The Space of EDF Feasible Deadlines

Enrico Bini, **Giorgio Buttazzo**
Scuola Superiore Sant'Anna
Pisa, Italy
e.bini@sssup.it, giorgio@sssup.it

Abstract

It is well known that the performance of computer controlled systems is heavily affected by delays and jitter occurring in the control loops, which are mainly caused by the interference introduced by other concurrent activities. A common approach adopted to reduce delay and jitter in periodic task systems is to decrease relative deadlines as much as possible, but without jeopardising the schedulability of the task set.

In this paper, we formally characterise the region of admissible deadlines so that the system designer can appropriately select the desired values to maximise a given performance index defined over the task set. Finally we also provide a sufficient region of feasible deadlines which is proved to be convex.

1. Introduction

The software of control systems is typically implemented through a set of periodic activities performing data sampling, sensory processing, control, action planning, and actuation. Although not strictly necessary, periodic execution simplifies the design of control algorithms and allows using standard control theory to guarantee system stability and performance requirements.

When several of such activities execute concurrently in the same processor, however, each task may experience a variable delay and jitter, mainly due to the interference created by the other tasks. The amount of delay and jitter experienced by each task depends on several factors, including the scheduling algorithm running in the kernel, the overall workload, and the task parameters (computation times, periods, and deadlines). If not properly taken into account, delays and jitter may degrade the performance of the system and even jeopardise its stability [17, 18, 14].

The problem of jitter in real-time control applications has received increased attention during the last decade and several techniques have been proposed to cope with it. Nilsson [22] analysed the stability and performance of real-time control systems with random delays and derived an optimal, jitter-compensating controller. Martí el al. [21] proposed a compensation technique for controllers based on the pole placement design method. Di Natale and Stankovic [15] proposed the use of simulated annealing to find the optimal configuration of task offsets that minimises jitter, according to some user defined cost function. Cervin et al. [11] presented a method for finding an upper bound of the input-output jitter of each task under EDF scheduling, and introduced the concept of *jitter margin* to simplify the analysis of control systems and guarantee their stability when certain conditions on jitter are satisfied.

A common practice to reduce jitter in control applications is to separate each control task into three distinct subtasks performing data input, processing, and control output. Then, the input-output jitter is reduced by postponing the input-output subtasks to some later point in time, so trading jitter with delay [13]. Cervin [10] proposed to split the control algorithm into two parts (Calculate Output and Update State), which are scheduled as separate tasks. This method works fine for simple control applications, but introduces a number of problems. In particular, the jitter reduction is obtained by inserting extra delays in task execution, since input and output parts are always separated by exactly a period, while normally the average delay could be smaller. The effect of having a higher delay in the control loop has to be carefully analysed, since it could be more negative than the effect of jitter. A recent performance study involving several LQG controllers (with a direct term) indicates that delay is worse than jitter, except when the sampling rate is extremely low [20].

Another approach widely adopted for reducing jitter and delay is to limit the execution interval of each task by setting a suitable relative deadline. Working on this line, Baruah et al. [3] proposed two methods (with different complexity and performance) for assigning shorter relative deadlines to tasks and guaranteeing the schedulability of the task set. A comparative evaluation of different jitter reduction approaches has been presented by Cervin and Buttazzo [8].

Several authors [26, 9, 1, 16] independently proposed

19th Euromicro Conference on Real-Time Systems (ECRTS'07)
0-7695-2914-3/07 $20.00 © 2007 IEEE

different algorithms for computing the minimum deadline of a newly arrived task, assuming the existing task set is feasibly schedulable by EDF. The problem of these methods is that they can hardly be extended to reduce a set of arbitrary deadlines, but can only be applied to a single task at the time, following a given order, as suggested by [16]. In this way, however, the only task which experiences a significant deadline reduction is the first task in the sequence, since it can use all the slack available in the task set to minimise its deadline, leaving little margin for the remaining tasks. To apply a more uniform deadline reduction in the task set, Balbastre et al. [1] proposed an algorithm able to scale all deadlines by the same factor. The problem of this approach, however, is that a uniform deadline reduction may not achieve a significant improvement in terms of jitter and delays, because all deadlines are reduced and, in some cases, the schedule could even remain unchanged.

In this paper, we present a general analysis methodology for identifying the *feasibility region of the task deadlines*, when tasks are scheduled by the Earliest Deadline First algorithm [19]. The knowledge of such a region is very useful in the design process, since it allows the designer to perform sensitivity analysis and select the set of relative deadlines that maximises a given performance index defined over the task set.

Different methods for optimising the performance of periodic task sets have been proposed in the literature, both under fixed priority [6] and EDF [25], but only with respect to periods. Sensitivity analysis in the domain of computation times has also been addressed in [7], while sensitivity analysis of any parameters was proposed at the cost of high complexity by Racu et al. [23] using binary search. Hence, this paper fills the missing gap, allowing the designer to reason also in the space of task deadlines.

The rest of the paper is organised as follows. Section 2 presents the system model, the terminology and the basic assumptions. Section 3 formally defines the problem to be solved and provides a simplified explanation of the approach, deriving the feasibility region for two tasks. Section 4 describes the general method for n tasks, and presents the algorithm for deriving the region. Section 7 presents an approximate solution to reduce the complexity of the approach. Finally, Section 8 states our conclusions and future work.

2. Terminology and Assumptions

We consider a set $\mathcal{T} = \{\tau_1, \tau_2, \ldots, \tau_n\}$ of n periodic (or sporadic) tasks that have to be executed on a uniprocessor system under the Earliest Deadline First (EDF) scheduling algorithm [19]. Each task τ_i consists of an infinite sequence of task instances, or jobs, having the same worst-case execution time (WCET), the same relative deadline, and the same interarrival period. The following notation is used throughout the paper.

τ_{ik} denotes the k-th job of task τ_i, with $k \in \mathcal{N}$.

C_i denotes the worst-case execution time (WCET) of task τ_i, that is, the WCET of each job of τ_i. The vector of all the computation times $(C_1, \ldots, C_n)$ is denoted by $\mathbf{C}$.

T_i denotes the period of task τ_i (or the minimum interarrival time for sporadic tasks).

D_i denotes the relative deadline of task τ_i, that is, the maximum finishing time allowed for any job, relative to the its activation time. The vector of all the deadlines $(D_1, \ldots, D_n)$ is denoted by $\mathbf{D}$.

r_{ik} denotes the release time of job τ_{ik}. If the first job is released at time $r_{i,1} = \Phi_i$, also referred to as the task phase, the generic k-th job is released at time

$$r_{ik} = \Phi_i + (k-1)T_i.$$

d_{ik} denotes the absolute deadline of job τ_{ik}, that is the maximum absolute time job τ_{ik} is allowed to complete.

U_i denotes the utilisation of task τ_i, that is, the fraction of CPU time used by τ_i ($U_i = C_i/T_i$).

U denotes the total utilisation of the task set, that is, the sum of all tasks utilisations ($U_i = \sum_{i=1}^{n} U_i$).

We assume all tasks are fully preemptive and are simultaneously activated at time $t = 0$ (that is, $\Phi_i = 0$, for all tasks).

3. Problem statement

We consider the problem of determining the region of the feasible task deadlines (also called the D-space), when tasks are scheduled by the Earliest Deadline First algorithm [19]. Reasoning in such a region allows the designer to perform sensitivity analysis or to select the set of relative deadlines that maximises a given performance index defined over the task set. Later in Section 7, we will also propose an approximate method, based on a simpler sufficient feasibility condition, to identify a convex region entirely enclosed in the D-space.

Before entering into the mathematical details of the proposed approach, we describe a simple example to visualise such a feasibility region of task deadlines. As we proceed with the explanation, this example will be considered throughout the paper as a sample application of the theoretical results.

Let us consider two periodic tasks, τ_1 and τ_2, with computation times $C_1 = 2$ and $C_2 = 3$, and periods $T_1 = 4$ and $T_2 = 7$, respectively. By setting the relative deadlines

0-7695-2914-3/07 $20.00

equal to periods, the task set is feasible by EDF, since the processor utilisation factor is less than one, in fact

$$U = \frac{2}{4} + \frac{3}{7} = \frac{13}{14} < 1.$$

Figure 1 illustrates the corresponding schedule (in the figures, consecutive jobs are drawn using alternated colours).

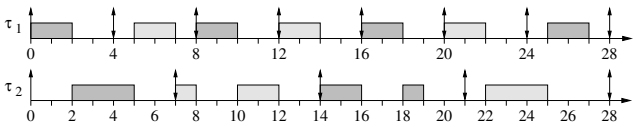

Figure 1. EDF schedule when $D_i = T_i$.

Now note that if we shorten the deadline of τ_1 as much as possible, that is if we set $D_1 = C_1$, the maximum response time of τ_2 becomes $R_2 = 7$, meaning that D_2 cannot be less than 7 if the task set must be feasible. The corresponding schedule produced by EDF with $D_1 = 2$ and $D_2 = 7$ is shown in Figure 2.

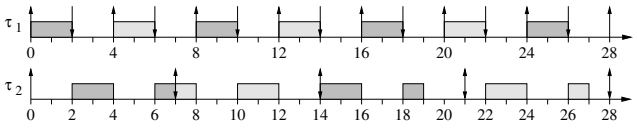

Figure 2. EDF schedule when minimising D_1.

Similarly, if we shorten the deadline of τ_2 back to its computation time, setting $D_2 = C_2 = 3$, the maximum response time for τ_1 becomes $R_1 = 5$, meaning that D_1 cannot be less than 5 to keep the task set schedulable. The corresponding schedule produced by EDF with $D_1 = 5$ and $D_2 = 3$ is shown in Figure 3.

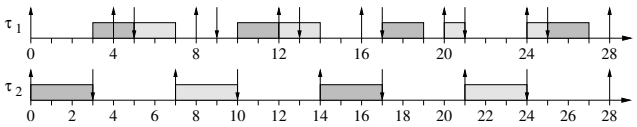

Figure 3. EDF schedule when minimising D_2.

Notice, however, that a feasible schedule can also be achieved with $D_1 = 3$ and $D_2 = 5$, as shown in Figure 4.

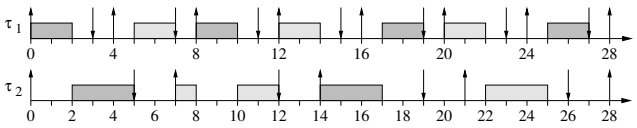

Figure 4. Schedule with $D_1 = 3$ and $D_2 = 5$.

Moreover, we know that EDF on a single processor is *sustainable* with respect to task deadlines [2], meaning that if EDF can feasibly schedule a task set $\mathcal{T}$, then it can also feasibly schedule any task set $\mathcal{T}'$ with the same computation times and periods as in $\mathcal{T}$ and larger deadlines.

From this observation and the cases reported above, we can state that all deadlines corresponding to points in the grey area depicted in Figure 5 generate a feasible EDF schedule (in the figure the deadline values previously discussed are indicated by a black thick dot). Hence, we can observe that;

Observation 1 *The exact EDF feasibility region is* larger than *or* equal to *the grey area shown in Figure 5.*

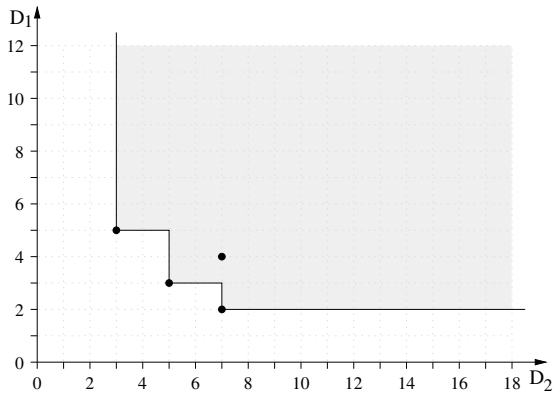

Figure 5. Space of feasible deadlines.

In the remainder of the paper we will show that the region of Figure 5 is necessary and sufficient and present a method for deriving such a region.

4. The space of EDF feasible deadlines

Unfortunately, the feasibility region cannot be described by a closed formula. In fact, as shown by Baruah, Rosier, and Howell [4], a set of periodic tasks simultaneously activated at time $t = 0$ is schedulable by EDF if and only if $U < 1$ and

$$\forall t \geq 0 \quad \sum_{i=1}^{n} \max\left\{0, \left\lfloor \frac{t + T_i - D_i}{T_i} \right\rfloor\right\} C_i \leq t \qquad (1)$$

Baruah, Mok, and Rosier [5] showed that the points in which the test has to be performed can be restricted to those deadlines within the hyperperiod H not exceeding the value

$$L_{\max} = \max\{D_1, \ldots, D_n, L^*\}$$

where

$$L^* = \frac{\sum_{i=1}^{n}(T_i - D_i)U_i}{1 - U}.$$

Hence, we can say that condition (1) has to be tested $\forall t \in \mathsf{dlSet}$, where

$$\mathsf{dlSet} = \{d_{ik} \mid d_{ik} \leq \min(L_{\max}, H)\}. \qquad (2)$$

Depending on the convenience, sometimes we will use the schedulability condition as expressed by equation (1), whereas some other times we will restrict the variable t to range in the set dlSet.

The difficulty of finding a closed formulation in terms of the deadlines is due to the presence of the floor operator in Equation (1). To overcome this problem we follow the same intuition used by Seto, Lehoczhy and Sha [24] to find all the admissible periods in a fixed priority scheduler.

We introduce a set of n functions $\mathsf{K}_i : \mathbb{R}_+ \times \mathbb{R}_+ \rightarrow \mathbb{N}$ defined as

$$\mathsf{K}_i(t, D_i) = \max\left\{0, \left\lfloor \frac{t + T_i - D_i}{T_i} \right\rfloor\right\}. \quad (3)$$

Each function $\mathsf{K}_i(t, D_i)$ compactly denotes the number of jobs of task τ_i entirely included in the interval $[0, t]$. We also observe that, by the definition of Equation (3), $\mathsf{K}_i(t, D_i)$ is the unique integer satisfying the following constraint:

$$\begin{cases} \mathsf{K}_i(t, D_i) = 0 & \text{if } t - D_i < 0 \\ \frac{t-D_i}{T_i} < \mathsf{K}_i(t, D_i) \leq \frac{t-D_i}{T_i} + 1 & \text{otherwise} \end{cases} \quad (4)$$

Finally, all the n integer functions can be condensed in a single function $\mathsf{K} : \mathbb{R}_+ \times \mathbb{R}_+^n \rightarrow \mathbb{N}^n$ defined as

$$\mathsf{K}(t, \mathbf{D}) = (\mathsf{K}_1(t, D_1), \ldots, \mathsf{K}_n(t, D_n)). \quad (5)$$

The introduction of function K is very convenient to write the schedulability conditions with a more compact notation. For example the necessary and sufficient schedulability condition by Baruah et al. expressed in Eq. (1) becomes

$$\forall t \geq 0 \quad \mathsf{K}(t, \mathbf{D}) \cdot \mathbf{C} \leq t. \quad (6)$$

We are now ready for a step forward. The condition resulting from Equation (1), and its equivalent Equation (6), has the disadvantage that it is not clear how to extract a constraint on task deadlines $\mathbf{D}$. The following lemma provides a necessary and sufficient condition from where deadlines constraints can be derived. Although apparently more complicated than condition (6), Lemma 1 allows to find a closed formulation of the EDF schedulability condition.

Lemma 1 *A set of periodic tasks $\mathcal{T}$ is feasibly schedulable by EDF* ***if and only if*** $U \leq 1$ *and:*

$$\forall\, t \geq 0,\ \forall\, \mathbf{k} \in \mathbb{N}^n \quad (\mathbf{k} = \mathsf{K}(t, \mathbf{D}) \ \wedge\ \mathbf{k} \cdot \mathbf{C} \leq t) \ \vee \mathbf{k} \neq \mathsf{K}(t, \mathbf{D}). \quad (7)$$

Proof. We prove the lemma by showing that Eq. (7) is equivalent to Eq. (1).

Eq. (7) $\Rightarrow$ Eq. (1) The proof is performed by contradiction. Let us assume that Eq. (1) is false, meaning that

$$\exists t^* > 0 \quad \sum_{i=1}^{n} \max\left\{0, \left\lfloor \frac{t^* + T_i - D_i}{T_i} \right\rfloor\right\} C_i > t^*. \quad (8)$$

Let us denote, for all i, $k_i^* = \max\left\{0, \left\lfloor \frac{t^* + T_i - D_i}{T_i} \right\rfloor\right\}$ which is also equal to $\mathsf{K}_i(t^*, D_i)$ by the definition of K given in Eq. (3), and $\mathbf{k}^* = (k_1^*, \ldots, k_n^*)$. We show that for such special t^* and $\mathbf{k}^*$ Eq. (7) is false as well. In fact, Eq. (8) can be rewritten as

$$\exists t^* > 0 \quad \mathbf{k} \cdot \mathbf{C} > t^*$$

and $\mathbf{k}^* = \mathsf{K}(t^*, \mathbf{D})$. Hence we have that

$$\exists\, t^* \geq 0,\ \exists\, \mathbf{k}^* \in \mathbb{N}^n \quad (\mathbf{k}^* = \mathsf{K}(t^*, \mathbf{D}) \ \wedge\ \mathbf{k}^* \cdot \mathbf{C} > t^*)$$

which implies that Eq. (7) is false, as required.

Eq. (1) $\Rightarrow$ Eq. (7) Again we proceed by contradiction showing that when Eq. (7) is false then Eq. (1) is false as well. From the negation of Eq. (7), we have that

$$\exists\, t^* \geq 0,\ \exists\, \mathbf{k}^* \in \mathbb{N}^n \quad (\mathbf{k}^* \neq \mathsf{K}(t^*, \mathbf{D}) \ \vee\ \mathbf{k}^* \cdot \mathbf{C} > t^*) \ \wedge\ \mathbf{k}^* = \mathsf{K}(t^*, \mathbf{D}) \quad (9)$$

which is equivalent to

$$\begin{aligned} &\exists\, t^* \geq 0,\ \exists\, \mathbf{k}^* \in \mathbb{N}^n \quad \mathbf{k}^* \cdot \mathbf{C} > t^* \ \wedge\ \mathbf{k}^* = \mathsf{K}(t^*, \mathbf{D}) \\ &\exists\, t^* \geq 0, \quad \mathsf{K}(t^*, \mathbf{D}) \cdot \mathbf{C} > t^* \\ &\exists\, t^* \geq 0, \quad \sum_{i=1}^{n} \mathsf{K}_i(t^*, \mathbf{D})\, C_i > t^* \\ &\exists\, t^* \geq 0, \quad \sum_{i=1}^{n} \max\left\{0, \left\lfloor \frac{t^* + T_i - D_i}{T_i} \right\rfloor\right\} C_i > t^* \end{aligned}$$

which contradicts Eq. (1) and proves the lemma. □

Lemma 1 has the clear advantage that it is possible to find a constraint on the task deadlines $\mathbf{D}$ starting from the relationship $\mathbf{k} = \mathsf{K}(t, \mathbf{D})$. In fact from Eq. (4) it follows that

$$\mathbf{k} = \mathsf{K}(t, \mathbf{D})$$

$$\begin{cases} k_i = 0 & \text{if } t - D_i < 0 \\ \frac{t-D_i}{T_i} < k_i \leq \frac{t-D_i}{T_i} + 1 & \text{otherwise} \end{cases}$$

$$\begin{cases} D_i > t & \text{if } k_i = 0 \\ t - k_i\, T_i < D_i \leq t - (k_i - 1)T_i & \text{otherwise} \end{cases} \quad (10)$$

which gives us the desired constraint on the deadline values.

The set of deadlines satisfying the constraint of Eq. (10), for a specific selection of t and $\mathbf{k}$, is denoted by $\mathsf{domD}(t, \mathbf{k})$, and its complement is denoted by $\mathsf{domD}^c(t, \mathbf{k})$. Due to the equivalence between Eq. (4) and (10), we have

$$\begin{aligned} \mathbf{k} = \mathsf{K}(t, \mathbf{D}) \quad &\Leftrightarrow \quad \mathbf{D} \in \mathsf{domD}(t, \mathbf{k}) \\ \mathbf{k} \neq \mathsf{K}(t, \mathbf{D}) \quad &\Leftrightarrow \quad \mathbf{D} \in \mathsf{domD}^c(t, \mathbf{k}) \end{aligned} \quad (11)$$

As it can be noticed from Eq. (10), the region $\mathsf{domD}(t, \mathbf{k})$ is a multi-dimensional box in the space of deadlines, which reduces to a simple rectangle if $n = 2$. If some k_i in $\mathbf{k}$ is

0-7695-2914-3/07 $20.00

equal to zero, then $\mathsf{domD}(t, \mathbf{k})$ is a degenerate box because its projection on the i^{th} axis is a right-unbounded interval. Figure 6 shows some examples of domD for the same task

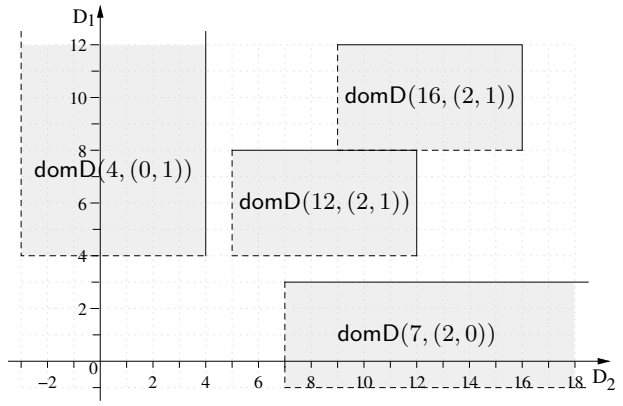

Figure 6. Samples of domD, when $\mathbf{T} = (4, 7)$.

parameters used in the previous example. In the figure it can be also noticed that, when some $k_i = 0$, then the corresponding deadline D_i is not upper bounded in domD. Another property that will be used later is that for the same value of $\mathbf{k}$, as t increases by a certain amount Δ, the corresponding box translates by Δ along all the coordinates. Finally, the complement $\mathsf{domD}^c(t, \mathbf{k})$ basically is all the space with a "boxed hole".

Using the definition of region domD, it is possible to formulate a result which describes the space of all feasible deadlines.

Theorem 1 *A set of periodic tasks $\mathcal{T}$ is feasibly schedulable by EDF* ***if and only if*** *$U \leq 1$ and:*

$$\forall\, \mathbf{k} \in \mathbb{N}^n,\ \forall\, t \in [0, \mathbf{k} \cdot \mathbf{C}) \quad \mathbf{D} \in \mathsf{domD}^c(t, \mathbf{k}) \tag{12}$$

where the set of deadlines $\mathsf{domD}(t, \mathbf{k})$ is defined by

$$\begin{cases} D_i > t & \text{if } k_i = 0 \\ t - k_i T_i < D_i \leq t - (k_i - 1)T_i & \text{otherwise} \end{cases} \tag{13}$$

and domD^c denotes its complement.

Proof. From Lemma 1 and the equivalence expressed by Equation (11) it follows that the task set $\mathcal{T}$ is feasible by EDF if and only if

$$\forall\, t \geq 0,\ \forall\, \mathbf{k} \in \mathbb{N}^n \quad \mathbf{D} \in \mathsf{domD}^c(t, \mathbf{k}) \vee (\mathbf{D} \in \mathsf{domD}(t, \mathbf{k}) \wedge \mathbf{k} \cdot \mathbf{C} \leq t) \tag{14}$$

Since the two "for all" quantifiers do not depend on each other, they can be exchanged obtaining the following condition

$$\forall\, \mathbf{k} \in \mathbb{N}^n,\ \forall\, t \geq 0 \quad \mathbf{D} \in \mathsf{domD}^c(t, \mathbf{k}) \vee (\mathbf{D} \in \mathsf{domD}(t, \mathbf{k}) \wedge \mathbf{k} \cdot \mathbf{C} \leq t).$$

Let us now study how the condition depends on t.

First of all, we remind that $\forall t \geq 0$ means that the condition must be **intersected** for all values of t greater than or equal to zero. For arbitrarily large values of t the condition $\mathbf{k} \cdot \mathbf{C} \leq t$ is always true. Hence, for arbitrarily large t the resulting space is the union of $\mathsf{domD}(t, \mathbf{k})$ with its complement $\mathsf{domD}^c(t, \mathbf{k})$, which is trivially the entire space. Thus, we can say that large values of t do not constrain vector $\mathbf{D}$ in any way. As t becomes smaller than $\mathbf{k} \cdot \mathbf{C}$, then the condition $\mathbf{k} \cdot \mathbf{C} \leq t$ becomes false and the region of the admissible deadlines becomes $\mathsf{domD}^c(t, \mathbf{k})$. Since we are performing the intersection as t varies, we can get rid of the big values of t by reformulating the necessary and sufficient EDF schedulability condition as follows:

$$\forall\, \mathbf{k} \in \mathbb{N}^n,\ \forall\, t \in [0, \mathbf{k} \cdot \mathbf{C}) \quad \mathbf{D} \in \mathsf{domD}^c(t, \mathbf{k})$$

as required. □

It is quite insightful to study Eq. (12). As explained earlier, region $\mathsf{domD}^c(t, \mathbf{k})$ is the entire space with a boxed hole described by Eq. (13) (remember that special care must be taken when some k_i is equal to 0). As t increases by Δ, the hole linearly translates by the same amount along all coordinates, in the space of deadlines. Figure 7 shows the

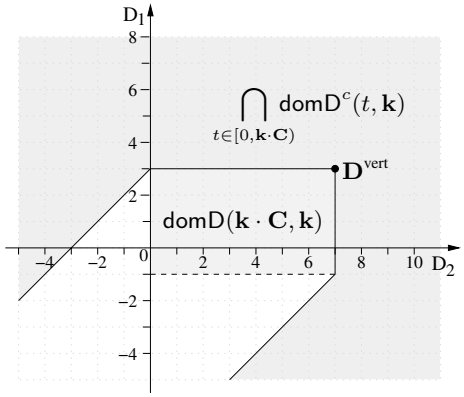

Figure 7. Intersection of $\mathsf{domD}^c(t, \mathbf{k})$.

intersection of the regions $\mathsf{domD}^c(t, \mathbf{k})$ for all t in the interval $[0, \mathbf{k} \cdot \mathbf{C})$, as indicated by Eq. (12). The figure is drawn assuming the same computation times and periods used in the previous example, that is, $\mathbf{C} = (2, 3)$ and $\mathbf{T} = (4, 7)$.

Figure 7 also highlights a special deadline vector, called *deepest vertex* $\mathbf{D}^{\text{vert}}(\mathbf{k}) = (D_1^{\text{vert}}, \ldots, D_n^{\text{vert}})$, associated with a given value of $\mathbf{k}$. The coordinates corresponding to a deepest vertex are the largest deadlines on the boundary of region $\mathsf{domD}(\mathbf{k} \cdot \mathbf{C}, \mathbf{k})$. In particular, each coordinate D_i^{vert} is the upper bound of the deadline interval from Eq. (13), when $t = \mathbf{k} \cdot \mathbf{C}$. As it can be noticed from Eq. (13), such an upper bound D_i^{vert} *exists only when the corresponding integer k_i is different than zero*. In fact, if $k_i = 0$, then

0-7695-2914-3/07 $20.00

the interval is right-unbounded. If we let I be the set of indexes of tasks whose number of jobs k_i is non-zero (formally, $k_i \neq 0 \Leftrightarrow i \in I$), then by replacing $t = \mathbf{k} \cdot \mathbf{C}$ in Eq. (13), we find that the coordinates of the deepest vertex $\mathbf{D}^{\text{vert}}(\mathbf{k})$ are

$$D_i^{\text{vert}}(\mathbf{k}) = \begin{cases} \mathbf{k} \cdot \mathbf{C} - (k_i - 1)T_i & i \in I \\ +\infty & i \notin I \end{cases} \quad (15)$$

Let us evaluate the vertex in some special cases. When $\mathbf{k} = (0, \ldots, 0, 1, 0, \ldots, 0)$, with a 1 at the i^{th} position, then $D_i^{\text{vert}} = C_i$ and the feasibility condition of Eq. (12) becomes $D_i \geq D_i^{\text{vert}} = C_i$, which means that the deadline must be not smaller than the computation time, as it is reasonable to expect.

More in general, as we intersect the regions $\bigcap_{t \in [0, \mathbf{k} \cdot \mathbf{C})} \mathsf{dom}\mathsf{D}^c(t, \mathbf{k})$ (also shown in Figure 7) for all possible integers $\mathbf{k}$, the resulting space of feasible deadlines becomes very similar to the one shown in Figure 5. Hence, the region of feasible deadlines can also be expressed by

$$\bigcap_{\mathbf{k} \in \mathbb{N}^n} \bigcup_{i: k_i \neq 0} D_i \geq \mathbf{k} \cdot \mathbf{C} - (k_i - 1)T_i. \quad (16)$$

From Eq. (16) it follows that the EDF schedulability region in the space of task deadlines can be derived by computing the deepest vertexes, for all vectors $\mathbf{k} \in \mathbb{N}^n$. However, as it will be shown in the next section, the number of vertexes to be computed can be drastically reduced, because certain regions (those associated with small k_i's) dominate all the others, due to the intersection operator.

The following example shows that, for the usual task set considered in this paper, only four deepest vertexes need to be computed for completely describing the region illustrated in Figure 5.

An example Let us take the usual task set considered throughout the paper, with two periodic tasks having $\mathbf{C} = (2, 3)$ and $\mathbf{T} = (4, 7)$. By applying Eq. (15), we can compute the deepest vertexes associated with a set of integer vectors purposely selected for the example. For each selected vector $\mathbf{k}$, Table 1 shows the two resulting coordinates of the corresponding deepest vertex $\mathbf{D}^{\text{vert}}(\mathbf{k})$ derived by Eq. (15). The deepest vertexes computed in Table 1 are also graphically illustrated in Figure 8, which shows the intersection of the 8 corresponding regions. Note that the first four vertexes *dominate* the others, meaning that the region described by them is not restricted by the other constraints. In fact, $\mathbf{D}^{\text{vert}}(1, 0)$ and $\mathbf{D}^{\text{vert}}(0, 1)$ are degenerate and corresponds to the thick horizontal and vertical lines, whereas $\mathbf{D}^{\text{vert}}(1, 1)$ and $\mathbf{D}^{\text{vert}}(2, 1)$ fall on the internal vertexes and are denoted by black dots. The other four vertexes (the three white dots and the dashed line) fall outside the region (or on its border) and do not alter it.

$\mathbf{k} = (k_1, k_2)$	$D_1^{\text{vert}}(\mathbf{k})$	$D_2^{\text{vert}}(\mathbf{k})$
$(1, 0)$	$C_1 = 2$	$+\infty$
$(0, 1)$	$+\infty$	$C_2 = 3$
$(1, 1)$	$C_1 + C_2 = 5$	$C_1 + C_2 = 5$
$(2, 1)$	$2C_1 + C_2 - T_1 = 3$	$2C_1 + C_2 = 7$
$(0, 2)$	$+\infty$	$2C_2 - T_2 = -1$
$(1, 2)$	$C_1 + 2C_2 = 8$	$C_1 + 2C_2 - T_2 = 1$
$(2, 2)$	$2C_1 + 2C_2 - T_1 = 6$	$2C_1 + 2C_2 - T_2 = 3$
$(3, 2)$	$3C_1 + 2C_2 - 2T_1 = 4$	$3C_1 + 2C_2 - T_2 = 5$

Table 1. Vertexes for the sample task set.

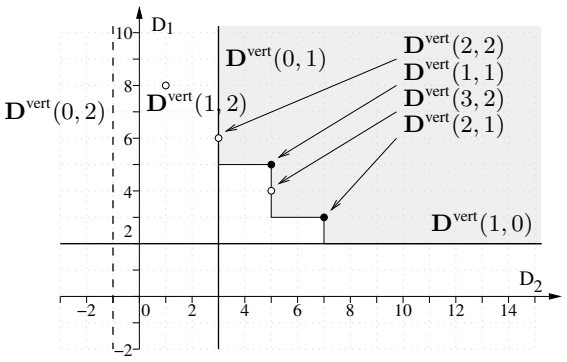

Figure 8. Feasible region resulting from the 8 deepest vertexes.

It is worth observing that continuing intersecting regions for other values of $\mathbf{k}$ may, eventually, only reduce the region depicted in Figure 8. As a consequence, we can observe that:

Observation 2 *The exact EDF schedulability region is* smaller than *or* equal to *the one shown in Figure 8.*

From Observation 1 and Observation 2 we conclude that the grey area illustrated in both Figure 5 and Figure 8 is the exact feasible region, for the considered task set.

This example suggests that the exact feasibility region can be simply computed from a finite set of integer vectors. The next section provides a method for determining such a set, by removing redundant points.

5. Reducing the set of $\mathbf{k}$'s

The derivation of the feasible deadline space based on Eq. (16) is still poorly effective, because it requires the intersection of infinite regions, derived from all possible $\mathbf{k}$'s ranging in $\mathbb{N}^n$.

However, as we observed in the previous section, a region associated to a vector $\hat{\mathbf{k}}$ can be neglected if its corresponding vertex $\mathbf{D}^{\text{vert}}(\hat{\mathbf{k}})$ is *dominated* by some other ver-

0-7695-2914-3/07 $20.00

tex. For instance, in Figure 8, $\mathbf{D}^{\text{vert}}(3,2) = (4,5)$ is dominated by $\mathbf{D}^{\text{vert}}(1,1) = (5,5)$. In general, an integer vector $\hat{\mathbf{k}}$ is *dominated* by another integer vector $\mathbf{k}$ if

$$\forall i = 1, \ldots, n \qquad D_i^{\text{vert}}(\hat{\mathbf{k}}) \leq D_i^{\text{vert}}(\mathbf{k}). \tag{17}$$

In fact, if all the coordinates of $\mathbf{D}^{\text{vert}}(\hat{\mathbf{k}})$ are smaller than or equal to the corresponding coordinates of $\mathbf{D}^{\text{vert}}(\mathbf{k})$, then intersecting the region associated with $\hat{\mathbf{k}}$ cannot eliminate any point that has not already been eliminated by the region associated with $\mathbf{k}$. To translate Eq. (17) as a constraint on the integers $\hat{\mathbf{k}} = (\hat{k}_1, \ldots, \hat{k}_n)$, let us define $I = \{i : k_i \neq 0\}$. From Eq. (17) we have that

$$\begin{cases} \forall i \in I & D_i^{\text{vert}}(\hat{\mathbf{k}}) \leq D_i^{\text{vert}}(\mathbf{k}) \\ \forall i \notin I & D_i^{\text{vert}}(\hat{\mathbf{k}}) \leq D_i^{\text{vert}}(\mathbf{k}) \end{cases}$$

$$\begin{cases} \forall i \in I & D_i^{\text{vert}}(\hat{\mathbf{k}}) \leq \mathbf{k} \cdot \mathbf{C} - (k_i - 1)T_i \\ \forall i \notin I & D_i^{\text{vert}}(\hat{\mathbf{k}}) \leq +\infty \end{cases}$$

$$\forall i \in I \qquad \hat{\mathbf{k}} \cdot \mathbf{C} - (\hat{k}_i - 1)T_i \leq \mathbf{k} \cdot \mathbf{C} - (k_i - 1)T_i$$

$$\forall i \in I \quad (\hat{k}_i - k_i)(T_i - C_i) - \sum_{j \neq i} (\hat{k}_j - k_j)C_j \geq 0 \tag{18}$$

For a given $\mathbf{k}$, Eq. (18) describes the region of the integer vectors $\hat{\mathbf{k}}$ that are dominated by $\mathbf{k}$. Such a region is a cone delimited by hyperplanes whose vertex is $\mathbf{k}$, the number of hyperplanes being equal to the number of non-zero coordinates of $\mathbf{k}$ (i.e., $|I|$). It is important to point out that when $|I|$ is small, the cone is less constrained, and hence a high number of dominated vectors $\hat{\mathbf{k}}$ can be neglected.

As a consequence, the cone eliminating the highest number of redundant integer vectors is the one corresponding to $\mathbf{k} = (0, 0, \ldots, 1, \ldots, 0)$, where the 1 is at the i^{th} position (in this case $I = \{i\}$). From Eq. (18) we find that the integers $\hat{\mathbf{k}}$ dominated by such a special value of $\mathbf{k}$ must satisfy the following constraint:

$$\hat{k}_i(T_i - C_i) - \sum_{j \neq i} \hat{k}_j C_j \geq T_i - C_i. \tag{19}$$

Hence, if $\mathsf{domK} \subseteq \mathbb{N}^n$ denotes the reduced set of integer vectors that have to be considered for deriving the deadline region, we can certainly assert that the n special vectors with only a single 1 must belong to domK, together with the remaining vectors not dominated by any of the n previous ones (i.e., those $\mathbf{k}$'s that do not satisfy Eq. (19) for all i). Formally, domK can be written as:

$$\mathsf{domK} = \bigcup_{i=1}^{n} \{\mathbf{k} : k_i = 1,\ k_j = 0,\ j \neq i\} \cup \bigcap_{i=1}^{n} \left\{ \mathbf{k} : k_i(T_i - C_i) - \sum_{j \neq i} k_j C_j < T_i - C_i \right\}. \tag{20}$$

It follows that the exact EDF feasibility region in the space

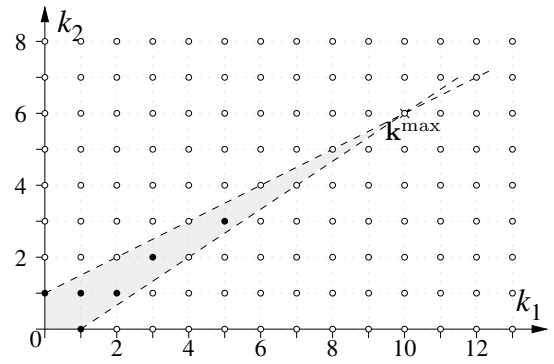

Figure 9. The set domK**.**

of deadlines can be restated as

$$\bigcap_{\mathbf{k} \in \mathsf{domK}} \bigcup_{i:k_i \neq 0} D_i \geq \mathbf{k} \cdot \mathbf{C} - (k_i - 1)T_i. \tag{21}$$

Now, it is worth showing how the previous result can be applied to compute the set domK for the sample task set used in this paper, consisting of two periodic tasks with $\mathbf{C} = (2, 3)$ and $\mathbf{T} = (4, 7)$.

From Equation (20), we start considering the points $(1, 0)$ and $(0, 1)$, and then all integer points satisfying the following constraint:

$$\begin{cases} 2\,k_1 - 3\,k_2 < 2 \\ -2\,k_1 + 4\,k_2 < 4 \end{cases} \tag{22}$$

Figure 9 depicts by a black dot the admissible values for $\mathbf{k}$ that need to be visited to determine the space of feasible deadlines. Note that the values determined in this way are sufficient but not necessary to derive the feasibility region, meaning that there may be still points that could be discarded. The figure also shows by white dots all the integer vectors that do not belong to domK, because they are dominated by some $\mathbf{k} \in \mathsf{domK}$.

To see whether the set domK is finite, it is useful to compute the intersection of the n hyperplanes described by Eq. (19), corresponding to those $\mathbf{k}$'s with a single 1 . Such an intersection is denoted as $\mathbf{k}^{\max}$ in Figure 9. The coordinates of $\mathbf{k}^{\max}$ are the solution of the following linear system:

$$\begin{vmatrix} T_1 - C_1 & -C_2 & \cdots & -C_n \\ -C_1 & T_2 - C_2 & \cdots & -C_n \\ \vdots & \vdots & \ddots & \vdots \\ -C_1 & -C_2 & \ldots & T_n - C_n \end{vmatrix} \cdot \mathbf{k}^{\max} = \begin{vmatrix} T_1 - C_1 \\ T_2 - C_2 \\ \vdots \\ T_n - C_n \end{vmatrix}$$

and, by the Cramer's rule, we find

$$k_i^{\max} = \frac{(T_i - C_i)(1 - \sum_{j \neq i} U_j) + \sum_{j \neq i} C_j(1 - U_j)}{T_i(1 - U)}. \tag{23}$$

0-7695-2914-3/07 $20.00

It is worth noticing that the coordinates of $\mathbf{k}^{\max}$ are inversely proportional to $1 - U$, meaning that the set domK grows as U approaches 1. In the extreme case of $U = 1$, $\mathbf{k}^{\max}$ diverges to infinity and the set domK, as defined in Eq. (20), is unbounded. In the next section we discuss further implications of the achieved results.

6. Potential Implications

This section discusses some potential implications of the results presented so far and points out some novel research directions for future investigation on the topic.

Intrinsic complexity of EDF tests The existence of the $\mathbf{k}^{\max}$ bound given in Eq. (23) allows enumerating domK using n nested loops, for each k_i ranging from 0 to $k_i^{\max}$, where we apply Equation (19) to check whether the candidate point belongs to domK or not.

As it can be argued, the complexity of such an algorithm is exponential in the number of tasks. Moreover, the complexity is tightly related to the value of $\mathbf{k}^{\max}$ which is inversely proportional to $1 - U$. This means that, for small values of U, $\mathbf{k}^{\max}$ is small and the enumeration of domK is more tractable. On the other hand, as the total utilisation U approaches 1, the size of domK increases and, consequently, the complexity of Eq. (21) grows exponentially with n. Note that the dependency of the complexity on the total processor utilisation is also typical of the processor demand test proposed by Baruah et al. [4], confirming that, as the total utilisation U approaches 1, the EDF schedulability guarantee tests become intrinsically more complex.

Reducing complexity The number of points in domK can be reduced further by eliminating additional redundant points in the region described by Eq. (20). In fact, from Figure 9 we can see that the candidate integers inside the region are more than the 4 strictly necessary reported in Table 1, namely $(1, 0)$, $(0, 1)$, $(1, 1)$ and $(2, 1)$.

From Eq. (18) we can see that the integer vectors dominated by $\mathbf{k}$ lay into a cone whose vertex is exactly $\mathbf{k}$. Hence, after examining a vector $\mathbf{k}$, all the integers laying into the cone of Eq. (18) can be removed from domK.

In our usual example, this means that, after we explored the points $(1, 0)$, $(0, 1)$, $(1, 1)$ and $(2, 1)$, we can neglect the other two points $(3, 2)$ and $(5, 3)$ (also represented by white dots in Figure 10), since they are both dominated by $(1, 1)$. Such an observation allows us to significantly reduce the number of points in domK and improve the efficiency of the algorithm.

U approaching 1 Although real-time systems should be designed to have a total utilisation well below 1, overload conditions can occur for several reasons, so it is interesting to analyse what happens when the total processor utilisation

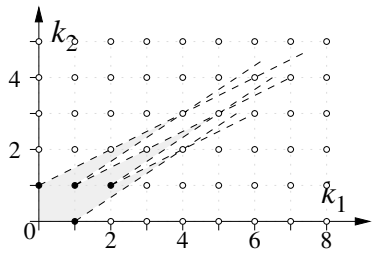

Figure 10. Reducing domK.

U approaches the value of 1.

We already observed that as U approaches 1 the point $\mathbf{k}^{\max}$ diverges to $+\infty$ and the set domK becomes larger and larger. When U is exactly 1, domK becomes unbounded. Hence, it seems that Eq. (21) becomes again impractical, since it requires to check an infinite number of vectors $\mathbf{k}$. Let us depict such a situation by an example. To achieve $U = 1$ we choose $\mathbf{C} = (2, 3.5)$ and $\mathbf{T} = (4, 7)$. Thus, the constraints of Eq. (20) become

$$\begin{cases} 2\,k_1 - 3.5\,k_2 < 2 \\ -2\,k_1 + 3.5\,k_2 < 3.5 \end{cases} \tag{24}$$

and the resulting region domK is drawn in Figure 11.

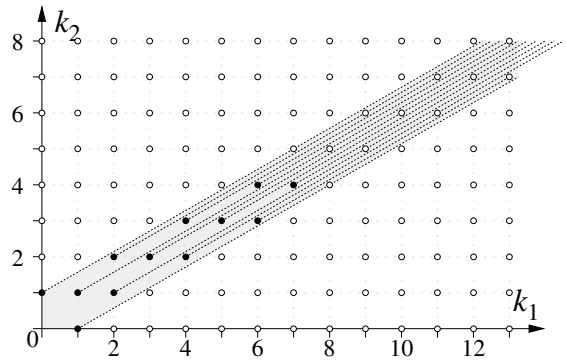

Figure 11. domK when $U = 1$.

When $U = 1$, the cone of the region dominated by $\mathbf{k}$ becomes degenerate, i.e., a half line (see dashed lines in Figure 11). Hence, the integers that can be removed from domK, because dominated by some $\mathbf{k}$, are those falling exactly on the half-line originated from $\mathbf{k}$. Thus, it is important to understand whether an integer vector falls on that half line or not.

By examining Eq. (18), and also from the coefficients of the constraint in Eq. (24), we see that the "slope" of the half-line is tightly related to the task parameters. When task parameters have a large greatest common divisor, many more integers fall on the half-line originating from a dominant $\mathbf{k}$, and fewer points need to be considered in domK.

0-7695-2914-3/07 $20.00

Such a remark is in accordance to the well-known property of the EDF guarantee test [4], which becomes more complex as the least common multiple of the periods increases.

7. Approximate solution

Considering the substantial complexity of the method for determining the exact EDF feasibility region in the space of deadlines, in this section we propose an alternative approach for deriving an approximate region with reduced complexity.

The complexity for finding the exact region mainly comes from the presence of the floor operator, which caused us to introduce the integer variables k_i to account for it.

Now we attempt to simplify the analysis by relaxing the floor operator. Clearly, the resulting analysis loses necessity, but it gains simplicity with respect to the exact approach.

We start from the classical processor demand tests, which says that a set $\mathcal{T} = \{\tau_1, \ldots, \tau_n\}$ of n periodic tasks is schedulable by EDF **if and only if**

$$\forall t \in \mathsf{dlSet} \quad \sum_{i=1}^{n} \max\left\{0, \left\lfloor \frac{t + T_i - D_i}{T_i} \right\rfloor\right\} C_i \leq t. \quad (25)$$

A first simplification can be done by removing the "max" operator. As shown by Chantem et al. [12], the max can be removed when the second term is greater than or equal to zero. That is when

$$\begin{aligned} &\forall t \in \mathsf{dlSet} \quad \forall i \qquad \left\lfloor \frac{t + T_i - D_i}{T_i} \right\rfloor \geq 0 \\ &\forall t \in \mathsf{dlSet} \quad \forall i \qquad \frac{t + T_i - D_i}{T_i} \geq 0 \\ &\forall t \in \mathsf{dlSet} \quad \forall i \qquad t + T_i - D_i \geq 0 \\ &\forall t \in \mathsf{dlSet} \quad \forall i \qquad D_i \leq t + T_i \\ &\qquad\qquad\quad \forall i \qquad D_i \leq T_i + D_{\min} \end{aligned} \quad (26)$$

where $D_{\min}$ denotes the minimum deadline. Under this assumption, the necessary and sufficient condition becomes:

$$\forall t \in \mathsf{dlSet} \quad \sum_{i=1}^{n} \left\lfloor \frac{t + T_i - D_i}{T_i} \right\rfloor C_i \leq t. \quad (27)$$

By removing the floor we easily find the following sufficient condition:

$$\forall t \in \mathsf{dlSet} \quad \sum_{i=1}^{n} \frac{t + T_i - D_i}{T_i} C_i \leq t \quad (28)$$

from which it directly follows that the feasible deadlines must satisfy

$$\forall t \in \mathsf{dlSet} \quad \sum_{i=1}^{n} U_i D_i \geq \sum_{i=1}^{n} C_i - t\,(1 - \sum_{i=1}^{n} U_i). \quad (29)$$

Now we observe that Eq. (29) must be intersected for all values of t in dlSet. The condition becomes more and more stringent as t decreases. Hence, the most stringent condition is achieved when $t = D_{\min}$, which is the minimum deadline. As a consequence, the sufficient schedulability condition results

$$\sum_{i=1}^{n} U_i D_i \geq \sum_{i=1}^{n} C_i - D_{\min}\,(1 - \sum_{i=1}^{n} U_i). \quad (30)$$

In Figure 12 we compare the approximated region (darker) with the exact region in the task set example $\mathbf{C} = (2, 4)$ and $\mathbf{T} = (4, 7)$.

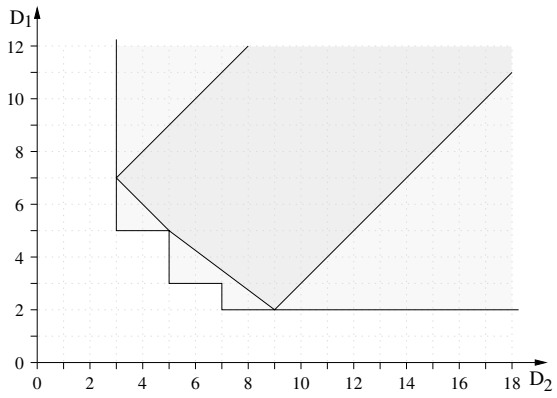

Figure 12. The approximated region.

Convexity We now like to investigate whether the resulting region is convex or not. First of all we recall that Eq. (30) is valid only in the hypothesis of Eq. (26), which can be restated as follows:

$$\begin{aligned} &\forall i \qquad D_i \leq T_i + D_{\min} \\ &\forall i, j \qquad D_i \leq T_i + D_{\min} \leq T_i + D_j \\ &\forall i, j \qquad D_i - D_j \leq T_i \end{aligned} \quad (31)$$

which is convex because intersection of half spaces. Similarly, the condition in Eq. (30) is equivalent to

$$\forall j \quad D_j\,(1 - \sum_{i=1}^{n} U_i) + \sum_{i=1}^{n} U_i D_i \geq \sum_{i=1}^{n} C_i \quad (32)$$

which is convex for the same reason. Since the overall region is determined by intersecting Eq. (31) and (32) it is also convex, and delimited by n^2 linear constraints.

8. Conclusions

In this paper we addressed the problem of finding the region of feasible deadlines for a periodic task set scheduled by EDF. In particular, we presented a general analysis technique to describe the exact region and we provided a method to reduce the number of points aimed at decreasing the complexity of the computation.

We finally presented an $O(n^2)$ method for identifying an approximate convex region, which can be efficiently used to apply performance optimisation.

We believe that the present approach is promising for enhancing the performance of delay/jitter sensitive applications and applying sensitivity analysis in the deadline domain.

References

[1] P. Balbastre, I. Ripoll, and A. Crespo. Optimal deadline assignment for periodic real-time tasks in dynamic priority systems. In *Proceedings of the* 18^{th} *Euromicro Conference on Real-Time Systems*, pages 65–74, Dresden, Germany, July 2006.

[2] S. K. Baruah and A. Burns. Sustainable schedulability analysis. In *Proceedings of the* 27^{rd} *IEEE Real-Time Systems Symposium*, pages 159–168, Rio de Janerio, Brazil, Dec. 2006.

[3] S. K. Baruah, G. Buttazzo, S. Gorinsky, and G. Lipari. Scheduling periodic task systems to minimize output jitter. In *Proceedings of the* 6^{th} *International Conference on Real-Time Computing Systems and Applications*, pages 62–69, Hong Kong, Dec. 1999.

[4] S. K. Baruah, R. Howell, and L. Rosier. Algorithms and complexity concerning the preemptive scheduling of periodic, real-time tasks on one processor. *Real-Time Systems*, 2:301–324, 1990.

[5] S. K. Baruah, A. K. Mok, and L. E. Rosier. Preemptively scheduling hard-real-time sporadic tasks on one processor. In *Proceedings of the* 11^{th} *IEEE Real-Time Systems Symposium*, pages 182–190, Lake Buena Vista (FL), U.S.A., Dec. 1990.

[6] E. Bini and M. Di Natale. Optimal task rate selection in fixed priority systems. In *Proceedings of the* 26^{th} *IEEE Real-Time Systems Symposium*, pages 399–409, Miami (FL), U.S.A., Dec. 2005.

[7] E. Bini, M. Di Natale, and G. C. Buttazzo. Sensitivity analysis for fixed-priority real-time systems. In *Proceedings of the* 18^{th} *Euromicro Conference on Real-Time Systems*, Dresden, Germany, July 2006.

[8] G. Buttazzo and A. Cervin. Analysis and evaluation of jitter control methods. Technical Report RETIS-TR06-01, Scuola Superiore Sant'Anna,, Pisa, Italy, Aug. 2006.

[9] G. Buttazzo and F. Sensini. Optimal deadline assignment for scheduling soft aperiodic task in hard real-time environments. *IEEE Transactions on Computers*, 48(10):1035–1052, Oct. 1999.

[10] A. Cervin. Improved scheduling of control tasks. In *Proceedings of the* 11^{th} *Euromicro Conference on Real-Time Systems*, pages 4–10, York, UK, June 1999.

[11] A. Cervin, B. Lincoln, J. Eker, K.-E. Årzén, and G. Buttazzo. The jitter margin and its application in the design of real-time control systems. In *Proceedings of the* 10^{th} *International Conference on Real-Time and Embedded Computing Systems and Applications*, Göteborg, Sweden, Aug. 2004.

[12] T. Chantem, X. S. Hu, and M. D. Lemmon. Generalized elastic scheduling. In *Proceedings of the* 27^{th} *IEEE Real-Time Systems Symposium*, pages 236–245, Rio de Janeiro, Brazil, Dec. 2006.

[13] A. Crespo, I. Ripoll, and P. Albertos. Reducing delays in rt control: the control action interval. In *Proceedings of the* 14^{th} *IFAC World Congress*, pages 257–262, Beijing, China, July 1999.

[14] C. Davidson. *Random sampling and random delays in optimal control*. PhD thesis, Department of Optimization and Systems Theory, Royal Institute of Technology, Sweden, 1973.

[15] M. Di Natale and J. A. Stankovic. Scheduling distributed real-time tasks with minimum jitter. *IEEE Transactions on Computers*, 49(4):303–316, Apr. 2000.

[16] H. Hoang, G. Buttazzo, M. Jonsson, and S. Karlsson. Computing the minimum edf feasible deadline in periodic systems. In *Proceedings of the 12th IEEE International Conference on Embedded and Real-Time Computing Systems and Applications*, Sydney, Australia, Aug. 2006.

[17] R. Kalman and J. Bertram. A unified approach to the theory of sampling systems. *J. Franklin Inst.*, 267:405–436, May 1959.

[18] H. J. Kushner and L. Tobias. On the stability of randomly sampled systems. *IEEE Transactions on Automatic Control*, 14(4):319–324, Aug. 1969.

[19] C. L. Liu and J. W. Layland. Scheduling algorithms for multiprogramming in a hard real-time environment. *Journal of the Association for Computing Machinery*, 20(1):46–61, Jan. 1973.

[20] M. Lluesma, A. Cervin, P. Balbastre, I. Ripoll, and A. Crespo. Jitter evaluation of real-time control systems. In *Proceedings of the* 12^{th} *IEEE International Conference on Embedded and Real-Time Computing Systems and Applications*, pages 257–260, Sydney, Australia, August 2006.

[21] P. Martí, J. M. Fuertes, K. Ramamritham, and G. Fohler. Jitter compensation for real-time control systems. In *Proceedings of the* 22^{nd} *IEEE Real-Time System Symposium*, pages 39–48, London, UK, Dec. 2001.

[22] J. Nilsson, B. Bernhardsson, and B. Wittenmark. Stochastic analysis and control of real-time systems with random time delays. *Automatica*, 34(1):57–64, Jan. 1998.

[23] R. Racu, A. Hamann, and R. Ernst. A formal approach to multi-dimensional sensitivity analysis of embedded real-time systems. In *Proceedings of the* 18^{th} *Euromicro Conference on Real-Time Systems*, pages 3–12, Dresden, Germany, July 2006.

[24] D. Seto, J. P. Lehoczky, and L. Sha. Task period selection and schedulability in real-time systems. In *Proceedings of the* 19^{th} *IEEE Real-Time Systems Symposium*, pages 188–198, Madrid, Spain, Dec. 1998.

[25] D. Seto, J. P. Lehoczky, L. Sha, and K. G. Shin. On task schedulability in real-time control systems. In *Proceedings of the* 17^{th} *IEEE Real-Time Systems Symposium*, pages 13–21, Washington (DC), U.S.A., Dec. 1996.

[26] Q. Zheng and K. G. Shin. On the ability of establishing real-time chennals in point-to-point packet-switched networks. *IEEE Transactions on Communications*, 42(2/3/4):1096–1105, Feb/Mar/Apr 1994.