Real-Time Syst (2008) 39: 5–30
DOI 10.1007/s11241-006-9010-1

Sensitivity analysis for fixed-priority real-time systems

Enrico Bini · Marco Di Natale · Giorgio Buttazzo

Published online: 12 April 2007

Abstract At early stages in the design of real-time embedded applications, the timing attributes of the computational activities are often incompletely specified or subject to changes. Later in the development cycle, schedulability analysis can be used to check the feasibility of the task set. However, the knowledge of the worst-case response times of tasks is often not sufficient to precisely determine the actions that would correct a non-schedulable design. In these situations, sensitivity analysis provides useful information for changing the implementation, by giving a measure of those computation times that must be reduced to achieve feasibility, or those that can be increased in case of a product extension, or providing the range of feasible periods for selecting the proper task activation rates.

In this work, we exploit the concept of feasibility region to propose a faster and more concise solution to the sensitivity analysis problem with respect to existing techniques based on binary search. Furthermore, we show how the formalization of other problems in the feasibility domain, such as managing overloads through elastic scheduling, can be extended to the exact analysis.

Keywords Sensitivity analysis · Fixed priority scheduler

Abbreviations

RM	Rate Monotonic
EDF	Earliest Deadline First
FP	Fixed Priorities

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

M. Di Natale
e-mail: marco@sssup.it

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

1 Introduction

Schedulability theory can be used at design time for checking the timing constraints of a real-time system, whenever a model of its software architecture is available. In specific cases, schedulability analysis techniques can significantly shorten the development cycle and reduce the time to market. Experiences on real-world applications show that design time validation of the schedulability properties can save over 50% of the time needed for the whole design process (Stankovic et al. 2003), by simply pruning non-feasible solutions from the design space.

In many cases, however, the system is characterized by a high degree of uncertainty on task activations and execution behaviour, which makes the use of schedulability analysis much less profitable. Uncertainty is tackled through a set of simplifications, typically based on worst-case assumptions, which make the system tractable at the price of an abundant resource allocation. Most schedulability analysis tools use a simplified view of the software architecture, which is modeled through a set of tasks, each characterized by a tuple (C_i, T_i, D_i) specifying its worst-case computation time, period and deadline, respectively. Determining the values of these task parameters is not trivial, because in real systems, external events and computation times are often characterized by a high degree of uncertainty.

Based on this model, companies such as TimeSys and TriPacific (among others) produce tools to be used at design time to check the timing properties of a real-time application. These tools can act as plug-ins for design environments, such as the IBM Rational Rose Technical Developer. Other vendors (ARTISAN and ILogix, among those) feature their own sets of tools/utilities for checking the schedulability of a real-time design.

In order to handle such a high degree of variability, research efforts have been devoted to propose models that provide enough expressive power for the definition of the semantics of task activation, communication and synchronization. Extensions of the model based on the triple (C_i, T_i, D_i) have been proposed to add flexibility and reduce pessimism. For example, the multi-frame task model proposed by Mok and Chen (1997) allows the user to specify different execution times for different task instances, through a sequence of numbers. The generalized multi-frame task model (Baruah et al. 1999) adds explicit deadlines to the multi-frame model and allows assigning distinct minimum separation values to the instances inside a multi-frame cycle. The recurring task model defined by Baruah (2003) allows modeling restricted forms of conditional real-time code and control flows.

To relax the pessimistic assumptions typically made in the evaluation of the task set attributes, some authors (Burns et al. 2003; Manolache et al. 2001) proposed a probabilistic model for task execution times and activations, which can be used to analyze soft real-time applications. Feasibility tests for these models are derived by applying stochastic analysis techniques and provide a probability for each task to meet its deadline.

Besides pessimism and insufficient flexibility, there is at least another problem with most current analysis methodologies: no informative result is provided to assist the designer in understanding how a change in the task parameters would affect the feasibility of the system. To be useful, the output of a schedulability analyzer should highlight the amount of lateness or slack available in each thread to plan for a corrective action. Desirable or allowable changes should be precisely expressed and measurable.

Information on how modifications of the task parameters may affect the system is indeed provided by *sensitivity analysis*, which is a generalization of feasibility analysis. In the definition of a new product, sensitivity analysis may be applied to a system model where computation times are defined with a degree of uncertainty. When a system implementation already exists, it can be used to define the bounds for possible function extensions. In both cases, starting from a feasible task set, sensitivity analysis provides the exact amount of change affordable in task computation times or periods to keep the task set feasible.

If the task set is not feasible, sensitivity analysis provides a quantitative indication of the actions required to bring the system back into a feasible state. If a system is not schedulable, there can be many causes and remedies: insufficient computing power from the CPU; excessive computational load from one or more real-time tasks, thereby requesting a faster (hence, often simpler and less accurate) implementation; excessive usage of shared resources; and, finally, too short periods for the execution of periodic tasks, which could be increased at the price of a performance degradation.

Being a generalization of feasibility analysis, sensitivity analysis heavily borrows from existing results. The most popular among them is the Response Time Analysis (RTA) (Joseph and Pandya 1986; Audsley et al. 1993). Although alternative efficient techniques have been proposed in the literature (Bini and Buttazzo 2004), RTA is currently the most adopted technique for deciding upon the schedulability of a task set, since it provides a necessary and sufficient condition. Efforts have been devoted to reduce its average complexity (Sjödin and Hansson 1998) and extend its applicability to more general task models (Palencia and González Harbour 1998). Unfortunately, as it will be shown in the next section, RTA does not directly provide the amount of parameter variations admissible for the task set, but it can only be used within a binary search algorithm.

1.1 A motivating example

In the following, we illustrate a simple example that highlights the main problem of RTA. Figure 1(a) shows the schedule generated by the Rate Monotonic algorithm on four periodic tasks, whose parameters (C_i, T_i) are indicated in the figure (relative deadlines are equal to periods). We are interested in computing, for example, the admissible values for the computation time C_1 of the highest priority task τ_1 such that the response time R_4 of the fourth task τ_4 does not exceed its deadline D_4, equal to 32.

The initial estimate for C_1 is 1 unit of time, which makes the task set schedulable and lets the lowest priority task τ_4 complete at time $t = 24$. Unfortunately, the 8 units of slack of τ_4 do not guarantee a *robust* design against changes in the computation time and/or period of the other tasks. In fact, even a small increase in the computation time of any task (suppose, for example $C_1 = 1.01$) leads to a non-feasible schedule, with the completion time of τ_4 equal to at least 34 units of time. Similarly, a small decrease in the periods of tasks τ_1 or τ_2 would compromise schedulability.

As shown in Fig. 1(b), the response time of task τ_4, denoted by R_4, is a discontinuous function of the computation time of the highest priority task, and it is impossible

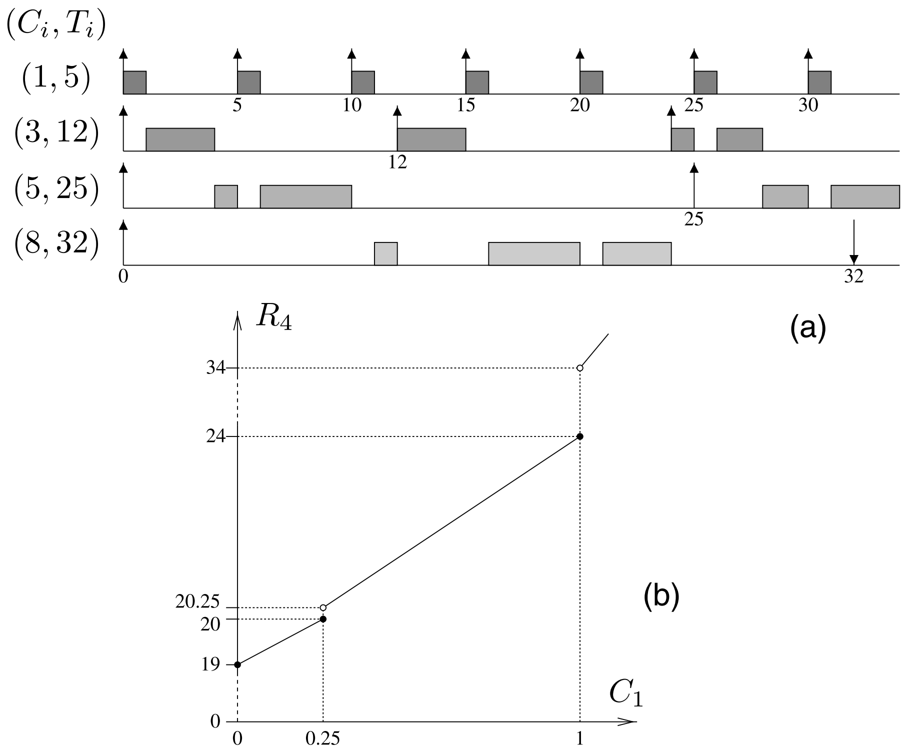

Fig. 1 **a** RM schedule of 4 periodic tasks. **b** Response time R_4 as a function of C_1

to analytically invert this relation to compute the exact value of C_1 that would allow R_4 to match the deadline of τ_4.

Discontinuities in the response time functions for system tasks are the main reason why response-time analysis is ill-fit for sensitivity analysis, except possibly by applying a binary search algorithm, which requires an estimate of the initial bounds and the evaluation of the response times of all tasks at each step.

1.2 Related work

Most of the existing solutions to the sensitivity problem have been provided in the domain of computation times. Early solutions to the sensitivity analysis problem can be traced back to the evaluation of the breakdown utilization (Lehoczky et al. 1989), where the critical scaling factor is computed for a fixed-priority scheduled system as the largest possible scaling value for task execution times still allowing the feasibility of the set.

The approach was later extended by Vestal (1994), who addressed sensitivity analysis of fixed priority task sets by introducing slack variables in the exact feasibility equations defined by Lehoczky et al. (1989) and solving the corresponding equality conditions. The approach suffers from the possibly very large number of conditions to be evaluated, and assumes deadlines equal to periods. Other techniques for sensitivity analysis are based on binary search over the schedulability property defined by Lehoczky et al. (1989), or other formulations of the algorithm for computing the response time of a task (Punnekkat et al. 1997; Tindell et al. 1994).

In the context of distributed scheduling, a system-level sensitivity analysis of the task computation times against end-to-end deadlines has been proposed (Racu et al. 2005). The method is based on binary search and has been implemented in a commercial tool by SymTAVision (Hamann et al. 2004). A problem of this method is that it performs a feasibility test at each iteration of the binary search, making the resulting complexity considerably high. Our algorithm does not suffer from this drawback, because it is as complex as a feasibility test. This greater simplicity comes at the price of a simpler task model. Other tools offering a built-in sensitivity analysis engine are RTA-OSEK by ETAS (formerly LiveDevices) and the analysis tool in MAST (Medina Pasaje et al. 2001).

In this paper we present a new type of sensitivity analysis that exploits the concept of *feasibility region* in the domain of the task variables, as introduced in (Bini and Buttazzo 2004; Bini and Di Natale 2005). By measuring the distance of a task configuration, expressed as a point in the space of the task variables, from the region delimiting the set of the feasible tasks, we can provide the variations that are required on each design parameter, if taken separately, for bringing the task set back into the schedulability region or for changing some parameters while preserving feasibility.

Unlike previous work, our sensitivity analysis also applies to the domain of task periods, providing a theoretical support for those control systems that perform rate adaptation to cope with overload conditions (Buttazzo et al. 2002, 2004). Moreover, it addresses very practical design issues, such as finding the fastest execution rate that can be assigned to one or more critical control loops. Alternatively, when fixing the defects of a non-schedulable configuration, it provides a measure of the required relaxation in the execution rate of one or more tasks to make the set schedulable.

The rest of the paper is organized as follows: Sect. 2 introduces the terminology and the system model; Sect. 3 describes the notion of feasibility region; Sects. 4 and 5 describe the sensitivity analysis of the computation times and the periods, respectively; Sect. 6 provides an example of application; Sect. 7 presents some experimental results aimed at evaluating the proposed approach with respect to a bisection method. Section 8 discusses the implementation of the proposed techniques in available tools; and finally, Sect. 9 states our conclusions and future work.

2 Terminology and system model

In this paper, we assume that an application consists of a set $\mathcal{T} = \{\tau_1, \tau_2, \dots, \tau_N\}$ of N real-time tasks running on a uniprocessor system. Each task τ_i is characterized by a period T_i, a worst-case execution time C_i and a relative deadline D_i. The notion of normalized deadline $\delta_i = D_i/T_i$ and activation rate $f_i = \frac{1}{T_i}$ are often used throughout the paper. Moreover, bold letters are used to denote vectors of parameters, such as $\mathbf{T} = (T_1, \dots, T_N)$, $\mathbf{f} = (f_1, \dots, f_N)$, and $\mathbf{C} = (C_1, \dots, C_N)$. Note that the task set utilization U can be written by the dot product $\mathbf{C} \cdot \mathbf{f}$. Finally, the response time (Joseph and Pandya 1986; Audsley et al. 1993) of task τ_i is denoted by R_i.

Tasks are scheduled by a fixed priority scheduler (not necessarily Rate Monotonic) and they are ordered by decreasing priorities, meaning that τ_1 has the highest priority, and τ_N the lowest one. The set of the first i higher priority tasks is denoted by $\mathcal{T}_i = \{\tau_1, \dots, \tau_i\}$, and the corresponding vectors of computation times, periods, and rates are denoted by $\mathbf{C}_i$, $\mathbf{T}_i$, and $\mathbf{f}_i$, respectively. Please notice that, while C_i is the computation time of τ_i, $\mathbf{C}_i = (C_1, C_2, \dots, C_i)$ is the vector of the computation times of the i highest priority tasks. Finally, although the proposed approach can be applied for arbitrary deadlines by extending the analysis to the future jobs within the busy period (Lehoczky 1990), in this paper we consider the case of deadlines less than or equal to periods, meaning that $0 < \delta_i \leq 1$ to reduce the time complexity of the analysis.

Each periodic task τ_i is considered as an infinite sequence of jobs $\tau_{i,j}$, where each job $\tau_{i,j}$ is activated at time $a_{i,j} = (j-1)T_i$ and must be completed by its absolute deadline $d_{i,j} = a_{i,j} + D_i$.

Task execution is modeled by the *linear compute time model* introduced by Vestal (1994) because of its generality and its aptness at representing real-world applications. Within this framework, the computation time of each task consists of a set of software modules (or procedures) that are called by the task to perform its computations. If there is a set of M software modules in the system $\mathcal{M} = \{\mu_1, \mu_2, \dots, \mu_M\}$, and each module μ_j is characterized by a worst-case computation time m_j, the execution time of each task can be expressed as a linear combination of the m_j values:

$$\begin{bmatrix} C_1 \\ C_2 \\ \vdots \\ C_N \end{bmatrix} = \begin{bmatrix} a_{1,1} & a_{1,2} & \cdots & a_{1,M} \\ a_{2,1} & a_{2,2} & \cdots & a_{2,M} \\ \vdots & & \ddots & \vdots \\ a_{N,1} & a_{N,2} & \cdots & a_{N,M} \end{bmatrix} \begin{bmatrix} m_1 \\ m_2 \\ \vdots \\ m_M \end{bmatrix}$$

or, in a more compact expression,

$$\mathbf{C} = \mathbf{A} \cdot \mathbf{m} \tag{1}$$

where $\mathbf{A}$ is the $N \times M$ array of the $a_{i,j}$ coefficients. Notice that the computation model expressed in (1) is a generalization of the classical "C_i's" model, obtainable by simply setting $\mathbf{A}$ equal to the identity. Each $a_{i,j}$ can be conveniently thought as the product of an integer term $x_{i,j}$, expressing the number of times module m_j is invoked by τ_i, and a real part $\alpha_{i,j}$ expressing other architecture-related parameters. For example, the model is suited for power-aware speed scaling if the $\alpha_{i,j}$ terms in a given row represent the reciprocal of the processor speed selected for τ_i.

Finally, whenever some variation is applied to the task set $\mathcal{T}$, the altered task set will be denoted by $\mathcal{T}'$. For example, if computation times are modified by an amount $\Delta\mathbf{C}$, then the new computation times of the modified set $\mathcal{T}'$ are denoted as $\mathbf{C}' = \mathbf{C} + \Delta\mathbf{C}$.

3 Feasibility region

The options available for sensitivity analysis can be better understood in the context of a new representation of the feasibility condition as a region defined in the space of those task attributes $\mathbf{X}$ considered as design variables. More formally, the feasibility region can be defined as follows.

Definition 1 Let $\mathbf{X}$ be the set of design variables for a task set $\mathcal{T}$. Then, the feasibility region in the X-space is the set of values of $\mathbf{X}$ such that $\mathcal{T}$ is feasible.

In this work, the schedulability region will be defined in the space of the worst-case computation times ($\mathbf{X} = \mathbf{C}$), referred to as the C-space, and in the space of the task rates ($\mathbf{X} = \mathbf{f}$), referred to as the f-space.

This paper provides theory and algorithms for measuring the distance of a scheduling solution from the boundary of the feasibility region, when a fixed priority scheduler is adopted. Such a distance, expressed in the C-space or in the f-space, is an exact measure of the available slack or, conversely, of the correcting actions that must be taken on the computation times or on the activation rates to make the system schedulable.

For models that can be analyzed by a simple utilization-based test, the feasibility region is defined by the well known bound (Liu and Layland 1973)

$$\mathbf{C} \cdot \mathbf{f} \le U_{\mathsf{lub}}^{\mathsf{A}}$$

where $U_{\mathsf{lub}}^{\mathsf{A}}$ is the least upper bound of the algorithm A considered for scheduling the set of tasks.

As shown in the following, the feasibility region in both the C-space and the f-space is delimited by hyperplanes and the positive quadrant.

3.1 Feasibility region in the C-space

When working in the C-space, we assume that the design variables are only the task computation times, whereas the periods and the deadlines are fixed.

The schedulability condition formulated by Bini and Buttazzo (2004) (originally proposed by Lehoczky et al. (1989)), can be conveniently used to establish a relationship between the task set parameters, from which the feasible values of computation times C_i can be inferred.

Theorem 1 (Bini and Buttazzo 2004) *A periodic task set $\mathcal{T}$ is schedulable under fixed priorities if and only if*

$$\forall i = 1, \dots, N \ \exists t \in \mathsf{schedP}_i \quad C_i + \sum_{j=1}^{i-1} \left\lceil \frac{t}{T_j} \right\rceil C_j \le t \tag{2}$$

where schedP_i *is a set of* scheduling points *defined as* $\mathsf{schedP}_i = \mathcal{P}_{i-1}(D_i)$, *and* $\mathcal{P}_i(t)$ *is defined as follows*

$$\begin{cases} \mathcal{P}_0(t) = \{t\}, \\ \mathcal{P}_i(t) = \mathcal{P}_{i-1}\left(\left\lfloor \frac{t}{T_i} \right\rfloor T_i\right) \cup \mathcal{P}_{i-1}(t). \end{cases} \tag{3}$$

By using the more compact vectorial notation and some logical operators, (2) can be rewritten as

$$\bigwedge_{i=1,\dots,N} \; \bigvee_{t \in \mathsf{schedP}_i} \mathbf{n}_i \cdot \mathbf{C}_i \le t \tag{4}$$

where $\mathbf{n}_i = (\lceil \frac{t}{T_1} \rceil, \lceil \frac{t}{T_2} \rceil, \dots, \lceil \frac{t}{T_{i-1}} \rceil, 1)$.

Equation (4) represents the feasibility region in the C-space. Notice that the computation times $\mathbf{C}$ are subject to a combination of linear constraints.

An example of the feasibility region for two tasks in the C-space is shown later in Sect. 4, where we will show how the feasibility region can be exploited to perform sensitivity analysis.

3.2 Feasibility region in the f-space

When reasoning in the f-space the design variables are the task frequencies $\mathbf{f}$, whereas the computation times $\mathbf{C}$ and the normalized deadlines are fixed.

The formal definition of the feasibility region in the f-space can be conveniently derived from the schedulability condition provided by Seto et al. (1998) and later explicitly formalized in (Bini and Di Natale 2005).

Theorem 2 (Seto et al. 1998; Bini and Di Natale 2005) *A periodic task set* $\mathcal{T}$ *is schedulable under Fixed Priorities if and only if*

$$\forall i = 1, \dots, N \;\exists \mathbf{n}_{i-1} \in \mathbb{N}^{i-1} \tag{5}$$

such that:

$$\begin{cases} 0 \le f_i \le \dfrac{\delta_i}{C_i + \mathbf{n}_{i-1} \cdot \mathbf{C}_{i-1}}, \\ \dfrac{\mathbf{n}_{i-1} - 1}{C_i + \mathbf{n}_{i-1} \cdot \mathbf{C}_{i-1}} \le \mathbf{f}_{i-1} \le \dfrac{\mathbf{n}_{i-1}}{C_i + \mathbf{n}_{i-1} \cdot \mathbf{C}_{i-1}} \end{cases} \tag{6}$$

where $\mathbb{N}^{i-1}$ *is the set of* $i-1$ *tuples of positive integers.*

For each vector $\mathbf{n}_{i-1}$, each task activation frequency f_j has an upper and lower bound defining an N-dimensional parallelepiped. Hence, the feasibility region in the f-space is given by the union of the parallelepipeds corresponding to all the possible tuples $\mathbf{n}_{i-1}$.

To better understand the insight behind Theorem 2, the feasibility region in the f-space for a set of two tasks is illustrated in Fig. 2. The graph is obtained for $C_1 = 1$

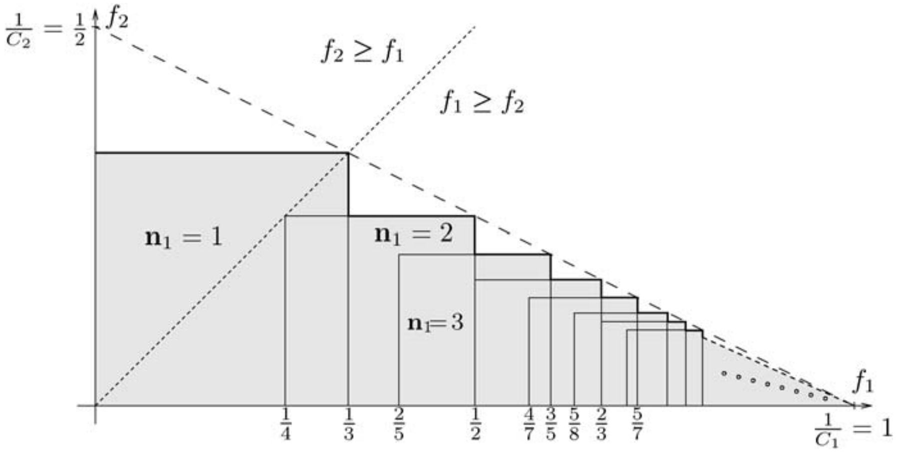

Fig. 2 Feasibility region in the f-space

and $C_2 = 2$ and deadlines equal to periods. For two tasks, the region is the union of rectangles given by (6) for each possible value of $\mathbf{n}_1$.

The EDF bound is also plotted in the figure (dashed line). In the f-space it is a linear constraint, which is, as expected, larger than the fixed priority region. Note that for harmonic frequencies (meaning that $f_1 = k\, f_2$ for some integer k) the boundary of the f-space coincides with the EDF boundary, as well known from the work by Kuo and Mok (1991).

The next section explains how the sensitivity analysis can be performed based on the feasibility region.

4 Sensitivity analysis in the C-space

The knowledge of the range of admissible variations of the computation times can be very useful in a number of practical situations. For example, the designer can often select among different implementations for the same functionality, corresponding to different computation times.

From the model expressed in (1), it follows that altering the module implementation (so affecting the m_j value) results in a linear scaling of the computation times. More formally, the modified computation times $\mathbf{C}'$ can be expressed as

$$\mathbf{C}' = \mathbf{C} + \lambda\, \mathbf{d} \tag{7}$$

where $\mathbf{d}$ is a non-negative vector setting the *direction of action* and λ measures the amount of variation imposed on the original task set. We expect $\lambda \geq 0$ if the original task set was schedulable, and $\lambda < 0$ if the task set was not schedulable. Moreover, since the only action examined in this section is a modification of the computation times, we assume $\mathbf{T}' = \mathbf{T}$, meaning that the periods are not changed.

A significant case occurs when $\mathbf{d} = \mathbf{C}$, which means that the direction of the action is equal to the computation times, which is equivalent to scaling all the computation times by the same factor. Another interesting action is to change only the

ith computation time, leaving the others unchanged. It corresponds to the direction $\mathbf{d} = (0, \dots, 0, 1, 0, \dots, 0)$ of all zeros and a 1 in the ith position.

Once the direction $\mathbf{d}$ is defined by the designer, the amount λ of admissible change can be found by applying the necessary and sufficient schedulability condition of (4) to the modified task set $\mathcal{T}'$. If we denote the first i elements of $\mathbf{d}$ by $\mathbf{d}_i$, then $\mathcal{T}'$ is schedulable if and only if

$$\begin{aligned}
&\bigwedge_{i=1,\dots,N} \ \bigvee_{t\in\mathsf{schedP}_i} \ \mathbf{n}_i \cdot \mathbf{C}'_i \le t, \\
&\bigwedge_{i=1,\dots,N} \ \bigvee_{t\in\mathsf{schedP}_i} \ \mathbf{n}_i \cdot (\mathbf{C}_i + \lambda\,\mathbf{d}_i) \le t, \\
&\bigwedge_{i=1,\dots,N} \ \bigvee_{t\in\mathsf{schedP}_i} \ \lambda \le \frac{t - \mathbf{n}_i \cdot \mathbf{C}_i}{\mathbf{n}_i \cdot \mathbf{d}_i}, \\
&\lambda \le \min_{i=1,\dots,N} \ \max_{t\in\mathsf{schedP}_i} \ \frac{t - \mathbf{n}_i \cdot \mathbf{C}_i}{\mathbf{n}_i \cdot \mathbf{d}_i} \triangleq \lambda^{\max}
\end{aligned} \tag{8}$$

which provides the amount of admissible linear alteration along the generic direction $\mathbf{d}$. As expected, if the original task set $\mathcal{T}$ is schedulable, then $\lambda_{\max} \ge 0$; whereas, if $\mathcal{T}$ is not schedulable, there exists an i such that for all $t \in \mathsf{schedP}_i$ the numerator is always negative and hence $\lambda_{\max}$ is negative as well. Relevant applications of (8) are discussed in the following.

Distance along the axes Given $\mathcal{T}$, it is interesting to find the amount of computation time variation that can be added (when $\mathcal{T}$ is schedulable) or subtracted ($\mathcal{T}$ not schedulable) to the computation time of task τ_k to guarantee the schedulability of the task set. These values can be obtained from (8) after the definition of the direction $\mathbf{d}$. Suppose we are changing the computation time C_k of task τ_k. This means that the direction of action is $\mathbf{d} = (0, \dots, 0, 1, 0, \dots, 0)$, where the only 1 is at the kth component in the vector. In order to relate this value to the task τ_k we refer to it as ΔC_k.

When searching for ΔC_k, we assume that all the higher priority tasks in $\mathcal{T}_{k-1}$ are schedulable. In fact, they are not affected by any variation of task τ_k, hence no solution could be found for ΔC_k if $\mathcal{T}'_{k-1}$ were unschedulable. This means that the schedulability condition must be guaranteed only for the tasks from the kth to the Nth.

From (8) we have

$$\Delta C_k^{\max} = \min_{i=k,\dots,N} \ \max_{t\in\mathsf{schedP}_i} \ \frac{t - \mathbf{n}_i \cdot \mathbf{C}_i}{\mathbf{n}_i \cdot \underbrace{(0, \dots, 0, 1, 0, \dots, 0)}_{i \text{ elements}}}$$

and, computing the dot product in the denominator, we have

$$\Delta C_k^{\max} = \min_{i=k,\dots,N} \ \max_{t\in\mathsf{schedP}_i} \ \frac{t - \mathbf{n}_i \cdot \mathbf{C}_i}{\lceil t/T_k \rceil}. \tag{9}$$

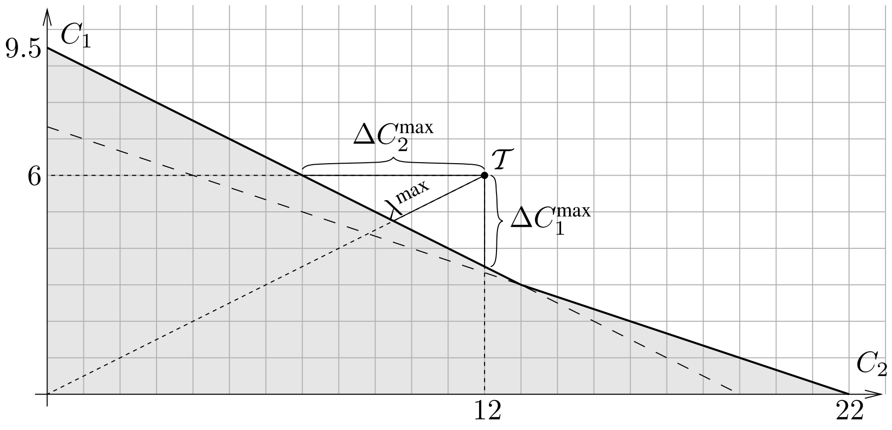

Fig. 3 Sensitivity analysis in the C-space

A representation of the sensitivity analysis in the C-space is shown in Fig. 3, in the case of two tasks when $T_1 = D_1 = 9.5$ and $D_2 = 22$ (the same simple example will be discussed in Sect. 6). The figure shows the geometrical interpretation of the distances to the feasibility region $\Delta C_k^{\max}$, as well as the scaling factor $\lambda^{\max}$ explained in the next section.

Scaling the computation times Suppose that all computation times are scaled proportionally. This situation occurs, for example, in variable speed processors when we want to find the minimum speed which makes the task set schedulable. In this scenario, the direction of action is given by the computation times, meaning that $\mathbf{d} = \mathbf{C}$. Hence, from (8), we obtain

$$\begin{aligned}\lambda^{\max} &= \min_{i=1\ldots n} \max_{t \in \mathsf{schedP}_i} \frac{t - \mathbf{n}_i \cdot \mathbf{C}_i}{\mathbf{n}_i \cdot \mathbf{C}_i}, \\ \lambda^{\max} &= \min_{i=1\ldots n} \max_{t \in \mathsf{schedP}_i} \frac{t}{\mathbf{n}_i \cdot \mathbf{C}_i} - 1.\end{aligned} \tag{10}$$

Notice again that a positive value of $\lambda^{\max}$ corresponds to an initially schedulable task set. Conversely, if $\lambda^{\max}$ is negative, the initial task set $\mathcal{T}$ is not schedulable, and the computation times must be decreased to achieve schedulability.

In the following, we show how an elastic compression (Buttazzo et al. 2002) can be used with the exact analysis of fixed priority systems, by a proper definition of the direction $\mathbf{d}$ in (8).

Linear transformation The elastic task model (Buttazzo et al. 2002), originally developed for a utilization based test, can be generalized to work under an exact condition, thanks to the proposed approach. The basic idea behind this model is that tasks can be viewed as flexible springs with rigidity coefficients and minimum length constraints, whose utilizations can be compressed to comply with a desired workload.

Each task is characterized by an elastic coefficient E_i, which represents the flexibility of the task to vary its utilization for adapting to a new workload condition. For instance, more important tasks could be assigned lower elastic coefficients, so that their rates are changed with greater difficulty during a system adaptation. Although the original model was devised to perform rate adaptation, the utilization can also be adjusted by changing the computation times. In general, the utilization can be tuned both in the C-space and in the f-space, provided that the design variables do not exceed the given boundaries.

To exploit the elastic model in the framework of sensitivity analysis, the direction of action can be simply set as $\mathbf{d} = (E_1, E_2, \dots, E_N)$, so that the new computation times $\mathbf{C}'$ resulting from the modification performed according to the elastic scheme can be derived from (8). Such a definition for the direction directly follows from the meaning of the elastic coefficient, which represents the flexibility of the task to a parameter variation: the higher an elastic coefficient, the higher the variation along that dimension.

Other feasibility problems defined in the literature can also be cast within such a framework. As shown in Sect. 2, changing the worst case execution m_j of a module μ_j corresponds to changes in the computation times given by the jth column $\mathbf{a}_j$ of the matrix $\mathbf{A}$. Then, setting $\mathbf{d} = \mathbf{a}_j$, from (8) we can find the admissible alteration of the module computation time m_j. In this case, the *linear compute task model* (Vestal 1994) is fully captured by properly setting the direction of action.

5 Sensitivity analysis in the f-space

When performing the sensitivity analysis in the f-space, the computation times $\mathbf{C}$ and the normalized deadlines δ_i are fixed.

The solutions of the sensitivity analysis problem in the C-space exploited the fact that the schedulability condition could be expressed by combining linear inequalities, as the ones in (4). Unfortunately, this is not the case in the f-space (or T-space), where the desired sensitivity values cannot be computed in a closed form as in (8). In fact, the feasibility region in the f-space is delimited by an infinite number of hyperplanes, thus an ad-hoc approach is required for each specific study.

Distance along the axes As done in the C-space, we first consider the sensitivity analysis when only one period can change. Let τ_k be the task whose period T_k is going to be modified. Hence, in the modified task set $\mathcal{T}'$ we have $T_i' = T_i$ for all $i \neq k$. Notice that, if the original task set $\mathcal{T}$ is schedulable, an increase of T_k will clearly preserve schedulability. Similarly, if $\mathcal{T}$ is not schedulable, reducing T_k will keep $\mathcal{T}'$ still unschedulable. Let $T_k^{(i)}$ denote *the minimum period of* τ_k *such that* τ_i *is schedulable*. Because of the priority ordering, period $T_k^{(i)}$ is meaningful only when $i \geq k$. In fact, if $i < k$ then any variation of the lower priority task τ_k does not affect the schedulability of the higher priority task τ_i. For this reason, $T_k^{(i)}$ is computed under the assumption that the task subset $\mathcal{T}_{k-1}$ is schedulable.

If we compute the periods $T_k^{(i)}$ for all $i \geq k$, then it is easy to compute *the minimum period* $T_k^{\min}$ *of* τ_k *such that* $\mathcal{T}'$ *is feasible* (see Fig. 4, where the maximum frequencies $1/T_k^{\min}$ are represented instead). In fact we have

$$T_k^{\min} = \max_{i \geq k} T_k^{(i)}, \tag{11}$$

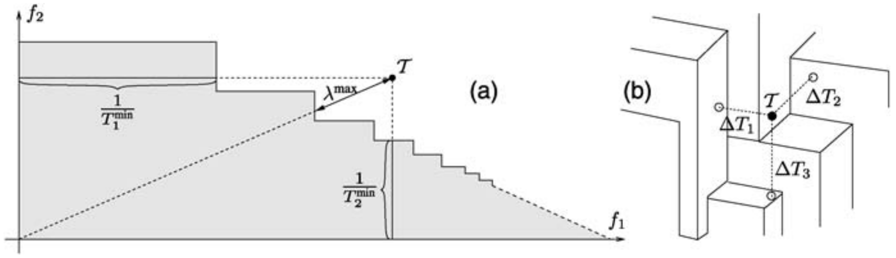

Fig. 4 Sensitivity analysis in the f-space: when $N = 2$ (**a**) and $N = 3$ (**b**)

because $T_k^{\min}$ must be large enough to guarantee the schedulability of all the tasks from τ_k to τ_N. Since the minimum period $T_k^{\min}$ ensures the schedulability of all the tasks from τ_k to τ_N, and the tasks from τ_1 to τ_{k-1} are schedulable by hypothesis, then we have

$$\mathcal{T} \text{ is schedulable} \quad \Leftrightarrow \quad T_k^{\min} \le T_k,$$
$$\mathcal{T} \text{ is not schedulable} \quad \Leftrightarrow \quad T_k^{\min} > T_k.$$

Now we address the problem of finding the value of $T_k^{(i)}$, when $i \ge k$. If $i = k$ the computation of $T_k^{(k)}$ is simple. Let R_k be the response time of τ_k. Then τ_k is schedulable if and only if:

$$R_k \le D_k \quad \Leftrightarrow \quad R_k \le \delta_k T_k \quad \Leftrightarrow \quad T_k \ge \frac{R_k}{\delta_k} \quad \Leftrightarrow \quad T_k^{(k)} = \frac{R_k}{\delta_k}. \tag{12}$$

The computation $T_k^{(i)}$ requires some extra effort. For this purpose we need to introduce the following definition, which is quite classical in FP analysis.

Definition 2 Let 0 be the critical instant at which all tasks are simultaneously activated. Given the subset of tasks $\mathcal{S}$, we define *level-$\mathcal{S}$ idle time at time t* the minimum amount of time such that no task in $\mathcal{S}$ is executed in $[0, t]$. We denote this amount by $Y(\mathcal{S}, t)$.

Although the interpretation of $Y(\mathcal{S}, t)$ may seem quite cryptic, this amount is tightly related to the notion of level-i busy period introduced by Lehoczky et al. (1989). For example, it is possible to assert that $Y(\{\tau_1\}, T_1) = T_1 - C_1$, which represents the idle left by the first task at time T_1. Another trivial property of $Y(\mathcal{S}, t)$ is that $Y(\varnothing, t) = t$, where $\varnothing$ denotes the empty set.

In order to find the value of $T_k^{(i)}$ it is necessary to evaluate $Y(\mathcal{T}_i \setminus \{\tau_k\}, D_i)$. In fact, this is the amount of idle before the deadline of τ_i which can accommodate jobs of τ_k. The smallest possible period T_k corresponds to the highest possible number of τ_k jobs. If $Y(\mathcal{T}_i \setminus \{\tau_k\}, D_i) < C_k$, then the execution of one single job of τ_k will push the execution of τ_i beyond its deadline D_i.

Notice that, thanks to previous results (Bini and Buttazzo 2004), it is possible to determine the exact amount of $Y(\mathcal{T}_i \setminus \{\tau_k\}, D_i)$ with the complexity of a schedulability test. In fact we have:

$$Y(\mathcal{T}_i \setminus \{\tau_k\}, D_i) = \max_{t \in \mathsf{schedP}_i^*} t - C_i - \sum_{\substack{j=i \\ j \neq k}}^{i-1} \left\lceil \frac{t}{T_j} \right\rceil C_j. \tag{13}$$

Note that in the previous expression the set of schedulability points schedP_i^* must be computed from (3) by considering the task set $\mathcal{T}_i$ without τ_k.

The maximum number of jobs of τ_k, which can interfere in the response time of the task τ_i is determined by the following lemma.

Lemma 1 *The maximum number $n_k^{(i)}$ of τ_k jobs that can interfere on τ_i, such that τ_i is schedulable, is equal to*

$$n_k^{(i)} = \left\lfloor \frac{Y(\mathcal{T}_i \setminus \{\tau_k\}, D_i)}{C_k} \right\rfloor. \tag{14}$$

Proof If $n_k^{(i)}$ jobs of τ_k interfere with τ_i, then the amount of level-$\mathcal{T}_i$ idle time at D_i is

$$Y(\mathcal{T}_i, D_i) = Y(\mathcal{T}_i \setminus \{\tau_k\}, D_i) - \left\lfloor \frac{Y(\mathcal{T}_i \setminus \{\tau_k\}, D_i)}{C_k} \right\rfloor C_k$$

which is greater than or equal to 0. Hence, τ_i is schedulable because it admits some non negative idle time in $[0, D_i]$.

If the number of interfering jobs is $n_k^{(i)} + 1$, then at the end of the $n_k^{(i)} + 1$ job of τ_k it will happen simultaneously that:

- the task τ_i has not finished yet, because of the interference of $n_k^{(i)} + 1$ jobs of τ_k;
- no more idle time is available in $[0, D_i]$, because τ_k has consumed more than $Y(\mathcal{T}_i \setminus \{\tau_k\}, D_i)$.

Hence, τ_i will miss its deadline and the maximum number of τ_k jobs interfering with τ_i is $n_k^{(i)}$, as given by (14). □

Once the maximum number of jobs $n_k^{(i)}$ is found, the minimum period $T_k^{(i)}$ can be determined as follows. We first compute the response time R_i from the following fixed point equation:

$$R_i = C_i + n_k^{(i)} C_k + \sum_{\substack{j=1 \\ j \neq k}}^{i-1} \left\lceil \frac{R_i}{T_j} \right\rceil C_j \tag{15}$$

which assumes that exactly $n_k^{(i)}$ interferences of τ_k occur. The period $T_k^{(i)}$ is the minimum period such that

$$n_k^{(i)} = \left\lceil \frac{R_i}{T_k} \right\rceil \quad \Rightarrow \quad T_k^{(i)} = \frac{R_i}{n_k^{(i)}}. \tag{16}$$

Equation (12) allows us to compute the value of $T_k^{(k)}$ that can schedule the task τ_k. Moreover, the period of τ_k can be computed by (16), which guarantees the schedulability of all the lower priority tasks τ_i. Hence, from (11), we have that the minimum period which guarantees the schedulability of the entire task set is

$$T_k^{\min} = \max\left\{\frac{R_k}{\delta_k}, \frac{R_{k+1}}{n_k^{(k+1)}}, \dots, \frac{R_N}{n_k^{(N)}}\right\}. \tag{17}$$

Scaling the rates Compared with the algorithm for finding the distance to the feasibility region along each component of **f**, finding the maximum value of λ such that the frequencies $\lambda\,\mathbf{f}$ result in an FP schedulable task set is relatively easy.

A previous result (please refer to Bini and Di Natale 2005 for its proof) establishes a relationship between the scaling operations in the C-space and in the f-space.

Theorem 3 (Bini and Di Natale 2005) *Given a task set* $\mathcal{T} = \{\tau_1, \tau_2, \dots, \tau_N\}$, *let* $\mathcal{T}^C$ *be the task set where task computation times are scaled by* λ *and let* $\mathcal{T}^T$ *be the task set where task periods are scaled by* $1/\lambda$. *Formally*:

$$\mathcal{T}^C = \{(\lambda C_1, T_1), \dots, (\lambda C_n, T_n)\},$$
$$\mathcal{T}^T = \{(C_1, T_1/\lambda), \dots, (C_n, T_n/\lambda)\}.$$

Then, task set $\mathcal{T}^C$ *is schedulable if and only if task set* $\mathcal{T}^T$ *is schedulable.*

From Theorem 3, we have that the scaling factor in the C-space is exactly the same as in the f-space. Hence, we can apply the same equation (10), used for computing the maximum scaling factor $\lambda^{\max}$ in the C-space, to find exactly the same scaling factor that allows finding the solution onto the boundary of the feasibility region in the f-space.

Linear transformation In the f-space, a non-schedulable set is represented by a point outside the feasibility region, and a set of constants (for example, representing the elastic coefficients) defines a direction vector in the N-dimensional space showing the preferred direction for reducing the rates until the set becomes schedulable. The compression of the tasks can be viewed in the f-space as a trajectory from an initial position P_0 (corresponding to the initial rate configuration) along a vector characterized by the coefficients of all tasks.

Figure 5 illustrates an example where two elastic tasks are compressed (i.e., their rates are reduced) until the task set is schedulable. Notice that during compression, the rate of task τ_1 reaches its lower bound, so the compression continues by reducing the rate of τ_2 only.

In general, computing the sensitivity along a generic linear direction **d** in the f-space is very challenging and, at present, no efficient solutions have been proposed. However the problem can still be approached by setting up a binary search procedure between two values $\lambda_A, \lambda_B > 0$ defining two task sets, $\mathcal{T}_A$ and $\mathcal{T}_B$, where $\mathcal{T}_A$ is outside and $\mathcal{T}_B$ is inside the feasibility region ($\lambda_A < \lambda_B$), as shown in Fig. 6.

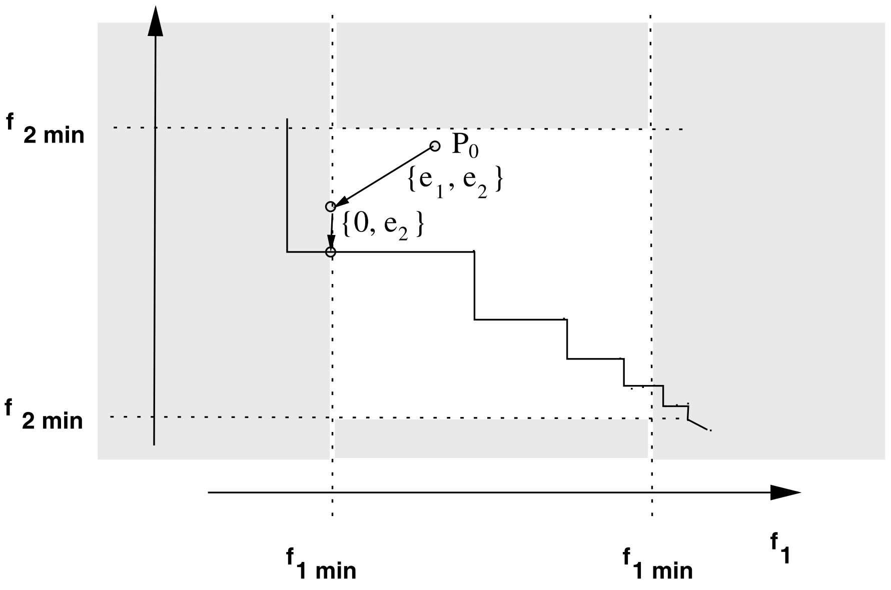

Fig. 5 Modification of the tasks' rates (periods) according to the elastic model

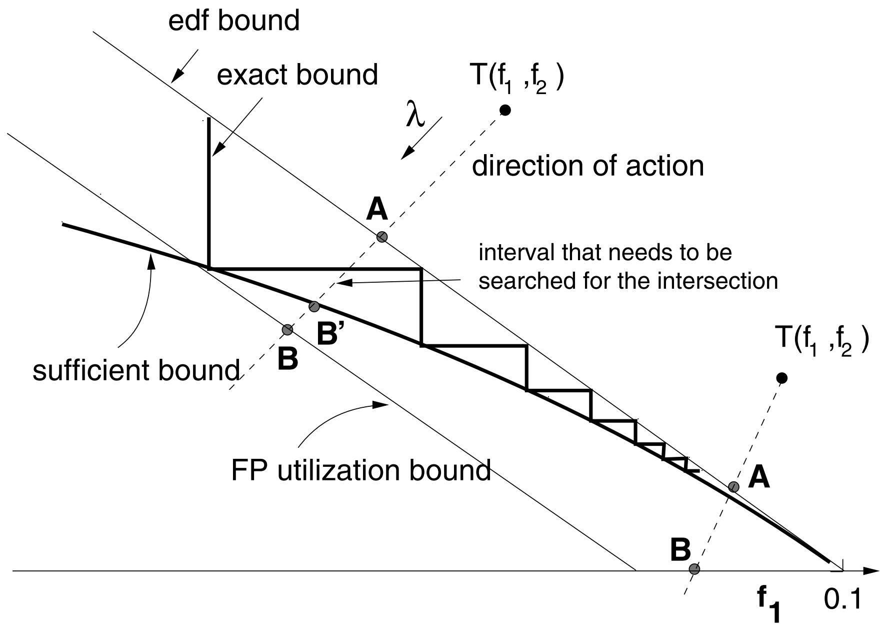

Fig. 6 Finding the solution $\mathcal{T}'$ according to the elastic model

Table 1 Task set parameters

i	C_i	T_i	D_i	δ_i	U_i	schedP_i
1	6	9.5	9.5	1.000	0.632	{9.5}
2	12	24	22	0.917	0.500	{22, 19}

6 An example

To explain all the possibilities offered by the sensitivity analysis, we propose a simple illustrative example of a two tasks set. The task parameters are reported in Table 1, which also shows the task utilizations U_i and the set of scheduling points schedP_i for each task.

As the reader can quickly realize, the task set is not schedulable, because the total utilization exceeds 1. We are now going to evaluate all the possible actions which can make the task set feasible. First, we consider modifying the individual components of the computation times vector, previously denoted by $\Delta C_k^t\max$. Since the task set is not schedulable, we expect that $\Delta C_k^{\max} < 0$, meaning that only a reduction of the computation time can bring the task set in the feasibility region. From (9), we have

$$\begin{aligned}
\Delta C_1^{\max} &= \min_{i=1,2} \max_{t \in \mathsf{schedP}_i} \frac{t - \mathbf{n}_i \cdot \mathbf{C}_i}{\lceil t/T_1 \rceil}, \\
\Delta C_1^{\max} &= \min\left\{(9.5-6), \max_{t \in \mathsf{schedP}_2} \frac{t - \mathbf{n}_2 \cdot \mathbf{C}}{\lceil t/T_1 \rceil}\right\}, \\
\Delta C_1^{\max} &= \min\left\{3.5, \max\left\{\frac{22-30}{3}, \frac{19-24}{2}\right\}\right\}, \\
\Delta C_1^{\max} &= \min\left\{3.5, \max\left\{-\frac{8}{3}, -\frac{5}{2}\right\}\right\}, \\
\Delta C_1^{\max} &= -2.5,
\end{aligned} \tag{18}$$

and

$$\begin{aligned}
\Delta C_2^{\max} &= \max_{t \in \mathsf{schedP}_2} \frac{t - \mathbf{n}_2 \cdot \mathbf{C}}{\lceil t/T_2 \rceil}, \\
\Delta C_2^{\max} &= \max\{(22-30), (19-24)\}, \\
\Delta C_2^{\max} &= -5.
\end{aligned} \tag{19}$$

As expected, both $\Delta C_1^{\max}$ and $\Delta C_2^{\max}$ are negative. The two values are depicted in Fig. 3.

The amount of scaling factor $\lambda^{\max}$ of the computation times can also be computed from (10). We have

$$\lambda^{\max} = \min_{i=1,2} \max_{t \in \mathsf{schedP}_i} \frac{t}{\mathbf{n}_i \cdot \mathbf{C}_i} - 1,$$

$$\lambda^{\max} = \max_{t \in \mathsf{schedP}_2} \frac{t}{\mathbf{n}_2 \cdot \mathbf{C}} - 1,$$

$$\lambda^{\max} = \max\left\{\frac{22}{30}, \frac{19}{24}\right\} - 1,$$

$$\lambda^{\max} = -0.20833 \tag{20}$$

which is also represented in Fig. 3.

It is now interesting to compute the variations that can be performed in the f-space to make the task set feasible. First, we evaluate the modification to each task period to reach feasibility. To do that, we compute $T_1^{\min}$ and $T_2^{\min}$ from (17). $T_1^{(1)}$ is equal to $R_1/\delta_1 = C_1 = 6$. In order to compute $T_1^{(2)}$, which is the minimum period of τ_1 that ensures the schedulability of τ_2, we compute $n_1^{(2)}$. The amount of available idle time in $[0, D_2]$ is simply $D_2 - C_2 = 10$. Hence, the maximum number of τ_1 jobs that interfere with τ_2 is $n_1^{(2)} = \lfloor \frac{10}{6} \rfloor = 1$ and the response time of τ_2 is $R_2 = n_1^{(2)} C_1 + C_2 = 18$. Hence, we have that $T_1^{(2)} = \frac{R_2}{n_1^2} = 18$. Finally, by applying (17), we find that

$$T_1^{\min} = \max\left\{\frac{R_1}{\delta_1}, \frac{R_2}{n_1^{(2)}}\right\}, \tag{21}$$

$$T_1^{\min} = \max\{6, 18\} = 18 \tag{22}$$

which is the minimum value of T_1 for which the task set is schedulable. The period of the second task is computed in a similar way and results to be

$$T_2^{\min} = \frac{R_2}{\delta_2} = \frac{36}{0.917} = 39.27. \tag{23}$$

A geometrical representation of the results is shown in Fig. 4.

We conclude this section by illustrating an application of the linear compute task model in the sensitivity analysis as expressed by (1). For this purpose, suppose task computation times are expressed as

$$\begin{bmatrix} C_1 \\ C_2 \end{bmatrix} = \begin{bmatrix} 2 & 2 & 0 \\ 1 & 4 & 3 \end{bmatrix} \begin{bmatrix} m_1 \\ m_2 \\ m_3 \end{bmatrix} \tag{24}$$

where the m_i's are the lengths of software modules, as it is expressed in (1). Suppose $m_1 = 2$, $m_2 = 1$ and $m_3 = 2$. From these values, task computation times become $C_1 = 6$ and $C_2 = 12$, which lead to a non-schedulable task set, as shown before. Now, we evaluate the variation Δm_i to the module length m_i which can make the task set feasible. As explained in Sect. 4, this is possible by setting the direction $\mathbf{d}$ equal to the ith column of the matrix in (24). By setting $\mathbf{d} = (2, 1)$ and computing (8), we find

$$\Delta m_1 = \min_{i=1,2} \max_{t \in \mathsf{schedP}_i} \frac{t - \mathbf{n}_i \cdot \mathbf{C}_i}{\mathbf{n}_i \cdot (2, 1)},$$

$$\Delta m_1 = \max\left\{\frac{22-30}{(3,1)\cdot(2,1)}, \frac{19-24}{(2,1)\cdot(2,1)}\right\}, \tag{25}$$

$$\Delta m_1 = \max\left\{-\frac{8}{7}, -\frac{5}{5}\right\} = -1.$$

This means that feasibility can be achieved by reducing the module length m_1 by one unit of time. Similarly, the length of the other modules m_2 and m_3 can be individually reduced to achieve feasibility. Using the same approach, the resulting amounts of reduction are $\Delta m_2 = -0.625$ and $\Delta m_3 = -5/3$.

7 Experiments

In order to evaluate the complexity of the proposed method with respect to the sensitivity analysis based on bisection (using the response time analysis as the inner formula) (Racu et al. 2005; Punnekkat et al. 1997), three sets of experiments have been performed. In the first set, the periods of the tasks have been randomly selected from the set $\{1, 2, 5, 10, 100, 200, 500, 1000\}$ representing a low rate and a high rate cluster. The second set consists of three pseudo-harmonic subgroups, with possible periods $\{1, 2, 5, 10\}$, $\{50, 100, 250, 500\}$ and $\{1000, 2000, 5000, 10\,000\}$, respectively. The third set is constructed from the specifications of an automotive application.

In the first two experiments, a number N of tasks (N between 45 and 400) with a total utilization $U = 1.0$ have been generated in such a way that the computation times are uniformly distributed (Bini and Buttazzo 2005). For each set and for each value of N, two sensitivity analysis algorithms have been tried to find the reduction of the computation time for each task that would make the set schedulable (if possible). The first algorithm is the bisection procedure, the other is the procedure presented in this paper. Similarly, for each set and each value of N, the sensitivity analysis on the periods has been evaluated for each task in the set to make the system feasible according to the two competing strategies. When computing the response times inside the bisection procedure, we adopted the improved algorithm proposed by Sjödin and Hansson (1998) to speed up the computation.

Our method resulted in much shorter times for all the experimental sets, both in the domain of the computation times and in the domain of the periods. Figure 7 shows the execution times for the first set when computing the sensitivity in the domain of the computation times. Figure 8 shows the results for the computation of the sensitivity in the domain of the periods. The higher efficiency of the proposed method can easily be explained by recalling that the sensitivity analysis methods presented in this paper leverage the definition of the points schedP_i and require one processing step for any such point in the set. In the case of groups of tasks with harmonic periods, the most interesting for most practical purposes, the number of points in the set schedP_i is much smaller, hence, the better performance of the method. The advantage of the proposed approach for the sensitivity analysis is also demonstrated for the second and third sets of experiments. For the second set, Fig. 9 shows the execution times for the sensitivity in the domain of the computation times and Fig. 10 shows the results for the periods domain.

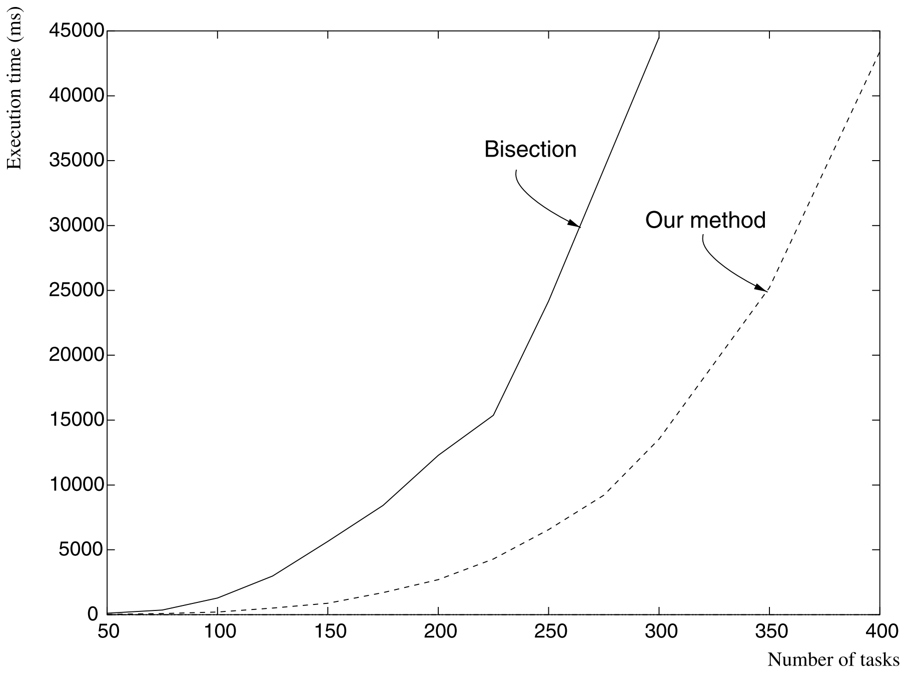

Fig. 7 Execution times required for the sensitivity analysis of computation times (first experiment)

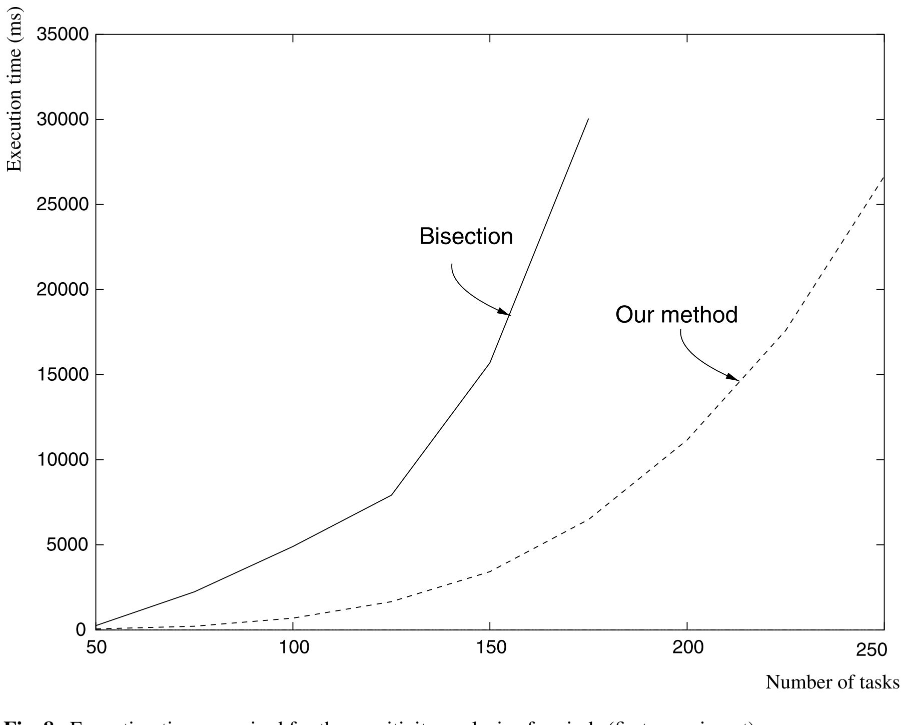

Fig. 8 Execution times required for the sensitivity analysis of periods (first experiment)

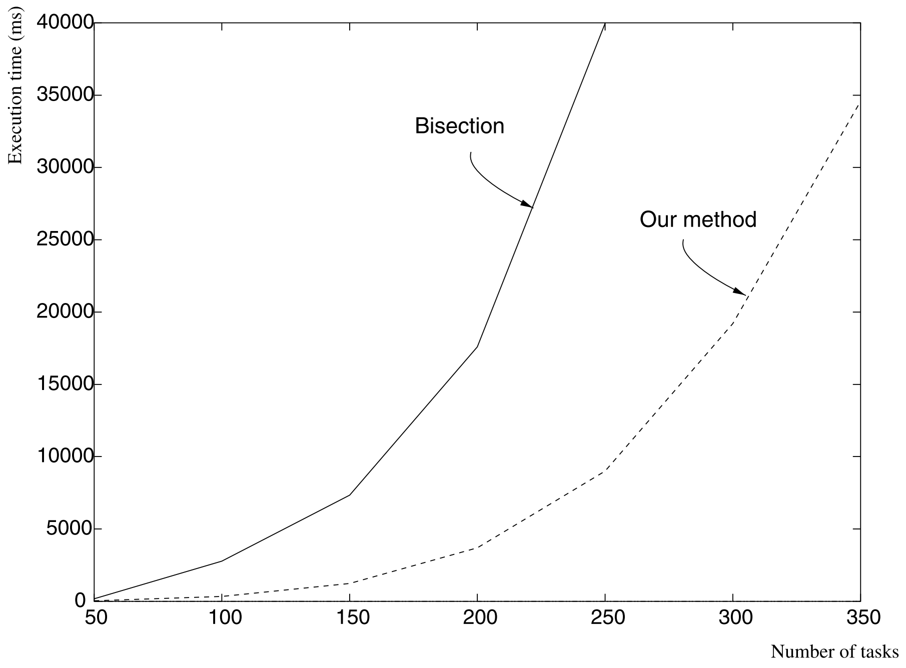

Fig. 9 Execution times required for the sensitivity analysis of computation times (second experiment)

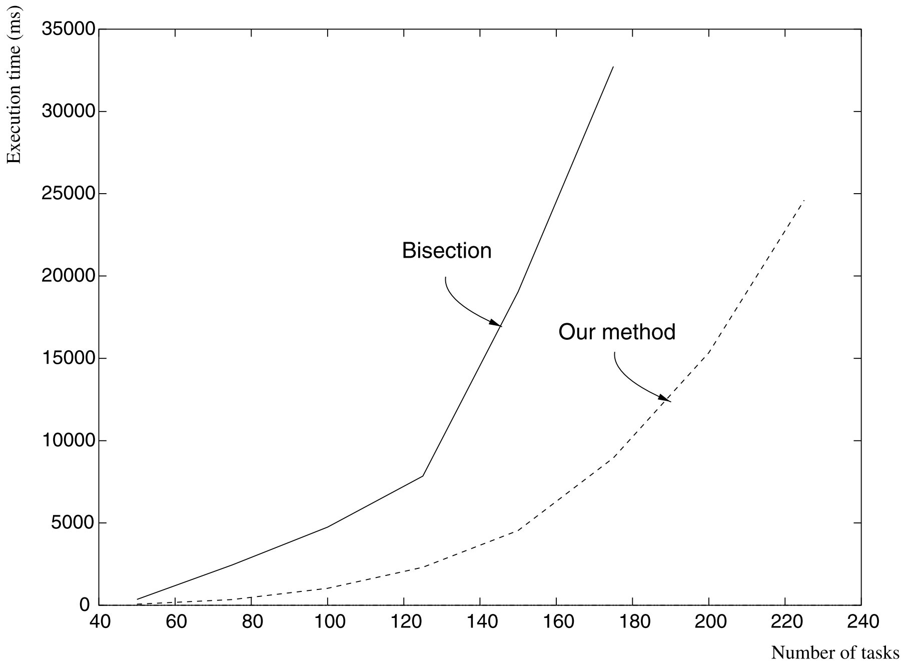

Fig. 10 Execution times required for the sensitivity analysis of periods (second experiment)

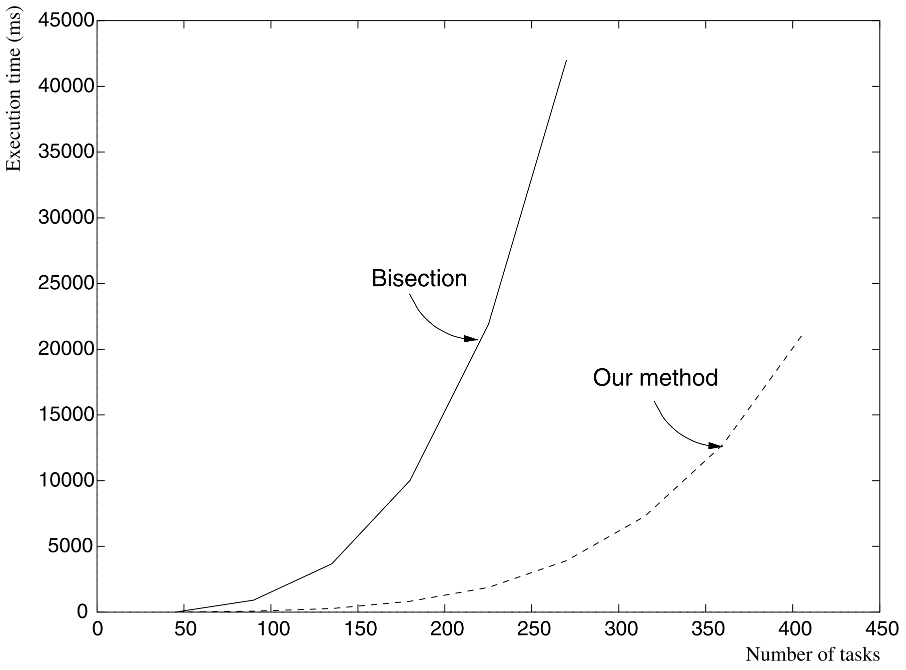

Fig. 11 Execution times required for the sensitivity analysis of computation times (third experiment)

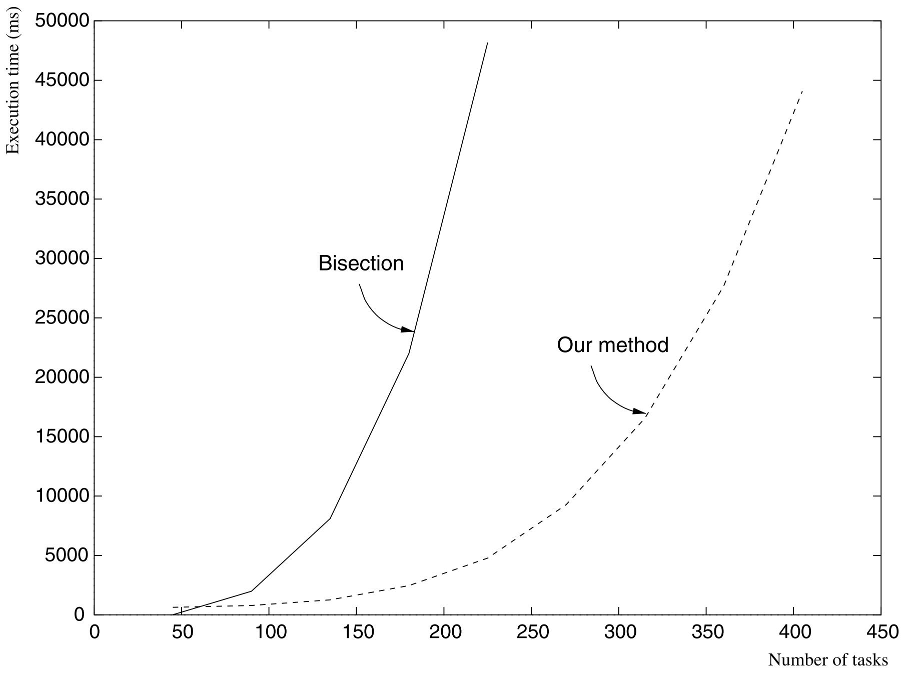

Fig. 12 Execution times required for the sensitivity analysis of periods (third experiment)

The task set of the third experiment has been derived from an distributed automotive application containing 90 tasks with execution periods ranging from 5 to 1000 milliseconds and belonging to two harmonic groups. The computation times have been scaled to obtain a total utilization $U = 1.0$. The results illustrated in Figs. 11 and 12 show a more clear advantage for the proposed method with respect to bisection.

8 Support for design cycles

The sensitivity analysis techniques discussed in this paper have been implemented in the RT-Druid toolset (Evidence s.r.l. 2004): a design and analysis tool developed for supporting the timing evaluation against uncertainties in the development cycle of embedded real-time applications. RT-Druid has been developed as a result of a cooperation with Magneti Marelli Powertrain and it is currently used to validate the scheduling properties of automotive real-time applications.

The tool is adopted in the context of a development process where software engineers map the functions/components developed in the functional stage (typically as Simulink or Ascet diagrams) into real-time threads, select the scheduler and the resource managers by exploiting the services of a Real-Time Operating System (RTOS), and ultimately perform schedulability and sensitivity analysis of the timing requirements upon the target (HW) architecture.

The variations of the computation times compatible with the schedulability constraints are used in the development of new products to avoid excessive iterations between mapping and schedulability analysis, when only imprecise specifications are available.

When extending the functionality of an existing product, sensitivity information is exploited by project managers to estimate, very early in the design flow, whether such an extension might have critical timing impacts, drastically reducing the risk of adopting a variation of the design.

The RT-Druid design environment has been implemented in Java, and it is integrated (as a set of additional plug-in modules) into the Eclipse open development framework. The entire design model is formally defined and represented by an XML schema, which defines the elements of the functional and architecture level design, the mapping relationships, the annotations adding timing attributes to the design objects and the schedulability-related information.

There are many possible ways in which the current work could be extended. The first immediate next step could be to allow fast sensitivity evaluation in the case of tasks sharing resources with predictable worst-case blocking times, as in the case of resources protected by priority ceiling semaphores. However, the most relevant extension for practical purposes would be to target distributed real-time applications and the case of deadlines larger than periods. In this case, unfortunately, the formulation of the feasibility region changes and the results in (Bini and Buttazzo 2004) are not valid and cannot be exploited for faster results with respect to bisection.

9 Conclusions

In this paper we presented a theoretical approach for performing sensitivity analysis of real-time systems consisting of a set of periodic tasks. The proposed method allows a designer not only to verify the feasibility of an application, but also to decide the specific actions to be done on the design variables to reach feasibility when the task set is not schedulable, or to improve resource usage when the task set is schedulable.

The analysis has been presented both in the C-space, where the design variables are the computation times, and in the f-space, where the design variables are the task rates. In both cases, the method allows computing the exact amount each variable can be varied to keep the task set in the feasibility region. We showed how the proposed framework can be conveniently adopted to generalize overload management methods, as the elastic scheduling approach (Buttazzo et al. 2002), which can be effectively extended to work with the exact analysis of fixed priority systems.

Simulation experiments showed that our approach is much more efficient that a classical bisection method, both in the domain of computation times and in that of periods. Finally, we presented an example illustrating how sensitivity analysis can be fruitfully integrated in a typical design cycle. We believe this approach can reduce the distance between the theory of feasibility analysis and the practice of real-time systems design.

References

Audsley NC, Burns A, Richardson M, Tindell KW, Wellings AJ (1993) Applying new scheduling theory to static priority pre-emptive scheduling. Softw Eng J 8(5):284–292

Baruah SK (2003) Dynamic- and static-priority scheduling of recurring real-time tasks. Real-Time Syst 24(1):93–128

Baruah SK, Chen D, Gorinsky S, Mok AK (1999) Generalized multiframe tasks. Real-Time Syst 17(1):5–22

Bini E, Buttazzo GC (2004) Schedulability analysis of periodic fixed priority systems. IEEE Trans Comput 53(11):1462–1473

Bini E, Buttazzo GC (2005) Measuring the performance of schedulability tests. Real-Time Syst 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, pp 399–409

Burns A, Bernat G, Broster I (2003) A probabilistic framework for schedulability analysis. In: proceedings of the EMSOFT, Philadelphia, PA, pp 1–15

Buttazzo GC, Lipari G, Caccamo M, Abeni L (2002) Elastic scheduling for flexible workload management. IEEE Trans Comput 51(3):289–302

Buttazzo GC, Velasco M, Martí P, Fohler G (2004) Managing quality-of-control performance under overload conditions. In: Proceedings of the 16th euromicro conference on real-time systems, Catania, Italy, pp 53–60

Evidence SRL (2004) RT-druid. Available at http://www.evidence.eu.com/RT-Druid.asp

Hamann A, Henia R, Jerzak M, Racu R, Richter K, Ernst R (2004) SymTA/S symbolic timing analysis for systems. Available at http://www.symta.org

Joseph M, Pandya PK (1986) Finding response times in a real-time system. Comput J 29(5):390–395

Kuo TW, Mok AK (1991) Load adjustment in adaptive real-time systems. In: Proceedings of the 12th IEEE real-time systems symposium, San Antonio, TX, pp 160–170

Lehoczky JP (1990) Fixed priority scheduling of periodic task sets with arbitrary deadline. In: Proceedings of the 11th IEEE real-time systems symposium, Lake Buena Vista, FL, pp 201–209

Lehoczky JP, Sha L, Ding Y (1989) The rate-monotonic scheduling algorithm: exact characterization and average case behavior. In: Proceedings of the 10th IEEE real-time systems symposium, Santa Monica, CA, pp 166–171

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

Manolache S, Eles P, Peng Z (2001) Memory and time-efficient schedulability analysis of task sets with stochastic execution time. In: Proceedings of the 13th euromicro conference on real-time systems, Delft, The Nederlands, pp 19–26

Medina Pasaje JL, González Harbour M, Drake JM (2001) MAST real-time view: a graphic UML tool for modeling object-oriented real-time systems. In: Proceedings of the 22nd IEEE real-time systems symposium, London, UK, pp 245–256

Mok AK, Chen D (1997) A multiframe model for real-time tasks. IEEE Trans Softw Eng 23(10):635–645

Palencia JC, González Harbour M (1998) Schedulability analysis for tasks with static and dynamic offsets. In: Proceedings of the 19th IEEE real-time systems symposium, Madrid, Spain, pp 26–37

Punnekkat S, Davis R, Burns A (1997) Sensitivity analysis of real-time task sets. In: Proceedings of the 3rd Asian computing science conference on advances in computing science, Kathmandu, Nepal, pp 72–82

Racu R, Jersak M, Ernst R (2005) Applying sensitivity analysis in real-time distributed systems. In: Proceedings of the 11th real time and embedded technology and applications symposium, San Francisco, CA, pp 160–169

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, pp 188–198

Sjödin M, Hansson H (1998) Improved response-time analysis calculations. In: Proceedings of the 19th IEEE real-time systems symposium, Madrid, Spain, pp 399–408

Stankovic JA, Zhu R, Poornalingam R, Lu C, Yu Z, Humphrey M, Ellis B (2003) VEST: an aspect-based composition tool for real-time systems. In: Proceedings of the 9th real-time and embedded technology and applications symposium, Washington, DC, pp 58–69

Tindell KW, Burns A, Wellings A (1994) An extendible approach for analysing fixed priority hard real-time tasks. J Real Time Syst 6(2):133–152

Vestal S (1994) Fixed-priority sensitivity analysis for linear compute time models. IEEE Trans Softw Eng 20(4):308–317

Enrico Bini is assistant professor at the Scuola Superiore Sant'Anna in Pisa. He received the PhD 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 Nederlands, 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.

Marco Di Natale is an Associate Professor at the Computer Science and Computer Engineering Dept. of the Scuola Superiore Sant'Anna in Pisa, Italy. He started his research work on real-time systems during his PhD at the Scuola Superiore Sant'Anna and later as a research assistant at the University of Massachusetts. His research interests include real time systems and embedded systems, optimization of software architectures for distributed systems, operating systems and design methodologies for embedded real-time systems. He has been principal investigator for its institution in more than ten projects supported by the Italian Ministry of University and Research, by industrial partners and by the European Union. He is an IEEE member and served in the program or technical committees of conferences in the embedded systems or real-time systems domain. He teaches courses on Operating Systems, Real-Time Systems and Models and Tools for the Design of Real-Time Embedded Systems. In 2006 he has been visiting researcher at UC Berkeley, in the EECS dept., in cooperation with GM Research, providing support for the evaluation of automotive architectures.

Giorgio Buttazzo is Full Professor of Computer Engineering at the Scuola Superiore Sant'Anna of Pisa. He graduated in Electronic Engineering at the University of Pisa in 1985, received a Master in Computer Science at the University of Pennsylvania in 1987, and a Ph.D. in Computer Engineering at the Scuola Superiore Sant'Anna of Pisa in 1991. From 1987 to 1988, he worked on active perception and real-time control at the G.R.A.S.P. Laboratory of the University of Pennsylvania, Philadelphia. His main research interests include real-time operating systems, dynamic scheduling algorithms, quality of service control, multimedia systems, advanced robotics applications, and neural networks. He has authored six books on real-time systems and over 200 papers in the field of real-time systems, robotics, and neural networks. Prof. Buttazzo is Senior Member of IEEE.