Parametric Polynomial-Time Algorithms for Computing Response-Time Bounds for Static-Priority Tasks with Release Jitters

Nathan Fisher, Thi Huyen Chau Nguyen, Joël Goossens, Pascal Richard

13th IEEE International Conference on Embedded and Real-Time Computing Systems and Applications (RTCSA'07), Daegu, Korea, 2007

Parametric Polynomial-Time Algorithms for Computing Response-Time Bounds for Static-Priority Tasks with Release Jitters

Nathan Fisher
Wayne State University
Detroit, MI (USA)
fishern@cs.wayne.edu

Thi Huyen Châu Nguyen
LISI, ENSMA and University of Poitiers (France)

Joël Goossens
Computer Science Department
Université Libre de Bruxelles (U.L.B.)
Brussels, Belgium
joel.goossens@ulb.ac.be

Pascal Richard
LISI, ENSMA and University of Poitiers (France)
pascal.richard@univ-poitiers.fr

Abstract

Feasibility analysis algorithms are based on particular metrics such as processor utilization, load factor, processor demand, response-times, etc. The design of efficient algorithms for computing these metrics is a major issue in real-time scheduling theory. In this paper we propose two FPTAS*s (fully-polynomial time approximation schemes) for checking feasibility of static-priority tasks subjected to release jitters executed upon a uniprocessor platform. We then use these* FPTAS*s for computing two upper bounds of worst-case response-times. Lastly, we show that these bounds do not achieve constant error bounds in comparison with values computed by an exact worst-case response-time analysis (performed in pseudo-polynomial time), and we present numerical experiments.*

1. Introduction

Improving feasibility tests is an important issue in real-time scheduling theory. For tasks having deadlines less than or equal to periods (i.e., constrained-deadlines), no polynomial-time feasibility tests are known for both static-priority and EDF scheduling. [13] shows that for static-priority scheduling, there exists task systems with only two tasks such that checking feasibility with an exact feasibility test requires $\mathcal{O}(k)$ steps, where k is an arbitrary integer which does not depend on task parameters (i.e., worst-case processing requirements, deadlines or periods). Many polynomial-time sufficient feasibility tests have been defined; see for instance [14, 9, 7, 5, 10] for a non exhaustive reference list.

Approximation algorithms allow the design of efficient feasibility tests (e.g. running in polynomial time) while introducing a small error in the decision process, that is controlled by an accuracy parameter. Such approaches have been developed for EDF scheduling [6, 1, 2] and for static-priority scheduling [8, 16]. Two different paradigms can be used to define approximate feasibility tests [6]: *pessimistic* and *optimistic*.

If a pessimistic test returns "feasible", then the task set is guaranteed to be feasible on a unit-speed processor. If the test returns "infeasible", the task set is guaranteed to be infeasible on a *slower processor*, of computing capacity $(1-\epsilon)$. In [8] is presented such a test for static-priority task with arbitrary-deadlines.

If an optimistic test returns "feasible", then the task set is guaranteed to be feasible on a $(1+\epsilon)$-speed processor. If the test returns "infeasible", the task set is guaranteed to be infeasible on a unit-speed processor [6]. To the best of our knowledge, such an approach has never been investigated for static-priority scheduling.

This research. The objective of this paper is to extend approximate feasibility analysis of static-priority tasks subjected to release jitters. We provide an alternative definition of the *Request Bound Function* (a characterization of work for static-priority tasks) which leads to a slight improvement of the pessimistic FPTAS presented in [8]. We define an optimistic FPTAS in the same scheduling context. Both FPTASs are then used for respectively computing upper bounds of worst-case response-times. We then analyse the maximum error of the approximate values for worst-case response-

times calculated in polynomial-time in comparison with exact values of worst-case response-times that are computed in pseudo-polynomial time. Lastly, we give numerical experiments to compare known methods for computing upper bounds of worst-case response-times.

Organization. Section 2 presents known results for validating static-priority tasks executed upon a uniprocessor platform. Section 3 presents an improvement of the pessimistic tests presented in [8] and an algorithm for computing upper bounds of task worst-case response-times. Section 4 presents an optimistic approximate feasibility test and its use for computing a second upper bound of task worst-case response-times. Section 5 presents results on worst-case error bounds of these approximate values of worst-case response-times. Lastly, we conclude in Section 7.

2. Definitions

2.1. Task model

A task τ_i, $1 \leq i \leq n$, is defined by a worst-case execution requirement C_i, a relative deadline D_i and a period T_i which is the time units between two consecutive job instances of task τ_i. The utilization factor of task τ_i is the fraction of time that τ_i requires the processor: $U_i \stackrel{\text{def}}{=} C_i/T_i$. The utilization factor of the task set is: $U \stackrel{\text{def}}{=} \sum_{i=1}^{n} \frac{C_i}{T_i}$. We assume that deadlines are constrained: $D_i \leq T_i$. Such an assumption is realistic in many real-world applications and also leads to simpler algorithm for checking feasibility of task sets [11]. In order to model delay due to the RTOS (in presence of system tick) or due to input data communications of tasks, we also consider that jobs are subjected to release jitters; that is, a job may not be ready to execute as soon as it is released. In fact, it may experience a variable delay between its release time and the first time instant such that the job is ready to execute. The release jitter J_i of a task τ_i is the *largest* delay between its release time and (first) ready time. We assume that $J_i < T_i$; we also assume that all parameters are integers (i.e., discrete time model).

We assume that all tasks to be run upon a same processor are independent, synchronously released. The presented results will also be valid for sporadic task systems (where there is *at least* T_i time units between two consecutive job instances of the task τ_i). All the tasks have static priorities that are set before starting the application and never changed at run-time. At any time, the highest priority task is selected among ready tasks. Without loss of generality, we assume that tasks are indexed in decreasing order of their priorities: τ_1 is the highest priority task and τ_n is the lowest one.

2.2. Approximate Response-Times

A common approach for checking the feasibility of a static-priority task set is to compute the exact worst-case response-time R_i. The worst-case response-time of τ_i is formally defined as:

Definition 1 *Assuming that the system is not overloaded (the utilization factor is strictly less than 1), the worst-case response-time of a task τ_i can be defined as follows:*

$$R_i \stackrel{\text{def}}{=} (\min\{t > 0 \mid W_i(t) = t\}) + J_i$$

($W_i(t)$ denotes the cumulative processor demand and will be defined formally in the next section.)

No polynomial-time algorithm is known for computing R_i for the considered task model. Computing efficiently such a metric will be addressed using approximation algorithms. We now formally define approximate worst-case response-times according to an accuracy parameter ϵ as follows:

Definition 2 *Let ϵ be a constant and R_i be the "exact" worst-case response-time of a task τ_i, then an approximate upper bound of its responses time $\widehat{R_i}$ satisfies:*

$$R_i \leq \widehat{R_i} \leq (1+\epsilon)R_i$$

If the approximation algorithm satisfies the following conditions: $0 < \epsilon < 1$ and the runtime is polynomial in the input size and $1/\epsilon$, then it is a Fully Polynomial Time Approximation Scheme (FPTAS).

In [17] is presented an upper bound of the worst-case response of a task τ_i:

$$R_i \leq \frac{\sum_{j=1}^{i} C_j}{1 - \sum_{j=1}^{i-1} \frac{C_j}{T_j}}$$

We have shown in [15] that this well-known upper bound does not have a constant error bound (i.e., there exists task sets such that the upper bound is c times greater than R_i where c is an arbitrary large number). Thus, the corresponding $\mathcal{O}(n)$ algorithm is not an approximation algorithm for computing upper bounds of worst-case response-times.

To improve the worst-case response-time approximate analysis, we need methods with a trade-off between computational time-complexity and sharpness of the upper bounds of worst-case response-times. In order to define our approximate response-time analysis, we review, in the next section, known feasibility tests that will be used to define our upper bounds.

2.3. Exact Analysis

The request-bound function of a task τ_i at time t (denoted $\text{RBF}(\tau_i, t)$) and the cumulative processor demand (denoted $W_i(t)$) of tasks at time t for tasks having priorities greater than or equal to i are (see [18] for details):

$$\text{RBF}(\tau_i, t) \stackrel{\text{def}}{=} \left\lceil \frac{t + J_i}{T_i} \right\rceil C_i \quad (1)$$

$$W_i(t) \stackrel{\text{def}}{=} C_i + \sum_{j=1}^{i-1} \text{RBF}(\tau_j, t) \quad (2)$$

Notice that deadline failures of τ_i (if any) occur necessarily in an interval of time where only tasks with a priority higher or equal to i are running. Such an interval of time is defined as a level-i busy period [12]. Using these functions, two distinct (but linked) exact feasibility tests can be defined. We now review both results that will be reused throughout the paper.

The processor-demand approach checks that the processor demand required by task executions is always less than or equal to the processor capacity. [12] presents a processor demand schedulability test for constrained-deadline systems (but has been extended for arbitrary deadline systems in [11]). It can be also easily extended to tasks subjected to release jitters as stated in the well-known following result:

Theorem 1 *A static-priority system with release jitters is feasible if, and only if:*

$$\max_{i=1\dots n} \left\{ \min_{t \in S_i} \frac{W_i(t)}{t} \right\} \leq 1$$

where S_i is the set of scheduling points defined as follows:

$$S_i \stackrel{\text{def}}{=} \{aT_j - J_j \mid j = 1 \dots i, a = 1 \dots \lfloor \frac{D_i - J_i + J_j}{T_j} \rfloor\} \cup \{D_i - J_i\}$$

Note that schedulability points S_i correspond to a set of time instants in the schedule where a higher-priority task can start its execution, after its release jitter delay.

In [16], we show that the worst-case response-time of a task with release jitter can be computed using Theorem 1. We defined the notion of the critical scheduling point for that purpose (under the assumption that the task τ_i will meet its deadline at execution time).

Definition 3 *The critical scheduling point for a feasible task τ_i is:*

$$t^* \stackrel{\text{def}}{=} \min\{t \in S_i \mid W_i(t) \leq t\}$$

We show in [16] that the cumulative request bound function at the critical scheduling point of a given task τ_i leads to its worst-case response-time.

Theorem 2 ([16]) *The worst-case response-time of a task τ_i, such that $W_i(t^*) \leq t^*$ (i.e., the task is feasible), is exactly $R_i = W_i(t^*) + J_i$.*

Thus, for all feasible tasks, it is possible to compute their worst-case response-times. But, for an infeasible task τ_i (i.e., $R_i > D_i$), there is no one scheduling point $t \in S_i$ satisfying $W_i(t) \leq t$. Since the size of S_i depends on $\sum_{j=1}^{i-1} \lfloor \frac{D_i + J_i}{T_j} \rfloor$, then the algorithm runs in *pseudo*-polynomial time. Note that computing the smallest fixed-point $W_i(t) = t$ using successive approximation of iterative convergence is also performed in pseudo-polynomial time (the utilization factor is assumed to be a constant).

3. A First Worst-Case Response-Time Bound

In order to define an upper bound of task response-times, we define a pessimistic FPTAS for analyzing feasibility of task sets. When such an algorithm returns feasible for a given task τ_i, then we shall be able to compute approximate upper bound of τ_i. The presented algorithm is a slight improvement of [8].

3.1. Pessimistic Approximation Scheme

The RBF function is a discontinuous function with a "step" of height C_i every T_i units of time. In order to approximate the request bound function according to an error bound $1 + \epsilon$ (accuracy parameter, $0 < \epsilon < 1$), we use the same principle as in [8]: we consider the first $(k-1)$ steps of $\text{RBF}(\tau_i, t)$, where k is defined as $k \stackrel{\text{def}}{=} \lceil 1/\epsilon \rceil$ and a linear approximation, thereafter. From this definition, we verify that $k \geq 1/\epsilon$.

By extending the approximate function presented in [8] to take into account release jitter, an approximate request bound function can be defined as follow [16]:

$$\delta(\tau_i, t) \stackrel{\text{def}}{=} \begin{cases} \text{RBF}(\tau_i, t) & \text{for } t \leq (k-1)T_i - J_i, \\ C_i + (t + J_i)\frac{C_i}{T_i} & \text{otherwise} \end{cases}$$

Thus, up to $(k-1)T_i - J_i$ no approximation is performed to evaluate the total execution requirement of τ_i, and after that it is approximated by a linear function with a slope equal to the utilization factor of task τ_i.

We propose next a new definition of the Request Bound Function (i.e., $\ddot{\text{RBF}}$) that will lead to an improved approximate feasibility algorithms in comparison with the results presented in [8, 16]. We propose the following alternative

definition and prove that it is equivalent to the definition presented in Equation (1).

Lemma 1 *Under the assumption that all the task parameters are integers, then the request-bound function* $\dddot{\text{RBF}}$ *for static-priority task subjected to release jitters can be defined as follows:*

$$\dddot{\text{RBF}}(\tau_i, t) \stackrel{\text{def}}{=} \left\lfloor \frac{t + T_i + J_i - 1}{T_i} \right\rfloor C_i$$

And with this definition we have that $\text{RBF}(\tau_i, t) = \dddot{\text{RBF}}(\tau_i, t) \quad \forall i, t$

Proof: We shall use the principle of indirect equality: let a, b, k be integers, then $a = b$ if, and only if, $\forall k, k \leq a \Leftrightarrow k \leq b$. Consider an arbitrary integer k such that $k \leq \left\lfloor \frac{t+J_i}{T_i} \right\rfloor$. Since $\lceil x \rceil < x + 1$ for any real number x, then

$$k \leq \left\lceil \frac{t + J_i}{T_i} \right\rceil < \frac{t + J_i}{T_i} + 1$$

Since $T_i > 0$, then:

$$T_i(k-1) < t + J_i$$
$$T_i(k-1) + 1 \leq t + J_i$$

This latter inequality exploits the fact that t is an integer. Then, we finally obtain that:

$$k \leq \frac{t + J_i + T_i - 1}{T_i}$$

Using the rule of indirect equality and since k is arbitrary, then it follows that:

$$\begin{aligned} \dddot{\text{RBF}}(\tau_i, t) &= \left\lfloor \frac{t + T_i + J_i - 1}{T_i} \right\rfloor C_i \\ &= \left\lceil \frac{t + J_i}{T_i} \right\rceil C_i = \text{RBF}(\tau_i, t) \end{aligned}$$

■

Using Lemma 1, we can define an improved approximate request bound function:

$$\gamma(\tau_i, t) \stackrel{\text{def}}{=} \begin{cases} \left\lfloor \frac{t+J_i+T_i-1}{T_i} \right\rfloor C_i & \text{for } t \leq (k-1)T_i - J_i, \\ (t + J_i + T_i - 1)\frac{C_i}{T_i} & \text{otherwise.} \end{cases} \quad (3)$$

From both definitions of $\delta(\tau_i, t)$ and $\gamma(\tau_i, t)$, it is easy to see that $\gamma(\tau_i, t)$ can only improve the approximate function $\delta(\tau_i, t)$, and thus the FPTAS feasibility algorithm proposed in [8].

Theorem 3 $\gamma(\tau_i, t)$ *is a tighter upper bound of* $\text{RBF}(\tau_i, t)$ *in comparison with* $\delta(\tau_i, t)$*:*

$$\forall t > 0, \delta(\tau_i, t) \geq \gamma(\tau_i, t)$$

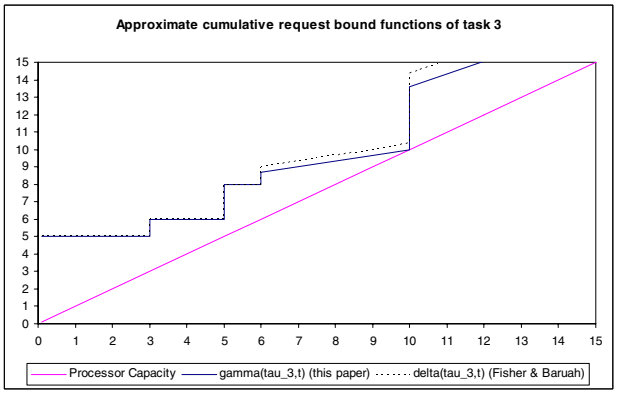


Figure 1. Approximate cumulative request bound functions on task set τ

We shall see that an FPTAS can be based on $\gamma(\tau_i, t)$. To define an approximate feasibility test based on the principle of Theorem 1, we define an approximate cumulative request bound function as:

$$\widehat{W}_i(t) \stackrel{\text{def}}{=} C_i + \sum_{j=1}^{i-1} \gamma(\tau_j, t)$$

According to the error bound $1+\epsilon$ leading to $k = \lceil 1/\epsilon \rceil$, we define the following testing set $\widehat{S}_i \subseteq S_i$:

$$\widehat{S}_i \stackrel{\text{def}}{=} \{bT_a - J_a \mid a = 1 \ldots i-1, b = 1 \ldots k-1\} \cup \{D_i - J_i\}$$

The principle of the algorithm is as follows:

- If there exists a time instant $t \in \widehat{S}_i$ such that $\widehat{W}_i(t) \leq t$, then τ_i is feasible (upon a unit speed processor),
- otherwise τ_i is infeasible on a processor of $(1 - \epsilon)$ capacity.

A simple implementation of this approximate feasibility test leads to a $\mathcal{O}(n^2/\epsilon)$ algorithm. This is an FPTAS since the algorithm is polynomial according the input size and the input parameter $1/\epsilon$.

To see the improvement in the approximation under $\gamma(\tau_j, t)$ consider the following task set τ, defined as $\tau_i(C_i, D_i = T_i), 1 \leq i \leq 3$ with $\tau_1(1, 3)$, $\tau_2(2, 5)$, $\tau_3(2, 12)$, is not proved feasible using the tests presented in [8]. Using the new definition of the Request-Bound Function $\dddot{\text{RBF}}$ (and more precisely $\gamma(\tau_3, t)$), the task τ_3 is now proved feasible since $\widehat{W}_i(10) = 10$ (i.e., see Figure 1 that presents both approximate cumulative request bound functions for task τ_3).

We now prove the correctness of this approximate feasibility test. The key point for the correctness of the approximation scheme is: $\delta(\tau_i,t)/\text{R}\ddot{\text{B}}\ddot{\text{F}}(\tau_i,t) \leq (1+\epsilon)$. This result will then be used to prove that if a task set is stated infeasible by the FPTAS, then it is infeasible under a $(1-\epsilon)$-speed processor.

Theorem 4 *$\forall t \geq 0$, we verify that:*

$$\begin{aligned} \dddot{\text{RBF}}(\tau_i,t) &\leq \gamma(\tau_i,t) \quad (4) \\ \gamma(\tau_i,t) &\leq (1+\epsilon)\dddot{\text{RBF}}(\tau_i,t) \quad (5) \end{aligned}$$

Proof: The Inequality (4) directly follows from the definitions of $\text{R}\ddot{\text{B}}\ddot{\text{F}}(\tau_i,t)$ and $\gamma(\tau_i,t)$. We now prove the Inequalities stated in (5):
If $\gamma(\tau_i,t) > \dddot{\text{RBF}}(\tau_i,t)$ then $t > (k-1)T_i - J_i$, since $\gamma(\tau_i,t) = \text{R}\ddot{\text{B}}\ddot{\text{F}}(\tau_i,t)$ prior to $(k-1)T_i - J_i$. Thus, there are $k-1$ steps in $\gamma(\tau_i,t)$ before approximating the request bound function; it thus follows that:

$$\dddot{\text{RBF}}(\tau_i,t) \geq kC_i \quad (6)$$

Furthermore, $\gamma(\tau_i,t) - \dddot{\text{RBF}}(\tau_i,t) \leq C_i$: this is obvious if $t \in [0,(k-1)T_i - J_i]$ since $\gamma(\tau_i,t) = \text{R}\ddot{\text{B}}\ddot{\text{F}}(\tau_i,t)$, and if $t > (k-1)t - J_i$, then using Lemma 1 and since $0 \leq x - \lfloor x \rfloor < 1$ for any real number x:

$$\gamma(\tau_i,t) - \dddot{\text{RBF}}(\tau_i,t) = \left(\frac{t+J_i+T_i-1}{T_i} - \left\lfloor \frac{t+J_i+T_i-1}{T_i} \right\rfloor\right) C_i \leq C_i$$

As a consequence:

$$\gamma(\tau_i,t) \leq \dddot{\text{RBF}}(\tau_i,t) + C_i$$

Using inequality (6), we obtain the result:

$$\gamma(\tau_i,t) \leq (1+\frac{1}{k})\dddot{\text{RBF}}(\tau_i,t) \leq (1+\epsilon)\dddot{\text{RBF}}(\tau_i,t)$$

As a consequence, both inequalities are verified. ■

Using the same approach presented in [8], we can establish the correctness of approximation.

Theorem 5 *If $\forall t \in (0, D_i - J_i]$, $\widehat{W}_i(t) > t$, then τ_i is infeasible on a processor of $(1-\epsilon)$ capacity.*

Proof: Assume that $\forall t \in (0, D_i - J_i]$, $\widehat{W}_i(t) > t$, but τ_i is still feasible on a $(1-\epsilon)$-speed processor. Since assuming τ_i to be feasible upon a $(1-\epsilon)$ speed processor, then there must exist a time t_0 such that τ_i:

$$W_i(t_0) \leq (1-\epsilon)t_0$$

But, from Theorem 4 and Lemma 1 we verify that $\widehat{W}_i(t) \leq (1+\epsilon)W_i(t)$, where $k \stackrel{\text{def}}{=} \lceil \frac{1}{\epsilon} \rceil$, then for all $t \in (0, D_i - J_i]$, the condition $\widehat{W}_i(t) > t$ becomes:

$$W_i(t) > \frac{t}{1+\epsilon}$$

Since $0 < \epsilon < 1$, then $\frac{1}{1+\epsilon} \geq (1-\epsilon)$ and:

$$W_i(t) > (1-\epsilon)t \quad \forall t \in (0, D_i - J_i]$$

As a consequence, a time t_0 such that $W_i(t_0) \leq (1-\epsilon)t_0$ cannot exist and τ_i is infeasible. ■

Now, if the approximate test concludes that a task τ_i is feasible, then it is feasible upon a unit-speed processor.

Theorem 6 *If there exists a time instant $t \in (0, D_i - J_i]$ such that $\widehat{W}_i(t) \leq t$, then $W_i(t) \leq t$*

Proof: Directly follows from Theorem 4 that allows to conclude that $\forall t \in (0, D_i - J_i], \widehat{W}_i(t) \geq W_i(t)$. ■

To conclude the correctness, we must prove that scheduling points are sufficient.

Theorem 7 *For all $t \in \widehat{S}_i$ such that $\widehat{W}_i(t) > t$, then we also verify that: $\forall t \in (0, D_i - J_i], \widehat{W}_i(t) > t$*

Proof: (Sketch) Let t_1 and t_2 be two *adjacent* points in $\widehat{S}_i$ (i.e., $\nexists\ t \in \widehat{S}_i$ such that $t_1 < t < t_2$). Since $\widehat{W}_i(t_1) > t_1$, $\widehat{W}_i(t_2) > t_2$ and the fact that $\widehat{W}_i(t)$ is an non-decreasing step left-continuous function we conclude that $\forall t \in (t_1, t_2)\ \widehat{W}_i(t) > t$. The theorem follows. ■

3.2. Approximate Worst-Case Response-Times

Using the FPTAS presented in previous section, we can check that a task is feasible upon a unit-speed processor or infeasible upon a $(1-\epsilon)$-speed processor. If it is feasible, then we are able to compute an upper bound of the worst-case response-time of a task as presented in Section 3. If the feasibility algorithm does not give a positive answer, then our approach is not able to derive any upper bound (but, we can use the one defined in [17] for instance).

Definition 4 *Consider a task τ_i such that there exists a time t satisfying $\widehat{W}_i(t) \leq t$, then an approximate upper bound of its worst-case response-time is defined by:*

$$\begin{aligned} t^* &\stackrel{\text{def}}{=} \min\left\{ t \in \widehat{S}_i \middle| \widehat{W}_i(t) \leq t \right\} \\ \widehat{R_i} &\stackrel{\text{def}}{=} \widehat{W}_i(t^*) + J_i \end{aligned}$$

We now prove that such a method defines an upper bound of the worst-case response-time of task τ_i (i.e., satisfying the condition presented in Definition 2).

Theorem 8 *For every task τ_i such that there exists a time $t \in \widehat{S}_i$ satisfying $\widehat{W}_i(t) \leq t$, then:*

$$R_i \leq \widehat{R_i}$$

Proof: We know that for all time instant t:

$$\widehat{W_i}(t) \quad \geq \quad W_i(t)$$

Let t_c be the first time instant such that $\widehat{W_i}(t_c) = t_c$ (i.e., t_c is critical) and t^* the time instant corresponding to the worst-case response-time of τ_i (i.e., such that $W_i(t^*) = t^*$), then we necessarily have $t^* \leq t_c$. Since $W_i(t)$ and $\widehat{W_i}(t)$ are non-decreasing functions, the result follows. ■

4. A Second Worst-Case Response-Time Bound

In order to define a second upper bound of worst-case response-times, we define an *optimistic* FPTAS for approximate feasibility analysis.

4.1. Optimistic Approximation Scheme

We use the same principle: $k-1$ steps of $\dddot{\text{RBF}}(\tau_i, t)$ will be considered, and after that a linear lower bound based an a simple relaxation of integral values of $\ddot{\text{RBF}}(\tau_i, t)$ is used. The number k of considered steps is defined according to the accuracy parameter ϵ:

$$k \stackrel{\text{def}}{=} \left\lceil \frac{1}{\epsilon} \right\rceil + 1$$

Note that k has a different definition in the pessimistic case than the optimistic case.

$$\lambda(\tau_i, t) \stackrel{\text{def}}{=} \begin{cases} \dddot{\text{RBF}}(\tau_i, t) & \text{for } t \leq (k-1)T_i - J_i, \\ (t + J_i)\frac{C_i}{T_i} & \text{otherwise.} \end{cases}$$

To define an optimistic approximate feasibility test, we define an approximate cumulative request bound function as:

$$\widetilde{W_i}(t) \stackrel{\text{def}}{=} C_i + \sum_{j=1}^{i-1} \lambda(\tau_j, t)$$

We define the following testing set $\widetilde{S}_i \subseteq S_i$:

$$\widetilde{S}_i \stackrel{\text{def}}{=} \{bT_a - J_a \mid a = 1 \ldots i-1, b = 1 \ldots k-1\} \cup \{D_i - J_i\}$$

The principle of the algorithm is the following:

- If there exists a time instant $t \in \widetilde{S}_i$ such that $\widetilde{W_i}(t) \leq t$, then τ_i is feasible on a $(1+\epsilon)$-speed processor (but no conclusion can be taken when a unit-speed processor is considered),
- otherwise τ_i is infeasible upon a unit speed processor.

As the approximate pessimistic version of the feasibility test, this optimistic one leads to an $\mathcal{O}(\frac{n^2}{\epsilon})$ algorithm (i.e., an FPTAS). We now prove the correctness of the optimistic approximation scheme.

Theorem 9 *$\forall t \geq 0$, we verify that:*

$$(1-\epsilon)\ddot{\text{RBF}}(\tau_i, t) \quad \leq \quad \lambda(\tau_i, t) \tag{7}$$

$$\lambda(\tau_i, t) \quad \leq \quad \dddot{\text{RBF}}(\tau_i, t) \tag{8}$$

Proof: The Inequality (8) follows from definitions of $\lambda(\tau_i, t)$ and $\ddot{\text{RBF}}(\tau_i, t)$. We now prove the Inequality (7). From definitions, $\lambda(\tau_i, t) < \ddot{\text{RBF}}(\tau_i, t)$ implies $t > (k-1)T_i - J_i$. There are exactly $(k-1)$ steps in $\lambda(\tau_i, t)$, before starting the linear approximation of the request bound function. As a consequence,

$$\ddot{\text{RBF}}(\tau_i, t) \geq kC_i \tag{9}$$

We now prove that $\dddot{\text{RBF}}(\tau_i, t) - \lambda(\tau_i, t) \leq C_i$. When $t \leq (k-1)T_i - J_i$, the result follows from definitions since $\dddot{\text{RBF}}(\tau_i, t) = \lambda(\tau_i, t)$; otherwise, if $t > (k-1)T_i - J_i$:

$$\left\lceil \frac{t + J_i}{T_i} \right\rceil C_i - (t + J_i)\frac{C_i}{T_i} = C_i \left(\left\lceil \frac{t + J_i}{T_i} \right\rceil - \frac{t + J_i}{T_i} \right) \tag{10}$$

Since $\lceil x \rceil < x + 1$ for any real number x, then the Inequality (10) becomes,

$$\dddot{\text{RBF}}(\tau_i, t) - \lambda(\tau_i, t) \quad \leq \quad C_i$$

To complete the proof, we use Inequality (9):

$$\dddot{\text{RBF}}(\tau_i, t) - \frac{\ddot{\text{RBF}}(\tau_i, t)}{k} \quad \leq \quad \lambda(\tau_i, t)$$

$$(1 - \frac{1}{k})\dddot{\text{RBF}}(\tau_i, t) \quad \leq \quad \lambda(\tau_i, t)$$

The result, $(1-\epsilon)\ddot{\text{RBF}}(\tau_i, t) \leq \lambda(\tau_i, t)$, follows since $k \geq 1/\epsilon$. ■

We now prove that if the approximation scheme returns *infeasible*, then the task is infeasible.

Theorem 10 *If $\forall t \in (0, D_i - J_i]$, $\widetilde{W_i}(t) > t$, then τ_i is infeasible on a unit-speed processor.*

Proof: From Theorem 9, we obtain that $\forall t > 0, \widetilde{W_i}(t) \leq W_i(t)$. Thus, if $\forall t \in (0, D_i - J_i]$, $\widetilde{W_i}(t) > t$ implies that $W_i(t) > t$. As a consequence, τ_i is infeasible. ■

Lastly, we prove that if the approximate test returns *feasible*, the tested task is feasible upon a processor with capacity of $(1+\epsilon)$.

Theorem 11 *If there exists a time instant $t \in (0, D_i - J_i]$ such that $\widetilde{W}_i(t) \leq t$, then τ_i is feasible on a processor with a capacity of $(1+\epsilon)$.*

Proof: Our proof obligation is:

$$\widetilde{W}_i(t) \leq t \Rightarrow W_i(t) \leq (1+\epsilon)t$$

Notice that $\frac{k}{k-1} \leq 1+\epsilon$. In the proof of the Theorem 9, we shown that for every task τ_j: $(1-\frac{1}{k})\ddot{\text{RBF}}(\tau_j, t) \leq \lambda(\tau_j, t)$. As a consequence considering the cumulative request bound functions will lead to $(1-\frac{1}{k})W_i(t) \leq \widetilde{W}_i(t)$. Then, $\widetilde{W}_i(t) \leq t$ can be rewritten:

$$\begin{aligned}(1-\frac{1}{k})W_i(t) &\leq t \\ W_i(t) &\leq \frac{k}{k-1}t \leq (1+\epsilon)t\end{aligned}$$

■

Using the same proof technique as in Theorem 7, we can show the necessity and sufficiency of the testing set $\widetilde{S}_i$ for analyzing task τ_i.

Theorem 12 *For all $t \in \widetilde{S}_i$ such that $\widetilde{W}_i(t) \leq t$, then we also verify that: $\forall t \in (0, D_i - J_i], \widetilde{W}_i(t) \leq t$*

4.2. Approximate Worst-Case Response-Time Bound

Using the optimistic FPTAS presented in the previous section, we can check that a task is feasible upon a unit-speed processor or infeasible upon a $(1-\epsilon)$-speed processor. If it is feasible, then we are able to compute an upper bound on the worst-case response-time of a task as presented in Section 2. As with the first upper bound, if the feasibility algorithm does not conclude that a given task is feasible, then we can use the bound defined in [17].

Definition 5 *Consider a task τ_i such that there exists a time t satisfying $\widetilde{W}_i(t) \leq t$ for a given accuracy parameter ϵ, then an approximate lower bound of its worst-case response-time is defined by:*

$$\begin{aligned}t^* &\stackrel{\text{def}}{=} \min\left\{t \in \widetilde{S}_i | \widetilde{W}_i(t) \leq t\right\} \\ \widetilde{R_i} &\stackrel{\text{def}}{=} \frac{k}{k-1}\widetilde{W}_i(t^*) + J_i\end{aligned}$$

where $k = \lceil \frac{1}{\epsilon} \rceil + 1$.

We now prove that such a method defines an upper bound the worst-case response-time of task τ_i.

Theorem 13 *For every task τ_i such that there exists a time t satisfying $\widetilde{W}_i(t) \leq t$, then:*

$$\widetilde{R_i} \geq R_i$$

Proof: The result is verified if the following condition holds: $\frac{k}{k-1}\widetilde{W}_i(t) \geq W_i(t)$. For task τ_i and all time t, Theorem 9 implies:

$$\left(1-\frac{1}{k}\right)\ddot{\text{RBF}}(\tau_i, t) \leq \lambda(\tau_i, t)$$

If we now consider cumulative request bound functions, the previous inequality becomes:

$$\begin{aligned}\widetilde{W}_i(t) &\geq \left(1-\frac{1}{k}\right)W_i(t) \\ \frac{k}{k-1}\widetilde{W}_i(t) &\geq W_i(t)\end{aligned}$$

Let t_c be the first time instant such that $\widetilde{W}_i(t_c) = t_c$ (i.e., t_c is critical for the approximate optimistic feasibility test), then the task is feasible upon a $(1+\epsilon)$-speed processor. As a consequence and t^* the time instant corresponding to the worst-case response-time of τ_i (i.e., such that $W_i(t^*) = t^*$). Since $W_i(t) \leq \frac{k}{k-1}\widetilde{W}_i(t)$ and both functions are non-decreasing, then we verify $t^* \leq t_c$. Thus, $\widetilde{R_i}$ is an upper bound of τ_i worst-case response time. ■

5. Worst-Case Analysis of Error Bounds

No upper bound of worst-case response times is known to have a constant error bound in comparison with exact values computed by a pseudo-polynomial time algorithm. We now show that the previously presented upper bounds do not achieve constant error bounds. Firstly, we show that the first upper bound dominates the second one. Secondly, we show using a (counter-)example that no worst-case performance guarantee can be achieved for these upper bounds of worst-case response times.

Theorem 14 *If both FPTASs conclude that a task τ_i is feasible, the second upper bound (Definition 5) is always dominated by the first one (Definition 4).*

Proof: Our proof obligation is $\widehat{R_i} \leq \widetilde{R_i}$ or equivalently $\widehat{W}_i(t) \leq \frac{k}{k-1}\widetilde{W}_i(t)$. We consider two cases according to $t > (k-1)T_i - J_i$. If $t \leq (k-1)T_i - J_i$, then the FPTAS do not start any linear approximation. As a consequence we

verify $\widehat{R_i} \leq \widetilde{R_i}$. Now, consider $t > (k-1)T_i - J_i$, then linear approximation are performed by both FPTAS:

$$\frac{k}{k-1}\widetilde{W_i}(t) - \widehat{W_i}(t) = \frac{C_i}{T_i}\left(\frac{t+J_i}{k-1} - T_i + 1\right)$$

Since $t > (k-1)T_i - J_i$, then $\widehat{W_i}(t) \leq \frac{k}{k-1}\widetilde{W_i}(t)$. So, joining both cases: $\widehat{R_i} \leq \widetilde{R_i}$. ■

We now prove that the approximate response time bounds can be far away from their related exact values.

Theorem 15 *For any accuracy parameter ϵ, there exists some task systems for which $cR_i \leq \widehat{R_i}$ for any integer c.*

Proof: Let k be defined by $k = \lceil 1/\epsilon \rceil$. Let us consider a task system with two tasks with the following parameters: τ_1 with $C_1 = (1-\lambda)K$ and $T_1 = K$; τ_2 defined as follows: $C_2 = \lambda K^2$ and $T_2 = K^2 + (K-1)\left\lceil \frac{1-\lambda}{\lambda} \right\rceil$, where $0 < \lambda < 1$, K is an arbitrary integer that is a multiple of k and such that λK is also an integer. Note that $(1-\lambda)K$ will be therefore an integer. We assume that both tasks are not subjected to release jitter constraints (i.e., $J_1 = J_2 = 0$). Using the Rate Monotonic Scheduling policy, the task τ_2 can only be executed λK units of time within every subsequent interval of time of length K. As a consequence, the exact worst-case response time of τ_2 is: $R_2 = K^2$. The approximation switches to a linear approximation at $(k-1)K$ which is strictly less than K^2. Thus we consider that $t > (k-1)K$. The approximate response time analysis leads to:

$$\widehat{W_2}(t) = \lambda K^2 + (t + K - 1)(1-\lambda)$$

The corresponding approximate worst-case response time will be achieved for $\widehat{W_2}(t) = t$. The fixed-point for τ_2 is:

$$t = K^2 + (K-1)\frac{1-\lambda}{\lambda}$$

Therefore, the approximate response-time is strictly larger than the exact one, and can be made arbitrarily large. Note that τ_2 is feasible according to the pessimistic approximate feasibility test. But, when λ tends to 0, then $\frac{1-\lambda}{\lambda}$ tends to infinity and so do the approximate worst-case response time, while the exact value is still bounded by a constant integer. So we can always define a constant c such that: $cR_i \leq \widehat{R_i}$ by setting λ using an appropriate value. ■

Since the second upper bound (obtained from the optimistic FPTAS) is dominated by the first one, then it achieves no performance guarantee neither.

6. Experiments

We randomly generated task sets in order to compare our best upper bound to other known ones. Unbiased utilizations were generated using the UUniFast algorithm [4]. Periods T_i are randomly generated in the interval $[1, 2500]$ and worst-case execution time C_i are computed as $C_i = u_i T_i$. Deadlines D_i are randomly generated within the interval $[C_i, T_i]$. A uniform law was used to generate random numbers. C_i, D_i, T_i was then rounded to the closest integer. The utilization factor varies from 0.5 to 0.9 (step 0.1) and for every value, the same number of task sets has been generated.

Experiment parameters are the task number n, the utilization factor U and the accuracy parameter ϵ. For fixed parameters, every experiment is replicated 100 times in order to achieve unbiased statistics. We compared our first bound (denoted $FHGR$) with two known upper bounds computed in linear time for every task τ_i: SH_i is an upper bound computed using the rounding principle presented in [17] and BB_i is an improvement of this upper bound presented in [3]:

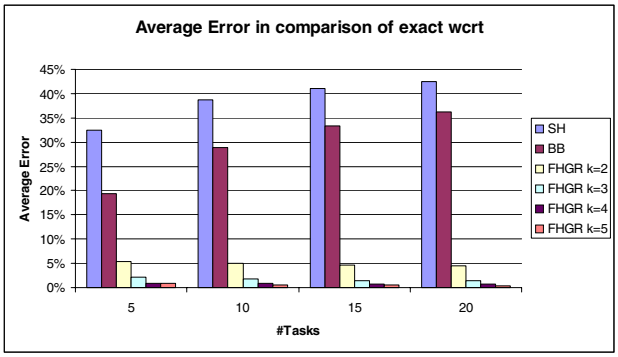
$$SH_i = \frac{\sum_{j=1}^{i} C_j}{1 - \sum_{j=1}^{i-1} U_j} \quad (11)$$

$$BB_i = \frac{C_i + \sum_{j=1}^{i-1} C_j(1-U_j)}{1 - \sum_{j=1}^{i-1} U_j} \quad (12)$$

In order to compare these bounds, only tasks accepted by our approximate feasibility tests have been considered (otherwise, no upper bound can be computed using our method). We monitored two indicators: the average error in comparison with exact values of worst-case response times (i.e., $(ub - R_i)/R_i$, where ub is an upper bound) and the rate of tasks stated "infeasible" using the upper bound (i.e., $ub > D_i$) for feasible tasks (i.e., $R_i \leq D_i$). Numerical results are presented in Figure 2. In the first graph, the average errors are presented for various values of k; the results show that our method clearly improves the previous known bounds, even if $k = 2$ (i.e., the smallest possible value since $k = \lceil 1/\epsilon \rceil$ and $0 < \epsilon < 1$). The average error is less than 1% when $k = 5$ (i.e., $0.2 \leq \epsilon < 0.25$). Concerning the second graph, we see that our approach is less pessimistic since only few feasible tasks are not accepted by our method and less sensitive to the task number.

7. Conclusion

We define two upper bounds of worst-case response time for static priority task subjected to release jitters. The corresponding algorithms are parametric in the sense that an accuracy parameter ϵ is used to define the time spent before starting an approximate analysis. If the accuracy parameter is a very small number, then our experiments show that upper bounds are very close to exact worst-case response times, but still computable in polynomial time according to task parameters and the constant $1/\epsilon$.



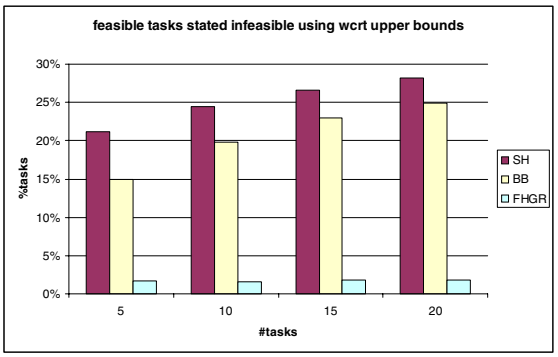


Figure 2. Simulation Results

We must mention that the existence of an algorithm for approximating worst-case response time in polynomial time is still an interesting open problem, even if basic task sets are considered (e.g., deadlines equal periods). Another interesting issue is to study if the approach presented in this paper can be extended to EDF scheduling.

References

[1] K. Albers and F. Slomka. An event stream driven approximation for the analysis of real-time systems. In *Euromicro Conference on Real-Time Systems*, pages 187–195, 2004.

[2] K. Albers and F. Slomka. Efficient feasibility analysis for real-time systems with edf scheduling. In *Design, Automation and Test in Europe*, 2005.

[3] E. Bini and S. Baruah. Efficient computation of response time bounds under fixed-priority scheduling. In *Real-Time and Network Systems*, 2007.

[4] E. Bini and G. Buttazzo. Measuring the performance of schedulability tests. *Journal of Real-Time Systems*, 30(1–2):129–154, 2005.

[5] E. Bini, G. Buttazzo, and G. Buttazzo. Rate monotonic scheduling: The hyperbolic bound. *IEEE Transactions on Computers*, 2003.

[6] S. Chakraborty, S. Kunzli, and L. Thiele. Approximate schedulability analysis. In *Real-Time Systems Symposium*, 2002.

[7] D. Chen, A. Mok, and T. Kuo. Utilization bound revisited. *IEEE Transactions on Computers*, 2003.

[8] N. Fisher and S. Baruah. A fully polynomial-time approximation scheme for feasibility analysis in static-priority systems with arbitrary relative deadlines. In *Euromicro Conference on Real-Time Systems*, pages 117–126, July 2005.

[9] C. Han and H. Tyan. A better polynomial-time schedulability test for real-time fixed-priority scheduling algorithm. In *Real-Time Systems Symposium*, 1997.

[10] S. Lauzac, R.Melhem, and D. Mossé. An improved rate-monotonic admission control and its applications. *IEEE Transactions on Computers*, 2003.

[11] J. Lehoczky. Fixed priority scheduling of periodic tasks with arbitrary deadlines. In *Real-Time Systems Symposium*, pages 201–209, 1990.

[12] J. Lehoczky, L. Sha, and Y. Ding. The rate monotonic scheduling algorithm: exact characterization and average case behavior. In *Real-Time Systems Symposium*, pages 166–171, 1989.

[13] Y. Manabee and S. Aoyagi. A feasible decision algorithm for rate monotonic and deadline monotonic scheduling. *Real-Time Systems Journal*, pages 171–181, 1998.

[14] D.-W. Park, S. Natarajan, A. Kanavsky, and M. Kim. A generalized utilization bound test for fixed-priority real-time scheduling. In *Workshop on Real-Time Systems and Applications*, 1995.

[15] P. Richard. Polynomial time approximate schedulability tests for fixed-priority real-time tasks: some numerical experimentations. In *Real-Time and Network Systems*, 2006.

[16] P. Richard and J. Goossens. Approximating response times for static-priority tasks with release jitters. In *Euromicro Conference on Real-Time Systems (WiP session)*, 2006.

[17] M. Sjodin and H. Hansson. Improved response time analysis calculations. In *Real-Time Systems Symposium*, 1998.

[18] K. Tindell. *Fixed Priority Scheduling of Hard Real-Time Systems*. PhD thesis, University of York, 1994.