Schedulability Analysis using Exact Number of Preemptions and No Idle Time for Real-Time Systems with Precedence and Strict Periodicity Constraints

Patrick Meumeu Yomsi, Yves Sorel

Schedulability Analysis using Exact Number of Preemptions and no Idle Time for Real-Time Systems with Precedence and Strict Periodicity Constraints

Patrick Meumeu Yomsi
INRIA Rocquencourt
Domaine de Voluceau BP 105
78153 Le Chesnay Cedex - France
Email: patrick.meumeu@inria.fr

Yves Sorel
INRIA Rocquencourt
Domaine de Voluceau BP 105
78153 Le Chesnay Cedex - France
Email: yves.sorel@inria.fr

Abstract

Classical approaches based on preemption, such as RM (Rate Monotonic), DM (Deadline Monotonic), EDF (Earliest Deadline First), LLF (Least Laxity First), etc, give schedulability conditions in the case of a single processor, but assume the cost of the preemption to be negligible compared to the duration of each task. Clearly the global cost is difficult to determine accurately because, if the cost of one preemption is known for a given processor, it is not the same for the exact number of preemptions of each task. Because we are interested in hard real-time systems with precedence and strict periodicity constraints where it is mandatory to satisfy these constraints, we give a scheduling algorithm which counts the exact number of preemptions for each task, and thus leads to a new schedulability condition. This is currently done in the particular case where the periods of all the tasks constitute an harmonic sequence.

1 Introduction

We address here hard real-time applications found in the domains of automobiles, avionics, mobile robotics, telecommunications, etc, where the real-time constraints must be satisfied in order to avoid the occurrence of dramatic consequences [1, 2]. Such applications based on automatic control and/or signal processing algorithms are usually specified with block-diagrams. They are composed of functions producing and consuming data, and each function has a strict period in order to guarantee the input/output rate as it is usually required by the automatic control theory. Consequently, in this paper we study the problem of scheduling tasks onto a single computing resource, i.e. a single processor, where each task corresponds to a function and must satisfy precedence constraints in addition to its strict period. This latter constraint implies that for such a system, any task starts its execution at the beginning of its period. We assume here that no jitter is allowed at the beginning of each task.

Traditional approaches based on preemption, such as RM (Rate Monotonic) [3], DM (Deadline Monotonic) [4], EDF (Earliest Deadline First) [5], LLF (Least Laxity First) [6], etc, give schedulability conditions but always assume the cost of the preemption to be negligible compared to the duration of each task [7, 8]. Indeed, this assumption is due to the Liu & Layland model [9], also called "the classical model", which is the pioneer model for scheduling hard real-time systems. With this model, the authors showed that a system of independent periodic preemptive tasks with the periods of all tasks forming an harmonic sequence [10] [1], is schedulable if and only if:

$$\sum_{i=1}^{n} \frac{C_i^{'}}{T_i} \leq 1 \qquad (1)$$

T_i denotes the period and $C_i^{'}$ the inflated worst case execution time (WCET) with the approximation of the cost of the preemption for task τ_i. It is worth noticing that most of the industrial applications in the field of automatic control, image and signal processing consist of tasks with periods forming an harmonic sequence. For example, the automatic guidance algorithm in a missile falls within this case. Actually, expression (1) takes into account the cost due to preemption inside the value of $C_i^{'}$. Thus, $C_i^{'} = C_i + \varepsilon_i^{'}$ where C_i is the value of the WCET without preemption, and $\varepsilon_i^{'}$ is an approximation of the cost ε_i of the preemption for this task, as explicitly stated in [9]. Thus, expression (1) becomes:

$$U + \varepsilon^{'} \leq 1 \qquad (2)$$

where

$$U = \sum_{i=1}^{n} \frac{C_i}{T_i}, \quad \text{and} \quad \varepsilon^{'} = \sum_{i=1}^{n} \frac{\varepsilon_i^{'}}{T_i}$$

The cost of the preemption for task τ_i is $\varepsilon_i = N_p(\tau_i) \cdot \alpha$, where α denotes the temporal cost of one preemption and $N_p(\tau_i)$ is the exact number of preemptions of task τ_i.

[1] A sequence $(a_i)_{1 \leq i \leq n}$ is harmonic if and only if there exists $q_i \in \mathbb{N}$ such that $a_{i+1} = q_i a_i$. Notice that we may have $q_{i+1} \neq q_i \quad \forall i \in \{1, \cdots, n\}$.

$N_p(\tau_i)$ may depend on the instance of the task according to the relationship between the periods of the other tasks in the system. For example, in the case where the periods of the tasks form an harmonic sequence $N_p(\tau_i)$ does not depend on the instance of τ_i. Therefore, since $\varepsilon_i^{'}$ is an approximation of ε_i and T_i is known, $\varepsilon^{'}$ is an approximation of the global cost ε due to preemption, defined by:

$$\varepsilon = \sum_{i=1}^{n} \frac{N_p(\tau_i) \cdot \alpha}{T_i}$$

If the temporal cost α of one preemption is known for a given processor, it is not the same for the exact number of preemptions $N_p(\tau_i)$ for each task τ_i during a period T_i. Consequently, it becomes difficult to calculate the global cost of the preemption, and thus to guarantee that expression (2) holds. Obviously the approximation of this latter may lead to a wrong real-time execution whereas the schedulability analysis concluded that the system was schedulable. To cope with this problem the designer usually allows margins which are difficult to assess, and which in any case lead to a waste of resources. Note that the worst-case response time of a task is the greatest time, among all instances of that task, it takes to execute each instance from its release time, and it is larger than the WCET when an instance is preempted. A. Burns, K. Tindell and A. Wellings in [11] presented an analysis that enables the global cost due to preemptions to be factored into the standard equations for calculating the worst-case response time of any task, but they achieved that by considering the maximum number of preemptions instead of the exact number. Juan Echagüe, I. Ripoll and A. Crespo also tried to solve the problem of the exact number of preemptions in [12] by constructing the schedule using idle times and counting the number of preemptions. But, they did not really determine the execution overhead incurred by the system due to these preemptions. Indeed, they did not take into account the cost of each preemption during the scheduling. Hence, this amounts to considering only the minimum number of preemptions since some preemptions are not considered: those due to the increase in the execution time of the task because of the cost of the preemptions themselves.

For such a system of tasks with strict periodicity and precedence constraints, we propose a method to calculate on the one hand the exact number of preemptions and thus the accurate value of ε, and on the other hand the schedule of the system without any idle time, i.e. the processor will always execute a task as soon as it is possible to do so. Although idle time may help the system to be schedulable, when idle time is forbidden it is easier to find the start times of all the instances of a task according to the precedence relation.

The proposed method leads to a much stronger schedulability condition than expression (1). Moreover, we do this in the case where tasks are subject to precedence and strict periodicity constraints, using our previous model [13] that is well suited to the applications we are interested in. Afterwards, to clearly distinguish between the specification level and its associated model, we shall use the term *operation* instead of the commonly used "*task*" [14] which is too closely related to the implementation level.

The paper is structured as follows: Section 2 describes the model and gives notations used throughout this paper. Section 3 restricts the study field thanks on the one hand to properties on the strict periods, and on the other hand to properties on WCETs. Section 4 proposes a scheduling algorithm which counts the exact number of preemptions, and derives a schedulability condition, in the case where the periods of all operations constitute an harmonic sequence. We conclude and propose future work in section 5.

2 Model

The model depicted in figure 1 is an extension, with preemption, of our previous model [13] for systems with precedence and strict periodicity constraints executed on a single processor.

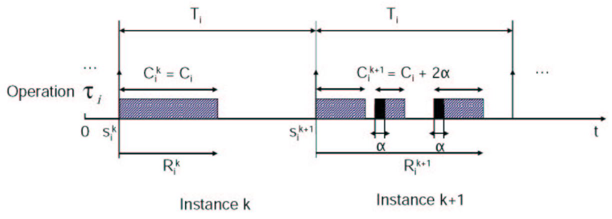


Figure 1. Model

Here are the notations used in this model assuming 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):

$\tau_i = (C_i, T_i)$: An operation
T_i: Period of τ_i
C_i: WCET of τ_i without preemption, $C_i \leq T_i$
τ_i^k: The k^{th} instance of τ_i
α: Temporal cost of one preemption for a given processor
$N_p(\tau_i^k)$: Exact number of preemptions of τ_i in τ_i^k
$C_i^k = C_i + N_p(\tau_i^k) \cdot \alpha$: Exact WCET of τ_i including its preemption cost in τ_i^k
s_i^0: Start time of the first instance of τ_i
$s_i^k = s_i^0 + (k-1)T_i$: Start time of the k^{th} instance of τ_i
R_i^k: Response time of the k^{th} instance of τ_i
R_i: Worst-case response time of τ_i
$T_i \wedge T_j$: The greatest common divisor of T_i and T_j, when $T_i \wedge T_j = 1$, T_i and T_j are co-prime
$\tau_i \prec \tau_j$: $\tau_i \longrightarrow \tau_j$, τ_i precedes τ_j

We denote by V the set of all systems of operations. Each system consists in a given number of operations, with precedence and strict periodicity constraints. Each operation τ_i of a system in V consists of a pair (C_i, T_i): C_i its WCET and T_i its period.

The precedence constraints are given by a partial order on the execution of the operations. $\tau_i \prec \tau_j$ means that the start time s_j^0 of the first instance of τ_j cannot occur before the first instance of τ_i, started at s_i^0, is completed. This precedence relation between operations also implies that $s_i^k \leq s_j^k, \forall k \geq 1$ thanks to the result given in [15]. In that paper it has been proven that given two operations $\tau_i = (C_i, T_i)$ and $\tau_j = (C_j, T_j)$:

$$\tau_i \prec \tau_j \Longrightarrow T_i \leq T_j$$

Regarding the latter relation from the practical point of view, it is worth noticing that when the precedence relations are due to data transfers and the periods of the operations exchanging data constitute an harmonic sequence, the number of operations producing data between two consecutive operations consuming the corresponding data, is constant. Consequently, the number of buffers used to actually achieve the data exchange is bounded, i.e. it cannot increase indefinitely.

The strict periodicity constraint means that two successive instances of an operation are **exactly** separated by its period: $s_i^{k+1} - s_i^k = T_i \quad \forall k \in \mathbb{N}, \quad \forall i \in \{1, \cdots, n\}$, and no jitter is allowed. In this model the start time is always equal to the release time, in contrast to Liu & Layland's classical model. A great advantage of the strict periodicity constraint for each task is that it is only necessary to focus on the start time of the first instance, the other being directly obtained from it.

It is fundamental to note that, because of the strict periodicity constraint and the fact that we are dealing with the single processor case, any two instances of any two operations of the system cannot start their executions at the same time.

3 Study field restriction

Firstly, we eliminate all the systems where the start times of any two instances of any two operations are identical. This will be achieved thanks to properties on the strict periods of the operations, using the *Bezout theorem*. This is formally expressed through both theorems given in section 3.1. Secondly, we eliminate all the systems where the start time of any instance of an operation occurs while the processor is occupied by a previously scheduled operation thanks to properties on WCETs of the operations. This is formally expressed through the theorem given in section 3.2. These three theorems give sufficient non-schedulability conditions. For the remaining systems of operations, we adopt a constructive approach which consists in building, i.e. in predicting, all the possible preemptive schedules without any idle time. In so far, as we are dealing with hard real-time systems whose main feature is predictability, constructive techniques are better suited than simulation techniques based on tests that are seldom exhaustive. In addition, an exhaustive simulation assumes that there exists a scheduling algorithm, e.g. RM or DM, which is used to perform the simulation. In our case we propose a scheduling algorithm which determines if the system is schedulable and provides the schedule.

3.1 Restriction due to strict periodicity

Theorem 1

Given a system of n operations in V, if there are two operations $\tau_i = (C_i, T_i)$ and $\tau_j = (C_j, T_j)$ with $(\tau_i \prec \tau_j)$ starting their executions respectively at the dates s_i^0 and s_j^0 such that

$$T_i \wedge T_j = 1 \tag{3}$$

then the system is not schedulable. Moreover, any additional assumption (for example preemption and idle times) on the system intending to satisfy all the constraints is of no interest in this case.

Proof: The proof of this theorem uses the Bezout theorem and is detailed in [16]. ∎

Theorem 2

Given a system of n operations in V, if there are two operations $\tau_i = (C_i, T_i)$ and $\tau_j = (C_j, T_j)$ with $(\tau_i \prec \tau_j)$ starting their executions respectively at the dates s_i^0 and s_j^0 such that

$$T_i \wedge T_j \mid (s_j^0 - s_i^0) \tag{4}$$

then the system is not schedulable. Moreover any additional assumption on the system intending to satisfy all the constraints is of no interest in this case.

Proof: The proof of this theorem also uses the Bezout theorem and is detailed in [16]. ∎

Theorems 1 and 2 give non-schedulability conditions for systems with strict periodicity constraints when both previous relations on the strict periods hold. Moreover, any additional assumption on the system would be useless because of the identical start times of two instances of at least two operations.

We denote by Ω_λ the sub-set of V excluding the cases where the strict periods of the operations verify both previous relations.

$$\Omega_\lambda = \{\{(C_i, T_i)\}_{1 \leq i \leq n} \in V \,/\, \forall i, j \in \{1, \cdots, n\} \\ \exists \lambda > 1,\, T_i \wedge T_j = \lambda \text{ and } \lambda \nmid (s_j^0 - s_i^0)\}$$

3.2 Restriction due to WCET

The scheduling analysis of a system of preemptive tasks (operations) has shown its importance in a wide range of applications because of its flexibility and its relatively easy implementation [17]. Although preemptions may allow schedules to be found that could not be found without it, it can, unfortunately, cause non schedulability of the system due to its global cost.

Since, given two operations $\tau_i = (C_i, T_i)$ and $\tau_j = (C_j, T_j)$ we have $\tau_i \prec \tau_j \Longrightarrow T_i \leq T_j$ thus, the operations must be scheduled in an increasing order of their periods corresponding to classical fixed priorities. In other words the smaller the period of an operation is, the greater its priority is, like in the RM scheduling. Note that the scheduling analysis of a system of preemptive tasks with fixed priorities has been a pivotal basis in real-time application development since the work of *Liu* and *Layland* [9]. Now, we assume that any operation of the system may only be preempted by those previously scheduled, and that any operation is scheduled as soon as the processor is free, i.e. no idle time is allowed between the end of the first instance of an operation and the start time of the first instance of the next operation relatively to $\prec$. This assumption about no idle time allows the greatest possible utilization factor of the processor to be achieved. Therefore, to schedule an operation τ_i relatively to those previously scheduled, amounts to filling available spaces in the scheduling with corresponding slices of the exact WCET of τ_i. Consequently, from the point of view of operation τ_i the start time s_i^0 of its first instance is yielded by the end of the first instance of τ_{i-1}. Thus, the notion of release time of τ_i is not relevant in this paper, or is equal to s_i^0.

A *potential schedule* $\mathcal{S}$ of a system is given by a list of the start times of the first instance of all the operations:

$$\mathcal{S} = \{(s_1^0, s_2^0, \cdots, s_n^0)\} \tag{5}$$

The start times s_i^k ($k \geq 1$, $i = 1 \cdots n$) of the other instances of operation τ_i are directly deduced from the first one, and this advantage derives directly from the model. The *response time* R_i^k of the k^{th} instance of operation $\tau_i = (C_i, T_i)$ is the time elapsed between its start time s_i^k and its end time. This latter takes into account the preemption thus,

$$R_i^k \geq C_i \quad \forall k.$$

We call R_i the *worst response time* of operation τ_i, defined as the maximum of the response times of all its instances.

These definitions enable us to say that, in order to satisfy the strict periodicity, any operation $\tau_i = (C_i, T_i)$ of a potentially schedulable system in Ω_λ must satisfy:

$$R_i \leq T_i \quad \forall i \in \{1, \cdots, n\} \tag{6}$$

We say that a system in Ω_λ has one *overlapping* when the start time of any instance of a given operation occurs while the processor is occupied by a previously scheduled operation. Such systems are not schedulable, as expressed in the following theorem.

Theorem 3

Given a system of n operations in Ω_λ, if there are two operations $\tau_i = (C_i, T_i)$ and $\tau_j = (C_j, T_j)$ with $(\tau_i \prec \tau_j)$ starting their executions respectively at the dates s_i^0 and s_j^0 such that for $k \geq 1$

$$\exists\, \beta < k \text{ and } 0 \leq (s_j^0 + \beta T_j) - (s_i^0 + (k-1)T_i) < R_i^k \tag{7}$$

then the system is not schedulable. Moreover any additional assumption on the system intending to satisfy all the constraints is of no interest in this case.

Proof: The proof of this theorem derives directly from the assumption that an operation may only be preempted by those previously scheduled, and it is detailed in [16]. An example is given below (see figure 2). ■

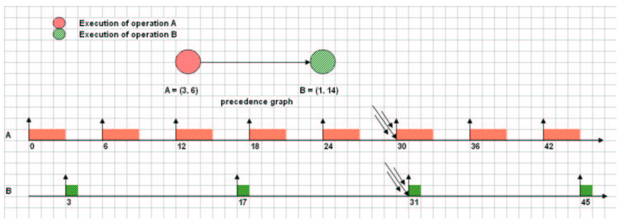


Figure 2. System with an overlapping

Now we can partition Ω_λ into the three following disjoint subsets: the subset V_c of systems with overlappings which are not schedulable thanks to theorem 3, the subset V_r of systems with regular operations, i.e. where the periods of all the operations constitute an harmonic sequence, and the subset V_i of systems with irregular operations. Thus, since the subset of operations where $T_i \wedge T_j = 1$ holds, the subset of operations where $T_i \wedge T_j \mid (s_j^0 - s_i^0)$ holds, and the subset V_c are not schedulable, only the remaining subsets V_r and V_i are potentially schedulable (see figure 3).

$$V_c = \{\{(C_i, T_i)\}_{1 \leq i \leq n} \in \Omega_\lambda \,/\, \exists i \in \{1, \cdots, n-1\},\ \exists j \in \{i+1, \cdots, n\} \quad \text{and} \quad 0 \leq (s_j^0 + \beta T_j) - (s_i^0 + (k-1)T_i) < R_i^k, \quad k \geq 1; \beta \in \mathbb{N}\}$$

$$V_r = \{\{(C_i, T_i)\}_{1 \leq i \leq n} \in \Omega_\lambda \,/\, T_1 \mid T_2 \mid \cdots \mid T_n\}$$

$$V_i = \Omega_\lambda \backslash (V_c \cup V_r)$$

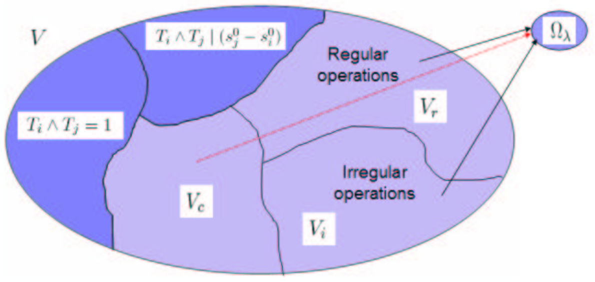


Figure 3. Ω_λ-partitioning

In the remainder of this paper, we restrict our scheduling analysis to the subset V_r.

4 Scheduling analysis for V_r

Given any system in V_r, both the exact WCET C_i^k and the response time R_i^k of the k^{th} instance of a given operation τ_i are the same for all its instances, $C_i^k = C_i^* =$

$C_i + N_p(\tau_i) \cdot \alpha$ and $R_i^k = R_i$ (equal to the worst response time R_i of the operation) because the number of available spaces left in each instance does not depend on the instance itself. Therefore it is worth, in this case, noticing that it is sufficient to give a schedulability condition for the first instance of each operation.

We call U_p (respectively U_p^*) the p^{th} *temporary load factor* (respectively the *exact* p^{th} *temporary load factor*) of the processor ($1 \leq p \leq n$) for a system of n operations $\{\tau_i = (C_i, T_i)\}_{1 \leq i \leq n}$ in V_r.

$$U_p = \sum_{i=1}^{p} \frac{C_i}{T_i} \quad \text{and} \quad U_p^* = \sum_{i=1}^{p} \frac{C_i^*}{T_i} = U_p + \sum_{i=1}^{p} \frac{N_p(\tau_i) \cdot \alpha}{T_i}$$

This system will be said to be *potentially schedulable* if and only if:

$$U_n \leq 1 \tag{8}$$

and *schedulable* if and only if:

$$U_n^* \leq 1 \tag{9}$$

Notice that in (8), C_i is the WCET of operation τ_i without preemption. From now on, we assume (8) is always satisfied.

We say that the exact WCET $C_i^* = C_i + N_p(\tau_i) \cdot \alpha$ of an operation $\tau_i = (C_i, T_i)$ of a system in V_r is a *critical* WCET if its scheduling causes a temporal *delay* to the start time of the first instance of operation $\tau_{i+1} = (C_{i+1}, T_{i+1})$, $\tau_i \prec \tau_{i+1}$. In other words, this means from the point of view of operation τ_i that C_i^* is critical when $s_{i+1}^0 > s_i^0 + R_i^1$, see figure 4. Indeed, in this case the last slice of the exact WCET of τ_i exactly fits the next available space in the scheduling, and thus the first instance of the next operation relatively to $\prec$ cannot start exactly at the end of the first instance of τ_i.

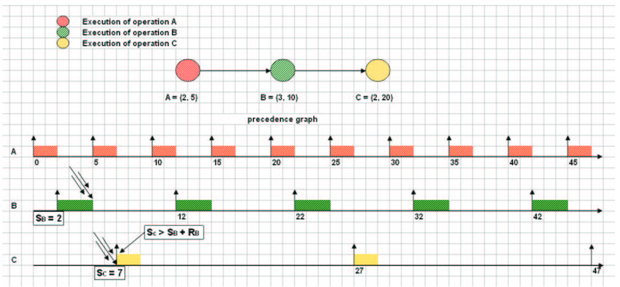


Figure 4. Operation with a critical WCET

In order to make it easier to understand the general case, we first study the simpler case of only two operations. Both cases are based on the same principle which consists, for an operation, in filling available spaces left in each instance with slices of its exact WCET taking into account the cost of the exact number of preemptions necessary for its scheduling.

4.1 System with two operations

We consider $\tau_1 = (C_1, T_1)$ and $\tau_2 = (C_2, T_2)$ to be a system with two operations in V_r such as $T_1 \mid T_2$.

To be consistent with what we have presented up to now, we will first schedule τ_1, and then τ_2, $\tau_1 \prec \tau_2$. Hence, since no idle time is allowed between the end of the first instance of τ_1 and the start time of the first instance of τ_2, we have:

$$C_1^* = C_1 \quad \text{and thus} \quad R_1 = C_1 \quad \text{and} \quad s_2^0 = s_1^0 + R_1 \tag{10}$$

Without any loss of generality, we assume in the remainder of this paper that $s_1^0 = 0$. Because the system is potentially schedulable, we have:

$$\left(\left\lceil \frac{R_1 + T_2}{T_1} \right\rceil - 1 \right) \cdot C_1^* + C_2 \leq T_2, \tag{11}$$

i.e. operation τ_2 is schedulable without taking into account the cost of the preemption.

Now, on the one hand, if:

$$C_1 + C_2 \leq T_1$$

then operation τ_2 is schedulable without any preemption, and we have:

$$C_2^* = C_2 \quad \text{and} \quad R_2 = C_2 \tag{12}$$

On the other hand, if:

$$C_1 + C_2 > T_1 \tag{13}$$

then the system requires at least one preemption of operation τ_2 to be schedulable. To compute the exact number of preemptions $N_p(\tau_2)$, we perform the algorithm below, using a sequence of Euclidean divisions.

We denote $e = T_1 - C_1$ and we initialize $C^1 = C_2$. The Euclidean division of C^1 by e gives:

$$C^1 = q_1 \cdot e + r_1 \text{ with } q_1 = \left\lfloor \frac{C^1}{e} \right\rfloor \text{ and } 0 \leq r_1 < e$$

For all $k \geq 0$, we compute

$$C^{k+1} = r_k + q_k \cdot \alpha \tag{14}$$

and at each step, we perform the Euclidean division of C^{k+1} by e which gives:

$$C^{k+1} = q_{k+1} \cdot e + r_{k+1} \text{ with } q_{k+1} = \left\lfloor \frac{C^{k+1}}{e} \right\rfloor \text{ and } 0 \leq r_{k+1} < e$$

We stop the algorithm as soon as: either there exists $m_1 \geq 1$ such that $\sum_{i=1}^{m_1} q_i \cdot e > T_2(1 - U_1^*)$, and thus the operation τ_2 is not schedulable in this case, or

$$\exists m_2 \geq 1 \quad \text{such that} \quad C^{m_2} \leq e \tag{15}$$

and thus, $N_p(\tau_2)$ is given by:

$$N_p(\tau_2) = \sum_{i=1}^{m_2 - 1} q_i \tag{16}$$

Hence

$$C_2^* = C_2 + N_p(\tau_2) \cdot \alpha \tag{17}$$

and the worst response time R_2 of the operation τ_2 is given by:

$$R_2 = R_2^0 - s_2^0 \tag{18}$$

where:

$$R_2^0 = C_2^* + \left\lceil \frac{R_2^0}{T_1} \right\rceil \cdot C_1^* \tag{19}$$

R_2^0 is easily obtained by using a fixed point algorithm according to:

$$\begin{cases} R_2^{0,l+1} = & C_2^* + \left\lceil \frac{R_2^{0,l}}{T_1} \right\rceil \cdot C_1^* \quad \forall l \geq 0 \\ R_2^{0,0} = & C_2^* \end{cases} \tag{20}$$

The algorithm is stopped as soon as two successive terms of the iteration are equal:

$$R_2^{0,l+1} = R_2^{0,l}, \quad l \geq 0 \tag{21}$$

To simplify the notation, the worst response time will be written as:

$$R_2 = \left\{ R_2^0 = C_2^* + \left\lceil \frac{R_2^0}{T_1} \right\rceil \cdot C_1^* \right\} - s_2^0 \tag{22}$$

Therefore a necessary and sufficient schedulability condition for operation τ_2, and thus for the system $\{\tau_1, \tau_2\}$ taking into account the cost of the preemption is given by:

$$U_2^* \leq 1 \quad \text{i.e.,} \quad U_2 + \frac{N_p(\tau_2) \cdot \alpha}{T_2} \leq 1 \tag{23}$$

Example 1

Let τ_1 and τ_2 be a system with two operations in V_r with the characteristics defined in table 1:

Table 1. Characteristics of example 4.1

	C_i	T_i
τ_1	2	5
τ_2	4	10

We have: $U_2 = \frac{2}{5} + \frac{4}{10} = 0.8$ and $e = 3$.

As operation τ_1 is never preempted, its worst response time R_1 is equal to its worst-case execution time: $R_1 = C_1^* = C_1 = 2$.

Because $\tau_1 \prec \tau_2$, these operations are schedulable if and only if preemption is used (is mandatory).

Although it is not realistic, let $\alpha = 1$ be the cost of one preemption for the processor in order to show clearly the impact of the preemption. Since $C_1 + C_2 = 6 > T_1 = 5$, the computation of $N_p(\tau_2)$ is summarized in the table below:

Therefore, there is only one preemption $N_p(\tau_2) = 1$ (see figure 5) and $C_2^* = 4 + 1 \cdot 1 = 5$

According to (20), $R_2^0 = 9$, and the worst response time R_2 of operation τ_2 is given by:

Table 2. computation of $N_p(\tau_2)$

Steps	q_i	C^i	r_i
1	1	4	1
2	0	2	2

$$R_2 = 9 - 2 = 7 \quad \text{and we have} \quad R_2 \leq T_2 = 10$$

Thus the system is schedulable because:

$$U_2^* = U_2 + \frac{N_p(\tau_2) \cdot \alpha}{T_2} = 0.9 \leq 1.$$

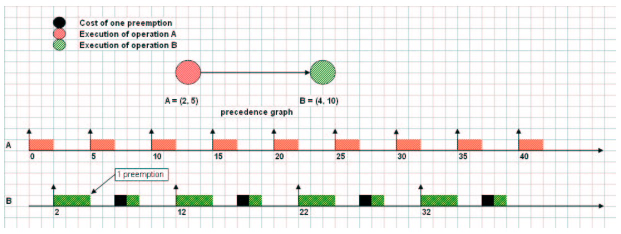


Figure 5. Scheduling of two operations

4.2 System with $n > 2$ operations

The strategy we will adopt in this section of calculating the exact number of preemptions for an operation is different from the one used in the previous section, because we can no longer perform a simple Euclidean division. Although, we can perform the Euclidean division to find the number of preemptions for the second operation, this technique cannot be usable for a third operation, and so on. Actually, the available spaces left after having scheduled the second operation may not be equal, as shown in example 4.2 below, see figure 6.

Example 2

Let $\alpha = 1$ and $\{\tau_1, \tau_2, \tau_3, \tau_4\}$ be a system with four operations in V_r with the characteristics defined in table 3:

Table 3. Characteristics of example 4.2

	C_i	T_i
τ_1	2	5
τ_2	1	10
τ_3	3	20
τ_4	3	40

The schedule is depicted in figure 6.

In figure 6, it can be seen that after the scheduling of the first operation, the available spaces left have equal lengths (3 time units) but it is no longer the case after the scheduling of the second operation, and thus for the third operation after the scheduling of the second operation, and so on.

The intuitive idea of our algorithm consists in two main steps for each operation, according to the precedence relation. First, determine the total number of available time units in each instance, and then the lengths of each available space (consecutive available time units). These data

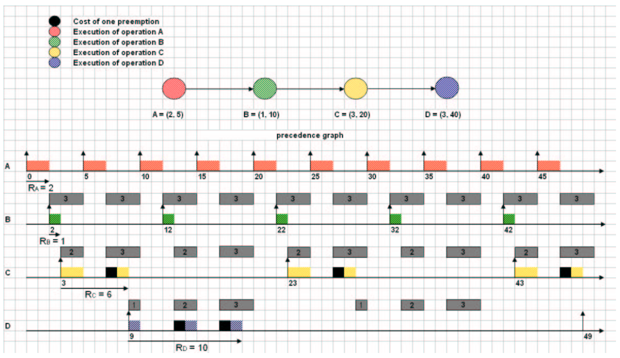


Figure 6. Difficulty of using a simple Euclidean division

allow the computation of the instants when the preemptions occur. A preemption occurence corresponds to the switch from an available time unit to an already executed one. Second, for each potentially schedulable operation, fill available spaces with slices of its WCET up to the value of its WCET, and then add the cost of the preemptions ($p \cdot \alpha$ for p preemptions) to the current inflated WCET, taking into account the increase in the execution time of the operation because of the cost of the preemptions themselves. Finally, the last inflated WCET corresponds to the exact WCET. Thus, it is possible to verify the schedulability condition and then whether this operation is schedulable.

Notice that the number of available spaces is the same for all the instances of an operation, thus it is only necessary to verify the schedulability condition in the first instance which is bounded by the period of the operation. In addition, this verification is performed only once for each operation. Consequently, the complexity of the algorithm even though it has not been yet computed precisely, will actually not explode.

Before going through our proposed algorithm, let us make some assumptions:

1. we will add the cost due to the preemptions to the scheduling analysis of a system if and only if the system is already schedulable without taking it into account, that is $\sum_{i=1}^{n} \frac{C_i}{T_i} \leq 1$.
2. we have scheduled the first $j-1$ $(2 \leq j \leq n-1)$ operations, and we are about to schedule the j^{th} operation,
3. we have potentially enough available spaces to schedule operation τ_j, that is to say:

$$\sum_{i=1}^{j-1} \left(\left\lceil \frac{s_j^0 + T_j}{T_i} \right\rceil - \left\lceil \frac{s_j^0}{T_i} \right\rceil \right) \cdot C_i^* + C_j \leq T_j$$

Under assumption 2, if F_j denotes the number of available time units left in each instance of the operation τ_j, then we have:

$$F_j = T_j \cdot (1 - U_{j-1}^*) \tag{24}$$

Therefore, the operation $\tau_j = (C_j, T_j)$ is schedulable if and only if:

$$0 < C_j^* \leq F_j \quad \text{i.e.,} \quad C_j^* \in \{1, \cdots, F_j\} \tag{25}$$

Let:

$$\mathcal{L}_j = \{1, \cdots, F_j\} \tag{26}$$

$\mathcal{L}_j$ denotes the set of all the possible exact WCET C_j^* of operation $\tau_j = (C_j, T_j)$. Thus, it also contains all the possible WCETs for operation τ_j. Once (25) is satisfied, the worst response time of τ_j is given by:

$$R_j = \left\{ R_j^0 = C_j^* + \sum_{i=1}^{j-1} \left\lceil \frac{R_j^0}{T_i} \right\rceil \cdot C_i^* \right\} - s_j^0 \tag{27}$$

and R_j is obtained by using a fixed point algorithm similar to the one given in the previous section, used to obtain R_2.

4.3 Scheduling algorithm

Hereafter is the scheduling algorithm which counts the exact number of preemptions in order to accurately take into account its cost in the schedulability condition. It has the twelve following steps.

1: Determine the start time s_j^0 of the first instance of operation $\tau_j = (C_j, T_j)$ according to whether the exact WCET $C_{j-1}^* = C_{j-1} + N_p(\tau_{j-1}) \cdot \alpha$ of operation $\tau_{j-1} = (C_{j-1}, T_{j-1})$ is critical or not.

2: Calculate the number of available time units F_j left in each instance of τ_j, and build the set $\mathcal{L}_j$ thanks to relations (24) and (25).

3: Make a first ordered partition of $\mathcal{L}_j$ in $k_{j-1} = \dfrac{T_j}{T_{j-1}}$ sub-sets of equal cardinals such that:

$$\mathcal{L}_j = \mathcal{L}_j^1 \cup \mathcal{L}_j^2 \cup \cdots \cup \mathcal{L}_j^{k_{j-1}} \quad \text{with}$$

$$\left\| \begin{array}{l} \mathcal{L}_j^1 = \left\{ 1, \cdots, \dfrac{F_j}{k_{j-1}} \right\} \\ \mathcal{L}_j^2 = \left\{ \dfrac{F_j}{k_{j-1}} + 1, \cdots, 2\dfrac{F_j}{k_{j-1}} \right\} \\ \vdots \\ \mathcal{L}_j^{k_{j-1}} = \left\{ (k_{j-1} - 1)\dfrac{F_j}{k_{j-1}} + 1, \cdots, F_j \right\} \end{array} \right.$$

4: For each subset $\mathcal{L}_j^i$ obtained in the previous step, make, if possible, a second ordered partition in h_{j-1} subsets such that:

$$\mathcal{L}_j^i = \mathcal{L}_j^{i,1} \cup \mathcal{L}_j^{i,2} \cup \cdots \cup \mathcal{L}_j^{i,h_{j-1}}; \quad i = 1, \cdots, k_{j-1}$$

where the cardinal of each $\mathcal{L}_j^{i,\sigma}$ with $2 \leq \sigma \leq h_{j-1}$ equals the cardinal of the subset at the same position in the partition of $\mathcal{L}_{j-1}$ starting from the subset with the greatest pair (k_{j-2}, h_{j-2}) of indices (the subset the furthest on the right).

To make this step clear, let us give an example with $(k_{j-2}, h_{j-2}) = (2,2)$.
Let the partition of $\mathcal{L}_{j-1}$ be such that:

$$\begin{aligned} \mathcal{L}_{j-1} &= \mathcal{L}_{j-1}^{1,1} \cup \mathcal{L}_{j-1}^{1,2} \cup \mathcal{L}_{j-1}^{2,1} \cup \mathcal{L}_{j-1}^{2,2} \\ &= \{1,2\} \cup \{3,4,5\} \cup \{6,7\} \cup \{8,9,10\} \end{aligned}$$

and let $\mathcal{L}_j$, and k_{j-1} be such that:

$$\begin{cases} \mathcal{L}_j = \{1,2,3,4,5,6,7,8,9,10,11,12\} \\ k_{j-1} = 2 \end{cases}$$

Thanks to step 3, we have:

$$\mathcal{L}_j = \mathcal{L}_j^1 \cup \mathcal{L}_j^2$$

where
$\mathcal{L}_j^1 = \{1,2,3,4,5,6\}$ and $\mathcal{L}_j^2 = \{7,8,9,10,11,12\}$
In step 4, we obtain:

$$\begin{aligned} \mathcal{L}_j^1 &= \mathcal{L}_j^{1,1} \cup \mathcal{L}_j^{1,2} \cup \mathcal{L}_j^{1,3} \\ &= \{1\} \cup \{2,3\} \cup \{4,5,6\} \\ \mathcal{L}_j^2 &= \mathcal{L}_j^{2,1} \cup \mathcal{L}_j^{2,2} \cup \mathcal{L}_j^{2,3} \\ &= \{7\} \cup \{8,9\} \cup \{10,11,12\} \end{aligned}$$

Thus, at the end of step 4, we can write:

$$\mathcal{L}_j = \bigcup_{i=1}^{k_{j-1}} \left\{ \bigcup_{\sigma=1}^{h_{j-1}} \mathcal{L}_j^{i,\sigma} \right\} \tag{28}$$

5: Set:

$$\left\| \begin{array}{l} \overline{0} = \mathcal{L}_j^{1,1} \\ \overline{1} = \mathcal{L}_j^{1,2} \\ \vdots \\ \overline{h_{j-1} - 1} = \mathcal{L}_j^{1,h_{j-1}} \\ \overline{h_{j-1}} = \mathcal{L}_j^{2,1} \\ \vdots \\ \overline{2h_{j-1} - 1} = \mathcal{L}_j^{2,h_{j-1}} \\ \overline{2h_{j-1}} = \mathcal{L}_j^{3,1} \\ \vdots \\ \overline{k_{j-1}h_{j-1} - 1} = \mathcal{L}_j^{k_{j-1},h_{j-1}} \end{array} \right.$$

$\overline{\theta}$ denotes the subset of the possible exact WCETs C_j^* of operation τ_j, preempted θ times. Because operation τ_j is potentially schedulable, thus:

$$\exists \theta_1 \in \{0, 1, \cdots, k_{j-1}h_{j-1} - 1\} \quad \text{and} \quad C_j \in \overline{\theta_1} \tag{29}$$

If $\theta_1 = 0$, then $N_p(\tau_j) = 0$. If it is not the case, i.e. $\theta_1 \neq 0$, thus we obtain for operation τ_j the exact number of preemptions $N_p(\tau_j)$ using the algorithm below:
We initialize

$$\begin{cases} C^1 = C_j \\ q_1 = \theta_1 \\ A^1 = \sum_{k=0}^{\theta_1 - 1} card(\overline{k}) \\ r_1 = C^1 - A^1 \end{cases}$$

For $l \geq 1$, we compute:

$$B^{l+1} = \sum_{k=1}^{l} A^k + (r_l + \theta_l \cdot \alpha) \tag{30}$$

If $B^{l+1} \leq F_j$, thus $\exists \theta_{l+1} \geq 0$ such that $B^{l+1} \in \overline{\theta_1 + \cdots + \theta_{l+1}}$. If $\theta_{l+1} = 0$, then expression (31) holds with $m_2 = l+1$ and $N_p(\tau_j)$ is given by (32), else we set:

$$\begin{cases} C^{l+1} = r_l + \theta_l \cdot \alpha \\ q_{l+1} = \theta_{l+1} \\ A^{l+1} = \sum_{k=\theta_1+\cdots+\theta_l}^{\theta_1+\cdots+\theta_{l+1}-1} card(\overline{k}) \\ r_{l+1} = C^{l+1} - A^{l+1} \end{cases}$$

The algorithm is stopped as soon as: either there exists $m_1 \geq 1$ such that $B^{m_1} > F_j$, and thus operation τ_j is not schedulable in this case, or

$$\exists m_2 \geq 1 \quad \text{such that} \quad \theta_{m_2} = 0 \tag{31}$$

and therefore:

$$N_p(\tau_j) = \sum_{k=1}^{m_2 - 1} q_k \tag{32}$$

We compute the exact WCET C_j^* of operation τ_j:

$$C_j^* = C_j + N_p(\tau_j) \cdot \alpha \tag{33}$$

6: Determine the set I_j of all the possible critical exact WCETs C_j^* of operation $\tau_j = (C_j, T_j)$. Each element of I_j is the maximum of each subset $\mathcal{L}_j^{i,\sigma}$, except F_j, with $(1 \leq i \leq k_{j-1})$ and $(1 \leq \sigma \leq h_{j-1})$ obtained in step 4.
We distinguish between two types of critical exact WCETs: *critical exact WCET of the first order* and *critical exact WCET of the second order*.
Critical exact WCET of the first order consists of the ordered set I_j^1 given by:

$$\begin{aligned} I_j^1 &= \left\{ max(\mathcal{L}_j^i) \quad \text{for} \quad 1 \leq i \leq k_{j-1} \right\} \setminus \{F_j\} \\ &= \left\{ \frac{F_j}{k_{j-1}}, 2\frac{F_j}{k_{j-1}}, \cdots, (k_{j-1} - 1)\frac{F_j}{k_{j-1}} \right\} \end{aligned}$$

Critical exact WCET of the second order consists of the ordered set I_j^2 given by:

$$\begin{aligned} I_j^2 &= \bigcup_{i=1}^{k_{j-1}} I_j^{2,i} \quad \text{with} \\ I_j^{2,i} &= \left\{ max(\mathcal{L}_j^{i,\sigma}) \quad \text{for} \quad 1 \leq \sigma \leq h_{j-1} \right\} \setminus I_j^1 \end{aligned}$$

Hence $I_j = I_j^1 \cup I_j^2$ and can be rewritten as the ordered set defined by:

$$\begin{aligned} I_j = I_j^{2,1} \cup \left\{ \frac{F_j}{k_{j-1}} \right\} \cup I_j^{2,2} \cup \left\{ 2\frac{F_j}{k_{j-1}} \right\} \cup \cdots \\ \cdots \cup \left\{ (k_{j-1} - 1)\frac{F_j}{k_{j-1}} \right\} \cup I_j^{2,k_{j-1}} \end{aligned} \tag{34}$$

Again, to make this step clear, let us give an example, using the same one as in step 4. In this step we obtain:

$$I_j^1 = \{max(\mathcal{L}_j^1), max(\mathcal{L}_j^2)\} \backslash \{12\} = \{6\}$$

$$I_j^{2,1} = \left\{max(\mathcal{L}_j^{1,\sigma}), \quad 1 \leq \sigma \leq 3\right\} \backslash \{6\} = \{1,3\}$$

$$I_j^{2,2} = \left\{max(\mathcal{L}_j^{2,\sigma}), \quad 1 \leq \sigma \leq 3\right\} \backslash \{6\} = \{7,9\}$$

Thus, by writing I_j like in expression (34) we obtain:

$$I_j = \{1,3\} \cup \{6\} \cup \{7,9\}$$

7: Determine whether C_j^* is a critical WCET, i.e. $C_j^* \in I_j$, or not, thanks to step 6.

8: Determine the delay $\Lambda_j(C_j)$ that operation τ_j will cause to the start time of the first instance of operation $\tau_{j+1} = (C_{j+1}, T_{j+1})$. There are three possible cases for C_j^*:

- $C_j^* \in \mathcal{L}_j \backslash I_j$, i.e. C_j^* is not a critical exact WCET, then:
$$\Lambda_j(C_j) = 0 \tag{35}$$
- $C_j^* \in I_j^1$, i.e. C_j^* is a critical exact WCET of the first order, then:
$$\Lambda_j(C_j) = s_j^0 \tag{36}$$
- $C_j^* \in I_j^2$, i.e. C_j^* is a critical exact WCET of the second order, then:
$$\Lambda_j(C_j) = \Lambda_{j-1}(C_{j-1}^0) \tag{37}$$
such that for each possible value $C_j^{0,i} \in I_j^{2,i}$ of C_j^* with $(1 \leq i \leq k_{j-1})$,
$$\Lambda_j(C_j^{0,i}) = \Lambda_{j-1}(C'_{j-1})$$
where $C'_{j-1} \in I_{j-1}$ and C'_{j-1} is at the same position in I_{j-1} written as in (34) as $C_j^{0,i}$ in $I_j^{2,i}$, starting in I_{j-1} from its maximum which belongs to the sub-set with the greatest pair $(2, k_{j-2})$ of indices $I_{j-1}^{2,k_{j-2}}$ (the subset the furthest on the right).

Again, to make this step clear, let us give an example, using that of step 4. Thanks to everything we have presented up to now,

$$I_{j-1} = I_{j-1}^{2,1} \cup \{5\} \cup I_{j-1}^{2,2} = \{2\} \cup \{5\} \cup \{7\}$$

if we assume we had:
$\Lambda_{j-1}(5) = s_{j-1}^0$ and $\Lambda_{j-1}(2) = \Lambda_{j-1}(7) = s_{j-2}^0$, then as

$$I_j = \{1,3\} \cup \{6\} \cup \{7,9\}$$

In this step we obtain:

$$\begin{cases} \Lambda_j(6) = s_j^0 \quad \text{because} \quad 6 \in I_j^1 \\ \Lambda_j(3) = \Lambda_j(9) = \Lambda_{j-1}(7) = s_{j-2}^0 \\ \Lambda_j(1) = \Lambda_j(7) = \Lambda_{j-1}(5) = s_{j-1}^0 \end{cases}$$

9: Calculate the worst response time R_j of operation τ_j thanks to expression (27).

10: Increment j: $j \leftarrow j+1$ and determine the start time s_{j+1}^0 of the first instance of operation $\tau_{j+1} = (C_{j+1}, T_{j+1})$ according to whether operation $\tau_j = (C_j, T_j)$ has a critical exact WCET C_j^*, or not.

$$s_{j+1}^0 = R_j + s_j^0 + \Lambda_j(C_j) \tag{38}$$

11: Go back to step 2 as long as there remain potentially schedulable operations.

12: Give the necessary and sufficient schedulability condition:

$$U_n^* \leq 1 \quad \text{i.e.,} \quad U_n + \sum_{i=2}^{n} \frac{N_p(\tau_i) \cdot \alpha}{T_i} \leq 1 \tag{39}$$

and the valid schedule $\mathcal{S}$ for the system taking into account the global cost due to preemptions:

$$\mathcal{S} = \{(s_1^0, s_2^0, \cdots, s_n^0)\} \tag{40}$$

Example 3

Let $\alpha = 1$ and $\{\tau_1, \tau_2, \tau_3, \tau_4\}$ be a system with four operations in V_r with the characteristics defined in table 4.

Table 4. Characteristics of example 4.3

	C_i	T_i
τ_1	2	5
τ_2	1	10
τ_3	3	20
τ_4	3	40

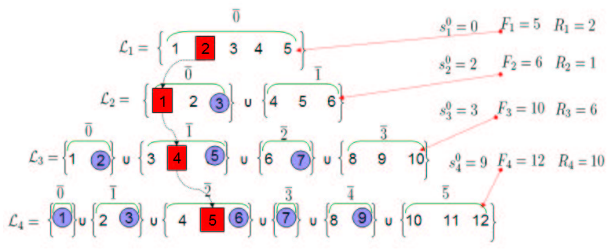
That system is potentially schedulable, indeed:

$$U_4 = \frac{2}{5} + \frac{1}{10} + \frac{3}{20} + \frac{3}{40} = 0.725$$

The scheduling algorithm that we introduced previously gives: $C_1^* = 2, C_2^* = 1, C_3^* = 4, C_4^* = 5$, thus:

$$U_4^* = \frac{2}{5} + \frac{1}{10} + \frac{4}{20} + \frac{5}{40} = 0.825$$

and we obtain (see figure 7):



$\mathcal{L}_1 = \{1\ 2\ 3\ 4\ 5\}$
$\mathcal{L}_2 = \{1\ 2\ 3\} \cup \{4\ 5\ 6\}$
$\mathcal{L}_3 = \{1\ 2\} \cup \{3\ 4\ 5\} \cup \{6\ 7\} \cup \{8\ 9\ 10\}$
$\mathcal{L}_4 = \{1\} \cup \{2\ 3\} \cup \{4\ 5\ 6\} \cup \{7\} \cup \{8\ 9\} \cup \{10\ 11\ 12\}$
$\bar{0}$ $\bar{1}$ $\bar{2}$ $\bar{3}$ $\bar{4}$ $\bar{5}$
$s_1^0 = 0$ $F_1 = 5$ $R_1 = 2$
$s_2^0 = 2$ $F_2 = 6$ $R_2 = 1$
$s_3^0 = 3$ $F_3 = 10$ $R_3 = 6$
$s_4^0 = 9$ $F_4 = 12$ $R_4 = 10$

Figure 7. Scheduling algorithm

In figure 7, for each operation, we can see its actual exact WCET (squared), its critical exact WCET (circled), and its exact number of preemptions.

The global cost due to preemption is given by:

$$pr = \frac{1}{10} \cdot 0 + \frac{1}{20} \cdot 1 + \frac{1}{40} \cdot 2 = 0.1$$

and therefore the schedulability condition is:

$$U_4^* = U_4 + pr = 0.825 \leq 1$$

The valid schedule of the system of operations obtained with our scheduling algorithm is given in figure 8, and is such that:

$$\begin{aligned} \mathcal{S} &= \{(s_1^0, s_2^0, s_3^0, s_4^0) = (0, R_1, R_2 + s_2^0, R_3 + s_3^0)\} \\ &= \{(0, 2, 3, 9)\} \end{aligned}$$

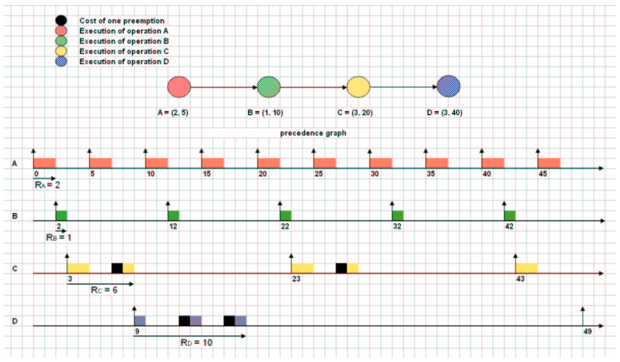


Figure 8. Preemptions taken into account

5 Conclusion and future work

We are interested in hard real-time systems with precedence and strict periodicity constraints where it is mandatory to satisfy these constraints. We are also interested in preemption which offers great advantages when seeking schedules. Since classical approaches are based on an approximation of the cost of the preemption in WCETs, possibly leading to a wrong real-time execution, we proposed a constructive approach so that its cost may be taken into account accurately. We proposed a scheduling algorithm which counts the exact number of preemptions for a system in V_r which is the subset of systems where the periods of all operations constitute an harmonic sequence as presented in section 3.2, and thus gives a stronger schedulability condition than Liu & Layland's condition.

Currently, we are seeking a schedulability condition for systems in V_i which is the subset of systems with irregular operations and we are planning to study the complexity of our approach in both V_r and V_i. Moreover, because idle time may increase the possible schedules we also plan to allow idle time, even though this would increase the complexity of the scheduling algorithm.

References

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

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

[3] J.P. Lehoczky, L. Sha, and Y Ding. The rate monotonic sheduling algorithm: exact characterization and average case bahavior. *Proceedings of the IEEE Real-Time Systems Symposium*, 1989.

[4] N.C. Audsley, Burns A., M.F. Richardson, and A.J. Wellings. Hard real-time scheduling : The deadline-monotonic approach. *Proceedings 8th IEEE Workshop on Real-Time Operating Systems and Software*, 1991.

[5] H. Chetto and M. Chetto. Some results on the earliest deadline scheduling algorithm. *IEEE Transactions on Software Engineering*, 1989.

[6] Patchrawat Uthaisombut. The optimal online algorithms for minimizing maximum lateness. *Proceedings of the 9th Scandinavian Workshop on Algorithm Theory (SWAT)*, 2003.

[7] M. Joseph and P. Pandya. Finding response times in real-time system. *BCS Computer Journal*, 1986.

[8] Burns A. Tindell K. and Wellings A. An extendible approach for analysing fixed priority hard real-time tasks. *J. Real-Time Systems*, 1994.

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

[10] Tei-Wei Kuo and Aloysius K. Mok. Load adjustment in adaptive real-time systems. *Proceedings of the 12th IEEE Real-Time Systems Symposium*, 1991.

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

[12] I. Ripol J. Echage 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] 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.

[14] J.H.M. Korst, E.H.L. Aarts, and J.K. Lenstra. Scheduling periodic tasks. *INFORMS Journal on Computing 8*, 1996.

[15] L. Cucu and Y. Sorel. Schedulability condition for systems with precedence and periodicity constraints without preemption. In *Proceedings of 11th Real-Time Systems Conference, RTS'03*, Paris, March 2003.

[16] P. Meumeu and Y. Sorel. Non-schedulability conditions for off-line scheduling of real-time systems subject to precedence and strict periodicity constraints. In *Proceedings of 11th IEEE International Conference on Emerging technologies and Factory Automation, ETFA'06, WIP*, Prague, Czech Republic, September 2006.

[17] Radu Dobrin and Gerhard Fohler. Reducing the number of preemptions in fixed priority scheduling. In *16th Euromicro Conference on Real-time Systems (ECRTS 04)*, Catania, Sicily, Italy, July 2004.