and extensive state of the art work has been performed in order to justify the considered assumptions, etc. Over the years, preemptive periodic task models [1], [2], [3] have proven remarkably useful for the modelling of hard real-time systems — systems where the failure to satisfy any constraint may have disastrous consequences [4], [2], [5], [6]. Unfortunately, none of the previously proposed models has been designed to take into account an issue such as the exact cost of preemption [7], [8]. This weakness in current existing models may lead to erroneous conclusions in terms of schedulability decisions [9], [10]. This in turn can affect the correct behavior of the system at run-time, or in any case leads to resources being wasted [11], [12], [13]. In this paper, we introduce a new model which unifies in one framework different models such as Liu & Layland's and Mok's models to solve the general scheduling problem of hard real-time systems while taking into account the exact cost of the Real-Time Operating System (*RTOS*) [14]. Indeed, the preemption cost represents only one half of the RTOS cost which consists of two parts. A constant part, easy to determine, which corresponds to the cost of the scheduler, is associated with the activation and termination of tasks. The activation of a task includes the context switch necessary to make possible the preemption of another task and the choice of the task with the highest priority. Thus, this cost only depends on the number of tasks. A variable part, which is more difficult to determine, is associated with the occurrence of every preemption of the current task in order for it to resume later on. Thus, this cost depends on the number of preemptions for every task. In order to handle all the cases we consider *simultaneous* as well as *non-simultaneous* scenarios of first activation for all the tasks. There currently exists a wide gap between the scheduling theory and its implementation in operating systems running on specific hardware platforms. This paper provides a first step toward bridging the gap between real-time scheduling theory and implementation realities. Surely, this gap must be bridged for any meaningful validation of timing correctness. Throughout the paper, we assume that all timing characteristics are non negative integers, i.e. they are multiples of some elementary time interval (for example the "CPU tick", the smallest indivisible CPU time unit). The general scheduling problem in this case consists in filling the available time units left after the schedule of some tasks with the execution time units of the other tasks. Based on our new model, we are able to propose a schedulability analysis which uses a binary operation $\oplus$ whose operands are called *otasks*.

The remainder of the paper is structured as follows: section 2 gives definitions and properties used throughout this paper. It also presents the new *"otask model"* that will help us to perform the schedulability analysis of a hard real-time system. Section 3 provides the correspondence between typical periodic tasks and periodic otasks. Section 4 addresses the schedulability interval issue for a set of periodic otasks. Section 5 defines the scheduling operation $\oplus$ when priorities are assigned according to a fixed-priority scheduling policy such as *Rate Monotonic*, *Deadline Monotonic*, *Audsley*, etc. [1], [15]. Section 6 points out the impact of the preemption cost on both the schedulability analysis and the schedule. It shows that the priority assignement according to *Audsley's algorithm* [15], [16] is no longer applicable, and thus not optimal in terms of feasibility. Section 7 defines the scheduling operation $\oplus$ with the exact preemption cost. Section 8 presents an application of the proposed approach on a periodic task set. Section 9 shows the impact of the choice of priorities of the otasks on the schedulability analysis. Section 10 provides an algorithm for choosing priorities which is optimal. In the end, section 11 shows the consequence of reducing the number of preemptions for some otasks. We conclude and propose future work in section 12.

# 2 Definitions and properties

In this section, inspired by the language theory [17], [18], [19], we introduce some definitions and properties in order to provide the reader with the framework of our new model for hard real-time systems.

First of all, we must specify a generator containing all the legal symbols which can be used. We define a *generator* $\Sigma$ as a finite set of symbols. As such, everywhere in this paper, the generator that will be considered is $\Sigma = \{a, e\}$. In the context of scheduling theory, we always associate the symbol "$a$" to a time unit which is *available* and the symbol "$e$" to a time unit which is either *executed* or *executable* depending on the cases we will detail later. Now, given this *generator* $\Sigma$, we define an *otask* as an *ordered multiset* consisting of a certain number of elements (possibly zero) all belonging to $\Sigma$. The fundamental difference we make here between the notion of *ordered multiset* and the common notion of *multiset* [20], [21], [22], [23] is that the order of elements in an otask is important in our case. In fact, this will allow us to make the difference between any two otasks and in particular between otasks obtained from permutations of the elements of another otask. We define an *otask system* $\Gamma$ as a set of otasks on the generator $\Sigma$. It is worth noticing that the definition of an *otask system* is quite general and allows us to consider highly structured otask systems such as periodic, aperiodic or hybrid otask systems. The proximity of the terminologies used here to those in the literature expresses the idea of a relationship between them which we will detail later on. Indeed we will consider an otask with specific properties to represent a *real-time task*, and thus a *system of real-time tasks* is a particular *system of otasks*. We consider a unique set of otasks where each real-time task corresponds to one and only one otask. This correspondence will allow us to derive results on real-time task systems from those obtained on otask systems. In order to illustrate the previous definitions here are some examples of otasks on $\Sigma$: $\tau_1 = \{a\}$, $\tau_2 = \{e, a, a\}$, $\tau_3 = \{a, e, a\}$ and $\tau_4 = \{a, a, e, e, a\}$. The otask with no symbols, is denoted by $\Lambda$ ($\Lambda = \{\} = \emptyset$). $\Lambda$ is always an otask on $\Sigma$.

The set of all possible otasks on $\Sigma$ will be denoted by $\Sigma^*$. Hence, any otask system $\Gamma$ is necessarily a subset of $\Sigma^*$. If $\tau$ is an otask on $\Sigma$, then the *cardinal* of $\tau$ is the number of elements in $\tau$ and will be denoted by $|\tau|$. Now, let $x$ and $y$ be two otasks on $\Sigma$. The *concatenation* of $x$ and $y$ is the otask $xy$ obtained by writing the symbols of $x$ and the symbols of $y$ consecutively. As an example, if $x = \{a, e, e\}$ and $y = \{e, a\}$ then the otask $xy$ is given by $xy = \{a, e, e, e, a\}$ and the otask $yx$ is given by $yx = \{e, a, a, e, e\}$. We have $xy \neq yx$ because of the importance of the order of the elements in an otask. Consequently, the *concatenation* operation is not *Commutative*, i.e. there are otasks $x$ and $y$ on $\Sigma$ such that $xy$ is different from $yx$. However, this operation is *associative*, i.e. for all otasks $x$, $y$ and $z$ on $\Sigma$, $(xy)z = x(yz)$. The advantage of the *associativity* is that it allows us to concatenate several otasks without worrying about the order in which the concatenation operations are carried out. Note that for any otask $x$, the concatenation of $x$ and $\Lambda$ equals $x$, i.e. $x\Lambda = \Lambda x = x$. If there exist two otasks $w$ and $z$ such that $y = wxz$, then $x$ is called a *sub-otask* of $y$. We call the operation leading to obtain a sub-otask from an otask an *extraction*. In order to illustrate the latter definition, the otask $\{a, e, e\}$ is a sub-otask of each otask $\{e, e, a, e, e, a, a\}$ and $\{a, a, a, e, e\}$ since we have for example $\{e, e, a, e, e, a, a\} = \{e, e\}\{a, e, e\}\{a, a\}$, but is not a sub-otask of $\{a, e, a, e\}$.

From now on, the superscripts will represent the number of times an element is concatenated. These elements can either be simple otasks, or even otask systems. Thus, if $x \in \Sigma^*$, and $\Gamma \subseteq \Sigma^*$, then:

$$
\begin{aligned}
&x^k = xx \cdots x \\
&\Sigma^k = \Sigma\Sigma \cdots \Sigma = \{x \in \Sigma^* / |x| = k\} \\
&\Gamma^k = \Gamma\Gamma \cdots \Gamma
\end{aligned}
$$

where in each case, there are $k$ factors that are concatenated.

Keeping in mind that we are interested in scheduling periodic task sets, we consider otasks with an infinite cardinal. An otask where a sub-otask with a finite cardinal can be extracted and the otask is an infinite concatenation of this sub-otask from a certain element relatively to the first one, will be termed *periodic*. An otask with an infinite number of the symbol "$e$", and with a minimum number of symbols between two consecutive sequence of symbols "$e$", will be termed *sporadic*. Finally, an otask that contains a finite number of symbols "$e$" will be termed *aperiodic*. Hereafter, we will only consider *periodic* otasks. An otask system $\Gamma$ consisting only of periodic otasks will be called a *system of periodic otasks*.

So far, the notion of "time", which is central to scheduling theory, has not yet been considered in this paper. In order to overcome this, we consider an index along an oriented time axis which is a temporal reference for all otasks. On this axis, we identify un instant of reference $t_0$, for example we can choose $t_0 = 0$. Hence, in addition to the cardinal and the order of elements that can help us to differentiate between any two otasks on the generator $\Sigma$, the *start date* $r \in \mathbb{Z}$ and *end date* $f \in \mathbb{Z}$ of each otask w.r.t. the reference time $t_0$ are important and may also help us to differentiate between two otasks. We will denote by $\tau_{(r,f)}$ the otask on $\Sigma$ which starts at date $r$ and finishes at date $f$. This notation allows us to describe both finite and infinite cardinal otasks. If $f = r$ then $\tau = \Lambda$. If $f = \infty$ then $|\tau| = \infty$ and in this case we denote by convention $\tau_{(r,\infty)} = \tau_r$ as there is no ambiguity concerning the end date of $\tau$.

By definition, a *periodic otask* $\tau_{per}$ on $\Sigma$ is an infinite cardinal otask with a start date $r$ such that there exists a finite cardinal sub-otask $\tau$, and $\tau_{per}$ is an infinite concatenation of $\tau$ from a certain time instant $\beta \geq r$. We denote each periodic otask by:

$$\tau_{per} = \zeta_{(r,\beta)}\, \tau_\beta^\infty \tag{1}$$

where the integer $\beta$ represents the smallest time instant such that relation 1 is satisfied, $\zeta_{(r,\beta)}$ represents a finite cardinal otask called *initial part* of $\tau_{per}$ and $\tau_\beta^\infty$ represents an infinite cardinal otask, infinite concatenation of $\tau$ from date $\beta$ called *periodic part* of $\tau_{per}$.

For the sake of clarity, the infinite cardinal otask

$$\tau_1 = \{a, a, e, \underbrace{e, e, a, e, a}_{\tau}, \underbrace{e, e, a, e, a}_{\tau}, \cdots, \underbrace{e, e, a, e, a}_{\tau}, \cdots\}_0 = \{a, a, e\}_{(0.3)}\{e, e, a, e, a\}_3^\infty$$

is a periodic otask on $\Sigma = \{a, e\}$ whose initial part is $\{a, a, e\}_{(0.3)}$ and the periodic part is $\{e, e, a, e, a\}_3^\infty$. A periodic otask system on $\Sigma$ is given for example by the set

$$\Gamma = \{\{a, a, e\}_{(0.3)}\{e, e, a, e, a\}_3^\infty, \{a, e, e, a\}_{(12.16)}\{e, a, a, e, e, a, a, a\}_{16}^\infty\}$$

Given a periodic otask, there are two different forms in which it may be written: the *factored form* and the *developed form*. For the case of otask $\tau_1$, $\{a, a, e\}_{(0.3)}\{e, e, a, e, a\}_3^\infty$ will be referred to as the *factored form* and $\{a, a, e, \underbrace{e, e, a, e, a}_{\tau}, \underbrace{e, e, a, e, a}_{\tau}, \cdots, \underbrace{e, e, a, e, a}_{\tau}, \cdots\}_0$ will be referred to as the *developed form*. Now, let $\tau_{per}$ be a periodic otask on $\Sigma$. If the existence of the integer $\beta$ such that $\tau_{per} = \zeta_{(r,\beta)}(\tau)_\beta^\infty$ is unique by definition, the existence of the finite sub-otask $\tau$ is not unique. Indeed, the otask $\tau_1$ can also be written

$$\tau_1 = \{a, a, e\}_{(0.3)}\{\underbrace{e, e, a, e, a}_{\tau}, \underbrace{e, e, a, e, a}_{\tau}\}_3^\infty$$

and we have $5 = |\{e, e, a, e, a\}| \neq |\{e, e, a, e, a, e, e, a, e, a\}| = 10$.

We define the *pattern* of a periodic otask $\tau_{per} = \zeta_{(r,\beta)}\, \tau_\beta^\infty$ to be the *minimum* finite cardinal sub-otask $\tau_{min}$ of the otask $\tau$ such that $\tau_{per} = \zeta_{(r,\beta)}(\tau_{min})_\beta^\infty$. The *pattern* of any periodic otask always exists and is unique. Consequently, any two periodic otasks $\tau_1 = \zeta_{i_{(r_i,\beta_i)}}(\tau_i)_{\beta_i}^\infty$ and $\tau_2 = \zeta_{j_{(r_j,\beta_j)}}(\tau_j)^\infty$ on $\Sigma$ are *equal* if and only if on the one hand $\zeta_{i_{(r_i,\beta_i)}} = \zeta_{j_{(r_j,\beta_j)}}$ and on the other hand $\tau_1$ and $\tau_2$ have the same *pattern*. In the same vein, $\tau_1$ and $\tau_2$ are said to be *equivalent* if and only if they have the same developed form. In the remainder of this paper, any periodic otask $\tau_{per}$ on $\Sigma$ will be denoted by $\tau_{per} = \zeta_{(r,\beta)}\tau_\beta^\infty$ where $\zeta_{(r,\beta)}$ is the initial part, $\tau$ is the pattern and $T = |\tau|$ is the *period* of $\tau_{per}$.

We assume that the pattern $\tau$ of $\tau_{per}$ contains at least one symbol "$e$". Indeed, if $\tau = \{a\}$ then we consider that $\tau_{per}$ equals the finite cardinal otask $\zeta_{(r,\beta)}$, i.e. $\tau_{per} = \zeta_{(r,\beta)}\{a\}_\beta^\infty = \zeta_{(r,\beta)}$ which is not interesting as it is not periodic. Similarly, we assume that the pattern $\tau$ contains at least one symbol "$a$". Indeed, if it is not the case, by identifying the symbol "$e$" to a time unit which is either *executed* or *executable*, then the periodic otask $\zeta_{(r,\beta)}\{e\}_\beta^\infty$ corresponds to one whose elements do not change from a certain date, the date $\beta$. This situation is not interesting since our goal is to compose otasks by replacing the available time units of a otask, i.e. the symbols "$a$", by the executable time units of another otask, the symbols "$e$".

It is worth noticing that any periodic otask $\tau_{per} = \zeta_{(r,\beta)}\tau_\beta^\infty$ of period $T$ on $\Sigma$ is equivalent to an infinite number of periodic otasks of period $T$ on $\Sigma$. Indeed, since we have $\tau \neq \Lambda$ and $|\tau|$ finite by definition, then there exists a finite otask $x \neq \Lambda$ and a finite otask $y$ such that $|\tau| = |x| + |y|$ and $\tau_{(\beta,\beta+|\tau|)} = x_{(\beta,\beta+|x|)}y_{(\beta+|x|,\beta+|\tau|)}$. Thus:

$$\begin{array}{rcl}
\tau_{per} & = & \zeta_{(r,\beta)}\tau_\beta^\infty \\
 & = & \zeta_{(r,\beta)}(xy)_\beta^\infty \\
 & = & \zeta_{(r,\beta)}(xy)(xy)(xy)\cdots(xy)(xy)\cdots \\
 & = & \zeta_{(r,\beta)}x(yx)(yx)(yx)\cdots(yx)(yx)\cdots \\
 & = & \left(\zeta_{(r,\beta)}x_{(\beta,\beta+|x|)}\right)(yx)_{\beta+|x|}^\infty = \tau_{per}^{'}
\end{array}$$

The otask $\tau_{per}^{'}$ is periodic of period $T$ and its initial part is $\zeta_{(r,\beta)}x_{(\beta,\beta+|x|)}$ and its periodic part is $(yx)_{\beta+|x|}^\infty$. As the otasks $x$ and $y$ are arbitrary, we can repeat this process as many times as we want.

In order to rewrite $\tau_{per}$ in terms of a concatenation of its initial part and $k \in \mathbb{N}$ times its pattern and its periodic part, it is sufficient to set $x = \tau$ and $y = \Lambda$. Thus we have:

$$\tau_{per} = \zeta_{(r,\beta)} \underbrace{\tau\tau\tau \cdots \tau}_{k\ times} \tau^{\infty}_{\beta+k|\tau|} = \zeta_{(r,\beta)} \tau^{k}_{(\beta,\beta+k|\tau|)} \tau^{\infty}_{\beta+k|\tau|} \qquad (2)$$

where $\tau^{k}_{(\beta,\beta+k|\tau|)}$ denotes the finite otask beginning at date $\beta$ and ending at date $\beta + k|\tau|$, it corresponds to the otask $\tau$ concatenated $k$ times.

We will say that $\tau_{per}$ is in the *canonical form* if and only if the first element of the finite cardinal otask $\tau$ is the symbol "$e$". Thanks to everything we have presented up to now, the periodic otask $\tau_2 = \{a, a, e\}_{(0,3)}\{a, a, e, e, a, e, a\}^{\infty}_{3}$ is not in the *canonical form*, but it is *equivalent* to the otask $\tau_2^{'} = \{a, a, e, a, a\}_{(0,5)}\{e, e, a, e, a, a, e\}^{\infty}_{5}$ which is in the *canonical form*. This transformation is useful as the schedule will consist in replacing symbols "$a$" belonging to an otask by symbols "$e$" belonging to another otask. From now on, for each periodic otask $\tau_{per} = \zeta_{(r,\beta)}\tau^{\infty}_{\beta}$, we consider the equivalent periodic canonical otask $\tau^{'}_{per} = \zeta_{(r,\beta')}\tau^{\infty}_{\beta'}$ where $\beta^{'}$ is the smallest integer greater than $\beta$. We define the *relative deadline* $D$ of a periodic otask $\tau_{per} = \zeta_{(r,\beta)}\tau^{\infty}_{\beta}$ to be an integer value equal to $\beta - r$ for the initial part of $\tau_{per}$, and equal to the cardinal of a single sub-otask, possibly the pattern itself, containing at least all the symbols "$e$" of the pattern for the periodic part of $\tau_{per}$. $D$ is at most equal to $T$.

Since the definition of the relative deadline does not present any ambiguity for the initial part, it is not necessary to represent it graphically. However, for the periodic part, the deadline will be represented by a checkmark: $\lrcorner$. At this point we have everything we need to introduce our model of periodic otasks.

Figure 1 illustrates a periodic otask with relative deadline $D$ and period $T$. Each shaded box corresponds to the symbol "$e$" and each non-shaded box to the symbol "$a$" in the generator $\Sigma$. The initial part which is finite, is between the dates $r$ and $\beta$. The pattern of the periodic part, which repeated infinitely, is comprised between $\beta$ and $\beta + T$. In this figure, $D$ can take 5 possible values relative to the position of the last symbol "$e$" in the periodic part of the otask. These values are $\{T, T-1, T-2, T-3, T-4\}$. Note that in our model, the value of the relative deadline for the periodic part of any periodic otask is *less than or equal* to its period.

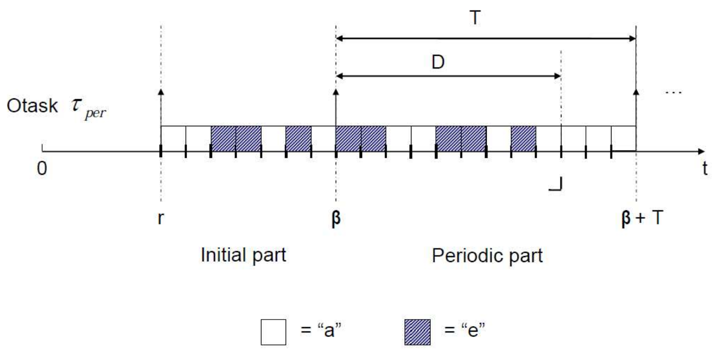


Figure 1: Model of a periodic otask.

We call *date of sub-activation of rank* $l$ *for the initial part* of $\tau_{per}$ denoted by $r^{l}_{in}$ (resp. *date of first sub-activation of rank* $l$ *for the periodic part* of $\tau_{per}$ denoted by $r^{l,1}_{p}$) the date of occurence of the first symbol "$e$" belonging to the sequence of rank $l$ in the initial part of $\tau_{per}$ relatively to $r$ (resp. the date of occurence of the first symbol "$e$" belonging to the sequence of rank $l$ of the pattern for the periodic part of $\tau_{per}$ relative to $\beta$). Identically, we call *sub-execution time of rank* $l$ *of the initial part* (resp. *sub-execution time of rank* $l$ *of the periodic part*) denoted $C^{l}_{in}$ (resp. $C^{l}_{p}$) the cardinal of the sub-otask which consists only of symbols "$e$" corresponding to the sequence of rank $l$. Figure 2 below clarifies these notions of *date of sub-activation* and *sub-execution time* for a periodic otask.

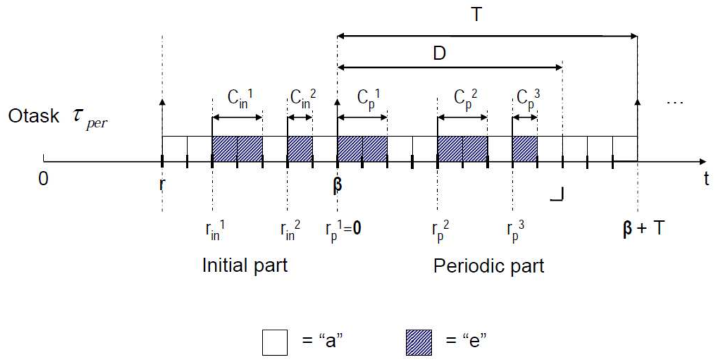


Figure 2: dates of sub-activations and sub-execution times for a periodic otask.

# 3 Model of periodic tasks

The study of a periodic real-time system by using a periodic otask system requires that each periodic task is describable uniquely as a periodic otask, that is to say that two distinct periodic tasks must match two distinct periodic otasks. In this section, we choose to build such a correlation by describing how each otask can be generated from the temporal characteristics of each real-time task and operations on simpler otasks.

Let $\Gamma_n = \{\tau_1, \tau_2, \cdots, \tau_n\}$ be a system of $n$ periodic tasks where $\tau_i = (r_i^1, C_i, D_i, T_i)$ and $C_i \le D_i \le T_i$. Based on the characteristics a periodic task, $r_i^1$ is the date of first activation, $C_i$ is the Worst Case Execution Time (WCET) without any approximation of the preemption cost, $D_i$ is the relative deadline and $T_i$ is the period of $\tau_i$. Relation 3 provides the periodic otask $o\tau_i$ which corresponds to the periodic task $\tau_i$.

$$o\tau_i = \left\{ \underbrace{\underbrace{\overbrace{e, e, \cdots, e}^{C_i}, a, a, a, \cdots, a}_{D_i \qquad \lrcorner}, a, \cdots, a, a}_{T_i} \right\}_{r_i^1}^{\infty} \tag{3}$$

where $r_i^1$ means that the pattern of otask $o\tau_i$ begins at the date $r_i^1$, corresponding to the date of first activation of task $\tau_i$. It thus follows that the otask $o\tau_i$ is a *particular otask* since it is *canonical*. It consists of a periodic part but has not got a non-trivial initial part, indeed its initial part equals $\Lambda$. Furthermore it is *regular*, that is to say that the pattern contains a single sequence of $C_i$ symbols "$e$" followed by a single sequence of $T_i - C_i$ symbols "$a$". The $D_i$ first symbols of the pattern represent the relative deadline of the otask $o\tau_i$. The value of $D_i$ delimits the interval before which $C_i$ symbols "$e$" of $o\tau_i$ must have been executed. In equality 3 each repetition of the pattern from the date $r_i^1$ corresponds to an instance of the task $\tau_i$. The pattern of rank $k$ starting at the date $r_i^k = r_i^1 + (k-1)T_i$ corresponds to the $k^{th}$ instance. Figure 3 illustrates a periodic task as a particular periodic otask given in figure 2.

Our main objective is the schedulability analysis of a system of periodic tasks by considering the corresponding otask system. For this purpose, we will combine otasks by using an *associative non commutative binary scheduling operation* that we denote by $\oplus$ in order to get an otask that will help us decide the schedulability.

$$\begin{array}{llll} \oplus : & \Sigma^* x \Sigma^* & \longrightarrow & \Sigma^* \\ & (x, y) & \longmapsto & z = x \oplus y \end{array}$$

When we write $x \oplus y$ where $x$ and $y$ are two periodic otasks, this means by convention that the left-hand operand (otask $x$) has a higher priority than the rignt-hand operand (otask $y$), therefore the operation $\oplus$ is *not commutative*, i.e.

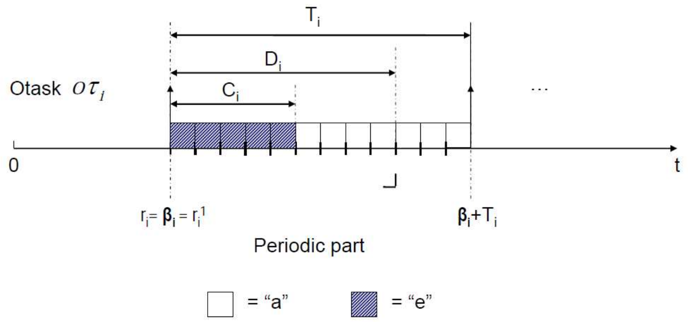


Figure 3: Correspondence between a periodic task and a periodic otask.

$x \oplus y \neq y \oplus x$. Now we have everything we need to explain the difference between *executed* and *executable* symbols "$e$". In the expression $x \oplus y$, the elements "$e$" of otask $x$ are called *executed* and those of otask $y$ are called *executable*. The intuitive idea that we propose to perform the operation $\oplus$ will therefore consist in replacing some elements "$a$" of a copy $z$ of otask $x$ by elements "$e$" of otask $y$, leading to the result $z = x \oplus y$. Although there are not enough "$a$" for all the *executable* "$e$", $x \oplus y = \Lambda$ is defined. When performing operation $\oplus$ the date of sub-activation of each sequence of executable symbols "$e$" of otask $y$ gives the earliest date of the symbols "$a$" to replace in otask $z$.

For any otask system $O\Gamma_n = \{o\tau_1, o\tau_2, \cdots, o\tau_n\}$ arranged according to decreasing priorities relative to an algorithm such as *Rate Monotonic* or *Deadline Monotonic*, since $\oplus$ is a binary operation, it will be used as many times as there are otasks in $O\Gamma_n$ in order to guarantee, or not, the schedulability of the system. The operations $\oplus$ will be applied from the otask with the highest priority to the otask with the lowest priority. This process will produce an intermediate result otask at each step which corresponds to the otask with the highest priority, i.e. the left-hand operand of the next the operation $\oplus$. Consequently, if $\mathcal{R}_n$ is the scheduling otask result of $O\Gamma_n$, then $\mathcal{R}_n$ is obtained by successive iterations:

$$\begin{cases} \mathcal{R}_1 = \Lambda \oplus o\tau_1 = o\tau_1 \\ \mathcal{R}_i = \mathcal{R}_{i-1} \oplus o\tau_i, \quad 2 \leq i \leq n \end{cases}$$

As such we have

$$\mathcal{R}_n = (((\Lambda \oplus o\tau_1) \oplus o\tau_2) \oplus \cdots \oplus o\tau_{n-1}) \oplus o\tau_n \tag{4}$$

that we will also denote by $\mathcal{R}_n = \bigoplus_{i=1}^{n} o\tau_i$.

The otask $o\tau_i$ will be said *schedulable* with respect to the considered priorities policy if and only if

$$\mathcal{R}_i \neq \Lambda \tag{5}$$

and the system $O\Gamma_n$ will be said *schedulable* if and only if all the otasks are schedulable. If this is not the case, then the system $O\Gamma_n$ is said *not schedulable*.

# 4 Schedulability interval

Since the otasks we are considering have an infinite cardinal, the goal of this section is to define a *finite interval* in order to perform the schedulability analysis. To do so, we need to extend the "schedulability interval theorem" of periodic tasks introduced by *J. Goossens* [24] to the case of periodic otasks.

**Theorem 1** *For a system $O\Gamma_n = \{o\tau_1, o\tau_2, \cdots, o\tau_n\}$, with $o\tau_i = \zeta_{i(r_i,\beta_i)}(\tau_{0,i})^{\infty}_{\beta_i}$, of $n$ periodic otasks arranged by decreasing priorities with respect to a fixed-priority scheduling policy, let $(s_i^{'})_{i\in\mathbb{N}^*}$ be the sequence defined by:*

$$\begin{cases} s_1^{'} = \beta_1 \\ \\ s_i^{'} = \beta_i + \left\lceil \dfrac{(s_{i-1} - \beta_i)^+}{T_i} \right\rceil \cdot T_i, \quad 2 \le i \le n \end{cases} \tag{6}$$

*If there exists a valid schedule of $O\Gamma_n$ until the time $s_n^{'} + H_n$ where $H_n = lcm\{T_i \mid i = 1, \cdots, n\}$, $T_i = |\tau_{0,i}|$ and $x^+ = max(x, 0)$, then this schedule is valid and periodic of period $H_n$ from $s_n^{'}$.*

**proof 1** *The proof of this theorem is similar to that performed by* J. Goossens *in his Ph.D. thesis [24].*

A direct consequence of the previous theorem is that in the case of a valid schedule, the otask $\mathcal{R}_i = \bigoplus_{j=1}^{i} o\tau_j$ is periodic of period $T_{\mathcal{R}_i} = lcm\{T_j \mid j = 1, \cdots, i\}$ from $s_i^{'}$. Thus, the interval which precedes $s_i^{'}$ necessarily contains the *transient phase*, corresponding to the initial part of $\mathcal{R}_i$ and the interval starting at time $s_i^{'}$ with length $H_i$ is isomorphic to the *permanent phase at level* $i$, corresponding to the periodic part of $\mathcal{R}_i$. Since the transient phase is finite due to the existence of the permanent phase which repeats identically from a certain time instant, the *earliest* start time (*minimum*) for the permanent phase will be derived from properties on periodic otasks. The permanent phase will be graphically depicted by using the *Dameid representation* [14].

# 5 Scheduling operation $\oplus$ and schedulability analysis

The goal of this section is to describe the steps to follow when performing the scheduling operation $\oplus$ for a set of periodic otasks when the preemption cost is zero, the extension which takes into account the exact preemption cost is described in section 7. For the sake of readability in the following mathematical layout, we introduce an intermediate operation for decomposing a periodic otask. This decomposition is usefull since it will help us, without loss of generality, to use only regular canonical otasks which are easier to manipulate when performing operation $\oplus$.

Let $\tau_{per} = \zeta_{(r,\beta)}\tau_\beta^\infty$ be a periodic otask on $\Sigma$ with the period $T$ and the relative deadline $D$. Let $n_p$ be the number of sequences of symbol "$e$" in the pattern of $\tau_{per}$. Let $r_p^l$, $l \in \{1, \cdots, n_p\}$ be the index such that $r_p^{l,1} = \beta + r_p^l$ is the time of first sub-activation of rank $l$ for the periodic part. The otask $\tau_{per}$ can be *decomposed* in $n_p$ canonical periodic otasks of period $T$ with the relative deadlines, for the periodic part of each *decomposed* otask, respectively given by:

$$D_{per,l} = D - r_p^l, \;\; l \in \{1, \cdots, n_p\} \tag{7}$$

As such, we obtain the decomposition of $\tau_{per}$ by using the following application $\pi$ from $\mathcal{P}$ to $\mathcal{P}^{n_p}$:

$$\begin{array}{rcl} \pi : \mathcal{P} & \longrightarrow & \mathcal{P}^{n_p} \\ \tau_{per} & \longmapsto & \tau_{per}^{\odot} = (x_1; x_2; \cdots; x_{n_p}) \end{array} \tag{8}$$

where $\mathcal{P}$ denotes the sub-set of all periodic otasks in $\Sigma^*$ and $x_l$ is the canonical periodic otask with the following characteristics: the periodic part starts at time $\beta + r_p^l$, the period is $T$ and the relative deadline for the periodic part is $D_{per,l} = D - r_p^l$. Since otask $x_l$ is regular, it contains a single sequence of symbols "$e$" with the cardinal of the corresponding sub-otask equal to $C_P^l$, this is the sub-execution time of rank $l$ for the periodic part of $\tau_{per}$. As an example, figure 5 shows the decomposition of the periodic otask illustrated in figure 4.

Now that the decomposition can be performed for any periodic otask, let us focus on operation $\oplus$ and the schedulability analysis of a periodic otask set.

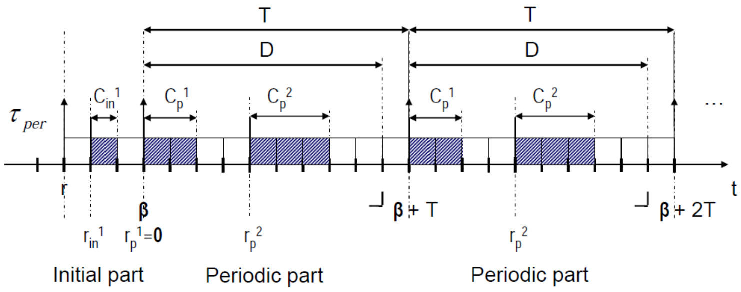


Figure 4: A periodic otask to be decomposed.

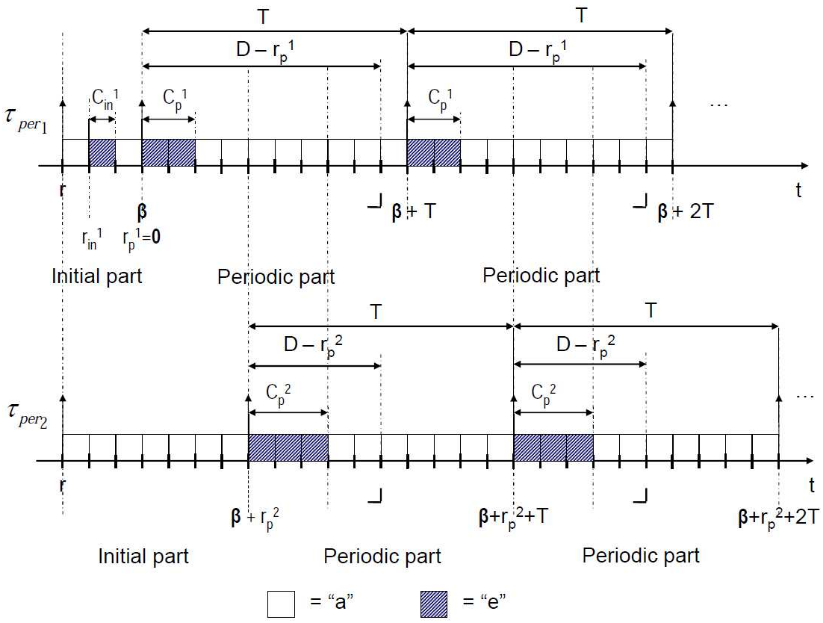


Figure 5: The decomposed periodic otask.

Let $O\Gamma_n = \{o\tau_1, o\tau_2, \cdots, o\tau_n\}$ be a system with $n$ periodic otasks. We recall that $x \oplus y$ means that the left-hand operand (that is, otask $x$) has a higher priority than the right-hand operand (that is, otask $y$). Now, it is worth noticing that operation $\oplus$ always involves two otasks with adjacent priorities (i.e. when it is written, there is no otask belonging to the system with a priority between that of $x$ and $y$). Thus we define by $\gamma_{ij} = r_j - r_i$ the *dephasing* between two otasks $\mathcal{R}_i = \zeta_{i(r_i,\beta_i)}(\tau_{0,i})_{\beta_i}^{\infty}$, which is the otask resulting of the application of $\oplus$ to the $i$ highest priorities otasks, and $o\tau_j = \zeta_{j(r_j,\beta_j)}(\tau_{0,j})_{\beta_j}^{\infty}$. We recall that when $\mathcal{R}_i \neq \Lambda$, then its period is $T_{\mathcal{R}_i} = lcm\{T_h \mid h = 1, \cdots, i\}$. With respect to the sign of the integer $\gamma_{ij} \in \mathbb{Z}$, the result of $z = x \oplus y$ differs.

The scheduling and the schedulability analysis of an otask system according to a given fixed-priority ordering for otasks consists of one main algorithm, which in turn calls three other algorithms. The second algorithm consists in *normalizing* the two operands, i.e. to reference them relative to the same origin and *rewriting* the left-hand operand to make it compatible with the right-hand operand. The third algorithm consists in *decomposing* the right-hand operand into a finite sequence of regular canonical otasks by applying application $\pi$ and *performing* the mesoidification and the scheduling operation $\oplus$ itself by calling Algorithm $4$, that is to say, replace the available time units "$a$" of the higher priority otask by the executable time units "$e$" of the lower priority otask. Details on these algorithms are given below (see Algorithms $Main$, 2, 3 and $4$). The Algorithm $4$ also performs the schedulability analysis of an otask. For the $j^{th}$ otask in the system ordered according to decreasing priorities, i.e. $o\tau_j$, we must compute the otask $\mathcal{R}_j = \mathcal{R}_i \oplus o\tau_j$ where $j = i+1$, $\mathcal{R}_i = \zeta_{i(r_i,\beta_i)}(\tau_{0,i})^{\infty}_{\beta_i} \neq \Lambda$ and $o\tau_j = \zeta_{j\,(r_j,\beta_j)}(\tau_{0,j})^{\infty}_{\beta_j}$. We set $\epsilon = min(r_i, r_j)$. Since $\epsilon$ always exists, we choose it to normalize $\mathcal{R}_i$ and $o\tau_j$.

**Algorithm Main: Scheduling and schedulability analysis of an otask system**

1: For the otask system $O\Gamma_n = \{o\tau_1, o\tau_2, \cdots, o\tau_n\}$ ordered according to decreasing priorities, the operations $\oplus$ will be applied from the otask with the highest priority to the otask with the lowest priority. Consequently, if $\mathcal{R}_n$ is the scheduling otask result of $O\Gamma_n$, then $\mathcal{R}_n$ is obtained by successive iterations:

$$\begin{cases} \mathcal{R}_1 = \Lambda \oplus o\tau_1 = o\tau_1 \\ \mathcal{R}_{i+1} = \mathcal{R}_i \oplus o\tau_{i+1}, \quad 1 \leq i < n \end{cases}$$

where $j = i+1 \in \{2, 3, \cdots, n\}$ is the index of the iteration for the computation of $\mathcal{R}_j$. As such we have $\mathcal{R}_n = (((\Lambda \oplus o\tau_1) \oplus o\tau_2) \oplus \cdots \oplus o\tau_{n-1}) \oplus o\tau_n$. At any iteration $j$, compute $\mathcal{R}_j = \mathcal{R}_i \oplus o\tau_j$ by calling Algorithm $2$, then decompose the right-hand operand by calling Algorithm 3 which in turn performs the mesoidification, the scheduling operation $\oplus$ and the schedulability analysis by calling Algorithm $4$. The otask $o\tau_j$ will be said *schedulable* with respect to the considered priorities policy if and only if $\mathcal{R}_j \neq \Lambda$.

2: The system $O\Gamma_n$ will be said *schedulable* if and only if all the otasks are schedulable. If this is not the case, then the system $O\Gamma_n$ is said *not schedulable*.

□

**Algorithm 2: Normalization and Rewriting**

1: Concatenate prefixes $\{a\}^{|\gamma_{ij}|}_{(\epsilon,r_i)}$ and $\{a\}^{|\gamma_{ij}|}_{(\epsilon,r_j)}$ respectively to otasks $\mathcal{R}_i$ and $o\tau_j$: this concatenation helps us to reference the two operands relative to the same origin, $\epsilon$.

$$\begin{aligned} \mathcal{R}_i \oplus o\tau_j &= \{a\}^{|\gamma_{ij}|}_{(\epsilon,r_i)}\zeta_{i(r_i,\beta_i)}(\tau_{0,i})^{\infty}_{\beta_i} \oplus \{a\}^{|\gamma_{ij}|}_{(\epsilon,r_j)}\zeta_{j\,(r_j,\beta_j)}(\tau_{0,j})^{\infty}_{\beta_j} \\ &= n\mathcal{R}_i \oplus no\tau_j \end{aligned}$$

$n\mathcal{R}_i$ and $no\tau_j$ are normalized.

2: Determine the instants $s_i^{'}$ of $\mathcal{R}_i$ and $s_j^{'}$ of $o\tau_j$ which delimit transient phase and permanent phase by using theorem 1 which leads to equation 9.

$$\begin{cases} s_i^{'} = \beta_i \\ \\ s_j^{'} = \beta_j + \left\lceil \dfrac{(s_i^{'} - \beta_j)^+}{T_j} \right\rceil \cdot T_j \end{cases} \tag{9}$$

The interval $[\epsilon, s_j^{'}]$ defines the transient phase and the interval $[s_j^{'}, s_j^{'} + lcm(T_{\mathcal{R}_i}, T_j)]$ defines the permanent phase.

3: Determine the following $\sigma_i$ which is the number of times the pattern of the left-hand operand is repeated by using equation 2 to rewrite $n\mathcal{R}_i$. This number will help us to make the left-hand operand compatible with right-hand operand. The first term of this computation is due to the dephasing between the operands and the second term is due to their periods.

$$\sigma_i = \left( \left\lceil \frac{s_j^{'} - s_i^{'}}{T_{\mathcal{R}_i}} \right\rceil - 1 \right) + \frac{lcm(T_{\mathcal{R}_i}, T_j)}{T_{\mathcal{R}_i}}$$

Thanks to the value of $\sigma_i$ and properties on periodic otasks, write $n\mathcal{R}_i$ as concatenation of a finite otask *ph-trans* with cardinal $|$*ph-trans*$| = s_j^{'} - \epsilon$ and a periodic otask *ph-perm* whose first time of activation is $s_j^{'}$ and whose period equals $lcm(T_{\mathcal{R}_i}, T_j)$. Otasks *ph-trans* and *ph-perm* respectively determine the transient phase and the permanent phase.

$$n\mathcal{R}_i = \textit{ph-trans ph-perm} \tag{10}$$

4: Since we have $|$*ph-trans*$| = s_j^{'} - \epsilon = (r_j - \epsilon) + (\beta_j - r_j) + (s_j^{'} - \beta_j)$, then the otask *ph-trans* can also be written as a concatenation of three finite cardinal otasks, *ph-trans* $= (tr_1)_{(\epsilon,r_j)}(tr_2)_{(r_j,\beta_j)}(tr_3)_{(\beta_j,s_j^{'})}$. Thus,

$$n\mathcal{R}_i = (tr_1)_{(\epsilon,r_j)}(tr_2)_{(r_j,\beta_j)}(tr_3)_{(\beta_j,s_j^{'})}\textit{ph-perm} \tag{11}$$

□

**Algorithm 3: Decomposition and schedulability analysis of otask $o\tau_j$**

1: At iteration $j = i + 1$, if $n_j$ is the number of sequences of symbol "$e$" in the pattern of the right-hand operand of $\oplus$, then $n\mathcal{R}_i \oplus no\tau_j = n\mathcal{R}_i \oplus no\tau_j^{\odot}$ where $no\tau_j^{\odot}$ is the decomposition of $no\tau_j$ by application $\pi$: $no\tau_j^{\odot} = \pi(no\tau_j) = (no\tau_{j,1}, no\tau_{j,2}, \cdots, no\tau_{j,n_j})$ obtained by using equation 8. Hence:

$$\begin{aligned}\mathcal{R}_j &= \mathcal{R}_i \oplus o\tau_j = n\mathcal{R}_i \oplus no\tau_j^{\odot} \\ &= (((n\mathcal{R}_i \oplus no\tau_{j,1}) \oplus no\tau_{j,2}) \oplus \cdots) \oplus no\tau_{j,l}) \cdots \oplus no\tau_{j,n_j}\end{aligned}$$

where $l \in \{1, 2, \cdots, n_j\}$ is the index of the sub-iteration for the computation of $\mathcal{R}_j$. Thus, we will have $\mathcal{R}_j = \mathcal{R}_{j,n_j}$. At any sub-iteration $l$ perform the mesoidification, the scheduling operation $\oplus$ and the schedulability analysis of otask $o\tau_j$ by calling Algorithm 4. If we have $\mathcal{R}_{j,l} = \Lambda$ for any $1 \leq l \leq n_j$ then the right-hand operand is said to be *not schedulable* and so is $o\tau_j$, otherwise the right-hand operand is said to be *schedulable*.

2: The *response times* of otask $o\tau_j$ when $\mathcal{R}_j = \mathcal{R}_{j,n_j} \neq \Lambda$ is written w.r.t. equation 12 always correspond to the cardinal of the prefix defined by the index of the last symbol "$a$" replaced. In other words, they are respectively given by $R_{r_j,\beta_j}$, $r_p^{n_j} + R_{tr_3,k}^{n_j}, k \in \{1, \cdots, \sigma_{tr_3}\}$, and $r_p^{n_j} + R_{perm,l}^{n_j}, l \in \{1, \cdots, \sigma_{perm}\}$ where we recall that $r_p^{n_j}$ denotes the release time of the last sequence of symbols "$e$" of $o\tau_j$. The *worst response time* of $o\tau_j$ is the maximum among all the different response times.

□

**Algorithm 4: Mesoidification, scheduling operation $\oplus$ and schedulability conditions**

Since we will repeat the same approach $n_j$ times meanwhile we are computing otask $\mathcal{R}_j$, let us explain the computation of $\mathcal{R}_{j,1} = n\mathcal{R}_i \oplus no\tau_{j,1}$.

1: For the left-hand operand of $\oplus$, write both the finite cardinal otasks $(tr_3)_{(\beta_j,s_j^{'})}$ and the pattern of otask *ph-perm* in equation 11 respectively as a concatenation of $\sigma_{tr_3} = \left\lceil \frac{(s_i^{'} - \beta_j)^+}{T_j} \right\rceil$ and $\sigma_{perm} = \frac{lcm(T_{\mathcal{R}_i}, T_j)}{T_j}$ finite cardinal otasks with cardinal $T_j$ each. We then obtain

$$n\mathcal{R}_i = (tr_1)_{(\epsilon,r_j)}(tr_2)_{(r_j,\beta_j)}\left(\mathcal{M}_{tr_3,1} \cdots \mathcal{M}_{tr_3,\sigma_{tr_3}}\right)\left(\mathcal{M}_{perm,1} \cdots \mathcal{M}_{perm,\sigma_{perm}}\right)_{s_j^{'}}^{\infty} \tag{12}$$

In equation 12 all otasks $\mathcal{M}_{tr_3,k}$, with ($k = 1, \cdots, \sigma_{tr_3}$) and all otasks $\mathcal{M}_{perm,l}$, with ($l = 1, \cdots, \sigma_{perm}$) are called $T_j$-*mesoids* because they involve the period of the right-hand operand $no\tau_j$. Each $T_j$-*mesoid* is an instance task $\tau_j$.

2: Determine all the so-called *deadline-bound-otasks*: $\mathcal{D}_{(r_j,\beta_j)}$, then $\mathcal{D}_{tr_3,k}$, with $(k = 1, \cdots, \sigma_{tr_3})$ and finally $\mathcal{D}_{perm,l}$, with $(l = 1, \cdots, \sigma_{perm})$ of each $T_j$-*mesoid* by using equation 12. This is performed by considering for $\mathcal{D}_{(r_j,\beta_j)}$ the finite cardinal otask $(tr_2)_{(r_j,\beta_j)}$ and for the other $\mathcal{D}_{tr_3,k}$, $\mathcal{D}_{perm,l}$, the prefix consisting of the $D_j$ first symbols of each $T_j$-*mesoid*.

3: Extract the *universes* which correspond to $\mathcal{U}_{r_j,\beta_j}$, then $\mathcal{U}_{tr_3,k}$, with $(k = 1, \cdots, \sigma_{tr_3})$ and $\mathcal{U}_{perm,l}$, with $(l = 1, \cdots, \sigma_{perm})$ consisting only of symbols "$a$" from each corresponding deadline-bound-otask.

4: If $\mathcal{E}_{r_j,\beta_j}$ is the sub-otask consisting only of symbols '$e$" of the initial part of $no\tau_{j,1}$, then otask $no\tau_{j,1}$ is schedulable without preemption cost if and only if

$$\begin{cases} |\mathcal{E}_{r_j,\beta_j}| \leq |\mathcal{U}_{r_j,\beta_j}|, \\ C_j^1 \leq \min\limits_{\substack{k \in \{1,\cdots,\sigma_{tr_3}\},\\ l \in \{1,\cdots,\sigma_{perm}\}}} (|\mathcal{U}_{tr_3,k}|, |\mathcal{U}_{perm,l}|) \end{cases} \tag{13}$$

5: If equation 13 holds, then otask $\mathcal{R}_{j,1} = n\mathcal{R}_i \oplus no\tau_{j,1}$ is obtained from equation 12 by replacing the first $|\mathcal{E}_{r_j,\beta_j}|$ symbols "$a$" of $(tr_2)_{(r_j,\beta_j)}$ and the $C_j^1$ first symbols "$a$" of each $T_j$-*mesoid* by symbols "$e$" respectively. We thus obtain $\mathcal{R}_{j,1} \neq \Lambda$ and then we can move on to the next iteration of operation $\oplus$ for otask $o\tau_j$ and so on until rank $n_j$.

6: In the process of replacing symbols, the *response times of rank* 1 for $o\tau_{j,1}$ match every time with the cardinal of the prefix defined by the index of the last symbol "$a$" replaced. We respectively denote by $R_{r_j,\beta_j}$, $R^1_{tr_3,k}, k \in \{1, \cdots, \sigma_{tr_3}\}$, and $R^1_{perm,l}, l \in \{1, \cdots, \sigma_{perm}\}$ the different values obtained. The *worst response time of rank* 1 for $o\tau_{j,1}$ is the maximum among all response times of rank 1.

7: If equation 13 does not hold, then $\mathcal{R}_{j,1} = \Lambda$. In this case, $o\tau_j$ and thus the otask system are declared *not schedulable* because of a deadline miss.

□

Until now, we considered the general scheduling problem of a set of periodic otasks, each consisting of an initial part and a periodic part, by using the binary operation $\oplus$ as many times as there are otasks. Our goal is to perform the schedulability analysis of a set of periodic tasks. We recall that a task is a particular otask which is regular and canonical with an intitial part equal to $\Lambda$. As such the right-hand operand of operation $\oplus$ is an otask which corresponds to a task when performing the schedulability analysis for a set of periodic tasks. In this case algorithm 4 is simplified in algorithm 5.

**Algorithm 5: Algorithm 4 simplified when the right-hand operand is a task**

1: For the left-hand operand of $\oplus$, write both the finite cardinal otasks $(tr_3)_{(\beta_j,s'_j)}$ and the pattern of otask *ph-perm* in equation 11 respectively as a concatenation of $\sigma_{tr_3} = \left\lceil \dfrac{(s'_i - \beta_j)^+}{T_j} \right\rceil$ and $\sigma_{perm} = \dfrac{lcm(T_{\mathcal{R}_i}, T_j)}{T_j}$ finite cardinal otasks with cardinal $T_j$ each. We then obtain

$$n\mathcal{R}_i = (tr_1)_{(\epsilon,r_j)}(tr_2)_{(r_j,\beta_j)} \left(\mathcal{M}_{tr_3,1} \cdots \mathcal{M}_{tr_3,\sigma_{tr_3}}\right) \left(\mathcal{M}_{perm,1} \cdots \mathcal{M}_{perm,\sigma_{perm}}\right)^{\infty}_{s'_j} \tag{14}$$

In equation 14 all otasks $\mathcal{M}_{tr_3,k}$, with $(k = 1, \cdots, \sigma_{tr_3})$ and all otasks $\mathcal{M}_{perm,l}$, with $(l = 1, \cdots, \sigma_{perm})$ are called $T_j$-*mesoids* because they involve the period of the right-hand operand $no\tau_j$. Each $T_j$-*mesoid* is an instance task $\tau_j$.

2: Determine all the so-called *deadline-bound-otasks*: $\mathcal{D}_{tr_3,k}$, with $(k = 1, \cdots, \sigma_{tr_3})$ and $\mathcal{D}_{perm,l}$, with $(l = 1, \cdots, \sigma_{perm})$ of each $T_j$-*mesoid* thanks to equality 12. This is performed by considering the prefix consisting of the $D_j$ first symbols of each $T_j$-*mesoid*.

3: Extract the *universes* which correspond to $\mathcal{U}_{tr_3,k}$, with $(k = 1, \cdots, \sigma_{tr_3})$ and $\mathcal{U}_{perm,l}$, with $(l = 1, \cdots, \sigma_{perm})$ consisting only of symbols "$a$" from each corresponding deadline-bound-otask.

4: Since $C_j$ corresponds to the execution time of the periodic part of otask $no\tau_j$, then otask $no\tau_j$ is schedulable without preemption cost if and only if

$$C_j \leq \min_{\substack{k \in \{1, \cdots, \sigma_{tr_3}\},\\ l \in \{1, \cdots, \sigma_{perm}\}}} (|\mathcal{U}_{tr_3,k}|, |\mathcal{U}_{perm,l}|) \tag{15}$$

5: If equation 15 holds, then otask $\mathcal{R}_j = n\mathcal{R}_i \oplus no\tau_j$ is obtained from the equality 12 by replacing the first $C_j$ symbols "$a$" of each $T_j$*-mesoid* by symbols "$e$". We thus obtain $\mathcal{R}_j \neq \Lambda$.

6: In the process of replacing symbols, the *response times* for $o\tau_j$ match every time with the cardinal of the prefix defined by the index of the last symbol "$a$" replaced. The *worst response time* for $o\tau_j$ is the maximum among all response times.

7: If equation 15 is not satisfied then $\mathcal{R}_j = \Lambda$. In this case, $o\tau_j$ and thus the otask system are declared *not schedulable* because of a deadline miss.

□

Thanks to a simple extension to a periodic otask, of the result on the occurrence of the worst response time of a periodic task in the permanent phase by *J. Goossens* in his thesis [24], we can take advantage of the modifications performed on Algorithm 4 to obtain Algorithm 5 in order to determine the worst response time of each otask without preemption cost relative to the schedulability analysis.

# 6 Impact of preemption cost

In section 5, we presented the scheduling operation $\oplus$ without preemption cost for a system of periodic otasks when the fixed-priorities are imposed to otasks (e.g. according to Rate Monotonic / RM, Deadline Monotonic / DM, Audsley or any other choice of priorities policy). In this section, we show the impact of preemption cost before extending the scheduling operation $\oplus$ to take the exact *RTOS cost* into account. To do so, we assume an *integrated interrupt event-driven scheduler* [25], [14] because in contrast to *nonintegrated interrupt event-driven scheduler* or to *interrupt timer-driven schedulers*, with this scheduler interrupts follow otasks priorities. Moreover, it does not introduce any blocking, nor does it introduce any priority inversion in the schedule, both due to activations of lower priority otask relative to the executing otask. Note that, since we intend to take the exact RTOS cost into account, execution and sub-execution times are considered without any approximation, unlike it is the case in the classical scheduling theory. Figure 6 details the cost of the RTOS for two tasks.

For the constant part of the RTOS cost which corresponds to the scheduler cost, we add to the sub-execution times the following quantities:

1. the time $C_i^{s_{in}}$ to save the context of the previous lower priority otask when an integrated interrupt, corresponding to each activation of otask $o\tau_i$, occurs, to execute the scheduler routine in order to choose the next otask to run, and to load the context of this otask,
2. the time $C_i^{s_{out}}$ to choose the next otask to run when an internal interrupt signals that otask $o\tau_i$ has completed its execution.

For the variable part of the RTOS cost which corresponds to the preemption cost, we denote by $\alpha_i$ the time to restore the context of otask $o\tau_i$ when a preemption has occurred. For otasks, a preemption corresponds to a switch from an "$a$" to an "$e$" in the left-hand operand of operation $\oplus$ while replacing "$a$"s of the left-hand operand by executable "$e$"s of the right-hand operand. It is worth noticing that the restoration of context for a task (resp. an otask) is *Atomic*, that is to say, if a task (resp. an otask) is preempted during this restoration of context which is partial in this case, then it will resume from the beginning of the restoration for its next execution. Since a task can be preempted several times during its execution, it is also the case for an otask when replacing symbols "$a$" by executable symbols "$e$". Therefore, it is obvious that a wrong quantification of additional symbols "$e$" which add to the number of executable symbols "$e$" of each otask when it is preempted, may lead to erroneous conclusions in terms of the schedulability of the system of tasks (resp. otasks).

Now, to distinguish the individuals symbols "$e$" related to a preemption to those corresponding to the otask itself, we denote them by "**ě**". Graphically, we denote "**ě**" by a *black slot* "■" in the linear representation of the schedule (*Gantt Chart*) and by a *black sector* in the circular representation of the schedule (*Dameid*).

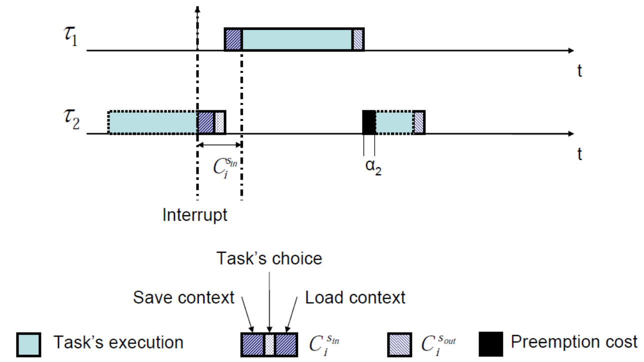


Figure 6: Details of the RTOS cost

We recall that we have the following results, concerning the impact of taking into account the exact RTOS cost [14] on both the schedule and the schedulability analysis, for a given task system $\Gamma_n$:

1. The *critical instant* of $\Gamma_n$ does not necessarily correspond to a simultaneous release for all tasks,

2. There does not exist any sequence of first release times for all tasks corresponding to the *critical instant* of $\Gamma_n$,

3. The *transient phase* can lead to the worst response time for a given task, whereas it usually occurs in the *permanent phase*,

4. The *Audsley algorithm* for choosing the task priorities is no longer *optimal*.

# 7 Scheduling operation $\oplus$ and schedulability analysis with preemption cost

In this section, we extend operation $\oplus$ in the case where the preemption cost is taken into account. The new obtained schedulability conditions will take into account the exact cost due to preemptions and these new conditions will always ensure the correct behavior of the system at run-time and eliminate waste of resources.

For the sake of clarity and without any loss of generality, even if this assumption is not realistic, we will consider *constant* preemption cost $\alpha$ for each task (resp. otask) in all examples. The cost $\alpha$ of one preemption may be arbitrarily large compared to execution times of tasks (resp. otasks). This high cost associated with the occurrence of a preemption will be used to illustrate the impact and especially the risk of not taking into account the temporal cost associated with preemptions in the schedulability conditions.

Because the *permanent phase* of a schedulable system repeats infinitely from a certain time instant $t$, then the interval preceding time $t$ necessarily contains the *transient phase*. Thus, it sufficient to perform the schedulability analysis in the interval preceding instant $t$ and in the interval $[t, t + H_n]$ where $H_n$ is the least common multiple of the periods of the tasks (resp. otasks). Since the worst response time of each task $\tau_j$ (resp. otask $o\tau_j$) may occur either in one or the other phase, we must consider all the instances (resp. all the $T_j$- *mesoids*) until the end of the first permanent phase for the schedulability analysis. Now, because our goal is to take into account the exact preemption cost, and as all tasks (resp. otasks) except the one with the highest priority can be preempted, then the analysis that we propose gives a schedulability condition for each task (resp. otask) individually with respect to those with a higher priority. If it seems clear that the number of preemptions of a task $\tau_j$ (resp. a canonical regular otask $o\tau_j$) may vary from one instance to another (resp. a $T_j$-*mesoid* to another), then it follows that the execution time also varies from one instance (resp. one $T_j$-*mesoid*) to another for the same task (resp. the same otask). For this reason, we introduce the *PET* (*Preempted Execution Time*) of a task $\tau_j$ (resp. of a canonical regular otask $o\tau_j$) in a given instance (resp. a given $T_j$-*mesoid*) as the sum of the execution time of the task (resp. the otask) and the exact cost due to preemption. Note that the PET is equivalent to the WCET augmented with the scheduler cost which is constant, and is without any preemption cost approximation when there are no preemptions.

**Notations** :

1. For a periodic task $\tau_j$ whose corresponding otask $o\tau_j$ is given by equation 3, $C_j^k$ denotes the *PET* of the $k^{th}$ instance (resp. the $T_j$-*mesoid* of rank $k$).

2. For a periodic otask $o\tau_i$ illustrated in figure 4, $C_i^{l,k}$ denotes the *PET* corresponding to the sub-execution time of rank $k \in \{1, \cdots, n_i\}$ in the $T_i$-*mesoid* of rank $l$.

Figure 7 illustrates the definition of the *PET* for a task (resp. an otask). It is therefore clear that its value depends on the number of preemptions in each instance (resp. in each $T_j$-*mesoid*). Its computation in a given instance (resp. in a $T_j$-*mesoid*) is explained in detail in Algorithm 10.

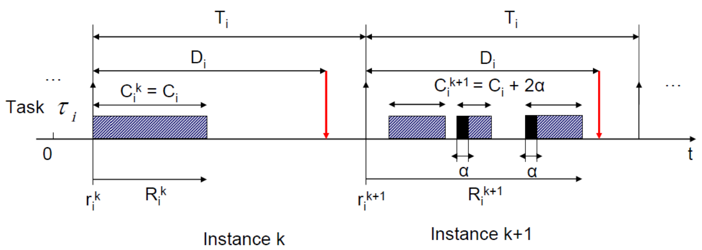


Figure 7: Illustration of the *PET* of a task (resp. of a regular canonical otask).

Since each regular canonical otask $o\tau_j$ corresponding to task $\tau_j$ can only be preempted by otasks with a higher priority, then the *hyperperiod at level* $j$ is given by $H_j = lcm\{T_l : o\tau_l \in hp(o\tau_j)\}$ where $T_l$ represents the period of otask $o\tau_l$ and $hp(o\tau_j)$ denotes the subset of otasks with a priority higher than that of $o\tau_j$. With this definition, the number of $T_j$-*mesoids* necessary to define the period of the otask result at level $j$, and therefore the permanent phase, when otask $o\tau_j$ is the second operand of the operation $\oplus$, is given by equation 16.

$$\sigma_{perm_j} = \frac{H_j}{T_j} = \frac{lcm\{T_l : o\tau_l \in hp(o\tau_j)\}}{T_j} \tag{16}$$

In the same vein of the notions we introduced in [8] in the case of periodic tasks, we here extend them in the case of periodic otasks. The *processor permanent utilisation factor without preemption cost* for a system $O\Gamma_n$ with $n$ periodic otasks where $o\tau_j = \zeta_{(r_j,\beta_j)}\tau_{\beta_j}^{\infty}$ has period $T_j$ is given by equation 17.

$$U_n = \sum_{j=1}^{n} \frac{C_j}{T_j} \text{ with } C_j = \sum_{k=1}^{n_j} C_{p_j}^k \tag{17}$$

In equation 17, $C_j$ and $n_j$ denote respectively the sum of the sub-execution times of rank $k \in \{1, \cdots, n_j\}$ and the number of sequences of symbol "$e$" in the pattern of otask $o\tau_j$. We recall that if $U_n > 1$, then whatever the algorithm used to select otask priorities of the system considered, this system can not be schedulable. This assertion implies that a necessary condition for the schedulability of the system is $U_n \leq 1$. In the latter case, we can define a $f_k$ function for each sub-execution time of rank $k \in \{1, \cdots, n_j\}$.

$$\begin{array}{rcl} f_k : \mathbb{N}^+ & \longrightarrow & \mathbb{N}^{+\sigma_{perm_j}} \\ C_{p_j}^k & \longmapsto & (C_{p_j}^{1,k}, C_{p_j}^{2,k}, \cdots, C_{p_j}^{\sigma_{perm_j},k}) \end{array}$$

where $C_{p_j}^{l,k}$ is the *PET* corresponding to the sub-execution time $C_{p_j}^k$ in the $T_j$-*mesoid* of rank $l \in \{1, \cdots, \sigma_{perm_j}\}$ of the permanent phase. Given this, the *exact permanent processor utilisation factor* $U_j^*$ of an otask $o\tau_j = \zeta_{(r_j,\beta_j)}\tau_{\beta_j}^{\infty}$ with period $T_j$, and with $n_j$ sequences of symbols "$e$" in his periodic part is given by equation 18.

$$U_j^* = \frac{C_j^*}{T_j} \text{ avec } C_j^* = \sum_{k=1}^{n_j} \frac{1}{\sigma_{perm_j}} \sum_{l=1}^{\sigma_{perm_j}} C_{p_j}^{l,k} \tag{18}$$

As such, the *exact permanent processor utilisation factor* of a system $\Gamma_n$ composed of $n$ periodic otasks $\tau_j$ with period $T_j$ is given by equation 19.

$$U_n^* = \sum_{j=1}^{n} U_j^* \text{ where } U_j^* = \frac{1}{T_j} \cdot \sum_{k=1}^{n_j} \frac{1}{\sigma_{perm_j}} \sum_{l=1}^{\sigma_{perm_j}} C_{p_j}^{l,k} \tag{19}$$

In equation 19, $U_j^*$ is given by equality 18. It therefore follows that the *exact permanent preemption cost* $\epsilon_n$ of a system consisting of $n$ periodic otasks is given by equation 20 when the system is schedulable.

$$\epsilon_n = U_n^* - U_n = \sum_{j=1}^{n} \left( \frac{1}{T_j} \cdot \sum_{k=1}^{n_j} \frac{1}{\sigma_j} \sum_{l=1}^{\sigma_j} C_{p_j}^{l,k} \right) - \sum_{j=1}^{n} \left( \frac{1}{T_j} \cdot \sum_{k=1}^{n_j} C_{p_j}^k \right) \tag{20}$$

Keeping in mind that the possible preemptions of otask $o\tau_j$ are identified by transitions $(a \to e)$ in each $T_j$-*mesoid* of the left-hand operand when performing operation $\oplus$ at level $j$, the computation of the *PET* of a regular canonical otask in a $T_j$-*mesoid* is summarized in Algorithm 10. In this algorithm, $\{\mathbf{expression}\}$ means a procedural step, the character "%" means the beginning of a comment to explain either a variable used in the algorithm or the main idea of a section of the algorithm.

Thanks to all the concepts that we have presented so far, since it is sufficient to consider the case where the left-hand operand of operation $\oplus$ is a periodic otask and the right-hand operand is a periodic otask with an initial part worthing $\Lambda$.

The scheduling and the schedulability analysis with the *exact preemption cost* of an otask system according to a given fixed-priority ordering for otasks consists of one main algorithm, which in turn calls three other algorithms. Details on these algorithms are given below (see Algorithms $Main2$, 7, 8 and 9).

**Algorithm Main2: Scheduling and schedulability analysis of an otask system with exact preemption cost**

1: For the otask system $O\Gamma_n = \{o\tau_1, o\tau_2, \cdots, o\tau_n\}$ arranged according to decreasing priorities, the operations $\oplus$ will be applied from the otask with the highest priority to the otask with the lowest priority. Consequently, if $\mathcal{R}_n$ is the scheduling otask result of $O\Gamma_n$, then $\mathcal{R}_n$ is obtained by successive iterations:

$$\begin{cases} \mathcal{R}_1 = \Lambda \oplus o\tau_1 = o\tau_1 \\ \mathcal{R}_{i+1} = \mathcal{R}_i \oplus o\tau_{i+1}, \quad 1 \leq i < n \end{cases}$$

where $j = i + 1 \in \{2, 3, \cdots, n\}$ is the index of the iteration for the computation of $\mathcal{R}_j$. As such we have $\mathcal{R}_n = (((\Lambda \oplus o\tau_1) \oplus o\tau_2) \oplus \cdots \oplus o\tau_{n-1}) \oplus o\tau_n$. At any iteration $j$, compute $\mathcal{R}_j = \mathcal{R}_i \oplus o\tau_j$ by calling Algorithm

$7$, then decompose the right-hand operand by calling Algorithm $8$, which in turn performs the mesoidification, the scheduling operation $\oplus$ with *exact preemption cost* and the schedulability analysis by calling Algorithm $9$. The otask $o\tau_j$ will be said *schedulable* with respect to the considered priorities policy if and only if $\mathcal{R}_j \neq \Lambda$.

2: The system $O\Gamma_n$ will be said *schedulable* if and only if all the otasks are schedulable. If this is not the case, then the system $O\Gamma_n$ is said *not schedulable*.

□

**Algorithm 7: Algorithm 2 modified to match the case where the left-hand operand is an otask and the right-hand operand has an initial part $\Lambda$**

1: Concatenate prefixes $\{a\}^{|\gamma_{ij}|}_{(\epsilon,r_i)}$ and $\{a\}^{|\gamma_{ij}|}_{(\epsilon,r_j^1)}$ respectively to otasks $\mathcal{R}_i = \zeta_{i(r_i,\beta_i)}(\tau_{0,i})^{\infty}_{\beta_i}$ and $o\tau_j = (\tau_{0,j})^{\infty}_{r_j^1}$: this concatenation helps us to reference the two operands relative to the same origin, $\epsilon = min(r_i, r_j^1)$.

$$\begin{array}{rcl} \mathcal{R}_i \oplus o\tau_j & = & \{a\}^{|\gamma_{ij}|}_{(\epsilon,r_i)}\zeta_{i(r_i,\beta_i)}(\tau_{0,i})^{\infty}_{\beta_i} \oplus \{a\}^{|\gamma_{ij}|}_{(\epsilon,r_j^1)}(\tau_{0,j})^{\infty}_{r_j^1} \\ & = & n\mathcal{R}_i \oplus no\tau_j \end{array}$$

$n\mathcal{R}_i$ and $no\tau_j$ are normalized.

2: Determine the instants $s_i^{'}$ of $\mathcal{R}_i$and $s_j^{'}$ of $o\tau_j$ which delimit transient phase and permanent phase by using theorem 1 which leads to equation 21.

$$\begin{cases} s_i^{'} = \beta_i \\ \\ s_j^{'} = r_j^1 + \left\lceil \dfrac{(s_i^{'} - r_j^1)^+}{T_j} \right\rceil \cdot T_j \end{cases} \tag{21}$$

The interval $[\epsilon, s_j^{'}]$ defines the transient phase and the interval $[s_j^{'}, s_j^{'} + lcm(T_{\mathcal{R}_i}, T_j)]$ defines permanent phase.

3: Determine the following $\sigma_i$ which is the number of times the pattern of the left-hand operand is repeated by using equation 2 to rewrite $n\mathcal{R}_i$. This number will help us to make the left-hand operand compatible with right-hand operand. The first term of this computation is due to the dephasing between the operands and the second term is due to their periods.

$$\sigma_i = \left( \left\lceil \frac{s_j^{'} - s_i^{'}}{T_{\mathcal{R}_i}} \right\rceil - 1 \right) + \frac{lcm(T_{\mathcal{R}_i}, T_j)}{T_{\mathcal{R}_i}}$$

Thanks to the value of $\sigma_i$ and properties on periodic otasks, write $n\mathcal{R}_i$ as concatenation of a finite otask *ph-trans* with cardinal $|$*ph-trans*$| = s_j^{'} - \epsilon$ and a periodic otask *ph-perm* whose first time of activation is $s_j^{'}$ and whose period equals $lcm(T_{\mathcal{R}_i}, T_j)$. Otasks *ph-trans* and *ph-perm* respectively determine the transient phase and the permanent phase.

$$n\mathcal{R}_i = \textit{ph-trans ph-perm} \tag{22}$$

4: Since $|$*ph-trans*$| = s_j^{'} - \epsilon = (r_j^1 - \epsilon) + (s_j^{'} - r_j^1)$, then the otask *ph-trans* can also be written as a concatenation of two finite cardinal otasks, *ph-trans* $= (tr_1)_{(\epsilon,r_j^1)}(tr_2)_{(r_j^1,s_j^{'})}$. Thus,

$$n\mathcal{R}_i = (tr_1)_{(\epsilon,r_j^1)}(tr_2)_{(r_j^1,s_j^{'})}\textit{ph-perm} \tag{23}$$

□

**Algorithm 8: Algorithm 3 modified when the exact preemption cost is taken into account**

1: At iteration $j = i + 1$, if $n_j$ is the number of sequences of symbol "$e$" in the pattern of the right-hand operand of $\oplus$, then $n\mathcal{R}_i \oplus no\tau_j = n\mathcal{R}_i \oplus no\tau_j^{\odot}$ where $no\tau_j^{\odot}$ is the decomposition of $no\tau_j$ by application $\pi$. Thanks to our restriction, $n_j = 1$ and $C_j$ is the execution time of otask $no\tau_j$. Hence:

$$\mathcal{R}_j = \mathcal{R}_i \oplus o\tau_j = n\mathcal{R}_i \oplus no\tau_j$$

Perform the mesoidification, the scheduling operation $\oplus$ and the schedulability analysis of otask $o\tau_j$ by calling Algorithm 9. If we have $\mathcal{R}_j = \Lambda$ then $o\tau_j$ is said to be *not schedulable*, otherwise $o\tau_j$ is said to be *schedulable*.

2: The *response times* of otask $o\tau_j$ when $\mathcal{R}_j \neq \Lambda$ is written w.r.t. equation 24 always correspond to the cardinal of the prefix defined by the index of the last symbol "$a$" replaced. The *worst response time* of $o\tau_j$ is the maximum among all the different response times.

□

**Algorithm 9: Algorithm 5 modified when the exact preemption cost is taken into account**

1: For the left-hand operand of $\oplus$, write both the finite cardinal otasks $(tr_2)_{(r_j^1, s_j')}$ and the pattern of otask *ph-perm* in equation 23 respectively as a concatenation of $\sigma_{tr_2} = \left\lceil \frac{(s_i' - r_j^1)^+}{T_j} \right\rceil$ and $\sigma_{perm} = \frac{lcm(T_{\mathcal{R}_i}, T_j)}{T_j}$ finite cardinal otasks with cardinal $T_j$ each. We then obtain

$$n\mathcal{R}_i = (tr_1)_{(\epsilon, r_j^1)} \left(\mathcal{M}_{tr_2,1} \cdots \mathcal{M}_{tr_2,\sigma_{tr_2}}\right) \left(\mathcal{M}_{perm,1} \cdots \mathcal{M}_{perm,\sigma_{perm}}\right)^{\infty}_{s_j'} \tag{24}$$

In equation 24 all otasks $\mathcal{M}_{tr_2,k}$, with $(k = 1, \cdots, \sigma_{tr_2})$ and all otasks $\mathcal{M}_{perm,l}$, with $(l = 1, \cdots, \sigma_{perm})$ are called $T_j$*-mesoids* because they involve the period of the right-hand operand $no\tau_j$. Each $T_j$*-mesoid* is an instance task $\tau_j$.

2: Determine all the so-called *deadline-bound-otasks*: $\mathcal{D}_{tr_2,k}$, with $(k = 1, \cdots, \sigma_{tr_2})$ and $\mathcal{D}_{perm,l}$, with $(l = 1, \cdots, \sigma_{perm})$ of each $T_j$*-mesoid* thanks to equation 24. This is performed by considering the prefix consisting of the $D_j$ first symbols of each $T_j$*-mesoid*.

3: Extract the *universes* which correspond to $\mathcal{U}_{tr_2,k}$, with $(k = 1, \cdots, \sigma_{tr_2})$ and $\mathcal{U}_{perm,l}$, with $(l = 1, \cdots, \sigma_{perm})$ consisting only of symbols "$a$" from each corresponding deadline-bound-otask.

4: Since $C_j$ corresponds to the execution time of the periodic part of otask $no\tau_j$, then otask $no\tau_j$ is *potentially schedulable* when the exact preemption cost is taken into account if and only if

$$C_j \leq \min_{\substack{k \in \{1, \cdots, \sigma_{tr_2}\},\\ l \in \{1, \cdots, \sigma_{perm}\}}} (|\mathcal{U}_{tr_2,k}|, |\mathcal{U}_{perm,l}|) \tag{25}$$

5: In each $T_j$*-mesoid*, compute the values of the *PET* $C_j^k$ with $(k = 1, \cdots, \sigma_{tr_2})$ and $C_j^l$ with $(l = 1, \cdots, \sigma_{perm})$ of otask $no\tau_j$ thanks to Algorithm 10.

6: Otask $no\tau_j$ is *schedulable* with exact preemption cost taken into account if and only if

$$\begin{cases} C_j^k \leq |\mathcal{U}_{tr_2,k}|, & \forall k \in \{1, \cdots, \sigma_{tr_2}\} \\ C_j^l \leq |\mathcal{U}_{perm,l}|, & \forall l \in \{1, \cdots, \sigma_{perm}\} \end{cases} \tag{26}$$

7: If equation 26 holds, then otask $\mathcal{R}_j = n\mathcal{R}_i \oplus no\tau_j$ is obtained from equality 24 by replacing the first $C_j^k$ with $k = 1, \cdots, \sigma_{tr_2}$ (resp. the first $C_j^l$ with $l = 1, \cdots, \sigma_{perm}$) symbols "$a$" of each $T_j$*-mesoid* by symbols "$e$". We thus obtain $\mathcal{R}_j \neq \Lambda$.

8: In the process of replacing symbols, the *response times* for $o\tau_j$ match every time with the cardinal of the prefix defined by the index of the last symbol "$a$" replaced. The *worst response time* for $o\tau_j$ is the maximum among all response times.

9: If equation 26 is not satisfied then $\mathcal{R}_j = \Lambda$. In this case, $o\tau_j$ and thus the otask system are declared *not schedulable* because of a deadline miss.

□

**Algorithm 10: Computation of the PET**

This algorithm provides the reader with the steps to follow for the computation of the PET in a *mesoid*. The main idea for the computation of the PET of otask $o\tau_j$ in a $T_j$*-mesoid* is a fixed-point algorithm based on a recursive function. The principle consists in adding $\alpha_j$ time units to the remaining execution time of otask $o\tau_j$ in that $T_j$*-mesoid* when a preemption has occured. The computation stops as soon as either two consecutive values of the PET are equal or there exists a time instant such that the current value of the PET is larger than the cardinal of the universe associated to that $T_j$*-mesoid*. In this latter case, otask $o\tau_j$ is not schedulable due to a deadline miss.

{***Execution of the algorithm***}
thanks to index $\phi$, move on to the first symbol "$a$" in the $T_j$*-mesoid*
call the function PET($C_j$, 0, 0)

{***Recursive PET computation in a*** $T_j$**-mesoid**}

{***Variables***}
% $rExecution$: remaining execution time to be scheduled,
% $rPreemption$: remaining time related to preemption cost to be scheduled,
% $vPET$: current value of the PET in the $T_j$-mesoid,
% $\mathcal{U}$: universe of the otask in the $T_j$-mesoid,
% $\phi$: index of enumeration of symbols in the $T_j$-mesoid,
% $\alpha$: cost of one preemption.

**function** PET($rExecution$, $rPreemption$, $vPET$)
% It remains to replace at least $rExecution + rPreemption$ symbols "$a$" by symbols "$e$" in the $T_j$*-mesoid* and we have replaced $vPET$ symbols "$a$".
**if** $rExecution = 0$ **then**
  % the otask is *schedulable* in this $T_j$*-mesoid*, the PET is $vPET$.
  return $vPET$
**else**
  **if** $vPET \leq |\mathcal{U}|$ **then**
    **if** $\phi$ = symbol "$e$" **then**
      $rPreemption \leftarrow \alpha$
      thanks to index $\phi$, skip the current sequence of symbols "$e$"
    **end if**
    **if** $rPreemption = 0$ **then**
      replace symbol "$a$" by an executable symbol "$e$": related to the otask
      thanks to index $\phi$, move on to the next symbol
      PET($rExecution - 1$, $rPreemption$, $vPET + 1$)
    **else**

replace symbol "$a$" by a symbol "**ě**": related to the preemption cost
thanks to index $\phi$, move on to the next symbol
PET($rExecution, rPreemption - 1, vPET + 1$)
**end if**
**else**
% the otask is *not schedulable* in this $T_j$*-mesoid*, due to a deadline miss.
return "*error*"
**end if**
**end if**
□

# 8 Application to a periodic task set

In this section we will apply the operation $\oplus$ when the exact preemption cost is taken into account for the schedulability analysis of a system of periodic real-time tasks. Let us consider the system $\Gamma_3 = \{\tau_1, \tau_2, \tau_3\}$ of 3 independent periodic preemptive tasks where $\tau_1$ is the task with the highest priority and $\tau_3$ is the task with lowest priority. The tasks' characteristics are summarized in table 1.

Table 1: Tasks' caracteristics

| Tasks | $r_i^1$ | $C_i$ | $D_i$ | $T_i$ |
|---|---|---|---|---|
| $\tau_1$ | 0 | 3 | 7 | 15 |
| $\tau_2$ | 5 | 2 | 6 | 6 |
| $\tau_3$ | 3 | 4 | 10 | 10 |

We recall that otasks $o\tau_1$, $o\tau_2$ and $o\tau_3$ which respectively correspond to tasks $\tau_1$, $\tau_2$ and $\tau_3$ w.r.t. equation 3 are given by the following relations:

$$\begin{cases} o\tau_1 = \{e, e, e, a, a, a, \underline{a}\rfloor a, a, a, a, a, a, a\}_0^\infty \\ \\ o\tau_2 = \{e, e, a, a, a, a\}_5^\infty \\ \\ o\tau_3 = \{e, e, e, e, a, a, a, a, a, a\}_3^\infty \end{cases}$$

Otask $o\tau_1$ is the otask with the highest priority and $o\tau_3$ is the otask with lowest priority. We consider the cost of one preemption to be one time unit for all tasks, that is to say $\alpha_i = \alpha = 1$ time unit, $i = 1, 2, 3$.
The processor permanent utilization factor without preemption cost is given by $U_3 = 3/15 + 2/6 + 4/10 = 28/30 = 0.9333$ and the result $\mathcal{R}_3$ of the scheduling problem is obtained by successive iteration:

$$\begin{cases} \mathcal{R}_1 = \Lambda \oplus o\tau_1 = o\tau_1 \\ \mathcal{R}_i = \mathcal{R}_{i-1} \oplus o\tau_i, \quad i = 2, 3 \end{cases}$$

**First iteration:** Computation of $\mathcal{R}_2 = o\tau_1 \oplus o\tau_2$

The normalized otasks $no\tau_1$ and $no\tau_2$ corresponding to otasks $o\tau_1$ and $o\tau_2$ are respectively given by $no\tau_1 = \{e, e, e, a, a, a, \underline{a}\rfloor\ a, a, a, a, a, a, a\}_0^\infty$ for the left-hand operand and $no\tau_2 = \{a, a, a, a, a\}_{(0,5)}\{e, e, a, a, a, a\}_5^\infty$ for the right-hand operand of $\oplus$. Thanks to equation 21, we have:

$$\begin{cases} s_1^{'} = 0 \\ s_2^{'} = 5 + \left\lceil \dfrac{(0-5)^+}{6} \right\rceil \cdot 6 = 5 \end{cases}$$

We have $H_2 = lcm(15,6) = 30$, thus interval $[0,5]$ determines the transient phase and interval $[5,35]$ determines the permanent phase.

The computation of $\sigma_1$ gives $\sigma_1 = \left(\left\lceil \dfrac{5-0}{15} \right\rceil - 1\right) + \dfrac{lcm(15,6)}{15} = 2$. Hence, we can write $no\tau_1$ as a concatenation of a finite cardinal otask *ph-trans* of cardinal $|$*ph-trans*$| = 5$ and a periodic part *ph-perm* whose instant of first activation is 5 and whose period is $lcm(15,6) = 30$. We then obtain:

$$\begin{aligned} no\tau_1 &= \{e,e,e,a,a,a,a,a,a,a,a,a,a,a,a\}_0^{\infty} \\ &= \{e,e,e,a,a\}_{(0,5)}\ \{a,a,a,a,a,a,a,a,a,a,e,e,e,a,a,a,a,a,a,a,a,a,a,a,a,e,e,e,a,a\}_5^{\infty} \end{aligned}$$

By identification relative to equality 23, the normalized otask $no\tau_1$ is of the form $no\tau_1 = (tr_1)_{(\epsilon,r_2^1)}(tr_2)_{(r_2^1,s_2^{'})}$*ph-perm* where the involved terms are respectively given by $(tr_1)_{(\epsilon,r_2^1)} = \{e,e,e,a,a\}_{(0,5)}$, $(tr_2)_{(r_2^1,s_2^{'})} = \Lambda$ since $s_2^{'} = r_2^1$ and then

$$\textit{ph-perm} = \{a,a,a,a,a,a,a,a,a,a,e,e,e,a,a,a,a,a,a,a,a,a,a,a,a,e,e,e,a,a\}_5^{\infty}$$

To perform the mesoidification of $no\tau_1$, the computations of $\sigma_{tr_2}$ and $\sigma_{perm_2}$ respectively give $\sigma_{tr_2} = \left\lceil \dfrac{(0-5)^+}{6} \right\rceil = 0$ and $\sigma_{perm_2} = \dfrac{lcm(15,6)}{6} = 5$. Consequently, we have:

$$\begin{aligned} no\tau_1 &= \{e,e,e,a,a\}_{(0,5)}\ \{a,a,a,a,a,a,a,a,a,a,e,e,e,a,a,a,a,a,a,a,a,a,a,a,a,e,e,e,a,a\}_5^{\infty} \\ &= \{e,e,e,a,a\}_{(0,5)}(\{a,a,a,a,a,a\}\{a,a,a,a,e,e\}\{e,a,a,a,a,a\}\{a,a,a,a,a,a\}\{a,e,e,e,a,a\})_5^{\infty} \\ &= \{e,e,e,a,a\}_{(0,5)}\ (\mathcal{M}_{perm,1}\mathcal{M}_{perm,2}\mathcal{M}_{perm,3}\mathcal{M}_{perm,4}\mathcal{M}_{perm,5})_5^{\infty} \end{aligned}$$

where the 6-*mesoids* $\mathcal{M}_{perm,1}$, $\mathcal{M}_{perm,2}$, $\mathcal{M}_{perm,3}$, $\mathcal{M}_{perm,4}$ and $\mathcal{M}_{perm,5}$ are given by $\mathcal{M}_{perm,1} = \{a,a,a,a,a,a\}$, $\mathcal{M}_{perm,2} = \{a,a,a,a,e,e\}$, $\mathcal{M}_{perm,3} = \{e,a,a,a,a,a\}$, $\mathcal{M}_{perm,4} = \{a,a,a,a,a,a\}$ et $\mathcal{M}_{perm,5} = \{a,e,e,e,a,a\}$. Each 6-*mesoid* corresponds to an instance of task $\tau_2$. Since $D_2 = T_2$ for task $\tau_2$, then the *deadline-bound-otasks* are also given by $\mathcal{D}_{perm,1} = \{a,a,a,a,a,a\}$, $\mathcal{D}_{perm,2} = \{a,a,a,a,e,e\}$, $\mathcal{D}_{perm,3} = \{e,a,a,a,a,a\}$, $\mathcal{D}_{perm,4} = \{a,a,a,a,a,a\}$ and $\mathcal{D}_{perm,5} = \{a,e,e,e,a,a\}$. Hence the universes $\mathcal{U}_{perm,1}$, $\mathcal{U}_{perm,2}$, $\mathcal{U}_{perm,3}$, $\mathcal{U}_{perm,4}$ and $\mathcal{U}_{perm,5}$ are:

$$\begin{cases} \mathcal{U}_{perm,1} &= \mathcal{D}_{perm,2} = \{a,a,a,a,a,a\} \\ \mathcal{U}_{perm,2} &= \{a,a,a,a\} \\ \mathcal{U}_{perm,3} &= \{a,a,a,a,a\} \\ \mathcal{U}_{perm,4} &= \mathcal{D}_{perm,4} = \{a,a,a,a,a,a\} \\ \mathcal{U}_{perm,5} &= \{a\}\{a,a\} = \{a,a,a\} \end{cases}$$

Thanks to the above equalities, $|\mathcal{U}_{perm,1}| = 6$, $|\mathcal{U}_{perm,2}| = 4$, $|\mathcal{U}_{perm,3}| = 5$, $|\mathcal{U}_{perm,4}| = 6$ and $|\mathcal{U}_{perm,5}| = 3$. Since $no\tau_2 = \{a,a,a,a,a\}_{(0,5)}\{e,e,a,a,a,a\}_5^{\infty}$, $C_2 = 2$ and consequently otask $o\tau_2$ is *potentially schedulable* as equation 25 holds.

$$C_2 = 2 \leq 3 = \min_{1 \leq j \leq 5}(|\mathcal{U}_{perm,j}|)$$

Thanks to Algorithm 5, we can compute the *PET* in each 6-*mesoid* and through application $f_5$ we obtain $f_5(2) = (2, 2, 2, 2, 3)$. Hence, $\mathcal{R}_2 = no\tau_1 \oplus no\tau_2$ is given by:

$$\begin{aligned}
\mathcal{R}_2 &= no\tau_1 \oplus no\tau_2 \\
&= \{e,e,e,a,a\}_{(0,5)}(\{a,a,a,a,a,a\}\{a,a,a,a,e,e\}\{e,a,a,a,a,a\}\{a,a,a,a,a,a\}\{a,e,e,e,a,a\})_5^{\infty} \\
&\qquad \oplus \{a,a,a,a,a\}_{(0,5)}\{e,e,a,a,a,a\}_5^{\infty} \\
&= \{e,e,e,a,a\}_{(0,5)}(\{\mathbf{e},\mathbf{e},a,a,a,a\}\{\mathbf{e},\mathbf{e},a,a,e,e\}\{e,\mathbf{e},\mathbf{e},a,a,a\}\{\mathbf{e},\mathbf{e},a,a,a,a\}\{\mathbf{e},e,e,e,\check{\mathbf{e}},\mathbf{e}\})_5^{\infty} \\
&= \{e,e,e,a,a\}_{(0,5)}\{e,e,a,a,a,a,e,e,a,a,e,e,e,e,e,a,a,a,e,e,a,a,a,a,e,e,e,e,\check{e},e\}_5^{\infty}
\end{aligned}$$

From the response times point of view, we have $R_2^1 = |\{e,e\}| = 2$, $R_2^2 = |\{e,e\}| = 2$, $R_2^3 = |\{e,e,e\}| = 3$, $R_2^4 = |\{e,e\}| = 2$ and $R_2^5 = \{e,e,e,e,\check{e},e\} = 6$ where $R_2^k$ denotes the response time of task $\tau_2$ in its $k^{th}$ instance. The worst response time of task $\tau_2$ thus is $R_2 = 6$ and it is obtained for the first time in the fifth instance of $\tau_2$.

**<u>Second iteration:</u>** Computation of $\mathcal{R}_3 = \mathcal{R}_2 \oplus o\tau_3$

Thanks to the result of the previous itération, the normalized otasks $n\mathcal{R}_2$ and $no\tau_3$ corresponding respectively to periodic otasks $\mathcal{R}_2$ and $o\tau_3$ are given by:
$n\mathcal{R}_2 = \{e,e,e,a,a\}_{(0,5)}\{e,e,a,a,a,a,e,e,a,a,e,e,e,e,e,a,a,a,e,e,a,a,a,a,e,e,e,e,\check{e},e\}_5^{\infty}$ and
$no\tau_3 = \{a,a,a\}_{(0,3)}\{e,e,e,e,a,a,a,a,a,a\}_3^{\infty}$. After a computation similar to that of the first iteration, we can conclude that otask $no\tau_3$ is schedulable and the result is given by:

$$\begin{aligned}
\mathcal{R}_3 &= n\mathcal{R}_2 \oplus no\tau_3 \\
&= \{e,e,e,a,a\}_{(0,5)}\ \{e,e,a,a,a,a,e,e,a,a,e,e,e,e,e,a,a,a,e,e,a,a,a,a,e,e,e,e,\check{e},e\}_5^{\infty} \\
&\qquad \oplus \{a,a,a\}_{(0,3)}\{e,e,e,e,a,a,a,a,a,a\}_3^{\infty} \\
&= \{e,e,e\}_{(0,3)}\{a,a,e,e,a,a,a,a,e,e\}_{(3,13)}\ \{a,a,e,e,e,e,e,a,a,a,e,e,a,a,a,a,e,e,e,e,\check{e},e,e,e,a,a,a,a,e,e\}_{13}^{\infty} \\
&\qquad \oplus \{a,a,a\}_{(0,3)}\{e,e,e,e,a,a,a,a,a,a\}_3^{\infty} \\
&= \{e,e,e\}_{(0,3)}\{a,a,e,e,a,a,a,a,e,e\}_{(3,13)}\ (\{a,a,e,e,e,e,e,a,a,a\}\{e,e,a,a,a,a,e,e,e,e\}\{\check{e},e,e,e,a,a,a,a,e,e\})_{13}^{\infty} \\
&\qquad \oplus \{a,a,a\}_{(0,3)}\{e,e,e,e,a,a,a,a,a,a\}_3^{\infty} \\
&= \{e,e,e\}_{(0,3)}\{\mathbf{e},\mathbf{e},e,e,\check{\mathbf{e}},\mathbf{e},\mathbf{e},a,e,e\}_{(3,13)}\ (\{\mathbf{e},\mathbf{e},e,e,e,e,e,\check{\mathbf{e}},\mathbf{e},\mathbf{e}\}\{e,e,\mathbf{e},\mathbf{e},\mathbf{e},\mathbf{e},e,e,e,e\}\{\ \check{\mathbf{e}},e,e,e,\mathbf{e},\mathbf{e},\mathbf{e},\mathbf{e},e,e\})_{13}^{\infty} \\
&= \{e,e,e,e,e,e,e,\check{e},e,e,a,e,e\}_{(0,13)}\ \{e,e,e,e,e,e,e,\check{e},e,e,e,e,e,e,e,e,e,e,e,e,\check{e},e,e,e,e,e,e,e,e,e\}_{13}^{\infty} \\
&= \{e,e,e,e,e,e,e,\check{e},e,e,a\}_{(0,11)}\ \{e,e,e,e,e,e,e,e,e,\check{e},e,e,e,e,e,e,e,e,e,e,e,e,\check{e},e,e,e,e,e,e,e\}_{11}^{\infty}
\end{aligned}$$

Thanks to Algorithm 10, the *PET* in each 10-*mesoid* equals $(5)$ for the 10-*mesoid* which starts at time 3, then through application $f_{10}$ is $f_{10}(4) = (5, 4, 4)$. From the point of view of the response times, $R_3^1 = |\{e,e,e,e,\ \check{e},e,e\}| = 7$, $R_3^2 = |\{e,e,e,e,e,e,e,\ \check{e},e,e\}| = 10$, $R_3^3 = |\{e,e,e,e,e,e\}| = 6$ and $R_3^4 = |\{\check{e},e,e,e,e,e,e,e\}| = 8$ where $R_3^2 = 10$, $R_3^3 = 6$ and $R_3^4 = 8$ denotes the response times of task $\tau_3$ in the permanent phase. Consequently, the worst response time of task $\tau_3$ is $R_2 = 10$ and it is obtained for the first time in the second instance of $\tau_3$. Figure 9 summarizes the results of this example and figure 8 illustrates the curve of the response time of each otask, and thus of each task relative to release times. In figure 9 the permanent phase corresponds to the highlighted zone of the schedule and the transient phase corresponds to the interval preceding that zone.

The *exact permanent processor utilisation factor* of the system is given by:

$$U_3^* = \frac{3}{15} + \frac{1}{6} \cdot \frac{(2+2+2+2+3)}{5} + \frac{1}{10} \cdot \frac{(5+4+4)}{3} = 1$$

Thanks to equation 20, the *exact permanent preemption cost* $\epsilon_3$ is given by:

$$\epsilon_3 = U_3^* - U_3 = 1 - \frac{28}{30} = 0.0667 \equiv 6.67\%$$

After this relatively simple example (the system consists of only 3 otasks and the preemption cost is $\alpha = 1$ time unit for all otasks) to explain the application of operation $\oplus$, figure 10 illustrates the summary of the analysis of a more

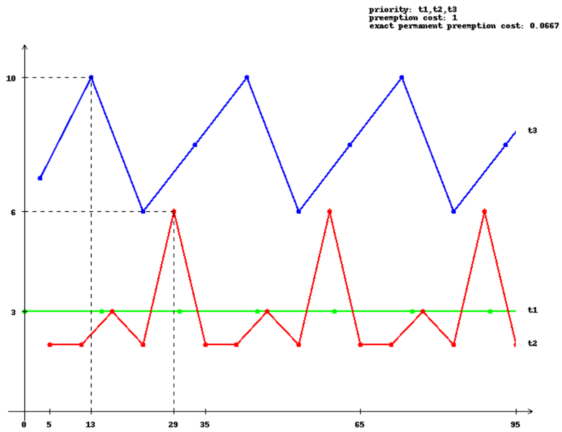


Figure 8: Response times of each task relative to release times.

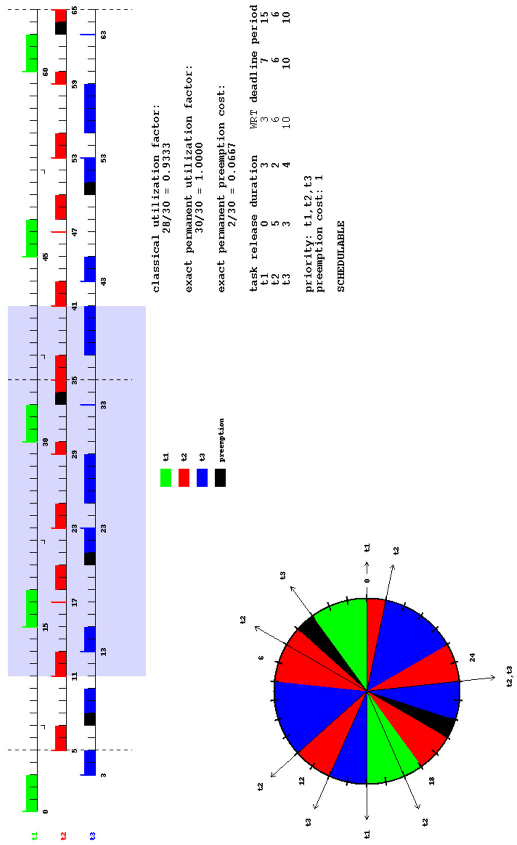

Figure 9: Results when the exact preemption cost is taken into account.

complex example by using our approach. The system under consideration consists of $10$ otasks. Characteristics and priorities of each otask are summarized in the table of the same figure. The preemption cost varies from one otask to another: for example, it is $\alpha_6 = 1$ time unit for otask $t_6$ and $\alpha_{10} = 3$ time units for otask $t_{10}$. The least common multiple (*lcm*) of the periods of all otasks is $H_{10} = 3600$ time units. The transient phase of the system begins at time $0$ and ends on time $1517$. Thus the permanent phase begins at time $t_{initial} = 1517$ and ends at time $t_{end} = 5117$ (blue zone). The classical permanent processor utilisation factor of conventional processor (without preemption cost) is $U_{10} = 80.08\%$, the exact permanent processor utilisation factor is $U^*_{10} = 94.83\%$, so the exact permanent preemption cost is $\epsilon_{10} = 14.75\%$. Figure 11 illustrates the curve of the response time relative to release times for each otask. In this figure, we see for example that the worst response time ($258$ time units) of otask $t_{10}$ is achieved for the first time at its fourth activation while that of otask $t_7$ ($108$ time units) is achieved for the first time at its tenth activation. The worst response time of otask $t_9$ ($350$ time units) is achieved for the first time at its ninth activation.

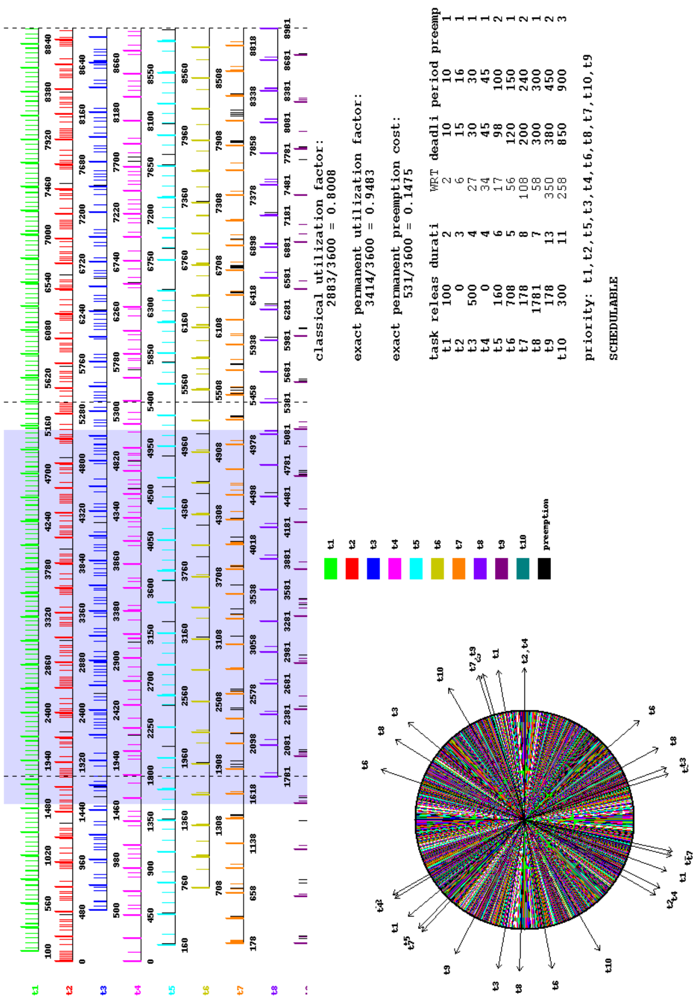

Figure 10: Schedule of $10$ tasks with exact preemption cost.

Figure 12 illustrates a *zoom* window in the linear representation (Gantt Chart) of the schedule given in figure 10. In this window, we clearly see the response time of otask $t_7$ whose activation occurs at the time $3058$. For this specific

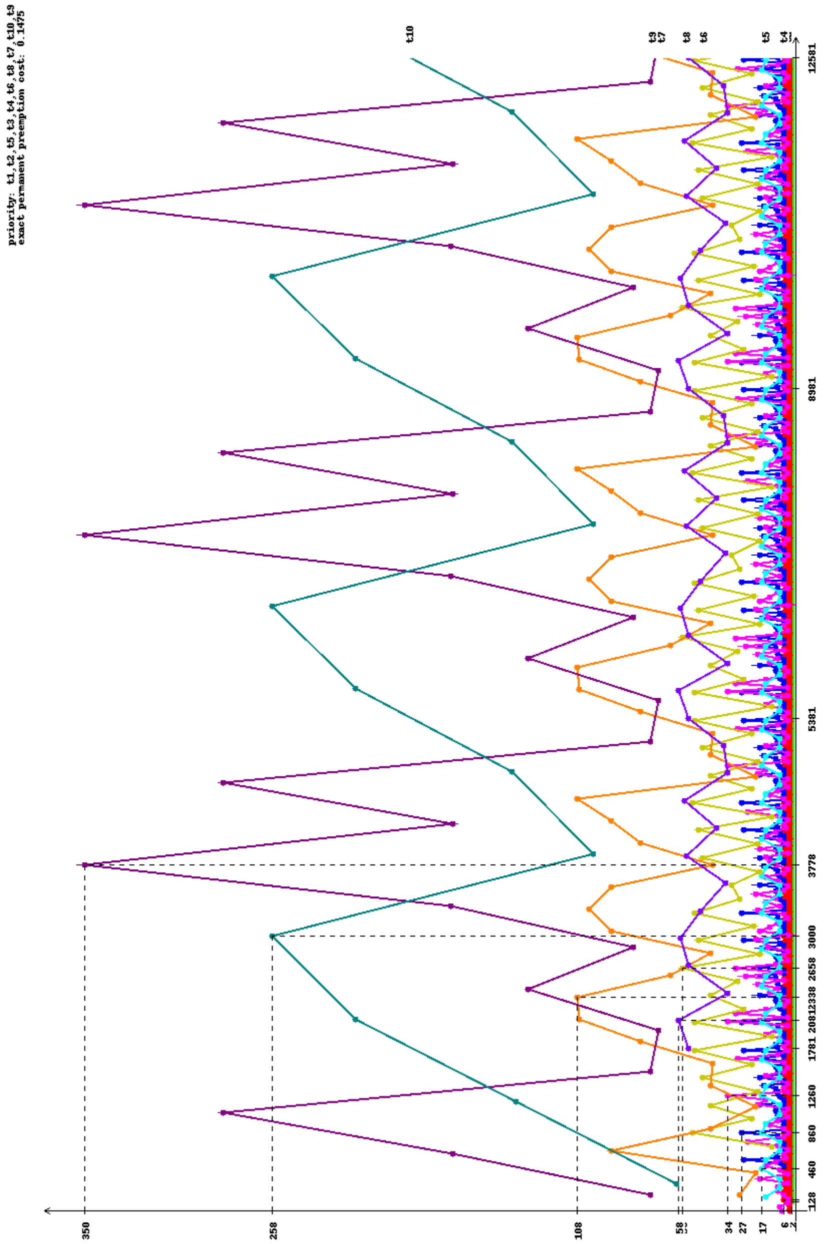


Figure 11: Response time of each task relative to release times.

activation, we note that after the first preemption of the otask at time $3060$, it can only resume its execution at time $3096$. This is due, on the one hand to otasks with a higher priority than that of $t_7$ and also, and on the other hand to the cost of each preemption. The atomic restoration of the context of an otask is illustrated for example at time $3130$. Indeed the cost of one preemption of otask $t_7$ in the example is $\alpha_7 = 2$ time units and at this time, otask $t_7$ is preempted by otask $t_1$ while it is restoring its context. The preemption takes place and at time $3132$, the restoration of the context of otask $t_7$ is resumed to the beginning again.

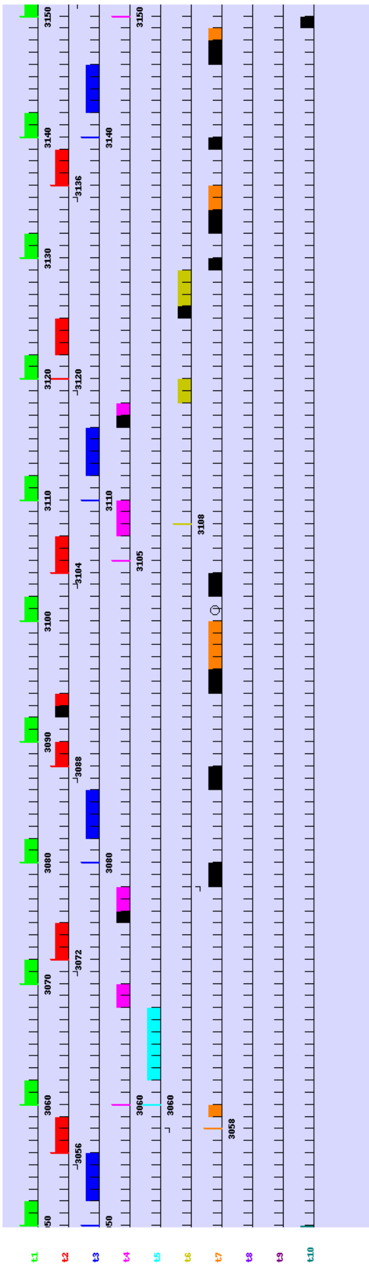

Figure 12: Zoom in the Gantt Chart.

# 9 Impact of otask priorities choice

Let $O\Gamma_n = \{o\tau_1, o\tau_2, \cdots, o\tau_n\}$ be a system consisting of $n$ periodic otasks. A *priorities choice* for otasks in $O\Gamma_n$ is a *permutation* of elements of $O\Gamma_n$ that orders otasks from the one with the highest priority to the one with the lowest priority. We denote a priorities choice by $\mathcal{S} =< o\tau_{g(1)}, o\tau_{g(2)}, \cdots, o\tau_{g(n)} >$ where the priority of $o\tau_{g(l)}$ is greater than that of $o\tau_{g(r)}$ as soon as $l < r$ and $\{g(1), \cdots, g(n)\}$ is the image of the set $\{1, \cdots, n\}$ by a permutation $g$. $\mathcal{S}$ is *schedulable* if and only if

$$\mathcal{R}_i = \bigoplus_{j=1}^{i} o\tau_{g(j)} \neq \Lambda, \ \ \forall\, i \in \{1, \cdots, n\}$$

We have already stressed in section 6 on the impact of otask priorities on the schedulability analysis which has two consequences. First, the result of the schedulability analysis of an otask according to a priorities choice $\mathcal{S}$ depends not only on all otask having a higher priority but also on their exact priorities. Second, if an otask is not schedulable, we can make it schedulable by lowering its priority. Both statements make the *Audsley* algorithm for choosing priorities no more *optimal* when the exact preemption cost is taken into account.

Since there may be several different priorities choices which make the system $O\Gamma_n$ schedulable, it remains to define a criterion for choosing a particular priorities choice among the choices (at least one) that satisfy all constraints during the application of Algorithms Main2, 7, 8, 9 and 10. Since we are interested in preemption cost we consider for this criterion the priorities choice which leads to the lower exact permanent preemption cost. If two schedulable priorities choices lead to the same exact permanent preemption cost, we consider the one that satisfies another criterion (e.g. the one for which the response times are lower, etc.). As the *Audsley* algorithm is no more optimal, the "*naïve*" algorithm that consists in testing all possible priorities permutations is very costly in terms of complexity since it requires $n!$ tests where $n$ is the number of otasks in the considered system. To circumvent this difficulty, we considerably reduce the number of priorities tests performed by using theorem 2 and theorem 3.

**Theorem 2** *Let $O\Gamma_n = \{o\tau_1, o\tau_2, \cdots, o\tau_n\}$ be a system with $n$ periodic otasks. We consider the priorities choice $\mathcal{S}$ = $< o\tau_{g(1)}, \cdots, o\tau_{g(i)}, o\tau_{g(i+1)}, \cdots, o\tau_{g(n)} >$ where $g$ is a permutation of $\{1, 2, \cdots, n\}$. If $\mathcal{S}$ is* not schedulable *due to otask $o\tau_{g(i)}$, i.e.*

$$\mathcal{R}_i = \bigoplus_{k=1}^{i} o\tau_{g(k)} = \Lambda \ \textit{and} \ \forall j < i, \ \ \mathcal{R}_j \neq \Lambda$$

*then any priorities choice $\mathcal{S}^{'}$ = $< o\tau_{g(1)}, \cdots, o\tau_{g(i)}, o\tau_{h(i+1)}, \cdots, o\tau_{h(n)} >$ where $h$ is a permutation of elements of the sub-set $\{g(i+1), g(i+2), \cdots, g(n)\}$, is also* not schedulable.

**proof 2** *– By contradiction –*

*Let us consider the priorities choice $\mathcal{S}$ = $< o\tau_{g(1)}, \cdots, o\tau_{g(i)}, o\tau_{g(i+1)}, \cdots, o\tau_{g(n)} >$ for otasks in $O\Gamma_n$ where $g$ is a permutation of $\{1, 2, \cdots, n\}$. Let us assume that $\mathcal{S}$ is* not schedulable *due to otask $o\tau_{g(i)}$ and that there exists at least one schedulable priorities choice $\mathcal{S}^{'}$ = $< o\tau_{g(1)}, \cdots, o\tau_{g(i)}, o\tau_{h_0(i+1)}, \cdots, o\tau_{h_0(n)} >$ where $h_0$ is a permutation of elements of the sub-set $\{g(i+1), g(i+2), \cdots, g(n)\}$. Then, in particular, we have:*

$$\mathcal{R}_i = \bigoplus_{k=1}^{i} o\tau_{g(k)} \neq \Lambda$$

*This contradicts the hypothesis that the priorities choice $\mathcal{S}$ is* not schedulable *due to otask $o\tau_{g(i)}$.*

**Theorem 3** *Let $O\Gamma_n = \{o\tau_1, o\tau_2, \cdots, o\tau_n\}$ be a system with $n$ periodic otasks. We consider the priorities choice $\mathcal{S}$ = $< o\tau_{g(1)}, \cdots, o\tau_{g(i)}, o\tau_{g(i+1)}, \cdots, o\tau_{g(n)} >$ where $g$ is a permutation of $\{1, 2, \cdots, n\}$. If otask $o\tau_{g(i)}$ is* schedulable *according to $\mathcal{S}$, i.e.*

$$\mathcal{R}_i = \bigoplus_{k=1}^{i} o\tau_{g(k)} \neq \Lambda$$

*then any priorities choice $\mathcal{S}^{'}$ = $< o\tau_{g(1)}, \cdots, o\tau_{g(i)}, o\tau_{h(i+1)}, \cdots, o\tau_{h(n)} >$ where $h$ is a permutation of elements of the sub-set $\{g(i+1), g(i+2), \cdots, g(n)\}$ is equivalent to the priorities choice $\mathcal{S}^{''}$ = $< \mathcal{R}_i, o\tau_{h(i+1)}, \cdots, o\tau_{h(n)} >$.*

**proof 3** *The proof of theorem 3 directly follows from the observation that operation $\oplus$ is an internal operation.*

Theorem 2 reduces the number of priorities permutations tests when the system is not schedulable according to a particular priorities choice and theorem 3 reduces the number of times you perform the operation $\oplus$ by reusing a partial result of a priorities choice for calculating the result of another priorities choice.

# 10 Optimal otask priorities choice

In this section we propose an *optimal* algorithm for choosing priorities of otasks. This algorithm is optimal in the sense that, for a given otask system, if there is a choice of priorities which leads to a valid schedule then the priorities choice generated by our algorithm will also lead to a valid schedule. Since our proposal for setting the priorities is based on statements made on the algorithm of priorities choice of *Audsley*, we will call $Audsley^{++}$ our algorithm for choosing priorities of otasks. This algorithm integrates on the one hand the exact cost of preemption by using Algorithms $Main2$, 8, 9, 10, and the other hand provides all the solutions of priorities choice leading to a valid schedule.

The $Audsley^{++}$ algorithm divides a given otask system in two groups: first, the otasks which have already been assigned a priority, and second the otasks with no priorities. Similarly, priorities are divided into two groups: first, those which are already assigned and second, those that are still not assigned. The algorithm always assigns the *highest* not assigned priority to any otask that satisfies its constraints with that priority. Thanks to theorems 2 and 3 which limit the number of tests to perform, the algorithm can terminate in two different ways. Either it assigns priorities to all otasks, in this case, a schedulable choice of priorities has been found: we keep this solution and through a *backtrack* algorithm, we repeat the process to find a new solution if there exists one, or at a certain step the highest not assigned priority (for example, $p_{max}$) cannot be assigned to any remaining otask. In this case, thanks once again to a backtrack algorithm, and theorems 2 and 3, we repeat the process to find a solution. If no solution is found, we can conclude that there is no choice of priorities leading to a valid schedule (that is, which satisfies all constraints).
The algorithm is therefore *optimal*, in that it always finds a choice of priorities which leads to a schedulable system, if such a choice exists, thanks to the *backtrack* algorithm. Note that the number of tests performed is significantly lower compared to the "*naïve*" algorithm which checks all possible permutations of priorities through the use of theorems 2 and 3.

Let us illustrate our algorithm on a simple example. Let $\Gamma_5 = \{t_1, t_2, t_3, t_4, t_5\}$ be a system consisting of five periodic real-time tasks. The characteristics of tasks are summarized in table 2. We assume that the priorites are assigned according to the $Audsley^{++}$ algorithm.

Table 2: Characteristics of tasks

| Tasks | $r_i^1$ | $C_i$ | $D_i$ | $T_i$ | $\alpha_i$ |
|---|---|---|---|---|---|
| $t_1$ | 9 | 1 | 6 | 6 | 0 |
| $t_2$ | 13 | 3 | 9 | 12 | 2 |
| $t_3$ | 5 | 2 | 15 | 15 | 2 |
| $t_4$ | 0 | 3 | 21 | 24 | 1 |
| $t_5$ | 15 | 5 | 47 | 60 | 1 |

Thanks to everything we have presented so far, note that this system is not schedulable according to the choice of priorities corresponding to *Deadline Monotonic / DM* (i.e. the choice of priorities where the shorter the relative deadline of a task, the higher its priority) or the choice of priorities corresponding to *Rate Monotonic / RM* (i.e. the choice of priorities where the shorter the period of a task, the higher its priority). This choice of priorities is given in both cases by $\mathcal{S}_{DM/RM} = < t_1, t_2, t_3, t_4, t_5 >$ and the summary of the results obtained for this priorities choice is illustrated in figure 13. After applying Algorithms $Main2$, 8, 9 and 10, $t_1$, $t_2$ and $t_3$ are schedulable but $t_4$ is not. Figure 14 illustrates a zoom window of the corresponding *Gantt Chart*. In this window, we can note the deadline miss of task $t_4$ at time

$t = 93$. This deadline miss is due on the one hand to the preemption cost of task $t_4$ itself but also on the other hand, to the preemption cost of tasks $t_2$ and $t_3$ which currently have a higher priority than that of task $t_4$.

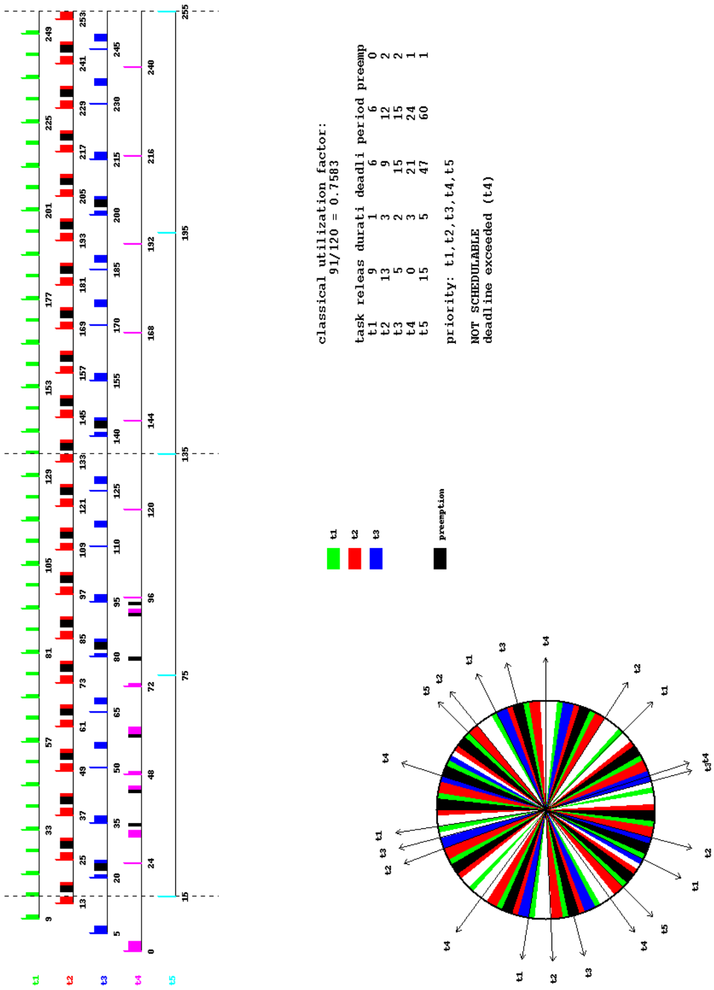

Figure 13: Results for the priorities choice $\mathcal{S}_{DM/RM} = < t_1, t_2, t_3, t_4, t_5 >$.

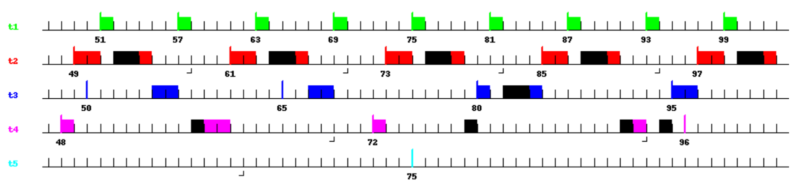

Figure 14: Zoom in the Gantt Chart.

Thanks to the $Audsley^{++}$ algorithm for choosing task priorities, four priorities choices lead to a valid schedule:

- $\mathcal{S}_1$ = $< t_2, t_1, t_3, t_4, t_5 >$,
- $\mathcal{S}_2$ = $< t_2, t_3, t_1, t_4, t_5 >$,
- $\mathcal{S}_3$ = $< t_3, t_2, t_1, t_4, t_5 >$,
- $\mathcal{S}_4$ = $< t_4, t_2, t_1, t_5, t_3 >$.

For these four schedulable priorities choices, the exact permanent preemption cost is $\epsilon_{\mathcal{S}_1} = 12.50\%$ for the priorities choice $\mathcal{S}_1$, $\epsilon_{\mathcal{S}_2} = 9.17\%$ for the priorities choice $\mathcal{S}_2$, $\epsilon_{\mathcal{S}_3} = 11.67\%$ for the priorities choice $\mathcal{S}_3$ and only $\epsilon_{\mathcal{S}_4} = 5.83\%$ for the priorities choice $\mathcal{S}_4$ by applying the Algorithms $Main2$, $8$, $9$ and $10$. Figures 15, 16, 17 and 18 summarize the results obtained for each of these four choices. Figures 19, 20, 21 et 22 illustrate the curves of the response time as a function of time for each task for each choice of priorities. The variation in terms of permanent preemption cost is due to the fact that on one hand the cost of preemption varies from one task to another and on the other hand to the fact that according to some choice of priorities, some tasks are preempted when they are not according to other choices of priorities.

Thanks to the criterion for choosing otask priorities we have proposed, that is to say, to always consider the otask priorities such that the exact permanent preemption cost is the lowest, we choose the priorities choice $\mathcal{S}_4$ = $< t_4, t_2, t_1, t_5, t_3 >$. For this choice of priorities, the worst response time of task $t_1$ is $R_1 = 4$ time units and it is reached for the first time at its fourth activation, the worst response time of task $t_2$ is $R_2 = 5$ time units and it is reached for first at its second activation, the worst response time of task $t_3$ is $R_3 = 14$ time units and it is reached for the first time at its seventh activation, the worst response time of task $t_4$ is $R_4 = 3$ time units and it is reached for the first time at its first activation ($t_4$ is the task with the highest priority) and finally, the worst response time of task $t_5$ is $R_5 = 16$ time units and it is reached for the first time at its second activation.

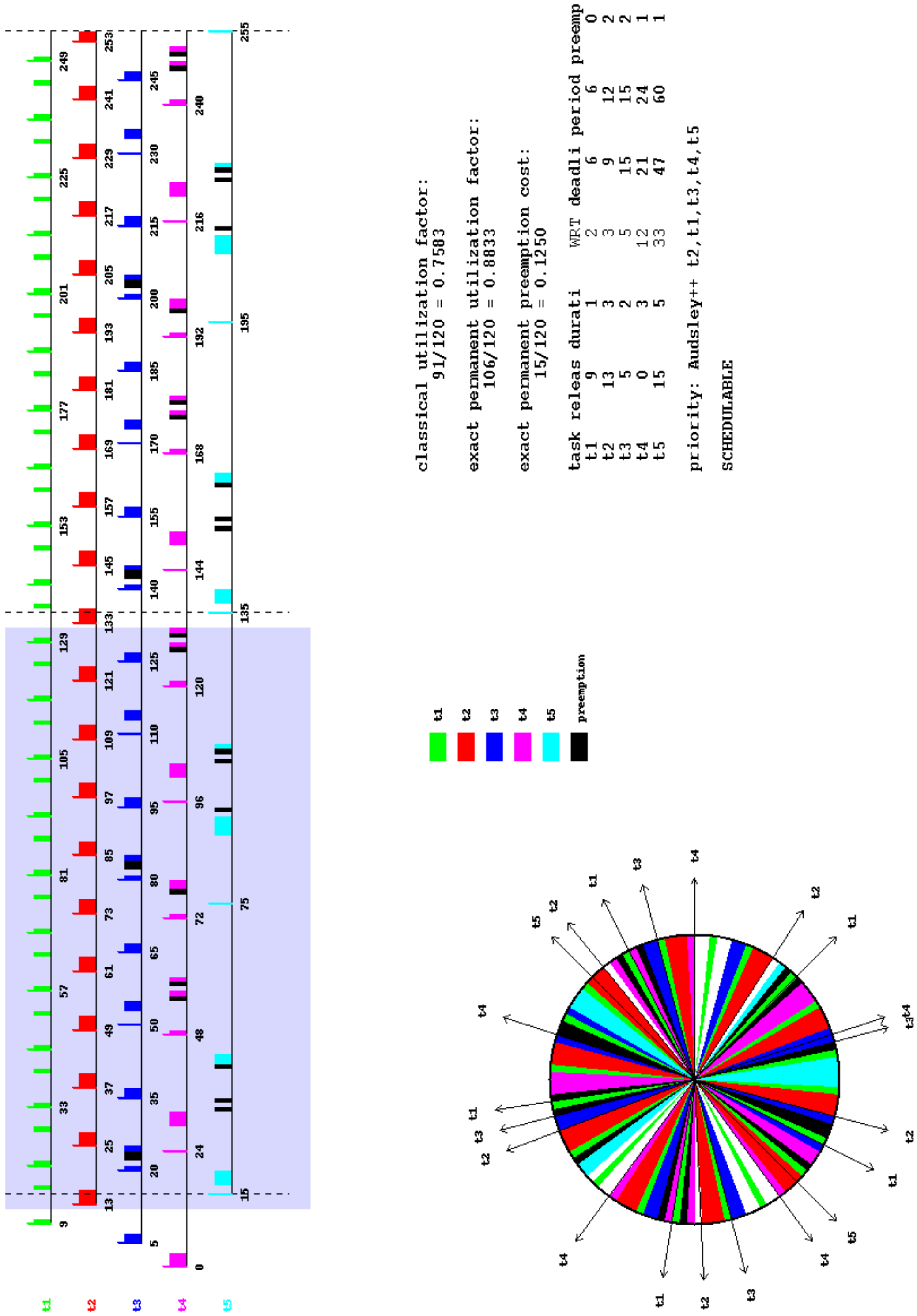


Figure 15: Results for $\mathcal{S}_1 = < t_2, t_1, t_3, t_4, t_5 >$.

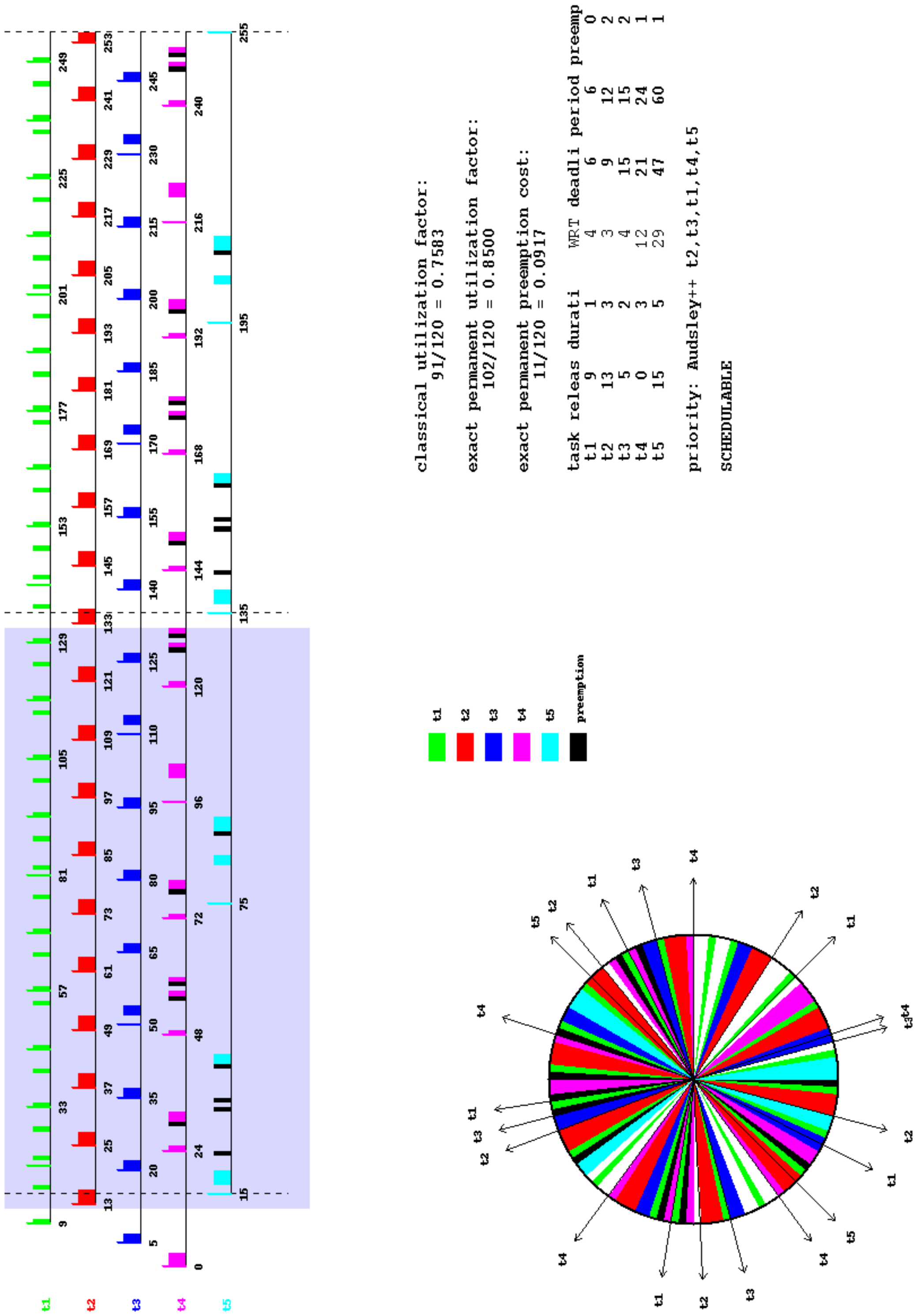


Figure 16: Results for $\mathcal{S}_2 = < t_2, t_3, t_1, t_4, t_5 >$.

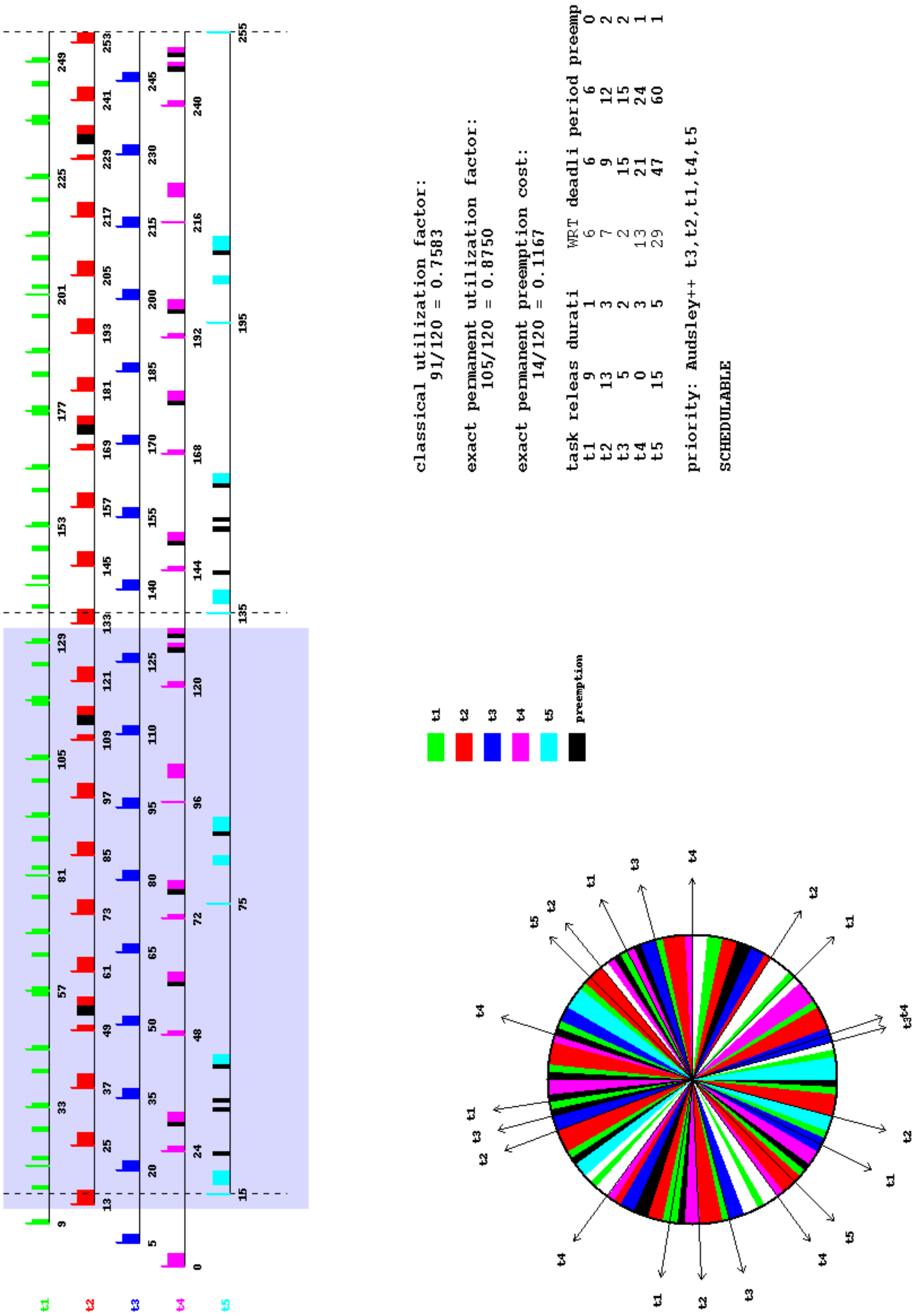


Figure 17: Results for $\mathcal{S}_3 = < t_3, t_2, t_1, t_4, t_5 >$.

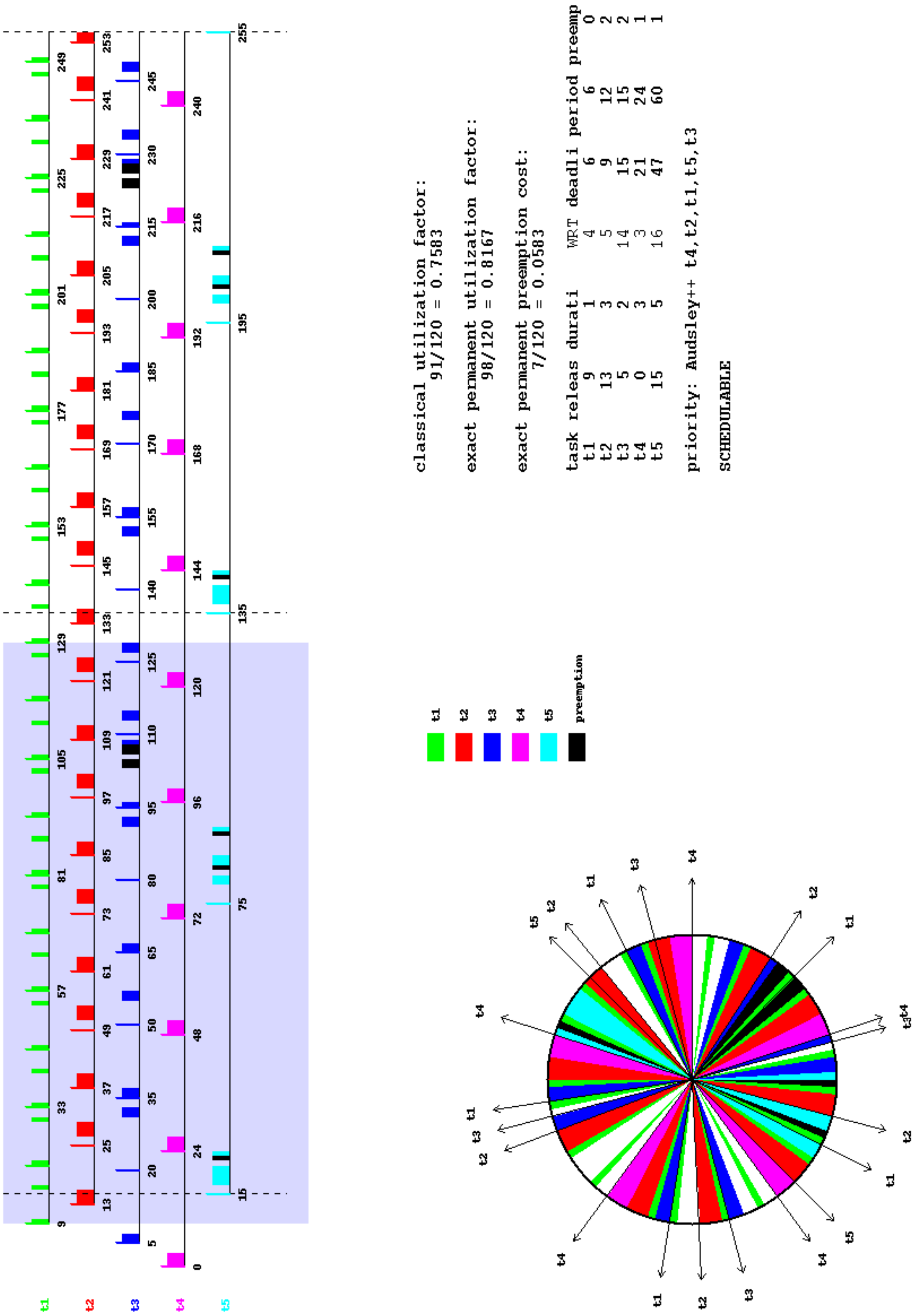


Figure 18: Results for $\mathcal{S}_4$ = $< t_4, t_2, t_1, t_5, t_3 >$.

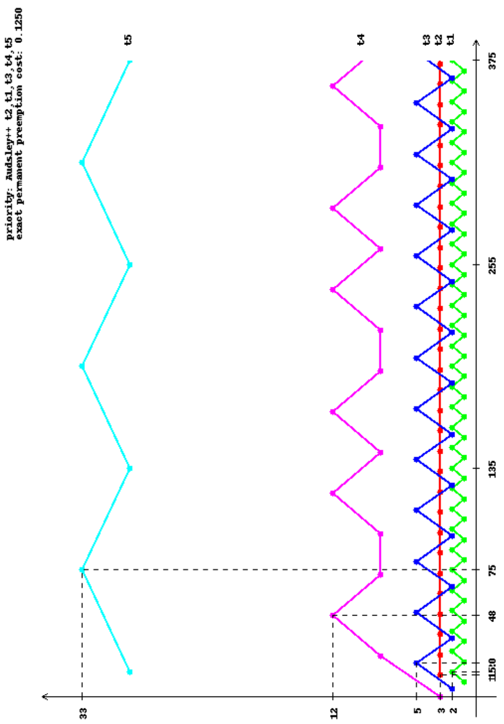

Figure 19: $\mathcal{S}_1 = < t_2, t_1, t_3, t_4, t_5 >$.

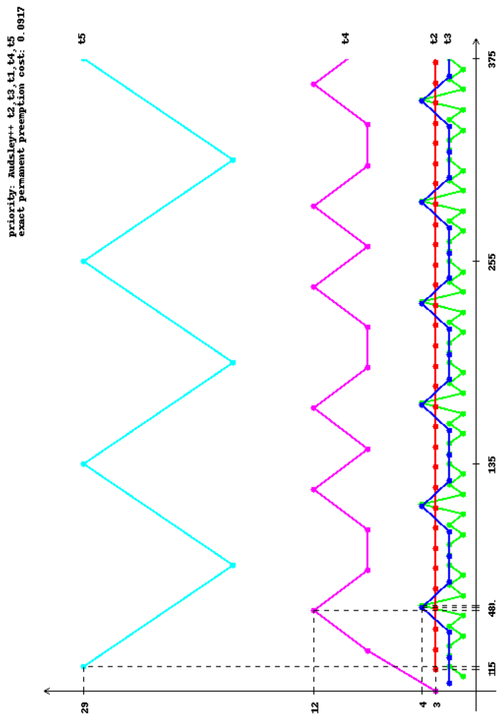

Figure 20: $\mathcal{S}_2 = < t_2, t_3, t_1, t_4, t_5 >$.

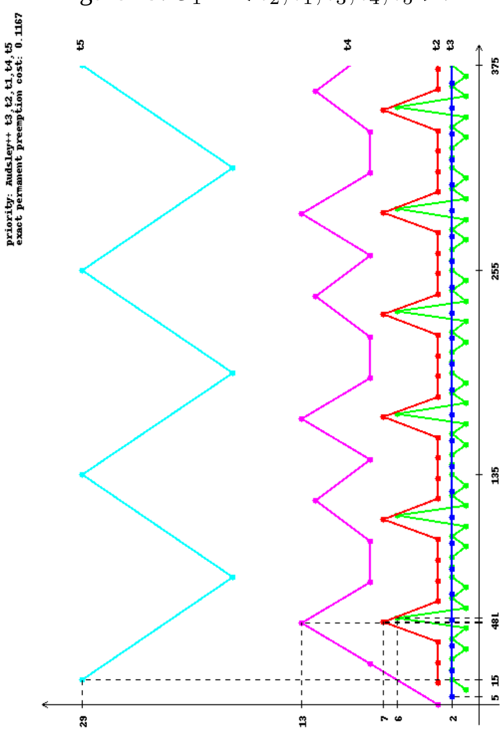

Figure 21: $\mathcal{S}_3 = < t_3, t_2, t_1, t_4, t_5 >$.

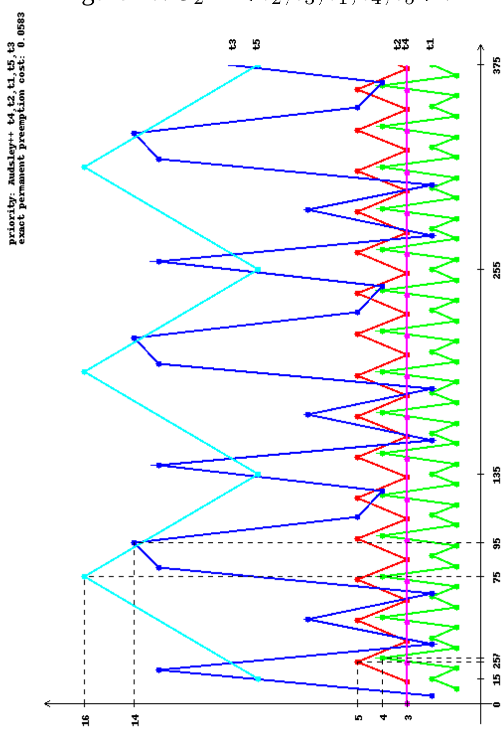

Figure 22: $\mathcal{S}_4 = < t_4, t_2, t_1, t_5, t_3 >$.

# 11 Consequence of reducing the number of preemptions

For a given priorities choice, an intuitive way to reduce the exact permanent preemption cost might be to force some otasks to be *non-delay non-preemptive* in order to reduce the number of preemptions. Here, when applying operation $\oplus$, we mean *non-delay non-preemptive* that we check if the cardinal of the first sequence of symbols "$a$" to replace in each *mesoid*, is higher than that of executable symbols "$e$" at this level. In this case the right-hand operand is schedulable. This notion of *non-delay non-preemptiveness* is quite different from the concept of the *classical non-preemptiveness* because in this case, once the priorities are assigned to otasks, a lower priority otask *cannot delay* the start time of the execution of another otask with a higher priority. Figures 23 and 24 illustrate the difference between the two concepts of non-preemptiveness. In this example, otask $o\tau_1$ has a higher priority than otask $o\tau_2$ and Otachi $o\tau_2$ is non-preemptive.

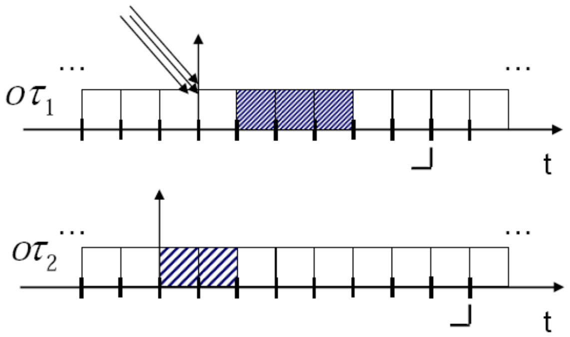

Figure 23: The classical non-preemptiveness.

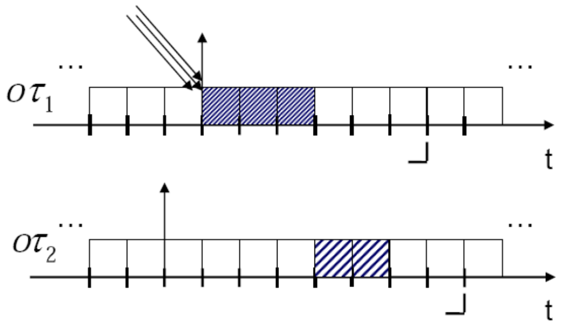

Figure 24: The non-delay non-preemptiveness.

In Figure 23, we can note that otask $o\tau_2$ delays the start time for the execution of otask $o\tau_1$, having a higher priority. Indeed, due to the activation of otask $o\tau_2$ one time unit before that of otask $o\tau_1$, there is no preemption at the activation time of $o\tau_1$ as $o\tau_2$ is non-preemptive. Thus, a disadvantage of the *classical non-preemptiveness* of an otask is a possible deadline miss of another otask with a higher priority due to these delays. In this case, if the system is not schedulable, the faulty otask may be very difficult to determine accurately.

In Figure 24, we can note that the effective start time of execution of otask $o\tau_2$ takes place after the end of execution of otask $o\tau_1$, having a higher priority. Indeed even if the activation of otask $o\tau_2$ occurs one time unit earlier than that of otask $o\tau_1$, it does not start executing at that time because it is *non-delay non-preemptive* on the one hand, and has a lower priority on the other hand. Indeed, there are not enough available time units to execute it before the next activation of $o\tau_1$. Therefore, an advantage of the *non-delay non-preemptiveness* of an otask is that it has no impact on the result of the schedulability of another otask with a higher priority.

With this approach, by forcing some otasks to be *non-delay non-preemptive*, some preemptions can be avoided, thus reducing the exact permanent preemption cost for a given choice of priorities. Figure 25 illustrates this assertion for the previous system consisting of $5$ otasks. In this figure the exact permanent preemption cost falls from $5.83\%$ to $2.50\%$ according to the choice of priorities $\mathcal{S}_4$. Figure 26 illustrates the curve of response times relative to release times for each otask.

If the *non-delay non-preemptiveness* of an otask can in the one hand reduce the exact permanent preemption cost of a system for a given choice of priorities, it may on the other hand, make it more important for a different choice of priorities. This is due to the fact that the *non-delay non-preemptiveness* has a *direct impact* on the preemption cost at a given level but an *indirect impact* on the overall exact preemption cost. Figure 27 illustrates this assertion for the previous system consisting of $5$ otasks. The exact permanent preemption cost goes from $2.50\%$ according to the choice of priorities $\mathcal{S}_4$ to $21.67\%$ according to the choice of priorities given by $\mathcal{S}_5 = < t_1, t_2, t_4, t_3, t_5 >$. Figure 28 illustrates the response times relative to release times for each otask following these priorities. Note that the worst response time otask $t_5$ is $46$ time units instead of $16$ according to the choice of priorities $\mathcal{S}_4$.

# 12 Conclusion and future work

In this paper, by using the otask model which is based on an algebraic approach, we have defined the binary scheduling operation $\oplus$ for the scheduling problem of periodic otask systems. This scheduling operation helped us to provide new schedulability conditions which take into account the exact cost due to the occurrence of each preemption for a given system. Since to the best of our knowledge there is not, in literature, any optimal algorithm for choosing priorities when the exact preemption cost is taken into account, we have shown the impact of the choice of priorities on both the schedulability analysis and the schedule for a given system. We have proposed an optimal algorithm called $Audsley^{++}$ for assigning priorities to otasks which leads to stronger schedulability conditions than those in the literature. These new conditions always guarantee a correct behavior of the system at run-time and eliminate the waste of the resource (CPU). Finally, we have shown the consequence of reducing the number of preemptions of some otasks. Future work will extend the proposed approach to take into account multiple real-time constraints such as precedence, strict periodicity, latencies, and jitter. On the other hand, we will address the scheduling problem of periodic otask systems when priorities are assigned to otasks according to dynamic priority policies such as Earliest Deadline First (EDF). Furthermore, by using the *non-delay non-preemptiveness* for some otasks, we will seek an optimization algorithm which minimizes the exact permanent preemption cost while maximizing the number of possible schedules.

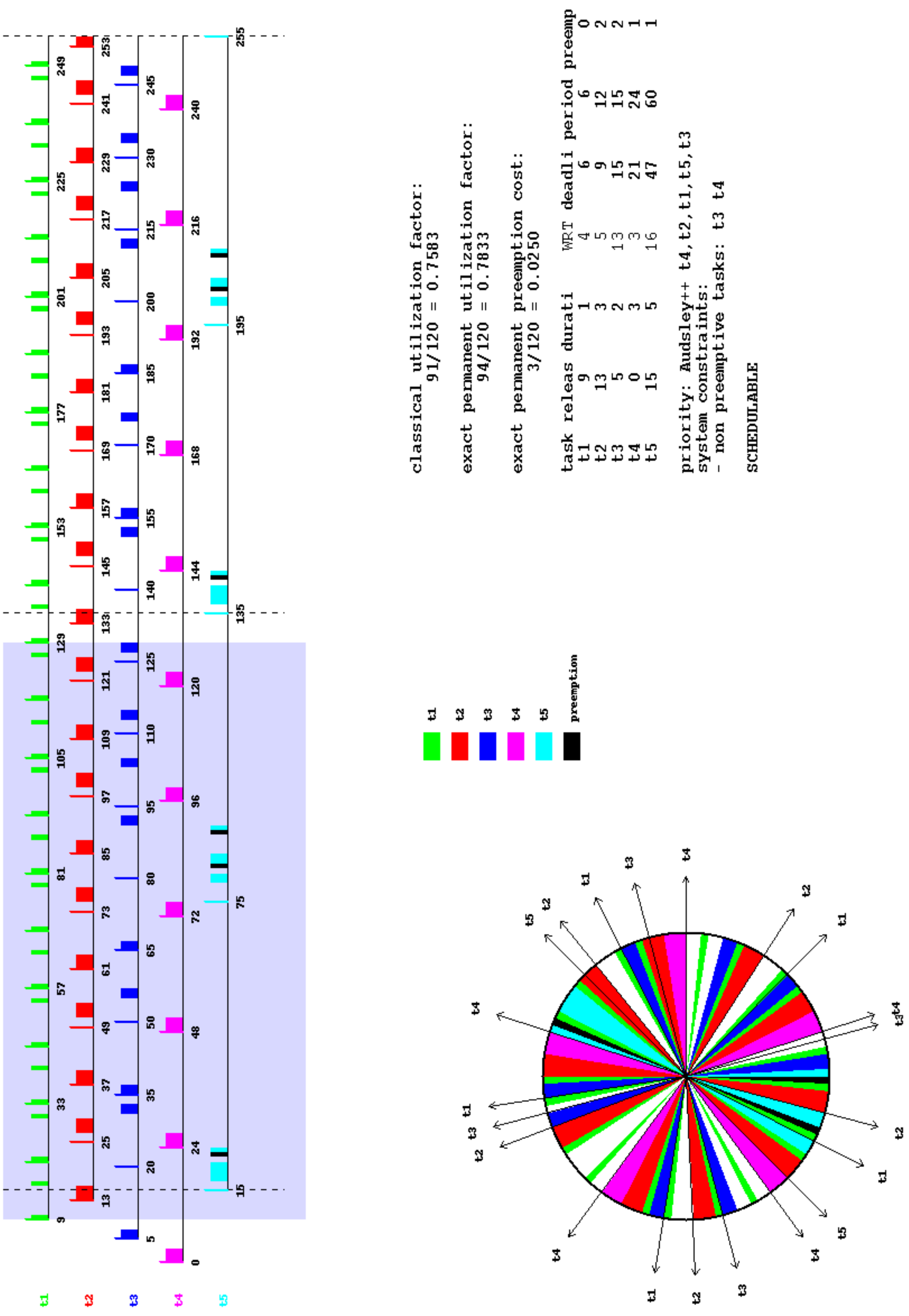


Figure 25: Results for $\mathcal{S}_4 = < t_4, t_2, t_1, t_5, t_3 >$.

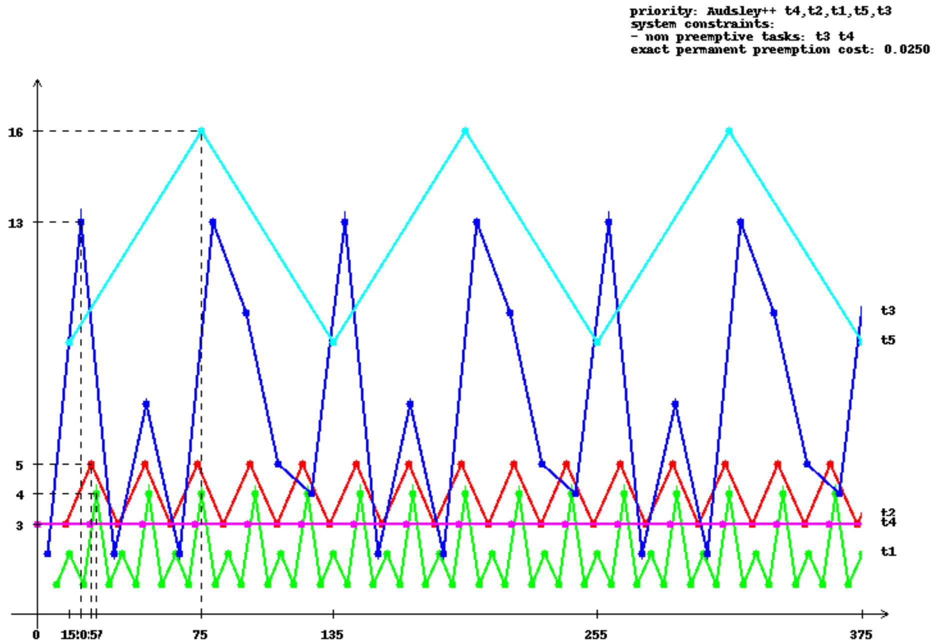


Figure 26: Response times relative to release times.

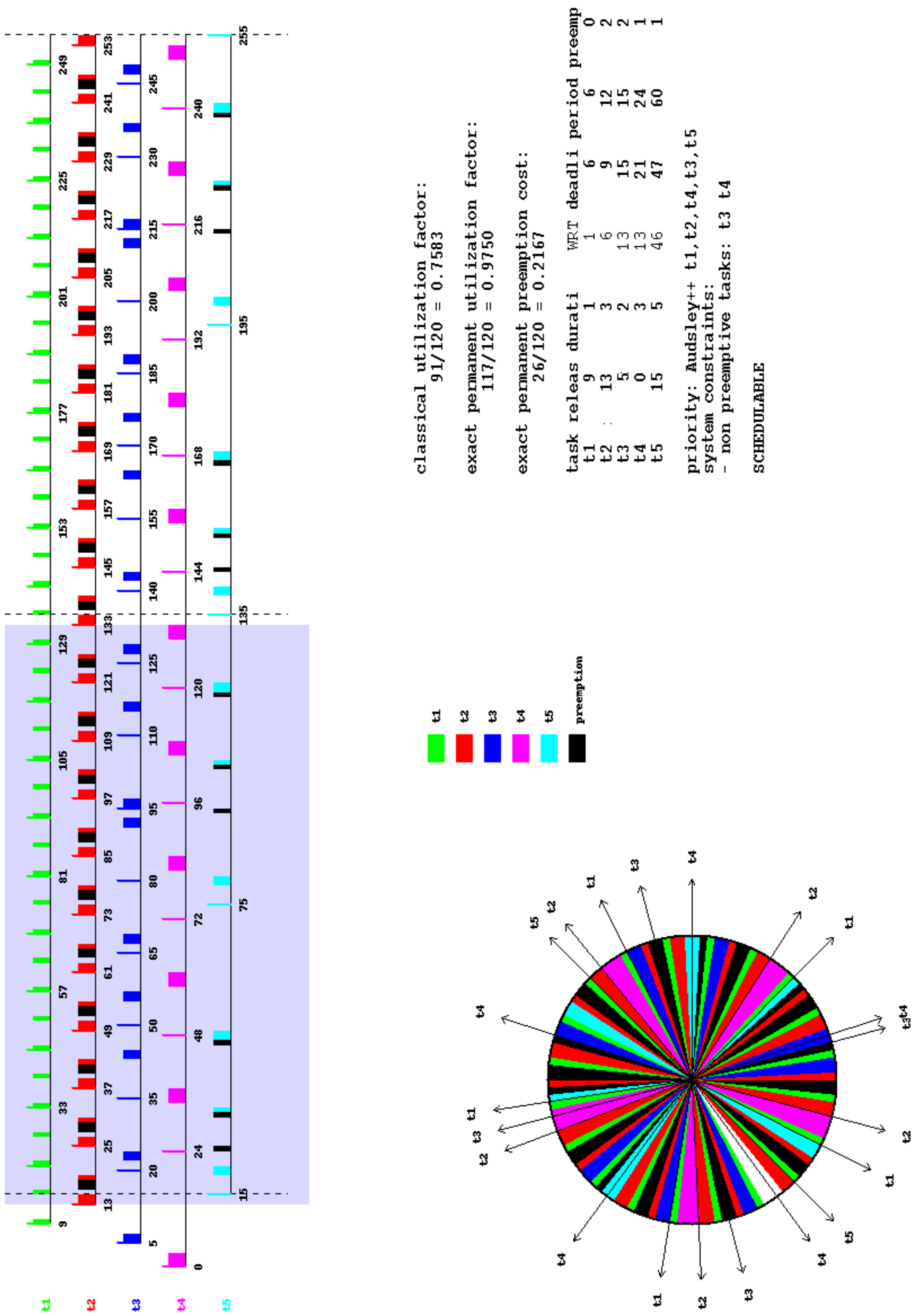


Figure 27: Results for $\mathcal{S}_5 = < t_1, t_2, t_4, t_3, t_5 >$.

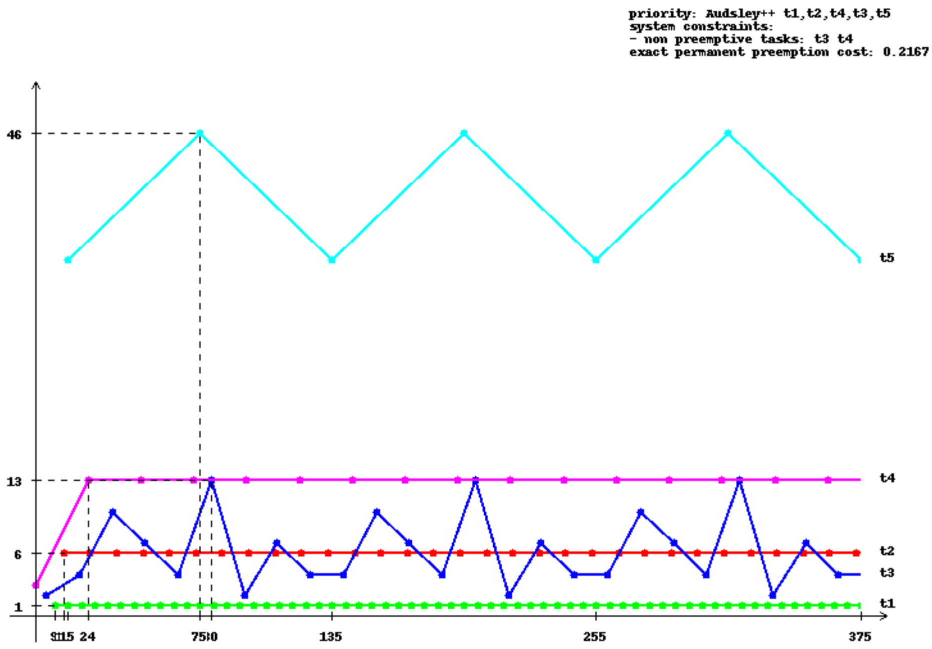


Figure 28: Response times relative to release times.

# References


[1] C.L. Liu and J.W. Layland. Scheduling algorithms for multiprogramming in a hard-real-time environment. *Journal of the ACM*, 1973.

[2] Joseph Y.-T. Leung and M. L. Merrill. A note on preemptive scheduling of periodic, real-time tasks. *Information Processing Letters*, 1980.

[3] R. Pellizzoni and G. Lipari. Feasibility analysis of real-time periodic tasks with offsets. *Real-Time Systems*, 2005.

[4] L. Cucu, R. Kocik, and Y. Sorel. Real-time scheduling for systems with precedence, periodicity and latency constraints. In *Proceedings of 10th Real-Time Systems Conference, RTS'02*, Paris, France, March 2002.

[5] A. Choquet-Geniet and E. Grolleau. Minimal schedulability interval for real time systems of periodic tasks with offsets. *Theoretical Computer Science*, 2003.

[6] Ray Obenza and Geoff. Mendal. Guaranteeing real time performance using rma. *The Embedded Systems Conference, San Jose, CA*, 1998.

[7] I. J. Bate. *Scheduling and Timing Analysis of Safety Critical Real-Time Systems*. PhD thesis, University of York, 1998.

[8] P. Meumeu Yomsi and Y. Sorel. Extending rate monotonic analysis with exact cost of preemptions for hard real-time systems. In *Proceedings of 19th Euromicro Conference on Real-Time Systems, ECRTS'07*, Pisa, Italy, July 2007.

[9] E. Bini and G. Buttazzo. The space of rate monotonic schedulability. *In Proc. of the 23rd IEEE Real-Time Systems Symposium, pp. 169-178*, 2002.

[10] Razvan Racu, Arne Hamann, and Rolf Ernst. Sensitivity analysis of complex embedded real-time systems. *Real-Time Syst.*, 39(1-3):31–72, 2008.

[11] Burns A., Tindell K., and Wellings A. Effective analysis for engineering real-time fixed priority schedulers. *IEEE Trans. Software Eng.*, 1995.

[12] J. Echagüe, I. Ripoll, and A. Crespo. Hard real-time preemptively scheduling with high context switch cost. *In Proceedings of the 7th Euromicro Workshop on Real-Time Systems*, 1995.

[13] F. Bimbard and L. George. Feasibility conditions with kernel overheads for mixed preemptive fp/fifo scheduling with priority ceiling protocol on an event driven osek system. *12th IEEE international Conference on Emerging Technologies and Factory Automation (ETFA'07)*, Patras, Grece, Sept 2007.

[14] P. Meumeu Yomsi. *Prise en compte du coût exact de la préemption dans l'ordonnancement temps réel monoprocesseur avec contraintes multiples*. PhD thesis, Université de Paris Sud, Spécialité Physique, 02/04/2009.

[15] N.C. Audsley, A. Burns, M.F. Richardson, Tindell K., and A.J. Wellings. Applying new scheduling theory to static priority pre-emptive scheduling. *Software Engineering Journal*, 1993.

[16] N.C. Audsley, A. Burns, and A.J. Wellings. Deadline monotonic scheduling theory and application. *J. Control Engineering Practice*, 1993.

[17] Noam Chomsky. Three models for the description of language. *I. R. E. transactions on information theory, Proceedings of the symposium on information theory, volume IT-2, pages 113-123*, September 1956.

[18] William C. Rounds, Alexis Manaster-Ramer, and Joyce Friedman. Finding natural languages a home in formal language theory. *John Benjamins Publishing Company, Amsterdam/Philadelphia, pages 349-359*, 1987.

[19] K. Lauri, J.P. Chanod, G. Grefenstette, and A. Schiller. Regular expressions for language engineering. *Natural Language Engineering, 2(4):305-328*, 1996.

[20] Z. Manna and R. Waldinger. *The Logical Basis for Computer Programming*. volume 1: Deductive Reasoning. Addison-Wesley, Reading, Massachusetts, 1985.

[21] Ronald R. Yager. *On the theory of bags*. Int. J. General Systems, 13:23-37, 1986.

[22] W.D. Blizard. *The Development of Multiset Theory*. Modern Logic, 1:319-352, 1991.

[23] Donald E. Knuth. *The art of computer programming*. Vol 2: Seminumerical algorithms, 3rd edition, Addison Wesley, 694, 1998.

[24] J. Goossens. *Scheduling of Hard Real-Time Periodic Systems with Various Kinds of Deadline and Offset Constraints*. PhD thesis, Université Libre de Bruxelles, 1998.

[25] D. Katcher, H. Arakawa, and J. Strosnider. Engineering and analysis of fixed priority schedulers. *IEEE Transactions on Software Engineering, vol. 19: pp 920-934*, 1993.

Centre de recherche INRIA Paris – Rocquencourt
Domaine de Voluceau - Rocquencourt - BP 105 - 78153 Le Chesnay Cedex (France)

Centre de recherche INRIA Bordeaux – Sud Ouest : Domaine Universitaire - 351, cours de la Libération - 33405 Talence Cedex
Centre de recherche INRIA Grenoble – Rhône-Alpes : 655, avenue de l'Europe - 38334 Montbonnot Saint-Ismier
Centre de recherche INRIA Lille – Nord Europe : Parc Scientifique de la Haute Borne - 40, avenue Halley - 59650 Villeneuve d'Ascq
Centre de recherche INRIA Nancy – Grand Est : LORIA, Technopôle de Nancy-Brabois - Campus scientifique
615, rue du Jardin Botanique - BP 101 - 54602 Villers-lès-Nancy Cedex
Centre de recherche INRIA Rennes – Bretagne Atlantique : IRISA, Campus universitaire de Beaulieu - 35042 Rennes Cedex
Centre de recherche INRIA Saclay – Île-de-France : Parc Orsay Université - ZAC des Vignes : 4, rue Jacques Monod - 91893 Orsay Cedex
Centre de recherche INRIA Sophia Antipolis – Méditerranée : 2004, route des Lucioles - BP 93 - 06902 Sophia Antipolis Cedex

Éditeur
INRIA - Domaine de Voluceau - Rocquencourt, BP 105 - 78153 Le Chesnay Cedex (France)
http://www.inria.fr
ISSN 0249-6399