Sensitivity Analysis for Fixed-Priority Real-Time Systems

Enrico Bini, Marco Di Natale, Giorgio Buttazzo
Scuola Superiore Sant'Anna
{e.bini,marco,giorgio}@sssup.it

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.

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, standard 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 [22], by simply pruning non-feasible solutions from the design space.

In many cases, however, the system is characterized by a high degree of heterogeneity and uncertainty, which makes the use of standard 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 plugins 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 cope with heterogeneity, 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, and possibly avoid the excessive simplification of modeling the execution time with a single attribute. 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 [16] allows the user to specify different execution times for different task instances, through a sequence of numbers. The generalized multi-frame task model [3] 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 [2] 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 [6, 14] 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

Proceedings of the 18th Euromicro Conference on Real-Time Systems (ECRTS'06)
0-7695-2619-5 /06 $20.00 © 2006 IEEE

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) [11, 1]. Although alternative efficient techniques have been proposed in the literature [4], 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 [21] and extend its applicability to more general task models [17]. Unfortunately, however, RTA is unfit for sensitivity analysis since it provides an aggregate value as the result of the application of a non invertible (and discontinuous) function to the task parameters.

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 Figure 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 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.

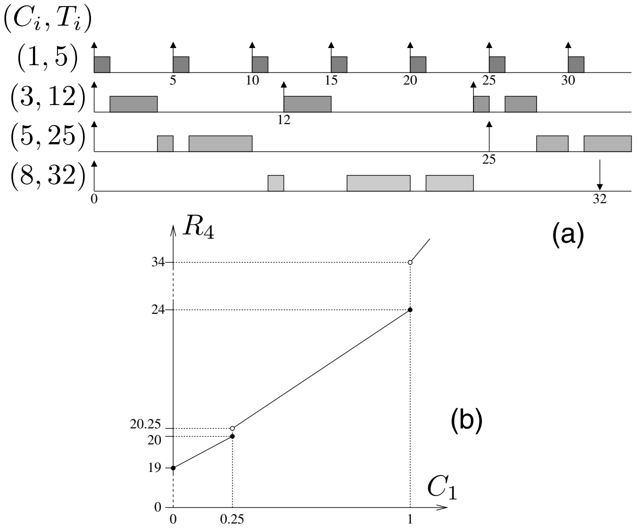

Figure 1. (a) RM schedule of 4 periodic tasks. (b) Response time R_4 as a function of C_1.

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 [12], 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 [24], who addressed sensitivity analysis of fixed priority task sets by introducing slack variables in the exact feasibility equations defined in [12] and solving the corresponding equality conditions. The approach suffers from the possibly very large number of conditions to be evaluated according to [12] and assumes deadlines equal to periods. Other techniques for sensitivity analysis are based on binary search over the schedulability property defined by Lehoczky [12], or other formulations of the algorithm for computing the response time of a task [18, 23].

In the context of distributed scheduling, a system-level sensitivity analysis of the task computation times against end-to-end deadlines has been proposed [19]. The method is based on binary search and has been implemented in commercial tools by SymTAVision [10]. 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 [15].

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 [4, 5]. 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 [7, 8]. 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: Section 2 introduces the terminology and the system model; Section 3 describes the notion of feasibility region; Section 4 and 5 describe the sensitivity analysis of the computation times and the periods respectively; Section 6 provides an example of application, whereas Section 7 discusses the implementation of the proposed techniques in available tools; finally Section 8 states our conclusions and future work.

2. Terminology and System Model

In this paper, we assume that an application consists of a set $\mathcal{T} = \{\tau_1, \tau_2, \ldots, \tau_N\}$ 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 $\mathbf{T} = (T_1, \ldots, T_N)$, $\mathbf{f} = (f_1, \ldots, f_N)$, and $\mathbf{C} = (C_1, \ldots, C_N)$. Note that the task set utilization U can be written by the dot product $\mathbf{C} \cdot \mathbf{f}$. Finally, the response time [11, 1] 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 $\mathcal{T}_i = \{\tau_1, \ldots, \tau_i\}$, and the corresponding vectors of computation times, periods, and rates are denoted by $\mathbf{C}_i$, $\mathbf{T}_i$, and $\mathbf{f}_i$, respectively. We consider the general case of deadlines less than or equal to periods, meaning that $0 < \delta_i \leq 1$.

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 [24] 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 $\mathcal{M} = \{\mu_1, \mu_2, \ldots, \mu_M\}$, 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 Equation (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 issues. For example, the model is suited for power-aware speed scaling operations 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 [13]

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

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

As it will be shown later, the feasibility region both in the C-space and in 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 [4] (originally proposed by Lehoczky et al. [12]), 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 (from [4]) *A periodic task set $\mathcal{T}$ is schedulable under fixed priorities **if and only if***

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

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

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

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

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

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

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 Section 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. [20] and later explicitly formalized in [5].

Theorem 2 (from [20, 5]) *A periodic task set $\mathcal{T}$ is schedulable under fixed priorities **if and only if***

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

such that:

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

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 Figure 2. The graph is obtained for $C_1 = 1$ and $C_2 = 2$ and deadlines equal to periods. For two tasks, the region is the union of rectangles given by Equations (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.

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

4. Sensitivity analysis in the C-space

As discussed above, it is very important to know the range of admissible variations of the computation times, because it often happens that the designer can select among

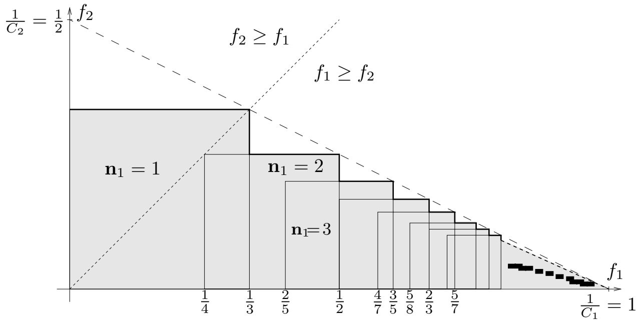

Figure 2. Feasibility region in the f-space.

different implementations for the same functionality, corresponding to different computation times.

From the computation model expressed in Equation (1), it follows that the action that the designer would undertake by altering the module implementation (so affecting the m_j value) is 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. Notice that 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 modified.

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 i^{th} computation time, leaving the others unchanged. It corresponds to the direction $\mathbf{d} = (0, \ldots, 0, 1, 0, \ldots, 0)$ of all zeros and a 1 in the i^{th} 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 Eq. (2) 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

$$\bigwedge_{i=1,\ldots,N} \bigvee_{t \in \mathsf{schedP}_i} \mathbf{n}_i \cdot \mathbf{C}'_i \leq t$$

$$\bigwedge_{i=1,\ldots,N} \bigvee_{t \in \mathsf{schedP}_i} \mathbf{n}_i \cdot (\mathbf{C}_i + \lambda\,\mathbf{d}_i) \leq t$$

$$\lambda \leq \min_{i=1,\ldots,N} \max_{t \in \mathsf{schedP}_i} \frac{t - \mathbf{n}_i \cdot \mathbf{C}_i}{\mathbf{n}_i \cdot \mathbf{d}_i} \triangleq \lambda^{\max} \tag{8}$$

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 $\lambda_{\max} \geq 0$; 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 Eq. (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 Eq. (8) after deciding 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 $\mathbf{d} = (0, \ldots, 0, 1, 0, \ldots, 0)$, where the only 1 in the vector is at the k^{th} position. 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 must be ensured only for the tasks from the k^{th} to the N^{th}.

From Eq. (8) we have

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

and computing the dot product in the denominator, we have

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

A representation of the sensitivity analysis in the C-space is shown in Figure 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 Section 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.

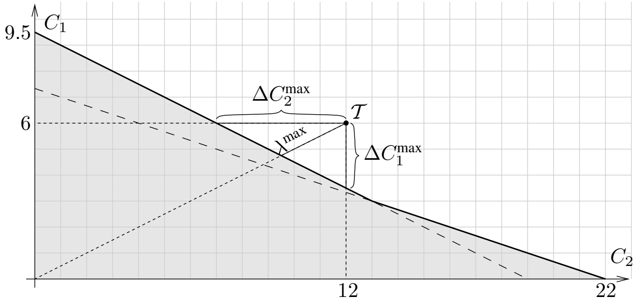

Figure 3. Sensitivity analysis in the C-space.

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 Eq. (8), we obtain

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

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

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 $\mathcal{T}$ was not schedulable, and the computation times must be decreased to achieve schedulability.

In the following, we show how an elastic compression [7] can be used with the exact analysis of fixed priority systems, by a proper definition of the direction $\mathbf{d}$ in Equation (8).

Linear Transformation The elastic task model [7], 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 not changed very much during a system adaptation. Although the original model was devised just 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 are in a given range.

To exploit the elastic model in the framework of sensitivity analysis, the direction of action can be simply set as $\mathbf{d} = \left(\frac{1}{E_1}, \frac{1}{E_2}, \ldots, \frac{1}{E_N}\right)$, so that the new computation times $\mathbf{C}'$ resulting from the modification performed according to the elastic scheme can be derived from Equation (8).

Other feasibility problems defined in the literature can also be cast within such a framework. As shown in Section 2, changing the duration m_j of a module μ_j corresponds to changes in the computation times given by the j^{th} column $\mathbf{a}_j$ of the matrix $\mathbf{A}$. Then, setting $\mathbf{d} = \mathbf{a}_j$, from Eq. (8) we can find the admissible alteration of the module length m_j. In this case, the *linear compute task model* [24] 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 Eq. (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 Eq. (8). In fact, it turns out that 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 may change. Let τ_k be the task whose period T_k is going to be modified. So 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, increasing T_k will clearly preserve schedulability. Similarly, if $\mathcal{T}$ is not schedulable, reducing T_k will keep $\mathcal{T}'$ still unschedulable. Let $T_k^{\min}$ be *the minimum period of τ_k such that $\mathcal{T}'$ is feasible* (see Figure 4). Then, it is easy to see that

$$\begin{array}{lcl} \mathcal{T} \text{ is schedulable} & \Leftrightarrow & T_k^{\min} \leq T_k \\ \mathcal{T} \text{ is not schedulable} & \Leftrightarrow & T_k^{\min} > T_k \end{array}$$

Also, any modification of period T_k does not affect the feasibility of the higher priority tasks. So, we assume that all the tasks in $\mathcal{T}_{k-1}$ are schedulable, otherwise no solution can be found.

Now we address the problem of finding the value of $T_k^{\min}$. The following theorem proves a very useful property of the minimum period $T_k^{\min}$.

Theorem 3 *Given a task set $\mathcal{T}$, let $T_k^{\min}$ be the minimum period among those T'_k that make the modified task set $\mathcal{T}'$ feasible. Then, we have that (a) either $T_k^{\min} = \frac{R_k}{\delta_k}$, (b) or there exists some lower priority task τ_i (with $i > k$) such that its response time R_i is an integer multiple of $T_k^{\min}$.*

Proof. We prove the theorem by contradiction. Suppose that there exists a task set $\mathcal{T}'$ such that

1. $\mathcal{T}'$ is feasible,
2. the period T'_k of the task τ_k cannot be further reduced without affecting the feasibility, meaning that $T'_k = T_k^{\min}$.

Also, by negating the hypothesis, we have

3. $T_k^{\min} \neq \frac{R_k}{\delta_k}$, and
4. for all lower priority tasks τ_i, $T_k^{\min}$ is not an integer multiple of R_i.

Since $\mathcal{T}'$ is feasible it must be have $R_k \leq D_k = \delta_k T_k$, thus the third hypothesis becomes

$$T_k^{\min} > \frac{R_k}{\delta_k}. \tag{11}$$

From the last hypothesis, it follows that for all the lower priority tasks τ_i we have

$$\frac{R_i}{T_k^{\min}} < \left\lceil \frac{R_i}{T_k^{\min}} \right\rceil \implies T_k^{\min} > \frac{R_i}{\lceil R_i / T_k^{\min} \rceil}. \tag{12}$$

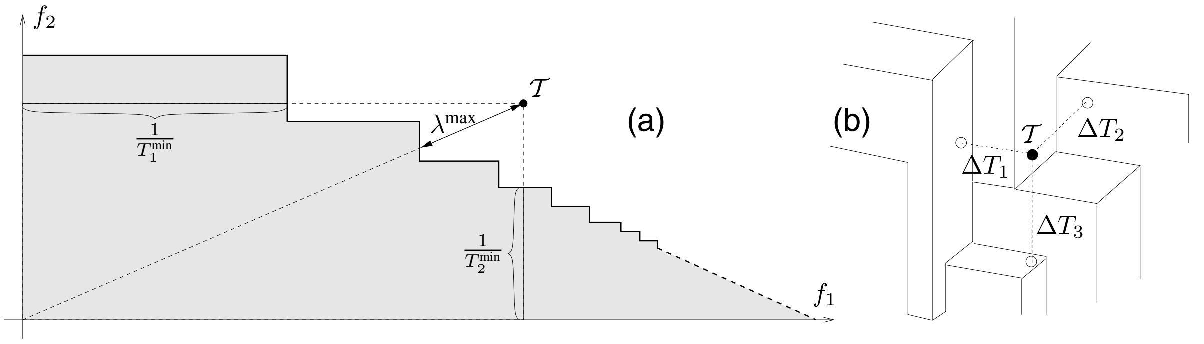

Figure 4. Sensitivity analysis in the f-space: when $N = 2$ (a) and $N = 3$ (b).

Notice that in Eq. (12) the equal sign does not hold. For this reason, the period $T_k^{\min}$ can be decreased until one of the Equations (11) or (12) holds with the equal sign (notice that Eq. (12) consists of $N - k$ equations). This contradicts the second hypothesis that $T_k^{\min}$ is the minimum feasible value, hence the theorem follows. □

It is now possible to devise an algorithm that computes the distance to the schedulability boundary along each frequency axis by computing the minimum feasible period $T_k^{\min}$ for each task τ_k.

First, notice that the response time of τ_k does not depend upon its period T_k'. It follows that any feasible period T_k' must satisfy the constraint

$$R_k \leq D_k \quad \Longrightarrow \quad T_k' \geq \frac{R_k}{\delta_k}. \tag{13}$$

Next, we investigate how the schedulability condition of all the lower priority tasks τ_i (with $i > k$) influences the bound of the period of the task τ_k.

From Theorem 3, we know that if Eq. (13) does not hold with the equal sign, then there exists a lower priority task τ_i whose response time R_i is an integer multiple of $T_k^{\min}$, that is

$$R_i = n_k \, T_k^{\min}. \tag{14}$$

Moreover, from the response time definition [11, 1], we must also have

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

where $R_i^k(n_k)$ is the response time of τ_i assuming that τ_k interferes exactly n_k times. Unfortunately, the number of interferences of τ_k in R_i is not known in advance. Hence, all the possible values for n_k must be checked.

The algorithm to search for $T_k^{\min}$ proceeds by computing $R_i^k(1)$ from Eq. (15), for all the lower priority tasks τ_i. If the fixed point equation returns a value greater than deadline D_i, then there is no value of T_k' that can possibly make the set schedulable, because τ_i cannot tolerate even a single interference from the higher priority task τ_k. However, if there is a first feasible value of $R_i^k(1)$, the algorithm increments the number of interferences n_k and computes all the response times $R_i^k(n_k)$ until for some n_k^*, the task τ_i becomes unschedulable. In this case, up to n_k^* interferences may occur, depending on the value of period T_k'. It follows from Eq. (14) that the periods T_k' that can feasibly schedule τ_i are such that

$$T_k' \geq \min_{n_k} \frac{R_i^k(n_k)}{n_k}. \tag{16}$$

Since the period T_k' must ensure the schedulability of all the lower priority tasks and, being larger than the bound of Eq. (13), the schedulability of τ_k itself, the minimum feasible period $T_k^{\min}$ is

$$T_k^{\min} = \max \left\{ \frac{R_k}{\delta_k}, \max_{i=k+1,\ldots,N} \min_{n_k} \frac{R_i^k(n_k)}{n_k} \right\}. \tag{17}$$

Computing the fixed point Equation (15) for each possible value of n_k is not as complex as it seems, since the computation of the candidate solution $R_i^k(n_k)$ can start from the previous solution $R_i^k(n_k - 1)$. The resulting complexity is, in the worst case, the same as the classical response time analysis [11, 1], performed for each task.

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

A previous result (please refer to [5] for its proof) establishes a relationship between the scaling operations in the C-space and in the f-space.

Theorem 4 (from [5]) *Given a task set $\mathcal{T} = \{\tau_1, \tau_2, \ldots, \tau_N\}$, 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 $1/\lambda$. Then, the task set $\mathcal{T}^C$ is schedulable **if and only if** the task set $\mathcal{T}^T$ is schedulable.*

From Theorem 4, 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.

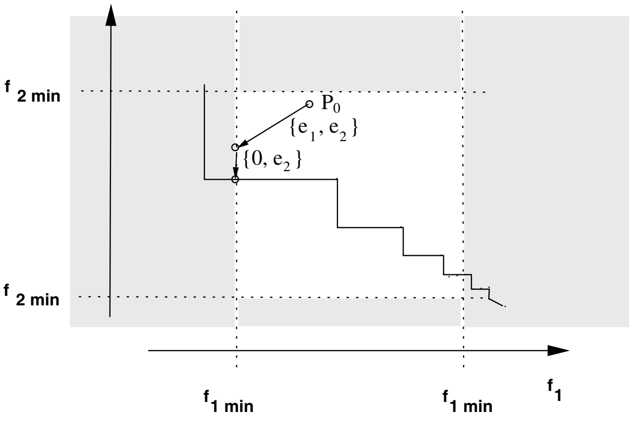

Figure 5. Modification of the tasks' rates (periods) according to the elastic model

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 $\mathbf{d}$ 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 the feasibility region ($\lambda_A < \lambda_B$), as shown in Figure 6.

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 schedulability 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 $\Delta C_k^{\max}$. 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 Eq. (9), we have

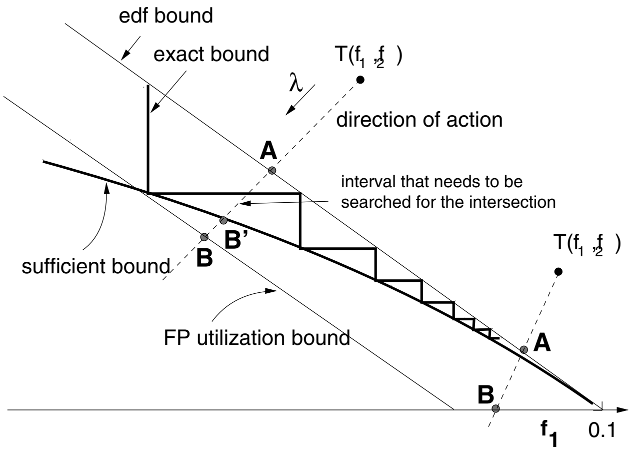

Figure 6. Finding the solution $\mathcal{T}'$ according to the elastic model.

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	$\{22, 19\}$

Table 1. Task set parameters.

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

and

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

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

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

$$\begin{aligned}
\lambda^{\max} &= \min_{i=1,2} \max_{t \in \mathsf{schedP}_i} \frac{t}{\mathbf{n}_i \cdot \mathbf{C}_i} - 1 \\
\lambda^{\max} &= \max_{t \in \mathsf{schedP}_2} \frac{t}{\mathbf{n}_2 \cdot \mathbf{C}} - 1 \\
\lambda^{\max} &= \max \left\{ \frac{22}{30}, \frac{19}{24} \right\} - 1 \\
\lambda^{\max} &= -0.20833 \qquad (20)
\end{aligned}$$

which is also represented in Figure 3.

It is now interesting to evaluate the variations that can be performed in the f-space to make the task set feasible. First, we evaluate the modification to apply to the individual task periods to reach feasibility. For this reason, we compute $T_1^{\min}$ and $T_2^{\min}$ from Eq. (17). We have

$$\begin{aligned} T_1^{\min} &= \max\left\{\frac{R_1}{\delta_1}, \min_{n_1} \frac{R_2^1(n_1)}{n_1}\right\} \\ T_1^{\min} &= \max\left\{C_1, \min\left\{\frac{R_2^1(1)}{1}, \frac{R_2^1(2)}{2}, \ldots\right\}\right\} \\ T_1^{\min} &= \max\{6, \min\{18, 24, \ldots\}\} = 18 \end{aligned} \tag{21}$$

which is the minimum value of T_1 which can schedule the task set. Notice that we stop computing the sequence of response times $R_2^1(n_1)$ as the value exceeds the deadline $D_2 = 22$. In a simpler way we proceed for the period of the second task, which amounts to

$$T_2^{\min} = \frac{R_2}{\delta_2} = \frac{36}{0.917} = 39.27. \tag{22}$$

The geometrical representation of the results is shown in Figure 4.

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

$$\begin{bmatrix} C_1 \\ C_2 \end{bmatrix} = \begin{bmatrix} 2 & 2 & 0 \\ 1 & 4 & 3 \end{bmatrix} \begin{bmatrix} m_1 \\ m_2 \\ m_3 \end{bmatrix} \tag{23}$$

where the m_i's are the length of software modules, as it is expressed in Eq. (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 want to evaluate the variation Δm_i to the module length m_i which can make the task set feasible. As explained in Section 4, this is possible by setting the direction $\mathbf{d}$ equal to the i^{th} column of the matrix in Eq. (23). By setting $\mathbf{d} = (2, 1)$ and computing Eq. (8), we find

$$\begin{aligned} \Delta m_1 &= \min_{i=1,2} \max_{t \in \mathsf{schedP}_i} \frac{t - \mathbf{n}_i \cdot \mathbf{C}_i}{\mathbf{n}_i \cdot (2,1)} \\ \Delta m_1 &= \max\left\{\frac{22-30}{(3,1)\cdot(2,1)}, \frac{19-24}{(2,1)\cdot(2,1)}\right\} \\ \Delta m_1 &= \max\left\{-\frac{8}{7}, -\frac{5}{5}\right\} = -1. \end{aligned} \tag{24}$$

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. Support for design cycles

The sensitivity analysis techniques discussed in this paper have been implemented in the RT-Druid toolset [9]: 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 the 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 a context where achieving the performance/cost trade-offs requires to setup a cyclic design process where control and software engineers work together providing the best possible quality and the best possible accuracy to system functions within the timing constraints and/or the performance target.

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 variation of the design.

The RT-Druid design environment has been implemented in Java, and it is integrated (as a set of additional plugin 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. The design environment supports models imported from Simulink, ASCET-SD and UML 2.0 object and component specifications. Furthermore, it fully supports the OSEK/VDX OIL specification, allowing the import of OIL architectural descriptions, and the export of the OIL application definitions compatible with third party OSEK RTOSs. RT-Druid also supports code generation from a software architecture (specified using OIL or using the internal RT-Druid graphical editor) to an OSEK/VDX RTOS.

8. 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 also showed how the proposed framework can be conveniently adopted to generalize overload management methods, as the elastic scheduling approach [7], which can be effectively extended to work with the exact analysis of fixed priority systems.

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

[1] N. C. Audsley, A. Burns, M. Richardson, K. W. Tindell, and A. J. Wellings. Applying new scheduling theory to static priority pre-emptive scheduling. *Software Engineering Journal*, 8(5):284–292, Sept. 1993.

[2] S. K. Baruah. Dynamic- and static-priority scheduling of recurring real-time tasks. *Real-Time Systems*, 24(1):93–128, Jan. 2003.

[3] S. K. Baruah, D. Chen, S. Gorinsky, and A. K. Mok. Generalized multiframe tasks. *Real-Time Systems*, 17(1):5–22, July 1999.

[4] E. Bini and G. C. Buttazzo. Schedulability analysis of periodic fixed priority systems. *IEEE Transactions on Computers*, 53(11):1462–1473, Nov. 2004.

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

[6] A. Burns, G. Bernat, and I. Broster. A probabilistic framework for schedulability analysis. In *Proceedings of the EMSOFT*, pages 1–15, Philadelphia (PA), U.S.A., Oct. 2003.

[7] G. C. Buttazzo, G. Lipari, M. Caccamo, and L. Abeni. Elastic scheduling for flexible workload management. *IEEE Transactions on Computers*, 51(3):289–302, Mar. 2002.

[8] G. C. Buttazzo, M. Velasco, P. Martí, and G. Fohler. Managing quality-of-control performance under overload conditions. In *Proceedings of the* 16^{th} *Euromicro Conference on Real-Time Systems*, pages 53–60, Catania, Italy, June 2004.

[9] Evidence s.r.l. RT-Druid. available at `http://www.evidence.eu.com/RT-Druid.asp`.

[10] A. Hamann, R. Henia, M. Jerzak, R. Racu, K. Richter, and R. Ernst. SymTA/S symbolic timing analysis for systems. available at `http://www.symta.org`, 2004.

[11] M. Joseph and P. K. Pandya. Finding response times in a real-time system. *The Computer Journal*, 29(5):390–395, Oct. 1986.

[12] J. P. Lehoczky, L. Sha, and Y. Ding. The rate-monotonic scheduling algorithm: Exact characterization and average case behavior. In *Proceedings of the* 10^{th} *IEEE Real-Time Systems Symposium*, pages 166–171, Santa Monica (CA), U.S.A., Dec. 1989.

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

[14] S. Manolache, P. Eles, and Z. Peng. Memory and time-efficient schedulability analysis of task sets with stochastic execution time. In *Euromicro Conference on Real-Time Systems*, pages 19–26, Delft, The Nederlands, June 2001.

[15] J. L. Medina Pasaje, M. González Harbour, and J. M. Drake. MAST real-time view: A graphic UML tool for modeling object-oriented real-time systems. In *Proceedings of the* 22^{nd} *IEEE Real-Time Systems Symposium*, pages 245–256, London, UK, Dec. 2001.

[16] A. K. Mok and D. Chen. A multiframe model for real-time tasks. *IEEE Transactions on Software Engineering*, 23(10):635–645, Oct. 1997.

[17] J. C. Palencia and M. González Harbour. Schedulability analysis for tasks with static and dynamic offsets. In *Proceedings of the* 19^{th} *IEEE Real-Time Systems Symposium*, pages 26–37, Madrid, Spain, Dec. 1998.

[18] S. Punnekkat, R. Davis, and A. Burns. Sensitivity analysis of real-time task sets. In *Proceedings of the* 3^{rd} *Asian Computing Science Conference on Advances in Computing Science*, pages 72–82, Kathmandu, Nepal, Dec. 1997.

[19] R. Racu, M. Jersak, and R. Ernst. Applying sensitivity analysis in real-time distributed systems. In *Proceedings of the* 11^{th} *Real Time and Embedded Technology and Applications Symposium*, pages 160–169, San Francisco (CA), U.S.A., Mar. 2005.

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

[21] M. Sjödin and H. Hansson. Improved response-time analysis calculations. In *Proceedings of the* 19^{th} *IEEE Real-Time Systems Symposium*, pages 399–408, Madrid, Spain, Dec. 1998.

[22] J. A. Stankovic, R. Zhu, R. Poornalingam, C. Lu, Z. Yu, M. Humphrey, and B. Ellis. VEST: An aspect-based composition tool for real-time systems. In *Proceesings of the* 9^{th} *Real-Time and Embedded Technology and Applications Symposium*, pages 58–69, Washington (DC), U.S.A., May 2003.

[23] K. W. Tindell, A. Burns, and A. Wellings. An extendible approach for analysing fixed priority hard real-time tasks. *Journal of Real Time Systems*, 6(2):133–152, Mar. 1994.

[24] S. Vestal. Fixed-priority sensitivity analysis for linear compute time models. *IEEE Transactions on Software Engineering*, 20(4):308–317, Apr. 1994.