Real-Time Syst (2009) 41: 27–51
DOI 10.1007/s11241-008-9060-7

The space of EDF deadlines: the exact region and a convex approximation

Enrico Bini · Giorgio Buttazzo

Published online: 1 October 2008

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 jeopardizing the schedulability of the task set.

In this paper, we formally characterize the region of admissible deadlines so that the system designer can appropriately select the desired values to maximize 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.

Keywords Earliest deadline first · Deadline assignment · Performance optimization

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 to use standard control theory to guarantee system stability and meet performance requirements.

When several of such activities execute concurrently in the same processor, however, each control 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

E. Bini (✉) · G. Buttazzo
Scuola Superiore Sant'Anna, Pisa, Italy
e-mail: e.bini@sssup.it

G. Buttazzo
e-mail: giorgio@sssup.it

by each task depends on several factors, including the dispatching policy in force, 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 jeopardize its stability (Kalman and Bertram 1959; Kushner and Tobias 1969; Davidson 1973).

The problem of jitter in real-time control applications has received increasing attention during the last decade and several techniques have been proposed to cope with it. Nilsson et al. (1998) analyzed the stability and performance of real-time control systems with random delays and derived an optimal, jitter-compensating controller. Martí et al. (2001) proposed a compensation technique for controllers based on the pole-placement design method. Di Natale and Stankovic (2000) proposed the use of simulated annealing to find the optimal configuration of task offsets that minimizes jitter, according to some user-defined cost function. Cervin et al. (2004) 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 sub-tasks 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, thereby trading jitter with delay (Crespo et al. 1999). Cervin (1999) 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, jitter reduction is obtained by inserting extra delays in task execution, since the input and output parts are always separated by exactly one period, while the average delay could be smaller. The effect of a longer delay in the control loop has to be carefully analyzed, since it could be worse 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 (Lluesma et al. 2006).

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. (1999) 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 Buttazzo and Cervin (2006).

Several authors (Zheng and Shin 1994; Buttazzo and Sensini 1999; Balbastre et al. 2006; Hoang et al. 2006) independently proposed different algorithms for computing the minimum deadline of a newly arrived task, assuming the existing task set is feasibly schedulable by EDF (Liu and Layland 1973). The problem with 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 a time, following a given order, as suggested by Hoang et al. (2006). 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 minimize its deadline, leaving little margin for the remaining tasks. To apply a more uniform deadline reduction in the task set, Balbastre et al. (2006) proposed an algorithm able to scale all deadlines by the same factor. The problem with

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 EDF. 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 maximizes a given performance index defined over the task set.

Different methods for optimizing the performance of periodic task sets have been proposed in the literature, both under fixed priority (Bini and Di Natale 2005) and EDF (Seto et al. 1996), but only with respect to periods. Sensitivity analysis in the domain of computation times has also been addressed in Bini et al. (2006), while sensitivity analysis of any parameters was proposed at the cost of high complexity by Racu et al. (2006) 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 organized 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. Sections 4 and 5 describe the general method for n tasks, and presents the algorithm for deriving the region. Section 6 presents an approximate solution to reduce the complexity of the approach. Finally, Sect. 8 presents our conclusions and our plans for future work.

2 Terminology and assumptions

We consider a set $\mathcal{T} = \{\tau_1, \tau_2, \ldots, \tau_n\}$ of n periodic tasks that have to be executed on a uni-processor system under the Earliest Deadline First (EDF) scheduling algorithm. Each periodic 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;

D_i denotes the relative deadline of task τ_i, that is, the latest completion 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 latest absolute time by which job τ_{ik} is allowed to complete;

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

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. Reasoning in such a region allows the designer to perform sensitivity analysis or to select the set of relative deadlines that maximizes a given performance index defined over the task set. Later in Sect. 6, we will also propose an approximate method, based on a simpler sufficient feasibility condition, to identify a convex region entirely contained in the D-space.

Before entering into the mathematical details of the proposed approach, we introduce a simple example to visualize such a feasibility region of task deadlines. This example will be used throughout the paper as a sample application of the theoretical results.

Example 1 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 equal to periods, the task set is feasible by EDF, since the processor utilization 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 colors).

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 Fig. 2.

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

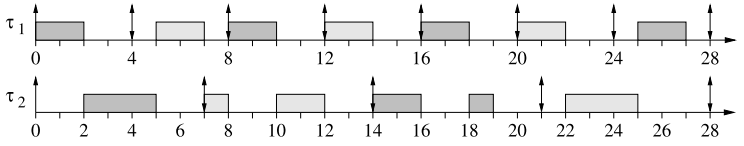


Fig. 1 EDF schedule when $D_i = T_i$

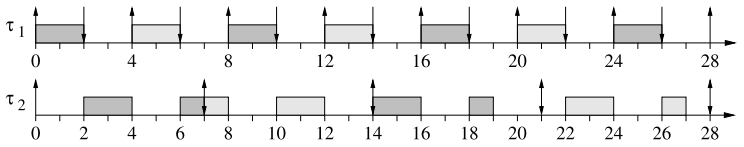


Fig. 2 EDF schedule when minimizing D_1

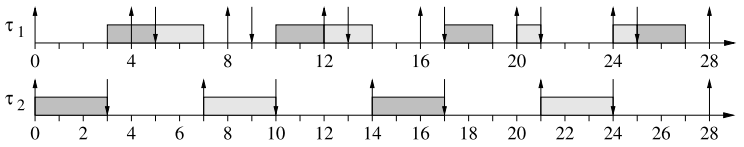


Fig. 3 EDF schedule when minimizing D_2

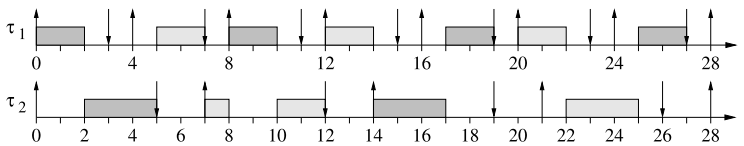


Fig. 4 Schedule with $D_1 = 3$ and $D_2 = 5$

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 Fig. 3.

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

Moreover, we know that EDF on a single processor is *sustainable* with respect to task deadlines, 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 (Baruah and Burns 2006).

From this observation and the cases reported above, we can state that all deadlines corresponding to points in the gray area depicted in Fig. 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:

Observation 1 When $\mathbf{C} = (2, 3)$ and $\mathbf{T} = (4, 7)$, the exact region of EDF feasible deadlines is *larger than or equal to* the gray area shown in Fig. 5.

In the remainder of the paper we will show that the region in Fig. 5 is necessary and sufficient, and then we will also present a method for deriving such a region.

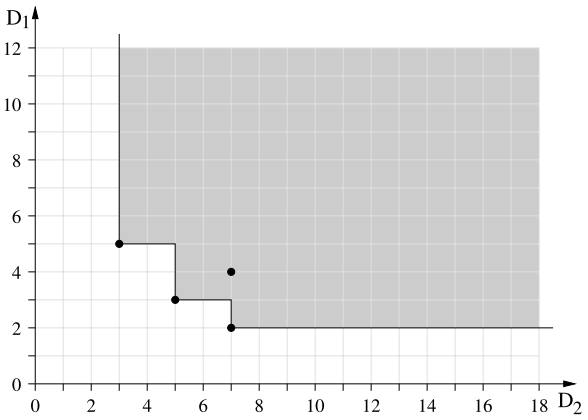


Fig. 5 Space of feasible deadlines

4 The space of EDF feasible deadlines

Unfortunately, the feasibility region cannot be described by a closed formula. In fact, as shown by Baruah et al. (1990), a set of periodic tasks simultaneously activated at time $t = 0$ is schedulable by EDF if and only if $U \leq 1$ and

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

The difficulty of finding a closed formulation in terms of the deadlines is due to the presence of the floor operator in (1). To overcome this problem we follow the intuition used by Seto et al. (1998) to find all the admissible periods in a fixed priority scheduler.

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

$$\mathsf{K}_i(t, D_i) = \max\left\{0, \left\lfloor \frac{t + T_i - D_i}{T_i} \right\rfloor\right\} \tag{2}$$

Each function $\mathsf{K}_i(t, D_i)$ denotes the number of jobs of task τ_i whose release time and absolute deadline are both within the interval $[0, t]$. We also observe that, by the definition of (2), $\mathsf{K}_i(t, D_i)$ is the unique integer satisfying the following constraint:

$$\begin{cases} \mathsf{K}_i(t, D_i) = 0 & \text{if } t - D_i < 0 \\ \frac{t-D_i}{T_i} < \mathsf{K}_i(t, D_i) \leq \frac{t-D_i}{T_i} + 1 & \text{otherwise} \end{cases} \tag{3}$$

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

$$\mathsf{K}(t, \mathbf{D}) = (\mathsf{K}_1(t, D_1), \ldots, \mathsf{K}_n(t, D_n)) \tag{4}$$

The introduction of function K is very convenient to write the schedulability conditions with a more compact notation. For example the necessary and sufficient schedu-

lability condition by Baruah et al. expressed in (1) becomes

$$\forall t \geq 0 \quad \mathsf{K}(t, \mathbf{D}) \cdot \mathbf{C} \leq t \tag{5}$$

We are now ready to move forward. The condition resulting from (1), and its equivalent (5), 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 which deadlines constraints can be derived. Although seemingly more complex than condition (5), 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*:

$$\forall t \geq 0, \ \forall \mathbf{k} \in \mathbb{N}^n \quad (\mathbf{k} = \mathsf{K}(t, \mathbf{D}) \wedge \mathbf{k} \cdot \mathbf{C} \leq t) \vee \mathbf{k} \neq \mathsf{K}(t, \mathbf{D}) \tag{6}$$

Proof We prove the lemma by showing that (6) is equivalent to (1).

Equation (6) $\Rightarrow$ (1). The proof is performed by contradiction. Let us assume that (1) is false, meaning that

$$\exists t^* > 0 \quad \sum_{i=1}^{n} \max \left\{ 0, \left\lfloor \frac{t^* + T_i - D_i}{T_i} \right\rfloor \right\} C_i > t^*. \tag{7}$$

Let us denote, for all i, $k_i^* = \max\{0, \lfloor \frac{t^*+T_i-D_i}{T_i} \rfloor\}$ which is also equal to $\mathsf{K}_i(t^*, D_i)$ by the definition of K given in (2), and $\mathbf{k}^* = (k_1^*, \ldots, k_n^*)$. We show that for such special t^* and $\mathbf{k}^*$, (6) is false as well. In fact, (7) 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 (6) is false, as required.

Equation (1) $\Rightarrow$ (6). Again we proceed by contradiction showing that when (6) is false then (1) is false as well. From the negation of (6), we have that

$$\exists t^* \geq 0, \ \exists \mathbf{k}^* \in \mathbb{N}^n \quad (\mathbf{k}^* \neq \mathsf{K}(t^*, \mathbf{D}) \vee \mathbf{k}^* \cdot \mathbf{C} > t^*) \ \wedge \ \mathbf{k}^* = \mathsf{K}(t^*, \mathbf{D}) \tag{8}$$

which is equivalent to

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

$$\exists t^* \geq 0, \quad \mathsf{K}(t^*, \mathbf{D}) \cdot \mathbf{C} > t^*$$

$$\exists t^* \geq 0, \quad \sum_{i=1}^{n} \mathsf{K}_i(t^*, \mathbf{D})\, C_i > t^*$$

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

which contradicts (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 (3) it follows that

$$\begin{aligned} &\mathbf{k} = \mathsf{K}(t, \mathbf{D}) \\ &\begin{cases} k_i = 0 & \text{if } t - D_i < 0 \\ \frac{t - D_i}{T_i} < k_i \leq \frac{t - D_i}{T_i} + 1 & \text{otherwise} \end{cases} \\ &\begin{cases} D_i > t & \text{if } k_i = 0 \\ t - k_i\, T_i < D_i \leq t - (k_i - 1)T_i & \text{otherwise} \end{cases} \end{aligned} \tag{9}$$

which gives us the desired constraint on the deadline values, when exactly k_i jobs of τ_i are in the interval $[0, t]$.

The set of deadlines satisfying the constraint of (9), for a specific selection of t and $\mathbf{k}$, is denoted by $\mathsf{dom}\,\mathsf{D}(t, \mathbf{k})$, and its complement is denoted by $\mathsf{dom}\,\mathsf{D}^c(t, \mathbf{D})$. Due to the equivalence between (3) and (9), we have

$$\begin{aligned} \mathbf{k} = \mathsf{K}(t, \mathbf{D}) \quad &\Leftrightarrow \quad \mathbf{D} \in \mathsf{dom}\,\mathsf{D}(t, \mathbf{k}) \\ \mathbf{k} \neq \mathsf{K}(t, \mathbf{D}) \quad &\Leftrightarrow \quad \mathbf{D} \in \mathsf{dom}\,\mathsf{D}^c(t, \mathbf{k}) \end{aligned} \tag{10}$$

As it can be noticed from (9), the region $\mathsf{dom}\,\mathsf{D}(t, \mathbf{k})$ 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 equal to zero, then $\mathsf{dom}\,\mathsf{D}(t, \mathbf{k})$ is a degenerate box because its projection on the ith axis is a right-unbounded interval. Figure 6 shows some examples of $\mathsf{dom}\,\mathsf{D}$ for the same task 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 $\mathsf{dom}\,\mathsf{D}$. 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

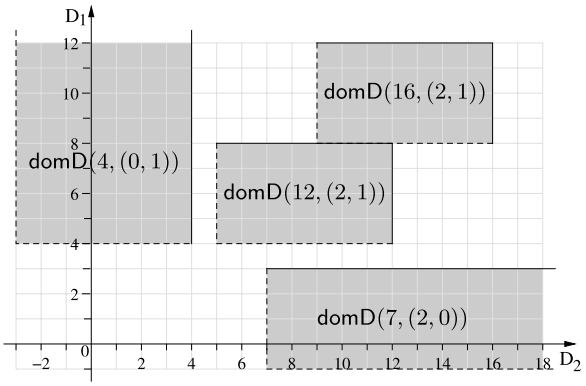


Fig. 6 Samples of $\mathsf{dom}\,\mathsf{D}$, when $\mathbf{T} = (4, 7)$

coordinates. Finally, the complement $\mathsf{dom}\,\mathsf{D}^c(t, \mathbf{D})$ basically is "all the space with a boxed hole" defined by $\mathsf{dom}\,\mathsf{D}(t, \mathbf{D})$.

Using the definition of region $\mathsf{dom}\,\mathsf{D}$, 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*

$$\forall \mathbf{k} \in \mathbb{N}^n, \quad \forall t \in [0, \mathbf{k} \cdot \mathbf{C}) \quad \mathbf{D} \in \mathsf{dom}\,\mathsf{D}^c(t, \mathbf{k}) \tag{11}$$

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

$$\begin{cases} D_i > t & \text{if } k_i = 0 \\ t - k_i T_i < D_i \leq t - (k_i - 1)T_i & \text{otherwise} \end{cases} \tag{12}$$

and $\mathsf{dom}\,\mathsf{D}^c$ *denotes its complement.*

Proof From Lemma 1 and the equivalence expressed by (10) it follows that the task set $\mathcal{T}$ is feasible by EDF if and only if

$$\forall t \geq 0, \ \forall \mathbf{k} \in \mathbb{N}^n \quad \mathbf{D} \in \mathsf{dom}\,\mathsf{D}^c(t, \mathbf{k}) \ \vee \ (\mathbf{D} \in \mathsf{dom}\,\mathsf{D}(t, \mathbf{k}) \wedge \mathbf{k} \cdot \mathbf{C} \leq t) \tag{13}$$

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

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

Let us now study how the condition depends on t.

First of all, we recall 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 $\mathsf{dom}\,\mathsf{D}(t, \mathbf{k})$ with its complement $\mathsf{dom}\,\mathsf{D}^c(t, \mathbf{k})$, which is trivially the entire space. Thus, we can say that large values of t do not constraint 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 $\mathsf{dom}\,\mathsf{D}^c(t, \mathbf{k})$. 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:

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

as required. □

It is quite insightful to study (11). As explained earlier, region $\mathsf{dom}\,\mathsf{D}^c(t, \mathbf{k})$ is the entire space with a boxed hole described by (12) (remember that the box becomes upper unbounded along the i coordinate when $k_i = 0$). 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 intersection of the regions $\mathsf{dom}\,\mathsf{D}^c(t, \mathbf{k})$ for all t in the interval $[0, \mathbf{k} \cdot \mathbf{C})$, as indicated by (11). The figure is drawn assuming the same computation times and periods as in Example 1, that is, $\mathbf{C} = (2, 3)$ and $\mathbf{T} = (4, 7)$.

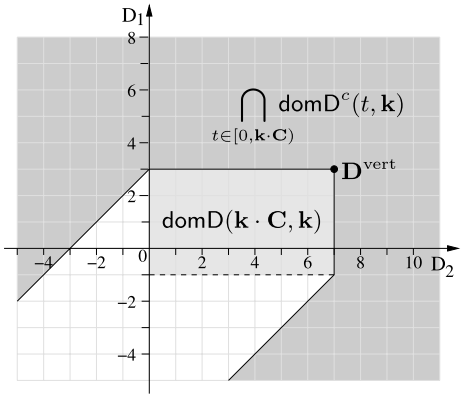


Fig. 7 Intersection of $\mathsf{dom}\,\mathsf{D}^c(t,\mathbf{k})$

Figure 7 also highlights a special deadline vector, called *deepest vertex* $\mathbf{D}^{\mathrm{vert}}(\mathbf{k}) = (D_1^{\mathrm{vert}},\ldots,D_n^{\mathrm{vert}})$, associated with a given value of $\mathbf{k}$. The coordinates corresponding to a deepest vertex are the largest deadlines on the boundary of region $\mathsf{dom}\,\mathsf{D}(\mathbf{k}\cdot\mathbf{C},\mathbf{k})$. In particular, each coordinate D_i^{vert} is the upper bound of the deadline interval from (12), when $t = \mathbf{k}\cdot\mathbf{C}$. As it can be noticed from (12), such an upper bound D_i^{vert} *is finite only when the corresponding integer* k_i *is strictly greater than zero*. In fact, if $k_i = 0$, then the interval is right-unbounded and we set $D_i^{\mathrm{vert}} = +\infty$. By replacing $t = \mathbf{k}\cdot\mathbf{C}$ in (12), we find that the coordinates of the deepest vertex $\mathbf{D}^{\mathrm{vert}}(\mathbf{k})$ are

$$D_i^{\mathrm{vert}}(\mathbf{k}) = \begin{cases} \mathbf{k}\cdot\mathbf{C} - (k_i - 1)T_i & \text{if } k_i \neq 0 \\ +\infty & \text{if } k_i = 0 \end{cases} \tag{14}$$

Let us evaluate the vertex in some special cases. When $\mathbf{k} = (0,\ldots,0,1,0,\ldots,0)$, with a 1 at the ith position, then $D_i^{\mathrm{vert}} = C_i$ and the feasibility condition of (11) becomes $D_i \geq D_i^{\mathrm{vert}} = C_i$, 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 Fig. 7) for all possible integers $\mathbf{k}$, the resulting space of feasible deadlines becomes very similar to the one shown in Fig. 5. Hence, the region of feasible deadlines can also be expressed by

$$\bigcap_{\mathbf{k}\in\mathbb{N}^n} \bigcup_{i:k_i\neq 0} D_i \geq \mathbf{k}\cdot\mathbf{C} - (k_i - 1)T_i. \tag{15}$$

From (15) it follows that the EDF schedulability region in the space of task deadlines can be derived by computing the deepest vertices, for all vectors $\mathbf{k} \in \mathbb{N}^n \setminus \{\mathbf{0}\}$, since when all $k_i = 0$ we have no constraint. However, as it will be shown in the next section, the number of vertices 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.

Below we show that only four deepest vertices need to be computed for completely describing the region illustrated in Fig. 5.

Table 1 Vertices for the sample task set

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

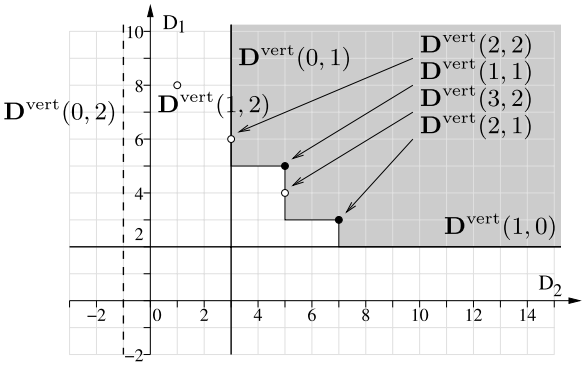


Fig. 8 Feasible region resulting from the 8 deepest vertices

Example 1 Let us take the initial task set considered throughout the paper, with two periodic tasks having $\mathbf{C} = (2, 3)$ and $\mathbf{T} = (4, 7)$. By applying (14), we can compute the deepest vertices 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 (14). The deepest vertices computed in Table 1 are also graphically illustrated in Fig. 8, which shows the intersection of the 8 corresponding regions. Note that the first four vertices *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 correspond 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 vertices and are denoted by black dots. The other four vertices (the three white dots and the vertical dashed line) fall outside the region (or on its border) and do not alter it.

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

Observation 2 When $\mathbf{C} = (2, 3)$ and $\mathbf{T} = (4, 7)$, the exact region of EDF feasible deadlines is *smaller than or equal to* the one shown in Fig. 8.

From Observations 1 and 2 we conclude that the gray area illustrated in both Figs. 5 and 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 k's

The derivation of the feasible deadline space based on (15) is impractical, because it requires the intersection of infinite regions, derived from all possible $\mathbf{k}$'s ranging in $\mathbb{N}^n \setminus \{\mathbf{0}\}$. 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 vertex. For instance, in Fig. 8, $\mathbf{D}^{\text{vert}}(3,2) = (4,5)$ is dominated by $\mathbf{D}^{\text{vert}}(1,1) = (5,5)$. In this section we formally define the idea of domination, and we will exploit this property to make an effective computation of the deadline space.

Definition 1 We say that an integer vector $\hat{\mathbf{k}}$ is *dominated* by the integer vector $\mathbf{k}$ if

$$\forall i = 1, \ldots, n \quad D_i^{\text{vert}}(\hat{\mathbf{k}}) \leq D_i^{\text{vert}}(\mathbf{k}) \tag{16}$$

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 the intersection with the region associated with $\hat{\mathbf{k}}$ cannot eliminate any point in the space of deadlines that has not already been eliminated by the region associated with $\mathbf{k}$. To translate (16) as a constraint on the integers $\hat{\mathbf{k}} = (\hat{k}_1, \ldots, \hat{k}_n)$, let us define $I_{\mathbf{k}} = \{i : k_i \neq 0\}$. From (16) we have that

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

$$\begin{cases} \forall i \in I_{\mathbf{k}} & D_i^{\text{vert}}(\hat{\mathbf{k}}) \leq \mathbf{k} \cdot \mathbf{C} - (k_i - 1)T_i \\ \forall i \notin I_{\mathbf{k}} & D_i^{\text{vert}}(\hat{\mathbf{k}}) \leq +\infty \end{cases} \tag{17}$$

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

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

For a given $\mathbf{k}$, (17) describes the region of the integer vectors $\hat{\mathbf{k}}$ that are dominated by $\mathbf{k}$. Such a region is a cone with vertex at $\mathbf{k}$, which can be formally defined by

$$\text{cone}(\mathbf{k}, I_{\mathbf{k}}) = \left\{ \hat{\mathbf{k}} \in \mathbb{N}^n : \forall i \in I_{\mathbf{k}} \; (\hat{k}_i - k_i)(T_i - C_i) - \sum_{j \neq i} (\hat{k}_j - k_j)C_j \geq 0 \right\} \tag{18}$$

These cones are important because they allow a significant reduction in the number of integer vectors to be tested. In fact, suppose there is a finite subset of integer vectors

$\mathsf{dom}\,\mathsf{K} \subseteq \mathbb{N}^n \setminus \{\mathbf{0}\}$ such that

$$\mathbb{N}^n \setminus \{\mathbf{0}\} = \bigcup_{\mathbf{k} \in \mathsf{dom}\,\mathsf{K}} \mathsf{cone}(\mathbf{k}, I_{\mathbf{k}}) \tag{19}$$

then the necessary and sufficient condition of (15) becomes *equivalent* to

$$\bigcap_{\mathbf{k} \in \mathsf{dom}\,\mathsf{K}} \bigcup_{i \in I_{\mathbf{k}}} D_i \geq \mathbf{k} \cdot \mathbf{C} - (k_i - 1)T_i. \tag{20}$$

In fact, the vectors in $\mathsf{dom}\,\mathsf{K}$ are explicitly checked by the test of (20), whereas the vectors in

$$\bigcup_{\mathbf{k} \in \mathsf{dom}\,\mathsf{K}} (\mathsf{cone}(\mathbf{k}, I_{\mathbf{k}}) \setminus \{\mathbf{k}\})$$

which are the points in the cones except the vertices, need not be tested because they are dominated by some vector in $\mathsf{dom}\,\mathsf{K}$. Hence, a key problem is to find the smallest $\mathsf{dom}\,\mathsf{K}$ which has the desirable property of (19), since a small $\mathsf{dom}\,\mathsf{K}$ results in a faster computation of the **D**-space from (20). It is quite clear that for large cones the set $\mathsf{dom}\,\mathsf{K}$ can be reduced, because a large cone permits to dominate more vectors, and consequently the number of integer vectors that need to be explicitly checked is smaller.

In order to derive the properties of the cones, it can be noticed that the coefficients of the hyperplanes delimiting the cone do not change with **k**, meaning that the "shape" of the cone remains unaltered for the same set of non-zero indexes $I_{\mathbf{k}}$. More formally, we can state the property of "invariance for translation" as follows

$$\mathsf{cone}(\mathbf{k}, I_{\mathbf{k}}) = \mathsf{cone}(\mathbf{0}, I_{\mathbf{k}}) + \mathbf{k} \tag{21}$$

with the implicit meaning that the plus sign at the right-hand side of (21) means a translation of the set.

The indexes in $I_{\mathbf{k}}$ simply denote the constraints delimiting $\mathsf{cone}(\mathbf{k}, I_{\mathbf{k}})$. Fig. 9 shows three examples of $\mathsf{cone}(\mathbf{k}, I_{\mathbf{k}})$. The black dots represent three possible vertices of the cones (resp. $(0, 2)$, $(1, 1)$, and $(4, 0)$), whereas the white dots represent the sets of integers which are dominated by $\mathsf{cone}((0, 2), \{2\})$, $\mathsf{cone}((1, 1), \{1, 2\})$, and $\mathsf{cone}((4, 0), \{1\})$ respectively.

Due to the translation property of (21), we can restrict the study to $\mathsf{cone}(\mathbf{0}, I)$ and then extend the results to the other cones by translation.

The following theorem allows the definition of a suitable finite subset $\mathsf{dom}\,\mathsf{K}$.

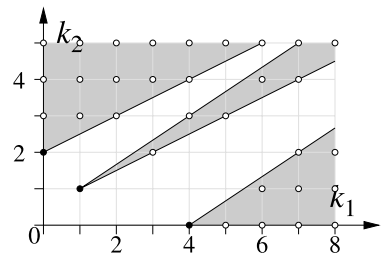


Fig. 9 Examples on $\mathsf{cone}(\mathbf{k}, I_{\mathbf{k}})$

Theorem 2 *Let* $\mathbf{k}^{\max} \in \mathsf{cone}(\mathbf{0}, \{1, \ldots, n\})$, $\mathbf{k}^{\max} \neq \mathbf{0}$. *Then*

$$\mathsf{dom}\,\mathsf{K} = \{\mathbf{k} \in \mathbb{N}^n \setminus \{\mathbf{0}\} : 0 \leq k_i \leq k_i^{\max}\} \tag{22}$$

satisfies the property of (19).

Proof We have to prove that $\mathbb{N}^n \setminus \{\mathbf{0}\}$ can be covered by cones whose vertices are in $\mathsf{dom}\,\mathsf{K}$. Given $\hat{\mathbf{k}} \in \mathbb{N}^n \setminus \{\mathbf{0}\}$ we will build the proper $\mathbf{k} \in \mathsf{dom}\,\mathsf{K}$ such that $\hat{\mathbf{k}} \in \mathsf{cone}(\mathbf{k}, I_{\mathbf{k}})$.

Using Euclidean division, let's define q_i and r_i as follows

$$\forall i = 1, \ldots, n \quad \hat{k}_i = q_i k_i^{\max} - r_i \quad q_i \in \mathbb{N},\ 0 \leq r_i \leq k_i^{\max} - 1$$

and let us set $q = \max_i\{q_i\}$. Since $\hat{\mathbf{k}} \neq \mathbf{0}$ we have $q \geq 1$. Let us also set $I = \{i = 1, \ldots, n : q_i = q\}$. We claim that the cone vertex $\mathbf{k}$ such that $\hat{\mathbf{k}} \in \mathsf{cone}(\mathbf{k}, I_{\mathbf{k}})$ is defined as follows

$$\begin{cases} k_i = k_i^{\max} - r_i & \text{if } i \in I \\ k_i = 0 & \text{if } i \notin I \end{cases} \tag{23}$$

First of all, we verify that $\mathbf{k} \in \mathsf{dom}\,\mathsf{K}$. Since $0 \leq r_i \leq k_i^{\max}$, it follows that $1 \leq k_i \leq k_i^{\max}$. Moreover $\mathbf{k} \neq \mathbf{0}$ because I is not empty. A posteriori we verify that such a definition of $\mathbf{k}$ permits to assert that $I = \{i : k_i \neq 0\} = I_{\mathbf{k}}$. In fact

$$i \in I \Rightarrow k_i = k_i^{\max} - r_i \Rightarrow k_i \neq 0 \Rightarrow i \in I_{\mathbf{k}}$$
$$i \notin I \Rightarrow k_i = 0 \Rightarrow i \notin I_{\mathbf{k}}$$

Finally, we verify that the constructed $\mathbf{k}$ dominates $\hat{\mathbf{k}}$, as required:

$$\hat{\mathbf{k}} \in \mathsf{cone}(\mathbf{k}, I_{\mathbf{k}}) = \mathsf{cone}(\mathbf{k}, I) \quad \Leftrightarrow \quad \forall i \in I\ (\hat{k}_i - k_i)(T_i - C_i) - \sum_{j \neq i} (\hat{k}_j - k_j) C_j \geq 0 \tag{24}$$

We will proceed by finding lower estimates of the previous inequality, for all $i \in I$:

$$\begin{aligned}
&(\hat{k}_i - k_i)(T_i - C_i) - \sum_{j \neq i,\, j \in I} (\hat{k}_j - k_j) C_j - \sum_{j \neq i,\, j \notin I} (\hat{k}_j - k_j) C_j \\
&\quad = (q_i k_i^{\max} - r_i - (k_i^{\max} - r_i))(T_i - C_i) \\
&\qquad - \sum_{j \neq i,\, j \in I} (q_j k_j^{\max} - r_j - k_j) C_j - \sum_{j \neq i,\, j \notin I} \hat{k}_j C_j \\
&\quad = (q-1) k_i^{\max} (T_i - C_i) - (q-1) \sum_{j \neq i,\, j \in I} k_j^{\max} C_j - \sum_{j \neq i,\, j \notin I} \hat{k}_j C_j \\
&\quad \geq (q-1) k_i^{\max} (T_i - C_i) - (q-1) \sum_{j \neq i,\, j \in I} k_j^{\max} C_j - \sum_{j \neq i,\, j \notin I} (q-1) k_j^{\max} C_j \\
&\quad = (q-1) \left(k_i^{\max} (T_i - C_i) - \sum_{j \neq i} k_j^{\max} C_j \right) \geq 0
\end{aligned}$$

because $q \geq 1$ and $\mathbf{k}^{\max} \in \mathsf{cone}(\mathbf{0}, \{1, \ldots, n\})$. Hence the inequality of (24) is proved and the theorem follows. □

Theorem 2 allows to reduce the problem of computing the EDF feasible deadlines to the problem of finding an integer vector $\mathbf{k}^{\max}$ in $\mathsf{cone}(\mathbf{0}, \{1, \ldots, n\})$ other than the vertex $\mathbf{0}$. If this point is found, then (20) permits to compute the space of EDF feasible deadlines. Unfortunately, the complexity is still proportional to the cardinality of $\mathsf{dom}\,\mathsf{K}$, which is $\prod_{i=1}^{n}(k_i^{\max}+1)-1$. How do we search for a suitable point $\mathbf{k}^{\max}$ that minimizes the complexity?

We propose to span onto all the integer vectors in $\mathsf{cone}(\mathbf{0}, \{1, \ldots, n\})$, moving to the direction of the vertex. This strategy can be effectively treated as an Integer Linear Programming (ILP) problem.

The first constraint of the ILP problem must translate the property that $\mathbf{k} \in \mathsf{cone}(\mathbf{0}, \{1, \ldots, n\})$. From (18), it follows that the constraint is expressed by the inequality

$$\forall i = 1, \ldots, n \quad k_i(T_i - C_i) - \sum_{j \neq i} k_j C_j \geq 0. \tag{25}$$

Then a second constraint must erase the vertex from the cone because we require (see Theorem 2) that $\mathbf{k} \neq \mathbf{0}$. Hence we add

$$\sum_{i=1}^{n} k_i \geq 1 \tag{26}$$

which erases only $\mathbf{0}$ from $\mathbb{N}^n$.

Finally, we set the minimization direction for reducing the cardinality of $\mathsf{dom}\,\mathsf{K}$. The goal of the problem is to minimize $\prod_i (k_i + 1)$. However this function is not well suited for the ILP problem because it is not linear. Hence we choose the following convenient linear cost function

$$\text{minimize} \sum_{i=1}^{n} k_i \tag{27}$$

which approximates $|\mathsf{dom}\,\mathsf{K}|$ at the first order in the point $\mathbf{0}$.

Now, it is worth showing how the previous result can be applied to compute the set $\mathsf{dom}\,\mathsf{K}$ for the task set of the Example 1. When $\mathbf{C} = (2, 3)$ and $\mathbf{T} = (4, 7)$, the solution of the ILP problem is $\mathbf{k}^{\max} = (2, 1)$. In Fig. 10, black dots represent the integers that belong to $\mathsf{dom}\,\mathsf{K}$. For each $\mathbf{k} \in \mathsf{dom}\,\mathsf{K}$ we also represent $\mathsf{cone}(\mathbf{k}, I_{\mathbf{k}})$. It can be seen that all the integers are dominated by some $\mathbf{k} \in \mathsf{dom}\,\mathsf{K}$, as proved in Theorem 2. It can also be noticed that $\mathsf{cone}((2, 0), \{1\}) \subseteq \mathsf{cone}((1, 0), \{1\})$ meaning that in this example it would be enough to consider $\mathsf{dom}\,\mathsf{K} = \{(1, 0), (0, 1), (1, 1), (2, 1)\}$, erasing $(2, 0)$. However this tighter reduction of points in $\mathsf{dom}\,\mathsf{K}$ is extremely challenging when it has to be generalized to an n-dimensional space.

In the next subsection we investigate what parameters affect the complexity of the set $\mathsf{dom}\,\mathsf{K}$.

5.1 Complexity of dom K

From Figs. 9 and 10 it is quite clear that the solution to the ILP problem described in (25), (26), and (27) depends on the width of the cone. The wider the cone, the smaller the solution, and consequently the fewer points in dom K.

For measuring the cone width we translated the n hyperplanes of the boundary in such way the ith hyperplane passes through the vector $\mathbf{e}_i$, which has all zeros except in the ith, position where it has 1 (see Fig. 11 for a graphical representation). Then, we computed the intersection point $\mathbf{k}^{\text{int}}$: if $\mathbf{k}^{\text{int}}$ has large coordinates then the cone is narrow, and vice versa. The coordinates of $\mathbf{k}^{\text{int}}$ are the solution of the following linear system:

$$\begin{pmatrix} T_1 - C_1 & -C_2 & \dots & -C_n \\ -C_1 & T_2 - C_2 & \dots & -C_n \\ \vdots & \vdots & \ddots & \vdots \\ -C_1 & -C_2 & \dots & T_n - C_n \end{pmatrix} \cdot \mathbf{k}^{\text{int}} = \begin{pmatrix} T_1 - C_1 \\ T_2 - C_2 \\ \vdots \\ T_n - C_n \end{pmatrix}$$

and, by Cramer's rule, we find

$$k_i^{\text{int}} = \frac{(T_i - C_i)(1 - \sum_{j \neq i} U_j) + \sum_{j \neq i} C_j (1 - U_j)}{T_i (1 - U)} \tag{28}$$

The coordinates of $\mathbf{k}^{\text{int}}$ are inversely proportional to $1 - U$, whereby the set dom K grows as U approaches 1. Therefore, for small values of U, the size of dom K is small. On the other hand, as the total utilization U approaches 1, the size of dom K increases and, consequently, the complexity of (20) grows exponentially with n.

Fig. 10 An example of dom K

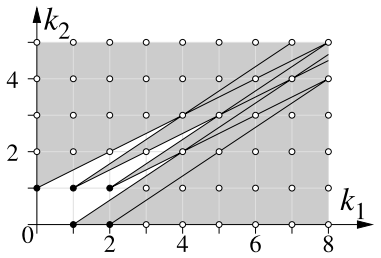
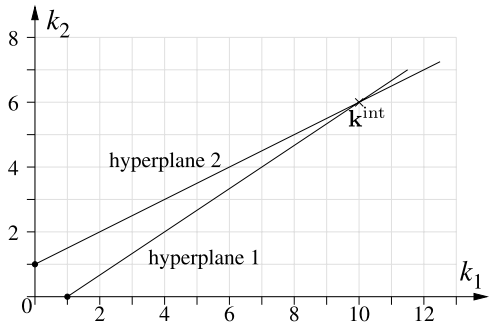


Fig. 11 Finding the width of the cone



Note that the dependency of the complexity on the total processor utilization U is also typical of the processor demand test proposed by Baruah et al. (1990), which requires to test all the absolute deadline not exceeding $\frac{\sum_i (T_i - D_i)U_i}{1-U}$. This confirms that, as the total utilization U approaches 1, the EDF schedulability guarantee tests become intrinsically more complex.

U approaching 1 Although real-time systems should be designed to have a total utilization well below 1, overload conditions can occur for several reasons, so it is interesting to analyze what happens when the total processor utilization U approaches the value of 1. We already observed that as U approaches 1, the cone $\mathsf{cone}(\mathbf{0}, \{1, \ldots, n\})$ becomes narrower. When U is exactly 1, the cone becomes an half-line and the solution of the ILP problem may become impractical.

Let us explicitly consider the case when $n = 2$ and $U = U_1 + U_2 = 1$. When formulating the ILP problem, the constraints of the cone (25) become

$$\begin{cases} (T_1 - C_1)k_1 - C_2 k_2 \geq 0 \\ -C_1 k_1 - (T_2 - C_2)k_2 \geq 0 \end{cases}$$

$$\begin{cases} T_1(1 - U_1)k_1 - T_2 U_2 k_2 \geq 0 \\ T_1 U_1 k_1 - T_2(1 - U_2)k_2 \leq 0 \end{cases} \tag{29}$$

$$\begin{cases} T_1 k_1 - T_2 k_2 \geq 0 \\ T_1 k_1 - T_2 k_2 \leq 0 \end{cases}$$

$$T_1 k_1 = T_2 k_2$$

The integers (k_1, k_2) which solves (29) are $k_1 = \frac{T_2}{m}$ and $k_2 = \frac{T_1}{m}$, where m is the greatest number such that $T_1 = r_1 m$, $T_2 = r_2 m$, and $r_1, r_2 \in \mathbb{N}$.

This solution suggests that when the total utilization U is equal to 1 then the complexity of the set $\mathsf{dom}\,\mathsf{K}$ becomes tightly related with the integer relationships among the task periods. Such a remark is in accordance to the well-known property of the EDF guarantee test (Baruah et al. 1990), which becomes more complex as the least common multiple of the periods increases.

In the next subsection we show some simulations which confirm the previous observations.

5.2 Simulations

In these experiments we explored the dependency of the cardinality of $\mathsf{dom}\,\mathsf{K}$ on the utilization U and the *harmonicity* of the periods, which we define as the ratio between the least common multiple and the greatest common divisor of all the periods. The cardinality of $\mathsf{dom}\,\mathsf{K}$ is measured by $\prod_i (k_i^{\max} + 1)$, where $\mathbf{k}^{\max}$ is the optimum of the ILP problem described in (25)–(27).

In all the experiment the number of tasks has been set equal to 5, since the dependency on the number of tasks is exponential and it didn't require further investigation. The task utilizations are real numbers extracted randomly using the UUniFast

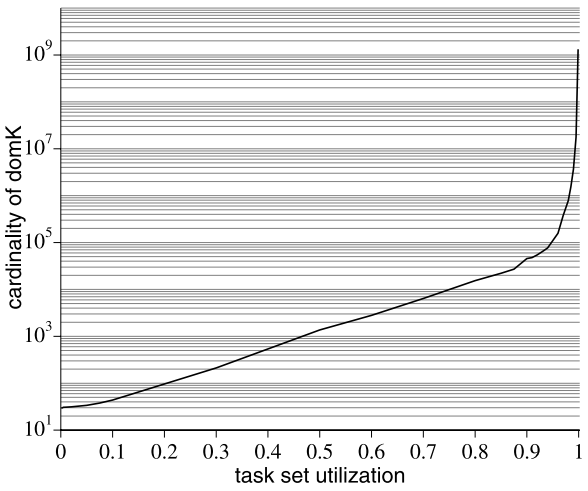


Fig. 12 Dependency of the utilization

algorithm (Bini and Buttazzo 2005), which generates uniform utilizations. The task periods are integers generated as the product of four factors extracted randomly in the set $\{1, 2, 3, 4, 5\}$, in order to reproduce different values of harmonicity.

Dependency on the utilization In the first experiment we explored the dependency of the cardinality of $\mathsf{dom}\,\mathsf{K}$ on the task set utilization U. 10000 random task sets have been generated for each utilization value. The result is reported in Fig. 12. It can be noticed that, as expected, the cardinality of $\mathsf{dom}\,\mathsf{K}$ (plotted in log scale), grows with U. As the utilization gets closer to 1, $|\mathsf{dom}\,\mathsf{K}|$ grows up to very high values.

Dependency on the harmonicity In the second experiment we explored the dependency on the harmonicity of the task set, defined as the ratio between the least common multiple (LCM) and the greatest common divisor (GCD) of the periods. We selected three possible utilizations (1, 0.95 and 0.5). For each utilization value we generated 1000 task set and we plotted the resulting value of $|\mathsf{dom}\,\mathsf{K}|$ (see Fig. 13. Both axes are in log scale). In the figure we also plot the linear interpolation of the points. We can notice a clear dependency on the harmonicity. In the extreme case $U = 1$ this dependency becomes very significant.

These experiments confirm the results regarding the complexity shown in Sect. 5.1.

6 A convex approximation of the D-space

Considering the high complexity of the method for determining the region of the feasible deadlines, in this section we propose a simpler approach for deriving an approximate region. We will prove that the resulting sufficient region is convex, allowing convex optimization routines to efficiently find the optimal deadline assignment.

The complexity of finding the exact region mainly comes from the floor operator, which forced us to introduce the integer variables k_i.

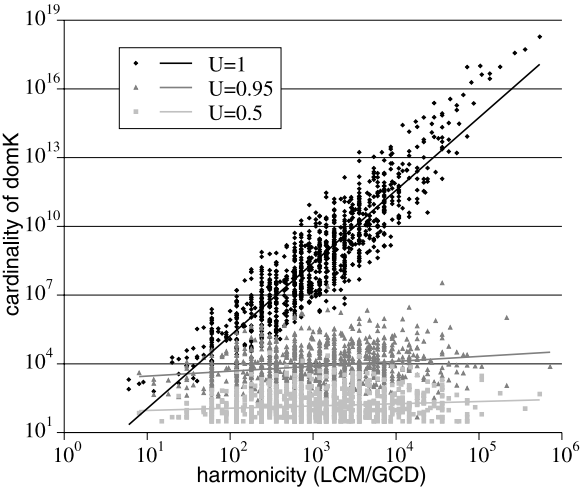


Fig. 13 Dependency on the harmonicity

Now we attempt to simplify the feasibility condition of (1) by relaxing the floor operator. Clearly, the resulting analysis loses necessity, but it gains greater simplicity with respect to the exact approach.

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

$$\forall t \in \mathsf{dlSet} \quad \sum_{i=1}^{n} \max\left\{0, \left\lfloor \frac{t + T_i - D_i}{T_i} \right\rfloor\right\} C_i \leq t \tag{30}$$

where dlSet is the set of deadlines within the least common multiple of the task periods H (called *hyperperiod*) not exceeding the value

$$L_{\max} = \max\left\{D_1, \ldots, D_n, \frac{\sum_{i=1}^{n}(T_i - D_i)U_i}{1 - U}\right\}$$

Hence, we can say that condition (30) has to be tested $\forall t \in \mathsf{dlSet}$, where

$$\mathsf{dlSet} = \{d_{ik} \text{ is an absolute deadline} \mid d_{ik} \leq \min(L_{\max}, H)\}. \tag{31}$$

A first simplification can be done by removing the "max" operator. As shown by Chantem et al. (2006), the max operator can be removed when the second term is greater than or equal to zero. That is when

$$\begin{aligned}
&\forall t \in \mathsf{dlSet}, \forall i \quad \left\lfloor \frac{t + T_i - D_i}{T_i} \right\rfloor \geq 0 \\
&\forall t \in \mathsf{dlSet}, \forall i \quad \frac{t + T_i - D_i}{T_i} \geq 0 \\
&\forall t \in \mathsf{dlSet}, \forall i \quad t + T_i - D_i \geq 0 \\
&\forall t \in \mathsf{dlSet}, \forall i \quad D_i \leq t + T_i \\
&\forall i \quad D_i \leq T_i + D_{\min}
\end{aligned} \tag{32}$$

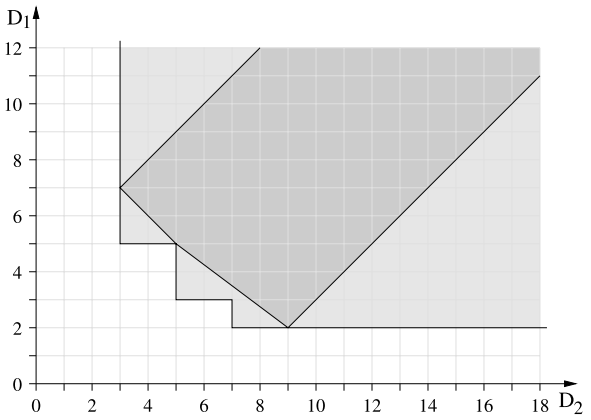


Fig. 14 The approximated region

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

$$\forall t \in \mathsf{dlSet} \quad \sum_{i=1}^{n} \left\lfloor \frac{t + T_i - D_i}{T_i} \right\rfloor C_i \leq t. \tag{33}$$

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

$$\forall t \in \mathsf{dlSet} \quad \sum_{i=1}^{n} \frac{t + T_i - D_i}{T_i} C_i \leq t \tag{34}$$

from which it directly follows that the feasible deadlines must satisfy

$$\forall t \in \mathsf{dlSet} \quad \sum_{i=1}^{n} U_i D_i \geq \sum_{i=1}^{n} C_i - t\left(1 - \sum_{i=1}^{n} U_i\right) \tag{35}$$

Now we observe that (35) 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 becomes

$$\sum_{i=1}^{n} U_i D_i \geq \sum_{i=1}^{n} C_i - D_{\min}\left(1 - \sum_{i=1}^{n} U_i\right) \tag{36}$$

In Fig. 14 we compare the approximated region (darker) with the exact region of Example 1.

Convexity We now investigate whether the resulting region is convex or not. First of all we recall that (36) is valid only in the hypothesis of (32), which can be restated

as follows:

$$\begin{aligned} &\forall i \quad && D_i \le T_i + D_{\min} \\ &\forall i, j \quad && D_i \le T_i + D_{\min} \le T_i + D_j \\ &\forall i, j \quad && D_i - D_j \le T_i \end{aligned} \tag{37}$$

which is convex because it is the intersection of half spaces. Similarly, the condition in (36) is equivalent to

$$\forall j \quad D_j \left(1 - \sum_{i=1}^{n} U_i\right) + \sum_{i=1}^{n} U_i D_i \ge \sum_{i=1}^{n} C_i \tag{38}$$

which is convex for the same reason. Since the overall region is determined by intersecting (37) and (38) it is also convex, and delimited by n^2 linear constraints.

Efficiency The convexity of the approximated region allows to efficiently find the solution of the optimal deadline assignment problem on the region described by (36). However, the cost of approximating the exact region by the convex sub-region is not clear. To evaluate such a penalty, let $J : \mathbb{R}^n \to \mathbb{R}$ be the system cost to be minimized expressed as a function of the task deadlines. We say that the cost J is *deadline increasing* when

$$\forall \mathbf{D}, \ \forall i \quad D_i \le D_i' \quad \Rightarrow \quad J(D_1, \ldots, D_i, \ldots, D_n) \le J(D_1, \ldots, D_i', \ldots, D_n) \tag{39}$$

This property is very natural, since it is expected that reducing any deadline can only reduce the system cost.

In the hypothesis of a deadline increasing system cost, then deadline assignment $\mathbf{D}$ is always superior to all the assignments within the region $\{\mathbf{D}' \in \mathbb{R}^n : \forall i \ D_i' \ge D_i\}$. Hence, under this assumption, a solution within the convex constraint of (36) is always superior to all the deadline assignments within the darker grey region depicted in Fig. 15. From the figure, we can see that the solution found on the convex approximation has a better performance of a large portion of the space of all feasible

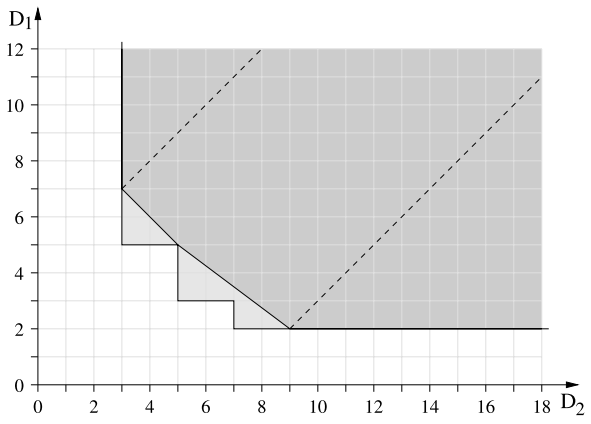


Fig. 15 Quality of the solution on the convex sub-region

deadlines. The only part that is not covered is the area around the staircase boundary that needs to be explored explicitly. However the solution on the convex sub-region can be an excellent starting point for further improvement.

7 Application of the D-space

The natural application of the **D**-space is the selection of feasible task deadlines that optimizes a given performance function. Hence the work presented in this paper can only be considered a first step toward the final goal of performing the optimization of task deadlines.

To clarify the benefit of the knowledge of the **D**-space, we propose a simple example with two tasks. We choose only two tasks to simplify the graphical representation. In the rest of the explanation we will stress the aspects that are negatively affected by the number of tasks.

Example 2 Let us consider a task set with parameters $T_1 = D_1 = 4$, $C_1 = 2$, $T_2 = D_2 = 7$, and $C_2 = 3.5$. Suppose that our goal is to modify the deadlines such that $D_1^2 + D_2^2$ is minimized. If we apply the existing deadline minimization algorithm (Balbastre et al. 2006; Hoang et al. 2006) starting from the given implicit deadline assignment, we find $D_1 = 3.5$ and $D_2 = 7$ if we start minimizing D_1, or $D_1 = 4$ and $D_2 = 6.5$ if we start minimizing D_2. In these two cases, we achieve a cost of $D_1^2 + D_2^2 = 61.25$ and 58.25, respectively. However previous research papers do not permit to determine whether a better deadline assignment is possible or not.

If we attempt to find the region of feasible deadlines, first we need to find a finite set $\mathsf{dom}\,\mathsf{K}$ that has the covering property of (19). In fact, this set of integer vectors allows us to define the feasible deadlines (20).

Theorem 2 suggests that if we find $\mathbf{k}^{\max} \in \mathsf{cone}(\mathbf{0}, 1, \ldots, n)$, $\mathbf{k}^{\max} \neq \mathbf{0}$, then the set $\mathsf{dom}\,\mathsf{K} = \{0 \leq k_1 \leq k_1^{\max}, 0 \leq k_2 \leq k_2^{\max}, \mathbf{k} \neq \mathbf{0}\}$ has the desired property. $\mathbf{k}^{\max}$ can be found by solving the ILP problem constrained by (25) and (26) whose goal is given by (27). In the example that we are considering, the ILP problem becomes

$$\text{minimize } k_1 + k_2$$

$$\text{subject to } \begin{cases} T_1 k_1 = T_2 k_2 \;\Rightarrow\; 4k_1 = 7k_2 \\ k_1 + k_2 \geq 0 \\ k_1, k_2 \in \mathbb{N} \end{cases}$$

whose solution is $(k_1, k_2) = (7, 4)$. Notice that if the number of tasks grows, solving the ILP problem can be very time consuming.

By applying Theorem 2, we have that $\mathsf{dom}\,\mathsf{K} = \{0, \ldots, 7\} \times \{0, \ldots, 4\} \setminus \{(0, 0)\}$ allows the description of the exact region of feasible deadlines, which is depicted in Fig. 16. In the figure, we denote by a white dot the nominal values of the deadlines $\mathbf{D} = (4, 7)$.

Since the goal of the deadline selection is the minimization of $D_1^2 + D_2^2$, the level curves of the cost function are circles centered in the origin. Hence, from the figure, we can notice that by assigning the deadlines $D_1 = 5.5$ and $D_2 = 5$ (or equivalently

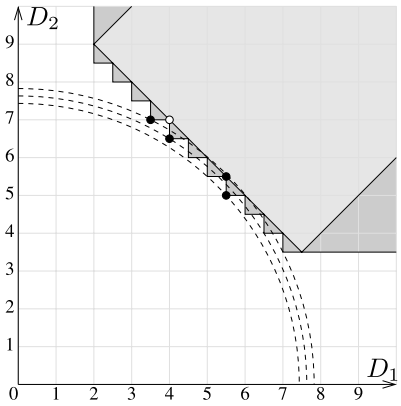


Fig. 16 Feasible deadlines of the example

$D_1 = 5$ and $D_2 = 5.5$) we achieve the minimum cost (55.25), which is lower than the cost found using the previous methods (Balbastre et al. 2006; Hoang et al. 2006).

In the figure it can be noticed also that, if the convex set of feasible deadlines is used, then the solution found is $D_1 = D_2 = 5.5$, achieving a cost of 60.5. The simplicity for finding the solution onto the convex constraint is paid in terms of an increase of cost.

However, we must say that for a large number of tasks, this process would be very inefficient, since it requires the enumeration of all the possible local minima. We are currently investigating whether it is possible to perform the optimal deadline assignment more efficiently.

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 so as to decrease 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 optimization.

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

Balbastre P, Ripoll I, Crespo A (2006) Optimal deadline assignment for periodic real-time tasks in dynamic priority systems. In: Proceedings of the 18th Euromicro conference on real-time systems, Dresden, Germany, July 2006, pp 65–74

Baruah SK, Burns A (2006) Sustainable schedulability analysis. In: Proceedings of the 27th IEEE real-time systems symposium, Rio de Janeiro, Brazil, December 2006, pp 159–168

Baruah SK, Howell R, Rosier L (1990) Algorithms and complexity concerning the preemptive scheduling of periodic, real-time tasks on one processor. Real-Time Systems 2:301–324

Baruah SK, Buttazzo G, Gorinsky S, Lipari G (1999) Scheduling periodic task systems to minimize output jitter. In: Proceedings of the 6th international conference on real-time computing systems and applications, Hong Kong, December 1999, pp 62–69

Bini E, Buttazzo GC (2005) Measuring the performance of schedulability tests. Real-Time Systems 30(1–2):129–154

Bini E, Di Natale M (2005) Optimal task rate selection in fixed priority systems. In: Proceedings of the 26th IEEE real-time systems symposium, Miami, FL, USA, December 2005, pp 399–409

Bini E, Di Natale M, Buttazzo GC (2006) Sensitivity analysis for fixed-priority real-time systems. In: Proceedings of the 18th Euromicro conference on real-time systems, Dresden, Germany, July 2006, pp 13–22

Buttazzo G, Cervin A (2006) Analysis and evaluation of jitter control methods. Technical report RETIS-TR06-01, Scuola Superiore Sant'Anna, Pisa, Italy

Buttazzo G, Sensini F (1999) Optimal deadline assignment for scheduling soft aperiodic task in hard real-time environments. IEEE Trans Comput 48(10):1035–1052

Cervin A (1999) Improved scheduling of control tasks. In: Proceedings of the 11th Euromicro conference on real-time systems, York, UK, June 1999, pp 4–10

Cervin A, Lincoln B, Eker J, Årzén K-E, Buttazzo G (2004) The jitter margin and its application in the design of real-time control systems. In: Proceedings of the 10th international conference on real-time and embedded computing systems and applications, Göteborg, Sweden, August 2004

Chantem T, Sharon Hu X, Lemmon MD (2006) Generalized elastic scheduling. In: Proceedings of the 27th IEEE real-time systems symposium, Rio de Janeiro, Brazil, December 2006, pp 236–245

Crespo A, Ripoll I, Albertos P (1999) Reducing delays in RT control: the control action interval. In: Proceedings of the 14th IFAC World Congress, Beijing, China, July 1999, pp 257–262

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

Di Natale M, Stankovic JA (2000) Scheduling distributed real-time tasks with minimum jitter. IEEE Trans Comput 49(4):303–316

Hoang H, Buttazzo G, Jonsson M, Karlsson S (2006) 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, August 2006, pp 125–134

Kalman RE, Bertram JE (1959) A unified approach to the theory of sampling systems. J Franklin Inst 267:405–436

Kushner HJ, Tobias L (1969) On the stability of randomly sampled systems. IEEE Trans Autom Control 14(4):319–324

Liu CL, Layland JW (1973) Scheduling algorithms for multiprogramming in a hard real-time environment. J Assoc Comput Mach 20(1):46–61

Lluesma M, Cervin A, Balbastre P, Ripoll I, Crespo A (2006) Jitter evaluation of real-time control systems. In: Proceedings of the 12th IEEE international conference on embedded and real-time computing systems and applications, Sydney, Australia, August 2006, pp 257–260

Martí P, Fuertes JM, Ramamritham K, Fohler G (2001) Jitter compensation for real-time control systems. In: Proceedings of the 22nd IEEE real-time system symposium, London, UK, December 2001, pp 39–48

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

Racu R, Hamann A, Ernst R (2006) A formal approach to multidimensional sensitivity analysis of embedded real-time systems. In: Proceedings of the 18th Euromicro conference on real-time systems, Dresden, Germany, July 2006, pp 3–12

Seto D, Lehoczky JP, Sha L, Shin KG (1996) On task schedulability in real-time control systems. In: Proceedings of the 17th IEEE real-time systems symposium, Washington, DC, USA, December 1996, pp 13–21

Seto D, Lehoczky JP, Sha L (1998) Task period selection and schedulability in real-time systems. In: Proceedings of the 19th IEEE real-time systems symposium, Madrid, Spain, December 1998, pp 188–198

Zheng Q, Shin KG (1994) On the ability of establishing real-time channels in point-to-point packet-switched networks. IEEE Trans Commun 42(2–4):1096–1105

Enrico Bini is assistant professor at the Scuola Superiore Sant'Anna in Pisa. He received the Ph.D. in Computer Engineering from the same institution in October 2004. In 2000 he received the Laurea degree in Computer Engineering from "Università di Pisa" and, one year later, he obtained the "Diploma di Licenza" from the Scuola Superiore Sant'Anna.
In 1999 he studied at Technische Universiteit Delft, in the Netherlands, by the Erasmus student exchange program. In 2001 he worked at Ericsson Lab Italy in Roma. In 2003 he was a visiting student at University of North Carolina at Chapel Hill, collaborating with prof. Sanjoy Baruah. His research interests cover scheduling algorithms, real-time operating systems, embedded systems design and optimization techniques.

Giorgio Buttazzo is Full Professor of Computer Engineering at the Scuola Superiore Sant'Anna of Pisa. His main research interests include real-time operating systems, dynamic scheduling algorithms, quality of service control, multimedia systems, advanced robotics applications, and neural networks. Prof. Buttazzo has been Program Chair and General Chair of the major international conferences on real-time systems. He is a member of the IEEE Technical Committee on Real-Time Systems and of the Euromicro Executive Board on Real-Time Systems. He has authored 6 books on real-time systems and over 200 papers in the field of real-time systems, robotics, and neural networks. For the importance of results achieved in his research, in 2005 Prof. Buttazzo received the title of Senior Member of IEEE.